ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

DIPLOMOVÁ PRÁCE

2019 **Diep Luong**

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra telekomunikační techniky

Systém Asterisk jako komponenta laboratorní IMS sítě

leden 2019

Diplomant: Bc. Diep Luong

Vedoucí práce: Ing. Pavel Troller, CSc.

Čestné prohlášení

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

Datum: 8.1.2019

.....
podpis diplomanta

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Luong** Jméno: **Diep** Osobní číslo: **373713**

Fakulta/ústav: **Fakulta elektrotechnická**

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

Studijní program: **Komunikace, multimédia a elektronika**

Studijní obor: **Sítě elektronických komunikací**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Systém Asterisk jako komponenta laboratorní IMS sítě

Název diplomové práce anglicky:

The Asterisk system as a component of laboratory IMS network

Pokyny pro vypracování:

Proveďte analýzu vhodnosti systému Asterisk pro možnost upravit jej tak, aby byl použitelný jako komponenta laboratorní IMS sítě. Pro určenou nejvhodnější komponentu proveďte potřebné úpravy / doplnění funkcionality a integrujte ji do laboratorní IMS sítě sestavené na bázi systému Kamailio.

Seznam doporučené literatury:

[1] Camarillo, G.; García-Martín, M.A.: The 3G IP multimedia subsystem (IMS) : Merging the Internet and the Cellular Worlds (2 ed.). Chichester [u.a.]: Wiley 2007. ISBN: 0-470-01818-6.
[2] Syed, A.A.; Mohammed, I.: IP multimedia subsystem (IMS) handbook. Boca Raton: CRC Press 2009. ISBN: 1-4200-6459-2.

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: **20.09.2017** Termín odevzdání diplomové práce: **09.01.2018**

Platnost zadání diplomové práce: **18.02.2019**

Ing. Pavel Troller, CSc. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen 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.

23. 2. 2018
Datum převzetí zadání

Podpis studenta

Anotace:

Tato diplomová práce se zabývá analýzou možnosti úpravy telekomunikačního systému Asterisk a implementaci hlasového dialogu VoiceXML. Návrh a realizace jednotlivých funkcí pro Asterisk. Testování funkčnosti navržených funkcí v testovacím nástroji SIPp. V diplomový práci se budeme také věnovat ukázky instalace, konfigurace, implementace Asterisku.

Klíčová slova: SIP, VoiceXML, Asterisk, XML, IMS, Festival

Summary:

This diploma thesis deals with the analysing of the possibilities of modifying Asterisk telecommunication system and implementation of VoiceXML voice dialogue. Desing and implementation of individual functions for Asterisk. Testing functionality of suggested functions in the SIPp test tool. In the diploma thesis we will also give examples of installation, configuration, implementation of Asterisk.

Index Terms: SIP, VoiceXML, Asterisk, XML, IMS, Festival

Obsah

Seznam obrázků

Kapitola 1

Úvod

Cílem této diplomové práce je integrace hlasového dialogu do telekomunikačního systému, popis jejich vlastností. Jedná se o opensource softwarový telekomunikační systém Asterisk od společenosti Digium. Nejprve je v teoreticky popsána telekomunikační struktura, její vlastnosti, funkce a využití. To samé je popsané pro telekomunikační systém a hlasový dialog. Dále následuje analýza zpracování VoiceXML a hledají se možností využití VoiceXML v systému Asterisk jakožto rozšíření funkcionalit. Po teoretické části navazuje praktická, jejíž součástí je instalace, konfigurace a úprava systému Asterisk. Implementace a zpracování vstupních parametrů VoiceXML. Rozšíření Asterisk o funkcionalitu TTS a další komponenty. Vytvoření scénářů testování služeb. V testování se testuje správnost implementovaných funkcionalit.

Kapitola 2

Stručný popis IMS sítě

IMS (IP Multimedia Subsystem) je jeden z mnoha projektů organizace 3GPP. IMS bylo definované na průmyslovém fóru 3G.IP v roce 1999. 3G.IP vyvinula počáteční architekturu IMS, která byla předána organizaci 3GPP. První znění o využití multimédia, které bylo založené na protokolu SIP (Session Initiation Protocol) je v release 5. Další důležitý milník přišel s release 6, který přidal podporu technologie WLAN. Mezi další rozšíření patří například IMS emergency services, detekce řeči a Push to Talk. [6]

Původní myšlenkou IMS byl bezdrátový standard pro 3GPP. S rozvojem sítí bude potřeba přejít z původní technologie přepojování okruhu (CS - circuit-switched-style) na systém přepojování paketů (PS - packet-switched). Důvodem tohoto přechodu je nahrazení starších signalizačních protokolů za nový signalizační protokol SIP, který je dnes široce rozšířen. A používá se v rámci IMS pro komunikaci mezi zařízeními a dalšími jednotlivými bloky. [6]

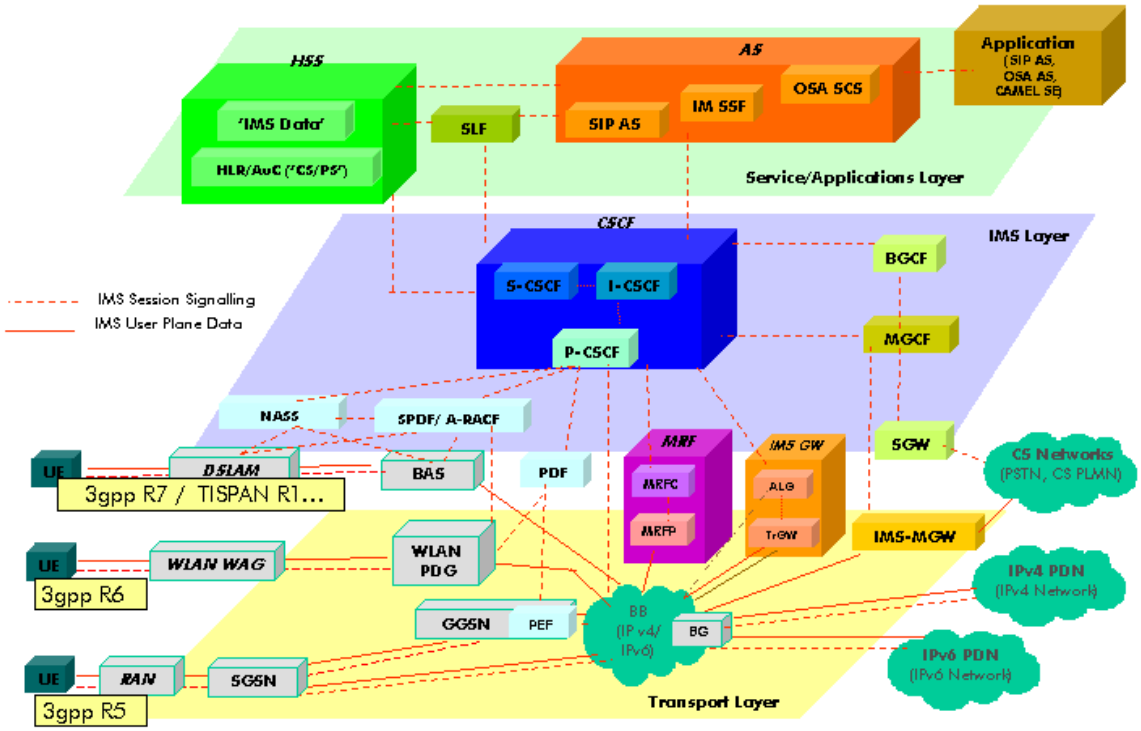
2.1 Architektura IMS

Na architekturu IMS lze z hlediska rozdělení vrstev nahlížet jako na model čtyřvrstvý nebo třívrstvý. Každá z těchto vrstev zajišťuje různé funkce. A každá vrstva je složena z několika bloků, které zajišťují určité funkce systému. [6] [4] Viz. 2.1

Architektura systému IMS může být rozdělena do několika hlavních bloků nebo vrstev:

- Transportní vrstva
- IMS vrstva
- Aplikační vrstva

Komunikace mezi jednotlivými bloky IMS sítě probíhá přes rozhraní (Interface). Jednotlivé bloky mohou komunikovat s několika jinými bloky. Jednotlivé komunikace mají vlastní rozhraní s vlastním označením.



Obrázek 2.1: Grafické znázornění jednotlivých vrstev IMS sítě a klíčových prvků sítě.[6]

2.1.1 Transportní vrstva

Transportní vrstva (přístupová vrstva) je nejnižší vrstvou architektury IMS dle 2.1. Uživatel se s její pomocí může připojit do IMS sítí ze širokého spektra koncových zařízení používající většinou standardní protokol IP. IMS zařízeními mohou být např. mobilní telefony, počítače, PDA systémy, digitální telefony atd. Samostatné připojení analogového telefonu není možné, protože se neumí přímo připojit do IP sítě. IMS však podporuje možnost přenosu analogových dat přes bránu PSTN, která umožňuje převést analogová data na digitální. Další možností připojení k IMS síti je přes pevné spojení, např. xDSL, Ethernet atd.

Do této vrstvy dále spadá například SGSN (Serving GPRS (General Packet Radio Service) Support Node) a GGSN (Gateway GPRS Support Node) mobilních sítí, DSLAM (Digital Subscriber Line Access Multiplexer), WLAN (Wireless Local Access Network), MPLS (Multiprotocol Label Switching), pateřní síť operátora atd. [6]

Úkolem této vrstvy je zahájit a ukončit signalizaci SIP. Dále má za úkol nastavení relací a služeb poskytovatele. [5]

2.1.2 IMS vrstva

IMS vrstva se nachází mezi transportní a aplikační vrstvou. Jedná se o jednu z nejdůležitějších vrstev, protože se zde nachází veškerá signalizace. Je složena ze čtyř bloků (čtvrtá je databáze uživatelů HSS (Home Subscribe Server)). Tyto části se nazývají CSCF (Call/Session Control Function) a podle funkce je dělíme do tří skupin:

1. P-CSCF - Proxy-Call Session Control Function
2. I-CSCF - Interrogating-Call Session Control Function
3. S-CSCF - Session-Call Session Control Function

IMS CSCF je část architektury, která poskytuje registraci uživatelů. Veškerá směrování a signalizace probíhají přes protokol SIP, což také podporuje QoS (Quality of Services) z transportní vrstvy.

2.1.2.1 P-CSCF

P-CSCF je prvním kontaktním bodem všech uživatelů v IMS síti. A všechna komunikace z UE je nejprve odeslána P-CSCF. Podobně jsou také ukončeny všechny SIP signalizace ze sítě. A jsou odeslány z P-CSCF do UE. Příklady realizací P-CSCF funkcí:

- SIP komprese

  SIP protokol je signalizačním protokolem v textové formě. Obsahuje velké množství hodnot v hlavičce a parametry hlaviček, což znamená, že SIP zpráva může být velká. 3GPP zajišťuje podporu SIP kompresi a dekompresi mezu UE a P-CSCF. Pokud chce UE přijímat komprimované zprávy, musí P-CSCF zprávu zkomprimovat.

- IPSec security association

  P-CSCF je zodpovědné za stanovení počtu přidružených IPSec směrem k IMS terminálům. Zabezpečení IPSec (SA) se používá pro ochranu integrity (tj. schopnost detekovat, zda byl změněn obsah zprávy od svého vzniku) a ochranu SIP signalizace. Toho je dosaženo v průběhu SIP registrace jak v UE, tak u P-CSCF při sjednávání IPSec (SA). Při ověřování uživatelů P-CSCF potvrzuje její identitu. Tyto informace předá ostatním uzlům v síti, které jsou využity pro další účely, jako je například poskytování individuálních služeb a záznam generovaných účtů.

- Interakce s Policy Decision Function (PDF)

  V rámci P-CSCF může být obsažen PDF, který může být integrován s P-CSCF nebo implementován jako samostatné zařízení. PDF autorizuje zdroje jsou v mediální rovině a v této spravuje QoS. Na základě získaných informací z PDF je schopen získat autorizované IP QoS informace, které budou předány GGSN v případě, že GGSN bude potřebovat provést Service-Based Local Policy. A navíc prostřednictvím PDF je IMS schopné dodávat informace pro srovnávání IMS účtování do sítě GPRS. Přijímací GPRS tyto informace srovnává s účtování ze sítě. Což umožňuje sloučení zpoplatněných datových záznamů z přicházející IMS a GPRS sítí ve fakturačním systému.

- Detekce nouzového spojení

  Relace nouzového hovoru v IMS síti stále nejsou plně specifikované. To je důležité, protože P-CSCF v IMS síti rozpoznává pokusy o tísňové relace, a provede UE pro použití tísňových volání. P-CSCF by mělo být schopné vybrat nouzové CSCF ke zpracování tísňových volání. To je zapotřebí, protože v případě roamingu je schopné IMS přiřadit S-CSCF, které je v domácí síti, a domácí S-CSCF není schopné směrovat požadavky na správné tísňové centrum.

IMS síť může obsahovat několik P-CSCF pro škálovatelnost a redundanci. Každá z nich slouží několika IMS terminálům, podle kapacity uzlu.

P-CSCF může být umístěn v domácí síti nebo navštívené síti. Když je základní síť založena na GPRS, tak P-CSCF se vždy nachází poblíž GGSN (Gateway GPRS Support Node). Předpokládá se, že první IMS síť bude nakonfigurována s GGSN a P-CSCF. [17]

2.1.2.2 I-CSCF

Interrogating-CSCF je kontaktní bod v síti operátora pro všechna připojení určená pro účastníka na provozované síti. Když SIP potřebuje odeslat SIP zprávu konkrétní doméně, provede DNS vyhledávání, aby získal adresu SIP serveru na této doméně. Záznamy DNS domény odkazují na I-CSCF domény. Proto I-CSCF zpracovává příchozí domény. Pro I-CSCF jsou přiděleny čtyři úkoly:

- Chce-li získat název následující skok (to může být S-CSCF nebo aplikační server) z Home Subscriber Server (HSS).

Rozhraní mezi I-CSCF a HSS, nebo SLF je založeno na Diameter protokolu.

- Přiřazené S-CSCF je na základě přijatých schopností od HSS.

Přiřazené S-CSCF probíhá v případě, že uživatel je zaregistrován v síti nebo uživatel obdržel SIP požadavek, zatímco neregistrovaní mají v síti jiné služby související s jejich neregistrovaným stavem (např. hlasová pošta).

- Pro směrování příchozích požadavků, které jsou přiřazeny S-CSCF nebo aplikačním servem.
- Zjištění funkčnosti Topology Hiding Inter-network Gateway (THIG), což znamená, že optimálně I-CSCF může šifrovat část SIP zpráv, které obsahují citlivé informace o doméně, například počet serverů v doméně, jejich DNS názvy nebo kapacity.

IMS síť může obsahovat několik I-CSCF pro škálovatelnost a redundanci.

I-CSCF je obvykle umístěn v domácí síti, ale v některých případech (jako je I-CSCF (THIG)) může být umístěn v navštívené síti. [17]

2.1.2.3 S-CSCF

S-CSCF je centrální uzel a lze jej nazvat jako "IMS mozek". S-CSCF je SIP server, který je zodpovědný za zpracování registračních procesů, rozhodování o směrovosti, udržování stavů relací a ukládání profilů služeb.

Když uživatel odešle žádost o registraci, bude přesměrován do S-CSCF. Následně si S-CSCF stáhne ověřovací data a profil služeb z HSS pomocí Diameter. Na základě ověřovacích dat generuje výzvu UE. Po obdržení odpovědi a ověření S-SCSF přijímá registraci, a začíná sledovat stav registrace. Po registraci uživatel může zahájit příjem IMS služeb.

Profil služeb je soubor uživatelsky-specifikovaných informací, které jsou uloženy v HSS. Jedná se o sadu aktivačních událostí, které mohou způsobit, že SIP zpráva prochází přes jednu nebo více aplikací. S-CSCF stahuje jednotlivé profily služeb spojené s konkrétní veřejnou uživatelskou identitou (např. emailová adresa), kdy je identita veřejně registrovaná v IMS. Informace z profilu služeb lze použít pro rozhodování S-CSCF, pro zvolení a kontaktování určitého aplikačního serveru, když uživatel odešle SIPový požadavek nebo obdrží žádost od někoho. Profily služeb mohou obsahovat informace o tom, jaký druh média musí S-CSCF používat.

Kromě toho je S-CSCF schopen odesílat informace týkající se účetnictví do Online Charging System pro online účtování (tj. podpora předplatné pro předplatitele) [14]

V IMS síti se obvykle vyskytuje několik S-CSCF pro škálovatelnost a redundanci. Pro každé S-CSCF slouží určitý počet terminálů, pod kapacity uzlu.

S-CSCF je vždy umístěn na domácí síti. [17]

2.1.2.4 Databáze

Existují dva typy definovaných databází v IMS síti: Home Subscriber Server (HSS) a Subscription Locator Function (SLF). [14]

2.1.3 Síťové funkce

Koncept IMS sítě by měl být univerzální. Jelikož není možné přemístit všechny uživatele na nová zařízení, která budou schopná se do IMS sítě připojit přes SIP v jeden okamžik (uživatel si musí koupit nový telefon), IMS síť musí umožnit spojení hovoru mezi uživateli, jejichž zařízení používá CS, či PS sítě se všemi možnými druhy signalizace a zároveň také opačně - ze starších sítí do IMS.

K tomu slouží hned několik entit. S-CSCF rozhoduje, jestli dojde k předání relace do jiné sítě. Pokud se zjistí, že došlo k předání relace do jiné sítě, tak se tato relace předá na BGCF (Breakout Gateway Control Function). Zde musí dojít k několika dalším rozhodnutím. A pokud je příslušná relace či hovor určen pro sítě stejného operátora a pouze starších typů sítě, pak v takovém případě relace putuje od BGCF k MGCF (Media Gateway Control Function), kde musí dojít k transformování signalizace z protokolu SIP na ISUP (ISDN User Part). MGCF kontroluje další prvky sítě jako jsou IMS-MGW (MGW). Odtud se převádí audio mezi protokoly v obou typech sítě. Pokud hovor přichází z CS sítě, prochází nejprve přes MGCF, kde jsou provedeny výše zmíněné úkony a dále je už SIP signalizace posílána na I-CSCF, který zajistí správné směrování.

Dalším případem může být předání do IMS sítě jiného operátora, poslední možností je starší typ sítě jiného operátora. V obou případech dochází k předání na BGCF cizí sítě a dále se pokračuje postupem přes MGCF popsaným výše.

2.1.4 Další funkce

IMS v sobě zahrnuje mnoho dalších funkcí, jejichž činnost může být okrajová nebo v závislosti na typu sítě ani nemusejí být implementovány. Jedná se například o vyhledávání geograficky nejbližších středisek pro tísňová volání, různé funkce pro technické zajištění spojení relací mezi sítěmi dvou operátorů nebo bezpečnostní funkce pro zabezpečení dat v síti operátora.

2.2 Application Layer - Aplikační vrstva

Na vrcholu architektury IMS sítě je Aplikační vrstva. Tři výše popsané vrstvy leží pod Aplikační vrstvou, poskytují jednotnou a standardizovanou síťovou platformu, která umožňuje nabízet velké množství multimediálních služeb. Tyto služby jsou provozovány aplikačními servery (AS - Application Server), které odpovídají na hostování a vykonávání služeb za použití SIP a Diameter protokolu. Jeden aplikační server může hostovat více služeb, což přináší flexibilitu. Další prvek které spadají do aplikační vrstvy jsou např. HSS, MRF atd.

2.2.1 HSS - Home Subscribe Server

HSS je hlavním úložištěm dat o uživatelích a uživatelských nastavení v rámci IMS. V této databázi jsou především ukládána data jako: uživatelské identity, registrační údaje, přístupové parametry, IP informace, informace o poloze uživatele, čísla atd. [2]. Pro komunikaci s ostatními prvky sítě HSS využívá protokol DIAMETER.

Uživatelské identity se dělí na dvě skupiny: privátní (IMPI - IP Multimedia Private Identity) a veřejné (IMPU - IP Multimedia Public Identity). Obě tyto identity nejsou ve formě telefonního čísla, ale jako URI (Uniform Resource Indentifier).

- Privátní identita je unikátní a pevně přidělena globální identita, kterou přiřadí operátor domácí sítě. Používá se k registraci, autorizaci, administraci a účetním účelům. Každý uživatel má jednu nebo více IMPI.
- Veřejnou identitu používají uživatelé k vyžádání komunikace s jinými uživateli v síti. K jedné privátní identitě může být přiřazeno více identit veřejných a zároveň IMPU může být sdílena na více koncových zařízení.

K navázání spojení slouží přístupové parametry, jako je autorizace uživatele, autorizace roamingu a jména přidělená S-CSCF. HSS také poskytuje údaje o specifických požadavcích každého uživatele a vlastnosti S-CSCF. Na základě těchto informací I-CSCF volí pro uživatele nejvýhodnější S-CSCF.

HSS neposkytuje informace S-CSCF umístěných v jiných sítích, což je důležité k zabezpečení uložených uživatelských dat před přístupem z nedůvěryhodných sítí. Tuto ochranu realizují entity P-CSCF a I-CSCF.

Kromě funkcí týkajících se funkčnosti IMS, HSS také obsahuje podskupinu HLR/AUC (Home Location Register and Authentication Center). HLR neboli domovský lokalizační registr je prvek nezbytný pro přístup do paketově spínaných sítí (nutná pro funkčnost entit SGSN a GGSN), do okruhově spínaných sítí nutné pro funkčnost entit SGSN a GGSN a do sítí okruhově spínaných obsahující zejména MSC/MSC servery. To uživateli umožňuje přístup k CS službám a podporuje roaming do GSM/IMTS CS sítí.

Autentizační centrum (AUC) zajišťuje řízení autentizace každého mobilního uživatele pomocí autentizační funkce ACF, která pro každého uživatele generuje dynamická zabezpečovací data. Data jsou použita pro vzájemnou autentizaci IMSI (International Mobile Subscriber Identity) a sítě. Tato data jsou zároveň používána pro poskytování ochrany integrity a šifrování komunikace skrz rádiovou přenosovou cestu mezi UE a sítí.

2.2.2 SLF - Subscription Locator Function

SLF je databáze, která mapuje uživatele HSS. Používá rozlišující mechanismus, který umožňuje I-CSCF, S-CSCF a AS určit adresu HSS obsahující informace konkrétního uživatele. Může existovat několik HSS v jedné IMS síti.

2.2.3 PDF - Policy Decision Function

Odpovídá za tvorbu pravidel pro rozhodování, která jsou generována na základě informací o relaci a multimediálním toku. Od release 5 je přímo součástí P-CSCF. Je založena na mechanismu SBLB (Service Based Local Policy), který plní funkci jako jsou:

- Ukládání informací o relaci a multimédiích (IP adresa, čísla portů, šířka pásma atd).
- Na výzvu GGSN poskytovat rozhodnutí o autorizaci založené na těchto informacích.
- Aktualizovat rozhodnutí o autorizaci po změně těchto informací a případně autentizaci odebrat.
- Generování znaku Media Authorization Token, jenž identifikuje relaci a PDF.

2.2.4 AS - Application server

Aplikační server (AS) není přímo IMS entita, ale spíš jde o funkci na vrcholu IMS struktury. Protože poskytuje v rámci IMS sítě důležité multimediální služby, jsou li zařazeny mezi její prvky. Mohou být umístěny v domácí či jiné síti nebo také jako samostatný AS. Hlavní funkce jsou:

- Možnost zpracování příchozích SIP relací z IMS sítě
- Schopnost vytvářet SIP dotazy

- Schopnost zasílat účetní údaje CCF (Charging Collection Function) a ECF (Event Changing Function).

Dle specifikací [ETSI TS 123 078] a [3GPP TS 23.228] poskytované služby AS nejsou omezeny služby založené na SIP, ale uživatel IMS také mohou využívat CAMEL (Customized Application for Mobile network Enhanced Logic) a OSA (Open Service Architecture). AS se dělí na tři druhy:

2.2.4.1 SIP AS (SIP Application Server)

Aplikační server založený na protokolu SIP, který může hostit široké spektrum multimediálních služeb - nejčastěji konferenční a prezenční služby nebo zasílání zpráv.

2.2.4.2 OSA-SCS (Open Service Access - Service Capability Server)

Tento server zprostředkovává vykonávání OSA služeb v IMS síti - slouží jako gateway (brána) a pomocí standardizovaného rozhraní umožňuje komunikaci IMS s externími OSA AS.

2.2.4.3 IM-SSF (IP Multimedia Service Switching Fuction)

Třetím typem je AS sloužící jako gateway pro spojení se sítěmi implementujícími služby CAMEL, hojně využívanými v GSM sítích. Chová se jako brána mezi SIP a CAMEL službami umožňující volání z IMS sítě CAMEL službám. [14]

2.2.5 MRF - Media Resource Function

Entita MRF je soubor funkcí pro obsluhu a zpracování médií. Zajišťuje například real-time převod multimediálních dat, rozpoznání hlasu, multimediální konference, přehrávání tónů a oznámení. Dále se dělí na:

- MRFC (MRF Controller) - interpretuje informace a zpracovává signalizaci mezi AS a S-CSCF, ovládá MRFP pomocí rozhraní H.248.
- MRFP (MRF Processor) - zpracovává, vytváří a slučuje toky médií, implementuje všechny funkce pro jejich zpracování a sdílení.

Kapitola 3

Asterisk

Asterisk je open source implementace telefonní pobočkové ústředny (PBX), který byl vytvořen v roce 1999 Markem Spencerem z firmy Digium. Tato pobočková ústředna umožňuje telefonům propojení s různými hardwarovými technologiemi, a vzájemně volat ostatním a připojovat telefonní služby, jako jsou veřejné telefonní síť (PSTN) a VoIP.

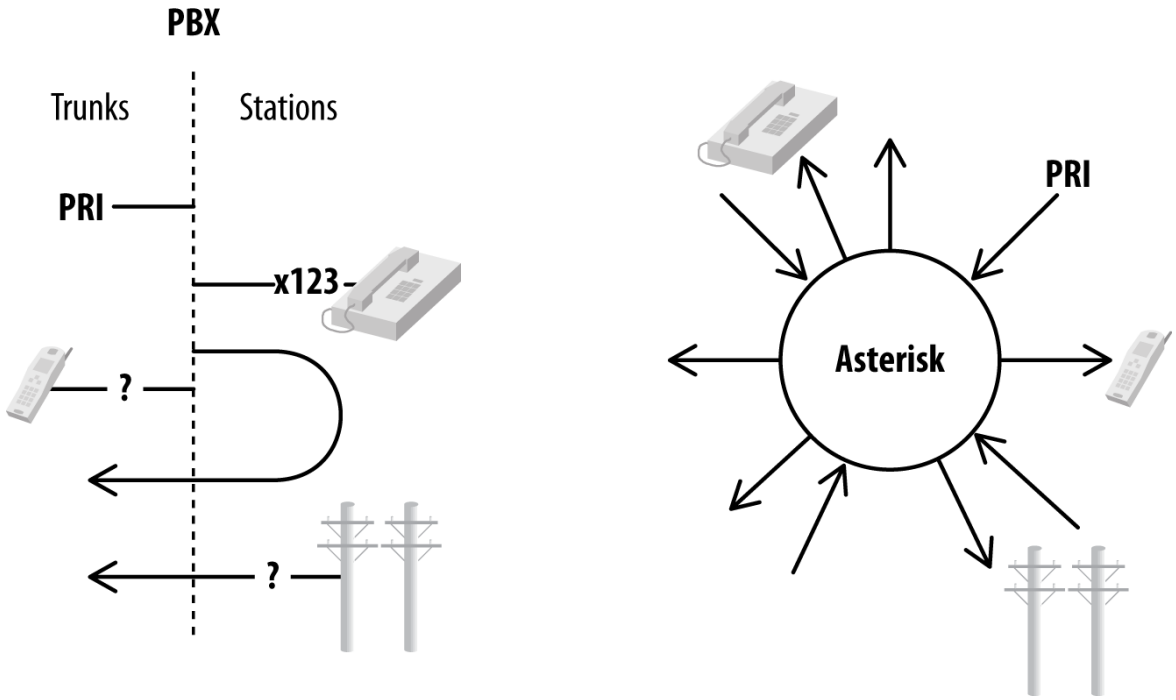
Asterisk je šířen pod dvěma licencemi. Jedna je GNU GPL a druhá je proprietární, která umožňuje využití uzavřených a patentovaných zdrojových kódů jako je např. G.729 kodek. Původně byl Asterisk navržen pro operační systém Linux (Unix), ale v současné době je možné provozovat tento systém i na dalších unixových systémech (NetBSD, OpenBSD, FreeBSD, Mac OS X, Solaris apod.). Systém obsahuje nadstavbu s grafickým rozhraním a je možné spustit i na operačním systému Windows. [3]

3.1 Popis architektury

Architektura Asterisku je velice odlišná od jiných tradičních pobočkových ústředen (PBX), v tom že obsahuje DialPlan, který zastřešuje všechny příchozí a odchozí hovory(kanály) stejným způsobem.

V tradičních pobočkových ústřednách je přístup rozdělen mezi účastnickou stanici (telefonní přístroje) a přenašeč (připojení vnějších zdrojů). To znamená problém v případě nainstalování externích bran na straně stanic nebo směrování externích hovorů. Také existuje koncept tzv. off-site, který je mnohem obtížnější na implementaci do tradičních pobočkových ústředen, protože systém neumožňuje externím zdrojům přístup k vnitřním funkcím. K tomu aby tradiční ústředny mohly nabízet tyto funkcionality, vyžaduje komplexní a proprietární software, který musí být nainstalován v pobočkové ústředně jako rozšíření protokolu.

Naproti tomu je Asterisk pouze softwarová ústředna, která je nainstalována na určitém serveru. Dále neobsahuje koncept přenášečů nebo koncových stanic. Příchozí a odchozí kanály, které procházejí systémem jsou řešeny v rámci Dialplanu stejným způsobem. To znamená, že například interní uživatel může také existovat na konci vnějším přenašeči (např. mobilní telefon) a je řešen stejným způsobem jako interní uživatel. Na obrázku 3.1 jsou znázorněny rozdíly mezi tradiční pobočkovou ústřednou a Asteriskem [18]



Obrázek 3.1: Rozdíl v architektuře mezi Asteriskem a pobočkovou ústřednou PBX [18]

3.2 Možnosti využití

3.2.1 Pobočková ústředna PBX

Asterisk umožňuje vytvoření pobočkové ústředny (PBX), která má stejné vlastnosti jako ústředna klasická.

Mezi hlavní vlastnosti patří:

- Podpora VoIP a možnosti dokoupení dodatečných karet pro přípojení analogové linky nebo ISDN.
- Možnost hlasové schránky
- Přesměrování hovorů a služeb "Find me/Follow me"
- Konferenční hovory
- Monitorování a reportování

3.2.2 VoIP a protokolová brána

Asterisk podporuje jak tradiční TDM, tak i VoIP protokoly, které umožňuje vytváření brány/mosty mezi dvěma či více telefonními standardy. Může obsahovat více typů signalizací, jako jsou např. mediální brána pro trans-kódování audia. Dále Asterisk umí odbírat či přidávat předdefinované číslice k volanému nebo volajícímu číslu při průchodu hovorem.

Podpora VoIP protokolů: SIP, IAX/IAX2, H.323, MGCP, Skinny

3.2.3 Server pro hlasovou schránku (Media Server)

V případě nezastižení volaného má účastník možnost zanechat hlasovou zpráva volanému. Problém hlasové schránky v současných digitálních ústřednách je jejich finanční náročnost, která může obsahovat nároky na výpočetní výkon a úložné místo pro archivaci všech aktivních hlasových zpráv či licencí. Asterisk obsahuje funkcionalitu nazvanou Voicemail, která umí nahrávat hlasovou zprávu a přeposílat ji rovnou uživateli na e-mail. Může být také integrovaná se staršími analogovými ústřednami nebo ústřednami, které touto funkcionalitou nedisponují.

3.2.4 Konferenční server

Konferenční server umožňuje skupině lidí uskutečnit tzv. konferenční hovor, který se odehrává na daném serveru. Tuto funkci má Asterisk plně integrovanou a podporuje základní vlastnosti konference. Jednotlivé konferenční místnosti mohou být zabezpečeny pomocí předurčeného PINu, který je nutný zadat pomocí DTMF pro přístup do místnosti

3.2.5 Call centrum

Pomocí Asterisku lze také definovat call centra, která umí využívat vlastnosti několika úrovní IVR. Systém je schopen inteligentně distribuovat hovory mezi agenty, nebo zařazovat volající do front a dále je dynamicky informovat o tom, v jakém stavu je jejich požadavek na vyřízení hovoru. V průběhu čekání ve frontě umí Asterisk přehrávat hudbu (Music on Hold) nebo předávat hlasovou zprávu agentům.

3.2.6 Interaktivní hlasový systém (IVR) server

Asterisk pomocí svého IVR umožňuje uživatelům využívat databázové menu předem nahraných hlasových zpráv. Volající může s ústřednou komunikovat pomocí tónové volby, nahrávat odpovědi, dotazovat se databáze a využívat AGI skripty k provedení určitých úkolů, jako je např. změna volaného čísla na základě výsledku z IVR, která dokáže hovor přesměrovat ke správnému uživateli, či skupině. [1]

3.3 Signalizace a podporované protokoly

Asterisk podporuje následující protokoly:

Signalizace

- IAX (Inter-Asterisk Exchange)
- H.323
- SIP (Session Initiation Protocol)
- MGCP (Media Gateway Control Protocol)

- SCCP (Skinny Call Control Protocol)

Podporované kodeky

- G.711 a/u
- G.726
- G.723.1
- G.729
- GSM
- iLBC
- Speex

3.4 Funkce systému (Moduly)

Asterisk je postaven na modulech. Moduly obsahují komponenty, které se načítají dle specifikace poskytovatele a jeho funkcionality. Ty jsou načteny na základě konfiguračního souboru modules.conf který je uložen v /etc/asterisk/. V případě úpravy/pozměnění těchto modulů stačí dle potřeby vytvořit takzvaný vlastní "patch". Protože je Asterisk open-source distribuce a je k dispozici stažitelný zdrojový kód jednotlivých modulů.

Systém Asterisk je možné spustit bez jakéhokoli modulu. V tomto případě systém nebude schopen nic vykonávat. Což je cílem architektury Asterisk a jeho modulární charakteristiky.

3.4.1 Rozdělení modulů

3.4.1.1 Kanálové ovladače

Komunikují se zařízeními mimo Asterisk a přenášejí tuto konkrétní signalizaci nebo protokol na jádro systému.

3.4.1.2 DialPlan aplikace

Tyto aplikace poskytují hovorovou funkcionalitu systému. Aplikace mohou přijímat hovor, přehrát hlášku, zavěsit hovor, nebo provést složitější funkcionalitu, jako jsou například funkce front, hlasovou schránky nebo možnost uskutečnit konferenční hovor.

3.4.1.3 DialPlan funkce

Funkce se používá pro načtení, nastavení nebo manipulaci s různými parametry hovoru. Funkce mohou být například použity pro nastavení volaného čísla pro odchozí hovor.

3.4.1.4 Zdroje

Jak název napovídá , zdroje poskytují zdroje pro Asterisk a jeho moduly. Mezi běžné příklady zdrojů lze uvést funkci "music on hold"nebo funkci pro parkování hovorů.

3.4.1.5 CODECs

CODEC je spojení zkratek COder/DECoder je modul pro kódování a dekódování zvuku nebo videa. Typicky je tento modul používaný ke kódování médií tak, aby zabíralo co nejmenší šířku pásma. Překlad audio/video.

3.4.1.6 Souborový ovladač

Používá se k ukládání médií na disk v určitém souborovém formátu nebo pro jejich opětovné čtení mediální soubory, nebo jejich převod pro přehrávání v hlasovém kanále.

3.4.1.7 Call Detail Record (CDR) ovladač

Zapisuje data o hovorech na disk nebo do specifické databáze. Ve výchozím stavu jsou data uloženy v souborovém formátu. Další možností uložení jsou databáze, Remote Authentication Dial In User Service (RADIUS), nebo v syslogu. Tyto data jsou následně využívány pro statistiky volání, či pro přeúčtování poplatku za hlasové služby.

3.4.1.8 Call Event Log (CEL) ovladač

Zapisuje event logy na disk či do specifikované databáze. Event logy jsou informace o tom, co se děje/stalo v Asterisk systému během příslušného hovoru.

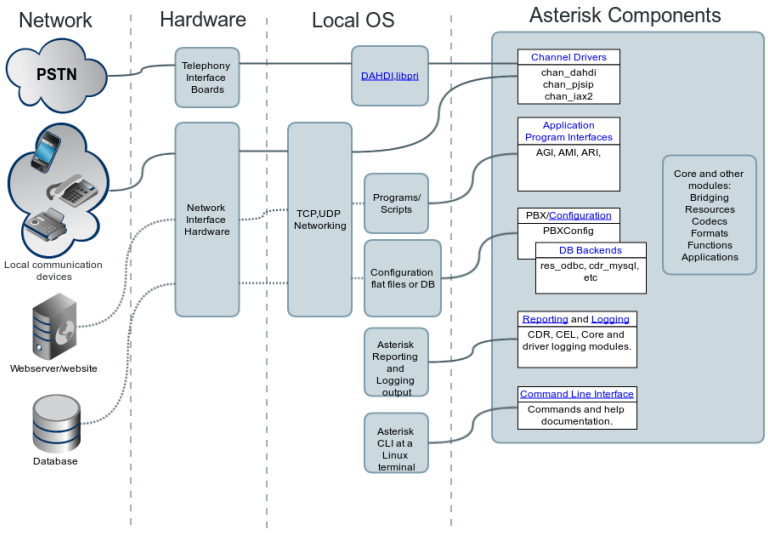
3.4.1.9 Bridge ovladač

Používají architekturu přemostění v aplikaci Asterisk a poskytují různé metody překlenování hovorových médií mezi účastníky hovoru. [8]

3.4.2 Konfigurační soubory

Konfigurační soubor je textový soubor, který obsahuje předdefinované konfigurační parametry jednotlivých modulů. Skrze tyto konfigurační soubory je administrátorem řízená celá ústředna. V případě, že Asterisk má uživatelské rozhraní, generují se tyto konfigurační soubory automaticky a mění se pouze jejich parametry.

Jedna z výhod konfiguračních souborů je možnost použití parametrů #include. Pomocí tohoto parametru je možné vkládat konfigurace z jiného souboru pro lepší strukturu konfiguračního souboru. V našem případě jsem použil pro konfiguraci vlastního VoiceXML [2]



Obrázek 3.2: Architektura Asterisku [9]

3.4.3 Popis základních konfiguračních souborů

3.4.3.1 Hlavní konfigurační soubor

- Asterisk.conf - Obsahuje cesty k adresářům a k veškerým souborům, které systém potřebuje. V základním nastavení jsou všechny konfigurační soubory nastavené pro adresář /etc/asterisk.

3.4.3.2 Asterisk kanály

- Sip.conf - Konfigurace SIP kanálu. Obsahuje všechny parametry potřebné ke konfiguraci jednotlivých SIP uživatelů, jejich proxy atd.
- Iax.conf - Konfigurace IAX kanálu. IAX je interní signalizace pro systém Asterisk, ve kterém se nastavují parametry jednotlivých uživatelů a přenašečů.

3.4.3.3 Konfigurace DialPlanu

- Extensions.conf - DialPlan obsahuje veškerou konfiguraci týkající se směrování hovorů či IVR funkce.

3.4.3.4 Konfigurace specifických funkcí DialPlanu

- Musiconhold.conf - Konfigurace Music on hold funkce
- Queues.conf - Konfigurace front v Asterisku
- Voicemail.conf - Konfigurace voicemailové služby Asterisku, tedy hlasový záznamník.

3.4.3.5 Ostatní konfigurační soubory

- Codecs.conf - Konfigurace jednotlivých podporovaných audio/video kodeků
- Features.conf - Konfigurace funkcionalit jako je parkování hovorů aj.
- Http.conf - Konfigurace vestavěného http serveru v Asterisku
- Modules.conf - Konfigurace nahrávání jednotlivých modulů do Asterisku
- Rtp.conf - Konfigurace portů pro RTP pakety. [2]

3.5 SIP

Popsání protokolu SIP není třeba, protože je již popsán v mnoha pracích a knížkách. [7] [19]

Kapitola 4

Voice XML

VoiceXML (Voice eXtensible Markup Language) jedná se o jazyk vyvinutý na bázi značkování XML, který v současné době vyvíjí konsorcium W3C jako součást projektu Voice Browser. VoiceXML je určen pro vytváření hlasových dialogů, které mohou obsahovat syntetizované řeči, digitalizované audio nebo rozpoznání vstupní uživatelské promluvy. Další z množnosti je také rozpoznání pomocí DTMF vstupu a vytvoření telefonního spojení. Hlavním cílem VoiceXML je přenesení dalších výhod vývoje webových aplikací a zpříjemnění webového obsahu pro její hlasovou interaktivitu. Výstupní textový obsah je možné převést do syntetizované podoby, která se vytváří pomocí technologie text-to-speech [10]

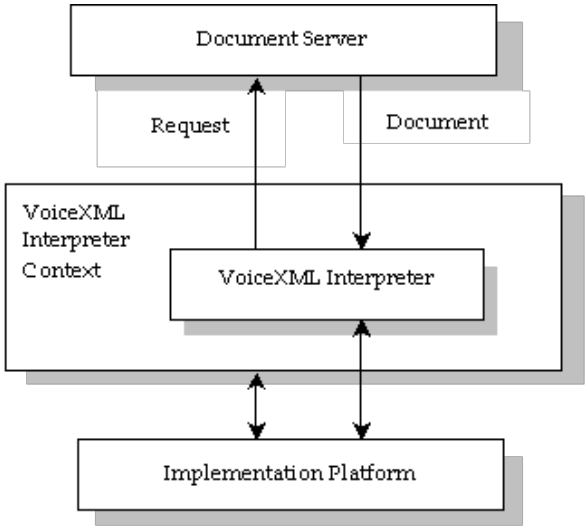
V roce 1998, W3C hostil konferenci o hlasovém prohlížeči, V této době již existovali různé varianty hlasových prohlížečů např. AT&T a Lucent spolu vyvíjeli různé varianty PML, zatímco Motorola vyvinula VoxML, a IBM si vyvinul vlastní SpeechML. Dalším z účastníků konference byla například firma HP, která pracovala na vývoji TalkML, ale na rozdíl od ostatních pracovala s dialogovým návrhem jazyka. VoiceXML Fórum byl založen společnostmi jako jsou AT&T, IBM, Lucent a Motorola. Cílem a úsilí VoiceXML Fóra je definování standardu dialogového jazyka, který budou moci vývojáři používat při tvorbě svých aplikací. [11]

VoiceXML Fórum v roce 2000 vydala VoiceXML 1.0, která byla přístupná pro veřejnost. Krátce poté, VoiceXML 1.0 předložili konsorciu W3C, jako základ pro vytvoření nové mezinárodní normy. Výsledkem je pracovní skupina W3C a veřejnosti je představeno VoiceXML 2.0. [16]

4.1 Architektura VoiceXML

Architektura VoiceXMl je rozdělena do jednotlivých komponent. První komponenta je dokumentový server (nebo-li Webový Server), který zpracovává požadavky od klientské aplikace, ty poskytuje VoiceXML Interpret přes Kontextový VoiceXML Interpret. V některých případech může kontext a interpret monitorovat uživatelský vstup a reagovat pouze na určité podměty na tomto vstupu. Kromě obsluhy vstupu může kontextový interpret stanovit spojení s uživatelem, a získávat informace ohledně kořenového aplikačního dokumentu, který se nachází v dokumentovém serveru pro VoiceXML interpret a dále řídí její dialogy.[16]

Poslední částí VoiceXML architektury je implementační platforma, která je řízená VoiceXML interpretem kontextu. Vytváří a generuje události podle uživatelských (např. mluvený nebo znakový vstup, odpověď, odpojení) nebo systémových akcí (např. vypršení časovače), jako jsou například interakce hlasové odezvy aplikace, detekce příchozích hovorů, získávání počátečního VoiceXML dokumentu a odpověď na volání.



Obrázek 4.1: VoiceXML Architektura [10]

4.2 Příklady aplikace VoiceXML

- Hlasový portál: Stejně jako Webový portál, tak hlasový portál může poskytovat individualizované služby pro přístup k informacím, jako jsou nabídky akcí, počasí, výpis restaurací, zprávy atd.
- Lokalizace základních služeb: možnost získávání cílených nebo specifických informací v závislosti odkud je voláno.
- Hlasové upozornění (např. na reklamu): VoiceXML může být používán k odesílání cíleného upozornění pro uživatele. Uživatelé, kteří se zaregistrovali k odběru speciálního upozornění, budou informováni o následujících událostech.
- Obchod: VoiceXML může být implementován s aplikací, která uživatelům umožňuje nákup po telefonu. Doporučuje se využívat VoiceXML pro obchod se zbožím, u kterého není zapotřebí velký detailní popis produktu (jako jsou například vstupenky, kancelářské potřeby apod.), jelikož hlas může předat zákazníkovi méně informací, než grafické vyobrazení. Pokud se nejedná o produkt s obsáhlým detailním popisem, funguje obchod velmi dobře.
- Auto Attendant: VoiceXML může nahradit operátora v případě přesměrování hovoru. Je možné použití Automatické rozpoznávaní řeči k rozpoznání jména uživatele ze seznamu kontaktů a přiřadí hovor k danému kontaktu.

4.3 VoiceXML pojmy

4.3.1 Dialogy

Aplikace VoiceXML je strukturována podle modelů konečného stavového automatu. Uživatel VoiceXML aplikace je vždy v jednom konverzačním stavu nebo dialogu pod daným časem. Každý dialog je následován dalším dialogem. Pokud dojde k situaci, kdy další dialogové okno není dostupné, VoiceXML provede ukončení aplikace. Existují dva typy dialogových oken: forms a menu. Formuláře obsahují uživatelské vstupy v podobě hodnot, stejně tak jak jsou zadávány ve formuláři HTML. Menu je prezentováno uživateli jako seznam možností, které si může vybrat.

4.3.2 Relace

Relace začíná, jakmile uživatel začne komunikovat s VoiceXML dokumentem.

4.3.3 Aplikace

Aplikace je sbírka VoiceXML dokumentu. Všechny dokumenty v aplikacích, mohou sdílet stejný kořenový adresář. Kořenový dokument lze používat a předávat proměnné z jednoho VoiceXML dokumentu do druhého. Všechny proměnné definované v kořenovém dokumentu jsou k dispozici všem dokumentům.

4.3.4 Gramatika

Gramatika určuje seznam povolených slovních zásob, ze kterých si uživatel může vybírat. Každý dialog obsahuje jjeden nebo více mluvených projevů nebo s ním spojenou gramatiku.

4.3.5 Události

Události jsou vyvolány VoiceXML platformou z několika důvodů. Například nereaguje-li uživatel na vstup,a nebo požádá-li uživatel o pomoc atd. Události jsou dále obsaženy například uvnitř Voice XML interpretu v případě, že existují nějaké sémantické chyby v dokumentu VoiceXML.

4.3.6 Odkazy

Odkaz určuje přechod, který je společný pro všechny dialogy v obsaženém dialogu. Když uživatelský vstup odpovídá na odkazovanou gramatiku, je kontrolován přenos tohoto cíleného odkazu. [20]

4.4 Základní struktura VoiceXML aplikace

VoiceXML je rozšíření jazyka XML, je zapotřebí dodržovat základní pravidla pro XML. VoiceXML aplikace, a která se skládá z jednoho nebo více textových dokumentů. Tyto souborové dokumenty jsou označovány příponou ".vxml"a obsahem jsou různé VoiceXML informace a pokyny pro aplikaci. Na základě jazyka XML je formát VoiceXML ve tvaru tag/atribut, syntaxe jsou ohraničeny instrukcemi. [13]

```
<xml> tag verze
```

Doporučuje se, aby první řádek libovolného VoiceXML dokumentu obsahoval položku ohledně XML verze tagu.

```
<?xml version="1.0"?>
```

VXML tag je obsažen v uzavřeném dokumentu instrukcí. Tag by měl obsahovat informaci ohledně verze, která se bude používat (v našem případě to je verze "2.0").

```
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
```

Základní struktura VoiceXML může vypadat následovně:

```
<?xml version="1.0"?>
<vxml version = "2.0" xmlns="http://www.w3.org/2001/vxml">
...
</vxml>
```

4.4.1 Příklad: Hello world!

V našem případě vytvoříme jednoduchý dokument, který bude obsahovat Hello world! pomocí VoiceXML. Dokument s kódem je zapotřebí nakopírovat do serveru VoiceXML. A je potřeba přiřadit tento dokument buď do Asterisku nebo do SIPového protokolu. Dále přidělit danému VoiceXML telefonní číslo, aby bylo možné otestovat tuto aplikaci. V našem případě, aby bylo něco přehráno bude potřebné buď předem připravit zvukový soubor a nebo použít technologii TextToSpeech (převod textu do mluveného slova). [13]

Instrukce "form"a "blok"budou popsány níže.

```
<?xml version="1.0"?>
<vxml version = "2.0" xmlns="http://www.w3.org/2001/vxml">
       <form>
              <blok>Hello world!</blok>
       </form>
</vxml>
```

4.5 Formulář

Formulář, který je ve VoiceXML dokumentu představuje informace a shromažďuje vstupy od uživatele. Formulář je reprezentován ve tvaru tagu <form>, a má ID atribut, který je s ním spojený. Atribut ID obsahuje název formuláře. Příklad použití tohoto prvku formuláře:

```
<form id="hello" >
  <blok>
       Hello world!
  </block>
</form>
```

V tomto příkladě je název formuláře "hello", ale uživateli ovšem bude prezentováno "Hello world!".

4.5.1 Položky formuláře

Existují dva druhy polože formuláře: pole položek a ovládací položka. V poli položek je uživatel vyzván k tomu co má říct. Klíčové informace, které jsou od uživatele získány jsou dále uloženy do pole proměnných. Pole může obsahovat také gramatiku, která definuje povolené vstupy. Obsloužené události jsou dále zpracovány do výsledné události. Parametr <filled> definuje požadavek na přijetí do požadované proměnné, která již byla naplněna. [13]

Seznam typu polí:

- <field>: hodnota pole je položka získávána od uživatele prostřednictvím řeči nebo DTMF
- <record>: hodnota pole je položka obsahující zvukový klip zaznamenaný uživatelem, jako je hlasová zpráva, která může obsahovat vybrané <record> prvky
- <transfer>: používá se pro přepojení uživatele na jiné telefonní číslo
- <object>: vyvolává platformu pro konkrétní objekt, s jedním nebo s více vlastnostmi
- <subdialog>: slouží jako volací funkce, vytváří volání na jiný dialog v aktuální stránce nebo v jiném VoiceXML dokumentu.

Ovládací položka má za úkol pomoci kontrolovat shromažďované pole ve formuláři. Jsou zde dva typy:

- <block>: posloupnost příkazů používaných pro zobrazování a výpočet
- <initial>: užitečné ve smíšených inicializovaných dialozích, které se dotazují uživatele k poskytnutí informace

4.5.2 Formulář s proměnnými položkami a podmínky

Formulář s proměnnými položkami je spojený s každým formulářem. Formulářová položka "ve výchozím stavu"je nastavena na "nedefinovanou"hodnotu. Obsahuje výsledek (získaný od uživatele) jakmile byla položka ve formuláři již interpretována, může být název její proměnné definován pomocí jména atributu.

Podmínky testují, zda má proměnná položka načtenou hodnotu. V případě, že hodnota existuje pak přeskočí tyto položky ve formuláři. [13]

4.6 Menu

Menu poskytuje uživateli seznam možností a na základě výběru možnosti uživatele dojde k přechodu na jiný dokument. Ukázka kódu s použitím menu jako seznam možností:

```
<menu>
 <prompt>Say what sports news you are interested in:
 <enumerate/></prompt>
  <choice next="http://www.news.com/hockey.vxml">
   Hockey
  </choice>
  <choice next="http://www.news.com/baseball.vxml">
   Baseball
  </choice>
  <choice next="http://www.news.com/football.vxml">
   Football
  </choice>
 <noinput>Please say what sports news you are interested in
 <enumerate/>
 </noinput>
</menu>
```

Podrobnější popis jednotlivých položek a funkcí je možné dohledat například na webových stránkách Voicexml v sekci tutoriál [12] nebo v normě VoiceXML 2.1. [13]

Kapitola 5

Návrh způsobu zpracování VoiceXML v systému Asterisk

V souvislosti s teoretickými znalostmi těchto technologií popsané v předchozích textu se tato kapitola bude zabývat analýzou, návrhem aplikací a funkcí, které se implementují do Asterisku podle doporučení standardu RFC 5552 (SIP Interface to VoiceXML Media Services). Hlavním rozdílem od ostatních implementací je to, že mé řešení je orientováno na využití lokálních VoiceXML skriptů, tedy bez volání dalšího VoiceXML serveru, což se ve standardu předpokládá.

Z toho vyplývají tyto hlavní cíle:

- Vytvořit a upravit systém Asterisk tak, aby bylo možné přijímat vstupní parametry od uživatele.
- Vytvořit funkcionalitu pro Asterisk, aby mohla zpracovávat vstupní parametry, které jsou obsaženy ve VoiceXML.
- Realizace jednotlivých funkcí ve VoiceXML skriptu.

Pro dosažení stanovených cílů bude nejdříve potřeba získat a nainstalovat systém Asterisk.

Existují dvě možnosti získání systému Asterisk. Tou první je stažení zkompilované nebo binární verze přímo ze stránek projektu. V současné době je dostupná ke stažení verze od 13 až po 16, která je poslední vydanou verzí systému. Tento typ systému, který je stažitelný na stránkách projektu, obsahuje obraz operačního systému a nazývá se AsteriskNOW. Pro instalaci není potřeba mít předem nainstalovaný operační systém a obsahuje grafické rozhraní, ve kterém se dají konfigurovat jednotlivé moduly systému.

Druhou z možností je získání zdrojového kódu systému Asterisk, který je také k dispozici na stránkách projektu. Výhodou této možnosti je plná volnost systému. V případě úpravy je k dispozici zdrojový kód, který je možní dle dle vlastní potřeby upravovat nebo přidávat další funkcionality. Nevýhodou je složitější instalace a potřeba mít předinstalovaný operační systém.

Pro řešení práce byla použita možnost se zdrojovým kódem, protože jsme potřebovali upravit jeden z klíčových modulů v systému Asterisk.

5.0.1 Doporučení RFC 5552

V dokumentu RFC 5552 je pod bodem 1.1.3 uvedena definice MRF (Media Resource Function) s použitím a zpracováním mediálních služeb, jako je schopnost nabízet VoiceXML přímo bez účasti Aplikačního serveru.

5.1 Načtení vstupních parametrů

Pro načítání vstupních parametrů je nejdříve potřeba tyto parametry nějak rozlišovat. Ve standardu je definované použití protokolu SIP, který je přímo standardem vyžadován.

5.1.1 Identifikace vstupních parametrů

Definovaný seznam vstupních parametrů, které jsou v hlavičce SIP protokolu dle RFC 5552 [15]

- **voicexml:** URI obsahuje HTTP URI VoiceXML dokument, který odkazuje na lokální nebo statický VoiceXML dokument. Pokud vstupní parametr nebude obsahovat "voicexml", tak VoiceXML Media Server může nastavit výchozí VoiceXML dokument ve výchozím stavu, nebo požadavek zamítne.
- **maxage:** Používá se pro nastavení maximální doby do kdy je systém schopen přijímat spojení s VoiceXML dokumentem, který je v hlavičce Cashe-Control. V případě, že "maxage"bude vynechán, VoiceXML Media Server nastaví parametr na výchozí hodnotu.
- **maxstale:** Používá se pro nastavení maximální doby od kdy je systém schopen přijímat spojení VoiceXML dokumentu, který je v hlavičce Cashe-Control. V případě, že "maxstale"bude vynechán, VoiceXML Media Server nastaví výchozí hodnotu.
- **method:** Používá se pro nastavení HTTP metody, která se aplikuje při načtení VoiceXML dokumentu. Povolené hodnoty jsou "get"nebo "post". Výchozí hodnota je nastavena na "get"
- **postbody:** Používá se pro nastavení "application/x-www-form-urlencoded"kódování (HTML4) HTTP těla, požadovaná hodnota musí být "post"(jinak je ignorován)
- **ccxml:** Používá se pro specifikace "JSON value", která je mapována v "session.connection.ccxml".
- **aai:** Používá se pro specifikace "JSON value", která je mapována v "session.connection.aai".

Příklad použití vstupních parametrů pro identifikaci využívaných služeb:

```
sip:dialog@mediaserver.example.com; \
voicexml=http://appserver.example.com/promptcollect.vxml; \
maxage=3600;maxstale=0
```

Parametr "voicexml"je odkaz, který ukazuje na VoiceXML dokument. Parametry "max-age/max-stale"jsou nastaveny na hodnotu 3600s/0s, to znamená že požadavek nesmí překročit uvedený čas 3600s a může být přijímán od 0s. [15]

```
dialog-param = "voicexml=" vxml-url ; vxml-url follows the URI
                                    ; syntax defined in [RFC3986]
maxage-param = "maxage=" 1*DIGIT
maxstale-param = "maxstale=" 1*DIGIT
method-param = "method=" ("get" / "post")
postbody-param = "postbody=" token
ccxml-param = "ccxml=" json-value
aai-param = "aai=" json-value
json-value = false /
             null /
             true /
             object /
             array /
             number /
             string ; defined in [RFC4627]
```

Výše je uvedený přehledný seznam všech možných vstupních parametrů pro identifikaci VoiceXML služby.

5.1.2 Inicializace VoiceXML relace

Pro inicializaci VoiceXML služeb je potřeba upravit hlavičku protokolu SIP, tak aby splňoval podmínku pro identifikaci vstupních parametrů. To znamená, že je zapotřebí také upravit v Asterisk SIPový kanál, aby byl schopen tyto parametry v systému číst a zpracovávat. V Asterisku existují dvě varianty kanálů SIP:

- Kanál SIP (chan_sip): Jedná se o kanálový ovládač, který je zaměřen na funkcionalitu SIP protokolu. Výhodou tohouo kanálového ovládače je jeho implementace, která je už od počátku systému Asterisk a je základním stavebním kamenem tohoto systému. Od verze 12 Asterisk umožňuje uživatelů používat novější variantu kanálového ovládače chan_pjsip, který bude vysvětlen níže. I přes příchod nové varianty, je obecně stále populárnější klasický starší chan_sip, protože je velmi známý, stabilní, zavedenější a má velkou podporu jednotlivých funkcí, které jsou již implementované a potřebné pro běžnou SIP komunikaci. Jeho největším negativem je obtížná implementace nových funkcí.

- Kanál PJSIP (chan_pjsip): Tento nový kanálový ovládač se začal implementovat do systému Asterisk od verze 12. Výhodou je velká modulárnost. Změna jednoho modulu nemá efekt na ostatní. Ovládač se používá buď samostatně nebo paralelně s chan_sip. V rámci vývoje Asterisku a budoucnosti je lepší ovladač chan_sip, je jednodušší na implementaci nových funkcí a méně riskantní v případě chyb. Bohužel je tento ovladač relativně nový, proto je jeho počet uživatelů nízký.

V našem případě navrhuji implementaci parametrů SIP URI do kanálového ovládače chan_sip, tato úprava bude schopna používat i starší systémy a její výhody s ní spojeny.

5.2 Parsing XML

SIP URI je pouze text, který obsahuje odkaz na VoiceXML dokument a jeho parametry. Asterisk neví co s tímto parametrem dělat proto je zapotřebí vytvořit skript, který je schopen najít VoiceXML dokument a zpracovat ho.

Veškerá komunikace v Asterisku je řešena přes DialPlan. Kdybychom začali vytvářet skript ohledně pasrování XML, konfigurační soubor by byl moc velký a nepřehledný. Dalším problémem je DialPlan, který umí zpracovávat XML soubor, ale je omezenější než kdybychom zpracovávali XML pomocí externího skriptu. Skript má výhodu, že umí vykonávat tu samou věc elegantněji a zbytečně tedy nemusíme zatěžovat systém Asterisk, který se bude zaměřovat pouze na vykonávání jednotlivých funkcí, které jsou obsaženy ve VoiceXML dokumentu.

5.3 Realizace jednotlivých funkcí XML skriptu

Po zpracovaní VoiceXML dokumentu, získáme jednotlivé funkce XML skriptu. Tyto funkce jsou dále potřebné vykonávat, ale samostatný Asterisk nemá tyto funkce implementované.

Ve VoiceXML existuje parametr "audio", který umožňuje funkci přehrát hudbu či zvukovou stopu. Asterisk umí přehrávat hudbu, ale nejdříve je potřeba upravit parametr do správné podoby. Implementace této funkce může například poskytnout hudbu dle výběru.

Jedna z neimplementovaných funkcí je například výstupní text, který je potřeba předat uživateli, aby mohl reagovat. Při hovoru není možné posílat text, jelikož hovor je zvuková interakce. Z toho vyplývá, že je zapotřebí vytvořit rozhraní, které dokáže text přehrát. Jednou z těchto technologií je TextToSpeech, která převádí text do mluveného slova. Asterisk v základu TextToSpeech nemá, a bez této implementace systém nemá smysl.

Další možností je implementace gramatiky, která bude schopna převádět mluvená slov na text. V tomto případě je potřebné omezit vstupní interakci při detekování mluveného slova. Tyto možnosti by byli velmi užitečné například pro interaktivní hlasovou službu, jako je nákup zboží či hlasový portál.

VoiceXML obsahuje další množství funkcí, které se dají postupně implementovat. Jejich implementaci je možné postupně přidat, není to ovšem hlavní cíl této práce..

Kapitola 6

Praktická realizace a výsledky navrženého řešení

Tato kapitola se zabývá popisem realizace navrženého řešení, které bylo popsané v průběhu této práce a potřebnou konfigurací prostředí s ní spojené.

6.1 Popis realizovaného prostředí

Celé realizované prostředí je vytvořené ve virtuálním prostředí VirtualBox, které je spustitelné na jakémkoliv operačním systému Windows, Linux/Unix či MacOS. Pro naší praktickou realizaci jsem použil virtualizovaný operační systém CentOS 7, který je instalován v základní verzi.

6.1.1 Příprava OS pro Asterisk

Před instalaci systému Asterisk je potřeba předpřipravit operační systém. Instalují se všechny potřebné knihovny a programy pro zajištění správné instalace a správný chod systému.

6.1.1.1 Použitá konfigurace PC

Nastavení virtuálního počítače

- Operační systém - Linux CentOS 7.3 s verzi jádra 3.10.0-514
- Procesor - 1 Virtuální procesor s možností využití lokálního procesoru - bez limitu
- Paměť - Dynamicky alokovaná paměť na 1 GB
- Pevný disk - Dynamicky se zvětšující, maximální velikost 8 GB

Minimální požadavky operačního systému Linux CentOS 7

- 1 jádrový procesor

- 256 MB operační paměti
- 4 GB lokální úložiště

6.1.1.2 Aktualizace operačního systému

Jednotlivé příkazy jsou určené pro operační systém CentOS ve verzi 7.3. Je potřeba být přihlášený pod správcem systému (root), jinak příkazy nebudou systémem vykonány..

```
# yum update
```

6.1.1.3 Instalace potřebných balíčků

Níže jsou uvedeny příkazy pro všechny potřebné nástroje, balíčky a programy. Které mají za úkol připravit operační pro instalaci Asterisku.

```
# yum install gcc gcc-c++ php-xml php php-mysql php-pear php-mbstring
mariadb-devel mariadb-server mariadb sqlite-devel lynx bison gmime-devel
psmisc tftp-server httpd make ncurses-devel libtermcap-devel sendmail
sendmail-cf caching-nameserver sox newt-devel libxml2-devel libtiff-devel
audiofile-devel gtk2-devel uuid-devel libtool libuuid-devel subversion
kernel-devel kernel-devel-$(uname -r) git subversion kernel-devel
php-process crontabs cronie cronie-anacron wget vim
```

Na pevném disku je potřebné mít k dispozici 125MB, což je celková velikost všech balíčků.

```
Transaction Summary
================================================================================
Install  31 Packages (+133 Dependent packages)
Upgrade   3 Packages (+ 25 Dependent packages)

Total download size: 125 M
Is this ok [y/d/N]: y
```

Obrázek 6.1: Součet všech instalovaných balíčku

6.1.1.4 Konfigurace MariaDB

Po dokončení instalace balíčků, je zapotřebí povolit databázovou a spustit ji. Na obrázku 6.2 jsou znázorněné jednotlivé příkazy.

V případě, že systém běží je nyní zapotřebí nastavit správcovské heslo a odstranit anonymní uživatele z testovací databáze. Jako další krok zakážeme vzdálené přihlášení.

```
# mysql_secure_installation
```

```
[root@centos-7 ~]# systemctl enable mariadb
ln -s '/usr/lib/systemd/system/mariadb.service' '/etc/systemd/system/multi-user.target.wants/mariadb.service'
[root@centos-7 ~]#
[root@centos-7 ~]# systemctl start mariadb
[root@centos-7 ~]# systemctl status mariadb
mariadb.service - MariaDB database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled)
   Active: active (running) since Tue 2015-09-08 12:41:51 BST; 11s ago
  Process: 20505 ExecStartPost=/usr/libexec/mariadb-wait-ready $MAINPID (code=exited, status=0/SUCCESS)
  Process: 20426 ExecStartPre=/usr/libexec/mariadb-prepare-db-dir %n (code=exited, status=0/SUCCESS)
 Main PID: 20504 (mysqld_safe)
   CGroup: /system.slice/mariadb.service
           ├─20504 /bin/sh /usr/bin/mysqld_safe --basedir=/usr
           └─20662 /usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib64/mysql/
```

Obrázek 6.2: Znázorněné příkazy a kontrola jestli databázový systém běží

6.1.1.5 Instalace libjansson

Jansson je knihovna v jazyce C, která slouží ke kódování, dekódování a manipulovaní s JSON daty. Nějdříve je potřeba stáhnout:

```
# wget http://www.digip.org/jansson/releases/jansson-2.7.tar.gz
```

```
[root@centos-7 ~]#
[root@centos-7 ~]# wget http://www.digip.org/jansson/releases/jansson-2.7.tar.gz
--2015-09-08 13:24:35--  http://www.digip.org/jansson/releases/jansson-2.7.tar.gz
Resolving www.digip.org (www.digip.org)... 217.30.184.170
Connecting to www.digip.org (www.digip.org)|217.30.184.170|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 445179 (435K) [application/x-gzip]
Saving to: 'jansson-2.7.tar.gz'

100%[======================================================================================>] 445,179

2015-09-08 13:24:36 (1.06 MB/s) - 'jansson-2.7.tar.gz' saved [445179/445179]

[root@centos-7 ~]#
[root@centos-7 ~]# ls
anaconda-ks.cfg  jansson-2.7.tar.gz
```

Obrázek 6.3: Stahování knihovny Jansson

Rozbalování archivovaného balíčku s Jansson knihovnou

```
# tar -zxvf jansson-2.7.tar.gz
```

```
[root@centos-7 ~]# ls
anaconda-ks.cfg  jansson-2.7  jansson-2.7.tar.gz
[root@centos-7 ~]# cd jansson-2.7
[root@centos-7 jansson-2.7]# ls
aclocal.m4  cmake           config.guess  configure.ac  install-sh                 LICENSE      Makefile.in  src
android     CMakeLists.txt  config.sub    depcomp       jansson.pc.in              ltmain.sh    missing      test
CHANGES     compile         configure     doc           jansson_private_config.h.in  Makefile.am  README.rst   test-driver
[root@centos-7 jansson-2.7]#
[root@centos-7 jansson-2.7]# ./configure --prefix=/usr
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
```

Obrázek 6.4: Změna adresáře a konfigurace balíčku

Kompilace a konfigurace balíčků s použitím příkazu "make"a "make install", které musí být ve stejném adresáři jako kompilovaný soubor.

```
# make clean
# make && make install
# ldconfig
```

6.1.2 Instalace Asterisk ze zdrojového kódu

Instalace Asterisku je jeden z nejdůležitější kroku naší práce. Nejdříve je potřeba stáhnout oficiální distribuci z oficiálních stránek projektu. Možností stažení je více, ale v případě Linuxového operačního systému, který nemá grafické rozhraní se dá stáhnout pomocí příkazu "wget". Má vybraná verze Asterisku je 13, která má aktuální dlouhodobou podporu a je certifikována v případě chyb.

Po stažení je potřeba soubor rozbalit a změnit cestu k adresáři, v něm spustit konfigurační příkaz k instalaci Asterisku.

```
# tar -zxvf asterisk-13.XX.XX.tar.gz
# cd asterisk-13.XX.XX
# ./configure --libdir=/usr/lib64
```

Po skončení instalace Asterisku se musí objevit znak Asterisku (logo) podle obrázku 6.5, který signalizuje že instalace proběhla v pořádku.

Obrázek 6.5: Úspěšná instalace systému Asterisk

6.1.2.1 Spuštění Asterisk

Posledním krokem je spuštění systému Asterisk, který se vykonává jednoduchými příkazy. V případě, že se Asterisk nespustí, je možné pomocí příkazu "safe_asterisk"pustit systém v bezpečnostním v módu. Jedná se o bezpečný způsob spuštění, v případě obav z narušení chodu systému. Dále stačí spustit příkaz "asterisk -r"nebo "rasterisk". Po vykonání příkazu se zobrazí příkazová řádka Asterisku (CLI). Zde se pak používají příkazy, které jsou definovány samotným Asteriskem.

```
# safe_asterisk
# asterisk -r
```

```
[root@centos-7 ~]# asterisk -r
Asterisk 13.5.0, Copyright (C) 1999 - 2014, Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.
This is free software, with components licensed under the GNU General Public
License version 2 and other licenses; you are welcome to redistribute it under
certain conditions. Type 'core show license' for details.
=========================================================================
Connected to Asterisk 13.5.0 currently running on centos-7 (pid = 26739)
centos-7*CLI>
centos-7*CLI>
```

Obrázek 6.6: Spuštění příkazové řádky Asterisku (CLI)

Další konfigurace a nastavení systému jsou uvedené v příloze A

6.2 Úprava funkce chan_sip v Asterisk

V této kapitole se budeme zabývat realizace prvního z vytyčených cílů. Podle doporučení RFC 5552 je potřeba upravit protokol SIP, aby splňoval podmínky tohoto standartu. Hlavní úprava protokolu spočívala v tom, že při odesílání INVITE musí být v hlavičce obsahovat parametr "voicexml". Níže je znázorněná implementace v hlavičce INVITE.

```
INVITE sip:[service]@[remote_ip]:[remote_port];voicexml=menu2.vxml SIP/2.0
```

V tom případě je potřeba tento parametr získat. Ale samostatný a Asterisk neumí s takhle upraveným INVITE pracovat. Když odesíláme INVITE, nejsou zpracované vstupní parametry za středníkem (";"). Tyto parametry potřebuje k volání daného VoiceXML dokumentu. Což je potřeba tyto parametry uložit do proměnné, aby se s nimi dalo později pracovat.

Veškerá SIPová komunikace probíhá přes funkci chan_sip ze zdrojové instalace je k dispozici i zdrojový kód této funkce. Nachází se adresáři asterisku /channel/chan_sip.c. Po dohodě s vedoucím práce jsme navrhli přidání nové funkce, která bude získaný parametr po středníku (";") ukládat do proměnné. Funkce se bude jmenovat SIP_URI_OPTION() a bude volána v dialplánu.

```
exten => service,1,Set(vxml=${SIP_URI_OPTIONS()})
```

Tedy při zahájení získání INVITE se vstupním parametrem VoiceXML, bude SIP_URI_OPTIONS() obsahovat například "voicexml=menu2.vxml". Obsah proměnné se předá proměnné "vxml", se kterou budeme dále pracovat v diálplánu.

6.2.1 Postup patchování funkce chan_sip

Nejdříve je mít již nainstalovaný program patch, který se použije pro patchování chan_sip.

```
# yum install patch
```

Zde je uveden postup patchování chan_sip pomocí vlastním zdrojovým kódem.

```
# cd /usr/local/src/asterisk-13.XX.X/channel/
# patch < sip_uri_opts.diff
# cd ..
# make
# make install
```

```
[root@localhost channels]# patch < /root/sip_uri_opts.diff -R
(Stripping trailing CRs from patch; use --binary to disable.)
patching file chan_sip.c
Hunk #1 succeeded at 378 (offset 18 lines).
Hunk #2 succeeded at 22999 (offset 448 lines).
Hunk #3 succeeded at 35355 with fuzz 2 (offset -138 lines).
Hunk #4 succeeded at 35455 with fuzz 2 (offset -135 lines).
```

Obrázek 6.7: Patchování souboru chan_sip.c o rozšíření programu sip_uri_opts.diff

Tímto postupem se patchuje chan_sip rozšířenou o naší funkci.

6.3 Realizace parsování XML

Po získání vstupní parametrů a cestu k VoiceXML dokumentu. VoiceXML v jazyce XML, který je potřeba roztřídit na jednotlivé příkazy, aby se dali v systému Asterisk vykonávat.

Pro parsování XML jsem použil linuxový shell, který Asterisk podporuje. Již dříve nevýhody parsování v Asterisku jsem vytvořil vlastní skript, který veškeré příkazy odstraní ostré závorky a roztřídí je dle naších potřeb. Skript přebírá cestu k VoiceXML dokumentu a ten po rozparsování předá jednotlivé parametry do proměnné, které se dále použije v dialplánu.

Příklad volání externího skriptu v dialplánu. Kde skript je uložen v adresáři "root"a skript se jmenuje "parse2.sh".

```
same => n,Set(parse=${SHELL(sh /root/parse2.sh ${vxml})})
```

Příklad jednoduchého VoiceXML dokumentu. Jaký bude výstup po vykonání skriptu "parse2.sh".

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.w3.org/2001/vxml
 http://www.w3.org/TR/voicexml20/vxml.xsd">
<menu>
 <property name="inputmodes" value="dtmf"/>
 <prompt>
  For sports press 1, For weather press 2, For Stargazer astrophysics press 3.
 </prompt>
 <choice dtmf="1" next="http://www.sports.example.com/vxml/start.vxml"/>
 <choice dtmf="2" next="http://www.weather.example.com/intro.vxml"/>
 <choice dtmf="3" next="http://www.stargazer.example.com/astronews.vxml"/>
</menu>
</vxml>
```

Zde je výstup po vykonání "parse2.sh", kde veškeré syntaxe XML jsou pouze jejich parametry a příkazy.

```
?xml * version="1.0" encoding="UTF-8"?
vxml * version="2.0" xmlns="http://www.w3.org/2001/vxml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.w3.org/2001/vxml
 http://www.w3.org/TR/voicexml20/vxml.xsd"
 menu
 property * name="inputmodes" value="dtmf"/
 prompt * For sports press 1, For weather press 2, For Stargazer astrophysics press 3.
 /prompt
 choice * dtmf="1" next="http://www.sports.example.com/vxml/start.vxml"/
 choice * dtmf="2" next="http://www.weather.example.com/intro.vxml"/
 choice * dtmf="3" next="http://www.stargazer.example.com/astronews.vxml"/
/menu
/vxml
```

6.4 Implementace TTS a dalších komponent Asterisku

Poslední z hlavní cíl je implementace technologie TTS (TextToSpeech) do Asterisku. Tedy realizace hlasová interakce s uživatelem. Jedno z možnosti je použití aplikace Festival Text-To-Speech, která byla vyvinuta Edinburskou Univerzitou. Festival je jeden z nejstarších běžících aplikací pro Text-To-Speech na operačním systému Linux. Další použitou funkci "Read()", která je již v Asterisku, a používáme jí pro čtení stisknuté volby.

6.4.1 Implementace TTS Festival

Nejdříve je potřeba si stáhnout a nainstalovat aplikaci Festival. Všechny uvedené postupy jsou pro uživatele root (správce) a operační systém CentOS 7. A v další kroku je její

implementace do systému Asterisk.

6.4.1.1 Instalace Festival

Příkaz pro instalaci Festival.

```
# yum install festival
```

Další krokem je konfigurace Festivalu, aby Asterisk mohl používat tuto aplikaci. Konfigurační soubor je "festival.scm"a je ulože v adresáři "/usr/share/festival/". Stačí editovat v jakémkoli textovém editoru, zde používám editační program "vim".

```
# vim /usr/share/festival/lib/festival.scm
```

Na konci souboru je poslední řádek "(provide 'festival)"vložíme nad ním tento text.

```
(define (tts_textasterisk string mode)
"(tts_textasterisk STRING MODE)
Apply tts to STRING. This function is specifically designed for
use in server mode so a single function call may synthesize the string.
This function name may be added to the server safe functions."
(let ((wholeutt (utt.synth (eval (list 'Utterance 'Text string)))))
(utt.wave.resample wholeutt 8000)
(utt.wave.rescale wholeutt 5)
(utt.send.wave.client wholeutt)))
```

Editovaný soubor uložíme a restartujeme Festival server.

```
# festival_server 2>&1 > /dev/null &
```

6.4.1.2 Implementace Festival do Asterisk

Ve zdrojové adresáři Asterisku je již uložená aplikace Festival "app_festival". Její aktivace je možné pomocí příkazu "make menuselect", kde vyskočí všechny povolené či zakázané implementace. Viz obrázek 6.8.

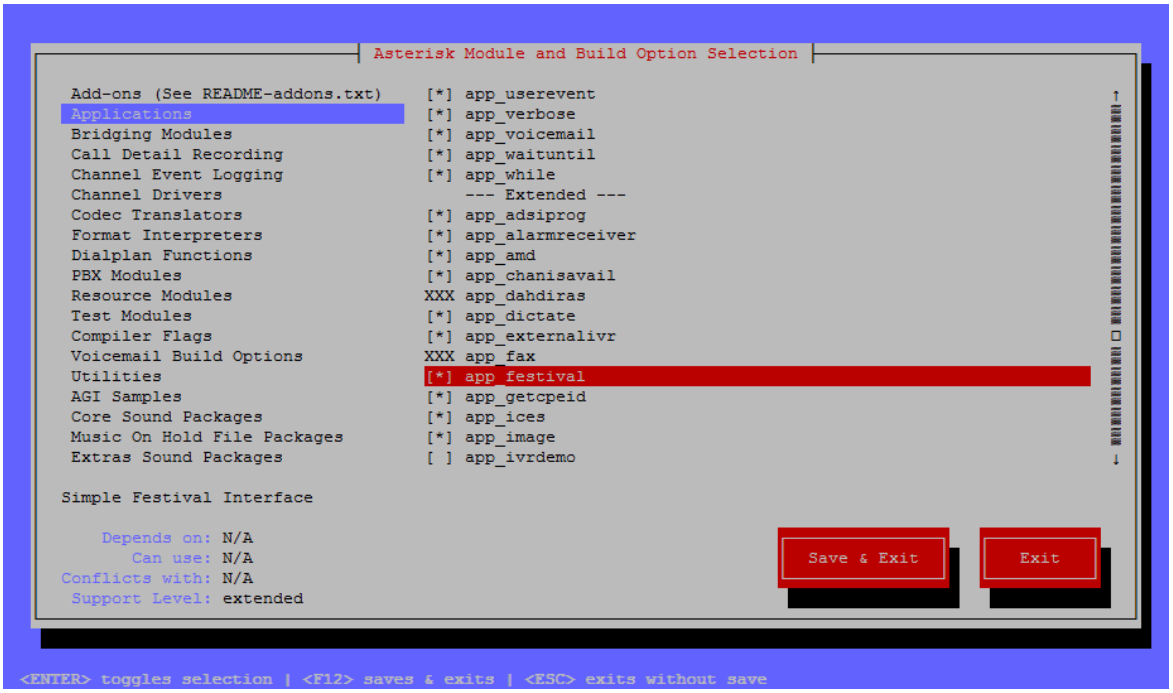
```
# cd /usr/local/src/asterisk-13.XX.X
# make menuselect
# make install
```

Po dokončení instalace modulu je potřeba překopírovat konfigurační soubor "festival.conf"do konfiguračního adresáře Astrisku.

```
# cd /usr/local/src/asterisk-13.XX.X
# cp configs/samples/festival.conf.sample /etc/asterisk/festival.conf
```

Po překopírování konfiguračního souboru je potřeba editovat zakomentované konfigurační parametry. Viz obrázek 6.9.

Příklad použití aplikace Festival v Dialplanu



Obrázek 6.8: Grafické rozhraní pro výběr používaných modulů

```
exten => 110,1,Verbose(2,Performing Festival test)
same => n,Answer()
same => n,System(echo "We are now doing Festival Test after successful
installation" | /usr/bin/text2wave -scale 1.5 -F 8000 -o /tmp/test.wav)
same => n,Playback(/tmp/test)
same => n,System(rm -f /tmp/test.wav)
same => n,Hangup()
```

6.4.2 Použití funkce Read

Funkce Read() slouží pro detekování příchozích DTMF volby. Příklad použití Read, který umí v průběhu přehrání zvuku zareagovat na příchozí DTMF volbu.

```
same => n,Read(test,"/tmp/${CDR(uniqueid)}",1,,,15)
```

6.5 Celkový popis řešení

Aktuálně je systém Asterisk upravený a je připraven přijímat vstupní parametry, které jsou definované podle standardu RFC 5552. Systém je schopen vykonávat jednotlivé příkazy uvedené ve VoiceXML dokumentu. Dále Asterisk umí používat technologii Text-To-Speech,

```
; Festival Configuration
;
[general]
;
; Host which runs the festival server (default : localhost);
;
host=localhost
;
; Port on host where the festival server runs (default : 1314)
;
port=1314
;
; Use cache (yes, no - defaults to no)
;
usecache=yes
;
; If usecache=yes, a directory to store waveform cache files.
; The cache is never cleared (yet), so you must take care of cleaning it
; yourself (just delete any or all files from the cache).
; THIS DIRECTORY *MUST* EXIST and must be writable from the asterisk process.
; Defaults to /tmp/
;
cachedir=/var/lib/asterisk/festivalcache/
;
; Festival command to send to the server.
; Defaults to: (tts_textasterisk "%s" 'file)(quit)\n
; %s is replaced by the desired text to say. The command MUST end with a
; (quit) directive, or the cache handling mechanism will hang. Do not
; forget the \n at the end.
;
festivalcommand=(tts_textasterisk "%s" 'file)(quit)\n
;
```

Obrázek 6.9: Konfigurační soubor festival.conf

tedy umí převádět text do zvukové podoby. Umí reagovat na DTMF volbu, a vykonat daný úkol dle VoiceXML dokumentu.

Vykonání jednotlivých příkazů v Asterisku je řešen přes DialPlan, který obsahuje funkci třídění příkazu na příkazy s parametrem či bez ní. A odstraní se zde dva počáteční příkazy, které jsou potřeba pro k vykonání. Ostatní jsou volány přes funkci Goto, která provede skok do externího konfiguračního souboru "voicexml.conf", kde jsou vytvořeny funkce.

```
#include voicexml.conf

[service]
exten => service,1,Set(vxml=${SIP_URI_OPTIONS()})
same => n,Set(vxml=${CUT(vxml,=,2)})

;Shell command
same => n,Set(parse=${SHELL(sh /root/parse2.sh ${vxml})})

exten => service,n,Set(context=${SHELL(echo "${parse}" | grep -w "*" |
 grep -v "?xml" | grep -v "vxml"| awk '{print $1}' | tr -d ' ' |
  tr '\r\n' '+')})

;Loop
same => n,MSet(x=$[0])
same => n(next),GotoIf($[ ${x}<${FIELDQTY(context,+,)}]?dosomething:continue)
same => n(dosomething),Set(tmp=${CUT(context,+,$[${x}+1])})
```

```
same => n,Goto(${tmp},s,1)

same => n(cc),MSet(x=$[${x}+1])
same => n,Goto(next)
same => n(continue),NoOp(End of Loop)

same => n,Hangup(102)
```

Veškeré funkce se jmenují stejně jako jednotlivé příkazy VoiceXML dokumentu. Tímto způsobem se dají jednoduše přidávat a další příkazy, které nejsou zatím implementovaná.

Příklad implementace příkazu audio

```
[audio]
exten => s,1,Set(audiovxml=${SHELL(echo "${parse}" | grep -e audio |
 tr -d ' ' | cut -d '=' -f 2 | sed 's/.$//' )})
same => n,Playback(${CUT(audiovxml,.,1)})
same => n,Goto(service,service,cc)
```

Příklad implementace příkazu prompt s Festivalem a Read s detekci přijatého čísla.

```
[prompt]
exten => s,1,Verbose(2,Performing Festival PROMPT)
same => n,Set(choice=${SHELL(echo "${parse}" | grep -w choice |
 awk '{printf $1}')})
same => n,Set(text=${SHELL(echo "${parse}" | grep -e prompt |
cut -d '*' -f 2 | rev | cut -d '/' -f 2 | rev | tr -d '\r\n')})

same => n,GotoIf($[ 0<${FIELDQTY(choice,choice,)}]?havechoice:nochoice)
;-----------------------------------------------------------------
same => n(havechoice),System(echo "${text}" |
 /usr/bin/text2wave -scale 1.5 -F 8000 -o /tmp/${CDR(uniqueid)}.wav)
same => n,Read(test,"/tmp/${CDR(uniqueid)}",1,,,15)
;same => n,Playback(/tmp/${CDR(uniqueid)})

same => n,MSet(i=$[0])
same => n,Set(numchoice=${SHELL(echo "${parse}" | grep -e choice |
 cut -d '=' -f 2 | awk '{print $1}' | tr '\r\n' '+')})
;same => n,Set(foof=${FIELDQTY(numchoice,+,)})
same => n(nextnum),GotoIf($[LEN(${test}) == 0]?nothing:gotdigit)
same => n(gotdigit),Set(iddd=${CUT(numchoice,+,$[${i}+1])})
same => n,GotoIf($[${test} != ${iddd}]?doit:gotit)

same => n(doit),NoOp(NEXT ONE)

same => n,MSet(i=$[${i}+1])
same => n,GotoIf($[ ${test} < ${FIELDQTY(numchoice,+,)}]?nexxt:nothing)
```

```
same => n(nexxt),Goto(nextnum)

same => n(nothing),NoOp(WRONG CHOICE !)
same => n,Goto(end)

same => n(gotit),NoOp(${test} == ${iddd} == ${i})
same => n,Set(tmp2=${SHELL(echo "${parse}" | grep -e choice |
 grep -w ${test} | cut -d '=' -f 3 | sed 's/.$//' )})
same => n,NoOp(${tmp2})

same => n,Goto(end)
;-------------------------------------------------------------------
same => n(nochoice),System(echo "${text}" |
 /usr/bin/text2wave -scale 1.5 -F 8000 -o /tmp/${CDR(uniqueid)}.wav)
same => n,Playback(/tmp/${CDR(uniqueid)})

same => n(end),System(rm -f /tmp/${CDR(uniqueid)}.wav)
same => n,Goto(service,service,cc)
```

Další příklady implementace jsou uvedeny v příloze

Kapitola 7

Testování realizovaného řešení

V téhle kapitole se budeme zabývat testování realizovaných řešení. Hlavní nástrojem pro testování byl použit program SIPp. Dalším programem je WireShark. Jednotlivé programy budou vysvětleny níže a použití a konfigurace.

7.1 SIPp

Sipp je Open Source nástroj pro testování provozu na protokol SIP. Pro testování je potřeba mít připravený testovací scénář. Kde jsou uvedeny všechny parametry pro testování. Program slouží pro optimalizaci provozu v ústřednách, kteří používají SIP protokol. Výhodou SIPp je velká variabilita testovacích scénářů a jednoduchost na ovládání, protože se program spouští v příkazové řádce, a běží pod operačním systémem Linux/Unix. Nevýhodou je že neumí pracovat s médií. Například odeslání zvukové stopy.

7.1.1 SIPp účet pro Asterisk

Konfigurace SIPp účtu pro Asterisk v souboru "sip.conf".

```
[sipp]
type=friend
context=service
dtmfmode=info
host=dynamic
user=sipp
canreinvite=no
```

7.1.2 SIPp scénař s VoiceXML parametrem

Část scénáře s odesíláním INVITE a parametr voicexml.

```
<send retrans="500">
<![CDATA[
```

```
INVITE sip:[service]@[remote_ip]:[remote_port];voicexml=menu2.vxml SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[call_number]
To: sut <sip:sipp@[remote_ip]:[remote_port]>
Call-ID: [call_id]
Cseq: 1 INVITE
Contact: sip:sipp@[local_ip]:[local_port]
Max-Forwards: 70
Subject: invite Test
Content-Type: application/sdp
Content-Length: [len]

]]>
</send>
```

Další částí je odesílání DTMF INFO s vybraným signálem.

```
<send>
<![CDATA[

INFO sip:[service]@[remote_ip]:[remote_port] SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[call_number]
To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
Call-ID: [call_id]
CSeq: 2 INFO
Max-Forwards: 70
Subject: info INFO
Content-Type: application/dtmf-relay
Content-Length: 22

Signal=2
Duration=250
]]>
</send>
```

7.1.3 Testování scénaře

Příkaz pro spuštění SIPp a testování scénáře

```
# sipp -sf invite.xml 192.168.0.111 -m 1
```

- Parametr ”-sf”je načtení scénáře ze souboru.
- IP adresa je adresa testovaného Asterisku.

```
Resolving remote host '192.168.0.111'... Done.
------------------------------ Scenario Screen -------- [1-9]: Change Screen --
  Call-rate(length)   Port   Total-time  Total-calls  Remote-host
  10.0(0 ms)/1.000s   5060       0.85 s            1  192.168.0.111:5060(UDP)

  Call limit reached (-m 1), 0.000 s period  0 ms scheduler resolution
  0 calls (limit 600)                  Peak was 1 calls, after 0 s
  0 Running, 3 Paused, 0 Woken up
  0 dead call msg (discarded)          0 out-of-call msg (discarded)
  1 open sockets

                                Messages  Retrans   Timeout   Unexpected-Msg
     INVITE ---------->         1         0         0
        100 <----------         1         0         0         0
        180 <----------         0         0         0         0
        200 <----------  E-RTD1 1         0         0         0
        ACK ---------->         1         0
       INFO ---------->         1         0
        200 <----------  E-RTD1 1         0         0         0
      Pause [    20.0s]        1                             1
------------------------------ Test Terminated --------------------------------


------------------------------ Statistics Screen ------- [1-9]: Change Screen --
  Start Time              | 2019-01-07   19:30:08.430779 1546885808.430779
  Last Reset Time         | 2019-01-07   19:30:09.287576 1546885809.287576
  Current Time            | 2019-01-07   19:30:09.287669 1546885809.287669
--------------------------+---------------------------+--------------------------
  Counter Name            | Periodic value            | Cumulative value
--------------------------+---------------------------+--------------------------
  Elapsed Time            | 00:00:00:000000           | 00:00:00:856000
  Call Rate               |    0.000 cps              |    1.168 cps
--------------------------+---------------------------+--------------------------
  Incoming call created   |        0                  |        0
  OutGoing call created   |        0                  |        1
  Total Call created      |                           |        1
  Current Call            |        0                  |
--------------------------+---------------------------+--------------------------
  Successful call         |        0                  |        0
  Failed call             |        0                  |        1
--------------------------+---------------------------+--------------------------
  Response Time 1         | 00:00:00:000000           | 00:00:00:453000
  Call Length             | 00:00:00:000000           | 00:00:00:750000
------------------------------ Test Terminated --------------------------------

2019-01-07      19:30:09.287366 154
```

Obrázek 7.1: Ukázka výsledku testování v Sipp

- Parametr ”-m 1”je maximální počet spojení

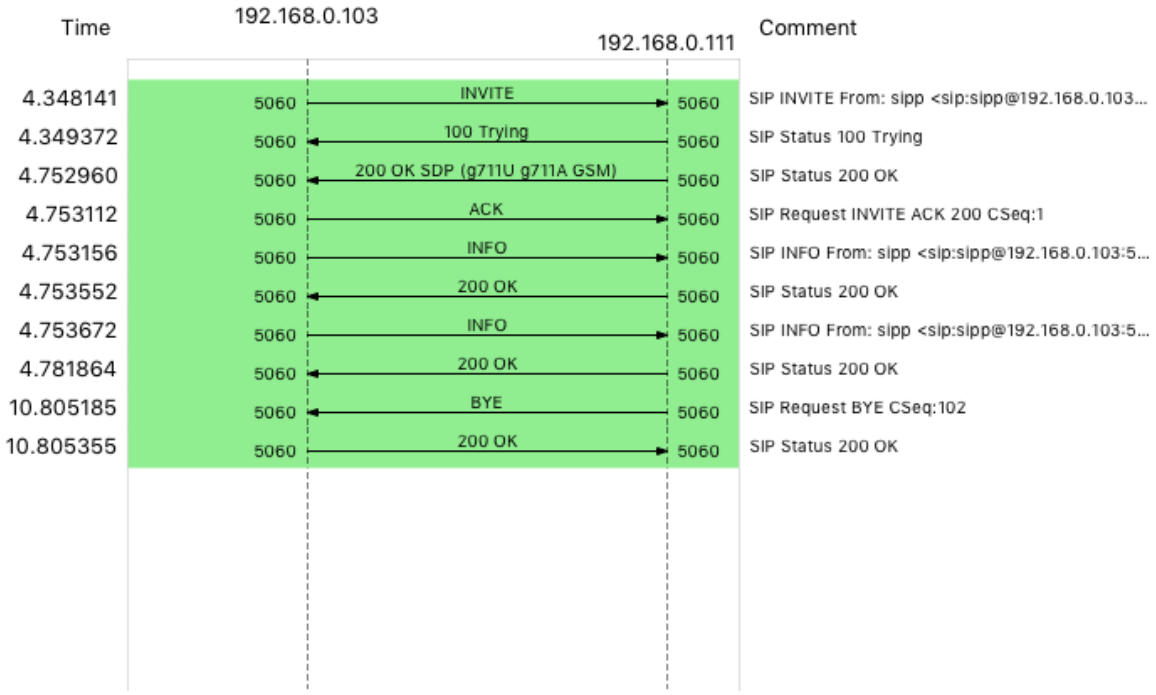
Celý scénář je uveden v příloze C.

7.2 WireShark

WireShark je software, který analyzuje veškerou komunikaci na daném síťovém zařízení. Software mí grafické rozhraní tak i terminálovou (bez GUI). Na obrázku 7.2 je znázorněná komunikace mezi SIPp a Asterisk. V prvním kroku se posílá INVITE s parametrem voicexml. Dale jsou odeslány 2x INFO s DTMF volbou, ve vrchním INFO je DTMF signál v hodnotě 2 a ve spodním je DTMF signál na hodnotě #.

7.3 Výpis z Asterisku

Výpis průběhu komunikace v Asterisk je uveden v příloze D



Obrázek 7.2: Příklad komunikace mezi SIPp a Asteriskem, na IP adrese 192.168.0.103 je SIPp a na 192.168.0.111 je Asterisk

Závěr

Cílem diplomové práce byla integrace hlasového dialogu VoiceXML do softwarové ústředny Asterisk. Hlavním cílem měla být implementace SIPového rozhraní s VoiceXML Media Službou. Veškeré existují VoiceXML jsou bud nefunkční nebo jsou z finančního hlediska drahé, a uzavřené.

Nejprve byla v práci popsána struktura IMS sítě, která nám dala základ k popsání systému Asterisk a hlasového dialogu VoiceXML. Na základě těchto znalosti byla provedena analýza možnosti implementace VoiceXML a funkcionalit do Asterisku. Tato funkcionalita byla nejprve analyticky popsána a bylo navrhnuto řešení její implementace. V poslední fázi byla provedena reálná konfigurace těchto funkcionalit, otestována a popsáná všemi omezeními a možnostmi řešení. Jedná se o funkci Festival a DialPlan, jsou v Asterisku a obsahují VoiceXML.

Nejvýhodnější volbou pro úpravu systému Asterisk je použití SIPové funkce Channel SIP. Přes kterou prochází veškerá komunikace, výhodou této úpravy je možnost využití ve starší verzích systému Asterisk. Problém funkci Channel SIP, který neuměl pracovat s SIP URI podle standardu RFC 5552. Tento problém se vyřešil úpravou funkce Channel SIP. Na základě této úpravy jsme mohly přijímat s SIP URI, která obsahovala vstupní parametr VoiceXML.

Dalším cílem práce bylo potřeba v Dialplan upravit a vytvořit skript, který by byl schopen vytáhnout potřebná data. Výhodou externího skriptu je lepší přehlednost a nižší zátěž na systém Asterisk. Což se mi podařilo a vytvořil způsob volání VoiceXML příkazu, který je modulární a dá se kdykoliv rozšířit o další příkazy.

Aplikace Festival je program co provádí syntézu řeči. Výhodou je lepší interakce s uživatelem. Jediný problém je, Festival je dobrý pouze pro testování. Existuje možnost jiného programu, jako například Cepstral má lepší syntézu, ale je k dispozici pouze za poplatek. Což v našem případě není třeba.

Existovali by další možnosti implementace VoiceXML do Asteriku, které by řešil hlasové rozpoznání textu, doprogramování dalších příkazů. V této práci jsme se zaměřili, základní implementaci hlasového dialogu VoiceXML do systému Asterisk. A otestovat funkčnosti základních příkazů.

Literatura

[1] WIJA, Tomáš, David ZUKAL a Miroslav VOZŇÁK. *Asterisk a jeho použití: Technická zpráva 12/2005* [online]. In: . Zikova 4: CESNET, 2005, 30.10.2005, s. 38 [cit. 2018-12-31]. Dostupné z: http://archiv.cesnet.cz/doc/techzpravy/2005/voip/asterisk.pdf

[2] Asterisk config files. *Voip-info.org* [online]. [cit. 2019-01-01]. Dostupné z: https://www.voip-info.org/asterisk-config-files

[3] Asterisk (PBX). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-12-31]. Dostupné z: https://en.wikipedia.org/wiki/Asterisk_(PBX)

[4] POOLE, Ian. IMS: IP Multimedia Subsystem Tutorial. *Radio-Electronics.com* [online]. Velká Británie: Adrio Communications, 2010 [cit. 2018-12-31]. Dostupné z: http://www.radio-electronics.com/info/telecommunications_networks/ims-ip-multimedia-subsystem/tutorial-basics.php

[5] IP IMS (IP Multimedia Subsystem). *IPv6* [online]. [cit. 2019-01-08]. Dostupné z: https://www.ipv6.com/general/ip-ims-ip-multimedia-subsystem/

[6] IP Multimedia Subsystem. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-07]. Dostupné z: https://en.wikipedia.org/wiki/IP_Multimedia_Subsystem

[7] Session Initiation Protocol. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-01-08]. Dostupné z: https://cs.wikipedia.org/wiki/Session_Initiation_Protocol

[8] DAVENPORT, Malcolm, NEWTON, Rusty, ed. Types of Asterisk Modules. *Asterisk Wiki* [online]. Digium, 2014-10-17 [cit. 2018-12-31]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Types+of+Asterisk+Modules

[9] DAVENPORT, Malcolm, NEWTON, Rusty, ed. Asterisk Architecture, The Big Picture. In: *Asterisk Wiki* [online]. Digium, Aug 06, 2014 [cit. 2019-01-08]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Asterisk+Architecture%2C+The+Big+Picture

[10] MCGLASHAN, Scott, ed. Voice Extensible Markup Language (VoiceXML) Version 2.0: W3C Recommendation. In: *W3C* [online]. 16.3.2004 [cit. 2019-01-03]. Dostupné z: https://www.w3.org/TR/voicexml20/Images/image005.gif

[11] VoiceXML. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-12-31]. Dostupné z: https://en.wikipedia.org/wiki/VoiceXML

[12] *VoiceXML Forum* [online]. 445 Hoes Lane [cit. 2018-12-31]. Dostupné z: http://www.voicexml.org/

[13] VoiceXML Tutorial. *Voximal* [online]. 4 Rue des Prairies: Ulex Innovative Systems, 21/09/2016 [cit. 2019-01-04]. Dostupné z: https://wiki.voximal.com/doku.php?id=tutorial:start#voicexml_examples

[14] POIKSELKA, Miikka, Aki NIEMI, Hisham KHARTABIL a Georg MAYER. *The IMS: IP multimedia concepts and services*. 2nd ed. Hoboken, NJ: J. Wiley, 2006. ISBN 04-700-1906-9.

[15] BRUKE, Dave a Mark SCOTT. *RFC 5552: SIP Interface to VoiceXML Media Services* [online]. Květen 2009, , 35 [cit. 2018-12-31]. DOI: 10.17487/RFC5552. Dostupné z: https://tools.ietf.org/html/rfc5552

[16] KUMAR APPARI, Kranti. Customized IVR Implementation Using Voicexml on SIP (Voip) Communication Platform. *International Journal of Modern Engineering Research* [online]. ijmer, Nov. - Dec., **2012**(Vol. 2, Issue. 6), 4239-4243 [cit. 2019-01-02]. ISSN 2249-6645. Dostupné z: http://www.ijmer.com/papers/Vol2_Issue6/BX2642394243.pdf

[17] CAMARILLO, Gonzalo a Miguel A. GARCIA-MARTIN. *The 3G IP multimedia subsystem (IMS): merging the Internet and the cellular worlds*. 2nd ed. Hoboken, NJ: J. Wiley, c2006. ISBN 978-047-0018-187.

[18] RUSSELL, Bryant, Madsen LEIF a Jim VAN MEGGELEN. *Asterisk: the definitive guide*. Fourth edition. Sebastopol: O'Reilly, [2013]. ISBN 978-1-449-33242-6.

[19] ROY, R.R. *Handbook on Session Initiation Protocol: Networked Multimedia Communications for IP Telephony: Networked Multimedia Communications for IP Telephony*. CRC Press, 2016. ISBN 9781498747714. Dostupné také z: https://books.google.cz/books?id=cHOmCwAAQBAJ

[20] OSHRY, Matt, ed. Voice Extensible Markup Language (VoiceXML) 2.1: W3C Recommendation. *W3C* [online]. 19.6.2007 [cit. 2019-01-08]. Dostupné z: https://www.w3.org/TR/voicexml21/

Příloha A

Další Konfigurace Asterisk

A.1 Konfigurace modulů

1. Asterisk hlavní menu select

```
# make menuselect
```

2. Načítání mp3 knihovny

```
# contrib/scripts/get_mp3_source.sh
```

3. Instalace modulů

```
# make install
```

A.2 Konfigurace uživatele Asterisku

```
# useradd -m asterisk
# chown asterisk.asterisk /var/run/asterisk
# chown -R asterisk.asterisk /etc/asterisk
# chown -R asterisk.asterisk /var/{lib,log,spool}/asterisk
# chown -R asterisk.asterisk /usr/lib64/asterisk
# systemctl restart asterisk
# systemctl status asterisk
```

A.3 Konfigurace pravidla firewall

```
# systemctl start firewalld
# systemctl enable firewalld
```

```
# firewall-cmd --zone=public --add-port=5060/udp --permanent
# firewall-cmd --zone=public --add-port=5060/tcp --permanent
# firewall-cmd --zone=public --add-port=5061/udp --permanent
# firewall-cmd --zone=public --add-port=5061/tcp --permanent
# firewall-cmd --zone=public --add-port=4569/udp --permanent
# firewall-cmd --zone=public --add-port=5038/tcp --permanent
# firewall-cmd --zone=public --add-port=10000-20000/udp --permanent

# firewall-cmd --reload
```

A.4 Nastavení databáze

```
# mysql -u root -p
Enter password:******
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 11
Server version: 5.5.44-MariaDB MariaDB Server

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create user 'asterisk'@'localhost' identified by '******';
MariaDB [(none)]> create database asterisk;
MariaDB [(none)]> create database cdrdb;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON asterisk.* TO
asterisk@localhost IDENTIFIED BY '******';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cdrdb.* TO
asterisk@localhost IDENTIFIED BY '******';
MariaDB [(none)]> flush privileges;
MariaDB [(none)]>
```

Příloha B

Parsování XML parse2.sh

```
#!/bin/bash
# Skrip for parse xml
read_dom() {
 local IFS=\>
 read -d \< ENTITY CONTENT
 local ret=$?
 TAG_NAME=${ENTITY%% *}
 ATTRIBUTES=${ENTITY#* }
 if [ "$ENTITY" == "/vxml" ]; then
  return 0
 else
  return $ret
 fi
}

while read_dom; do
CONTENT="$(echo $CONTENT |tr -d '\n[]' | sed -e 's/^[ \t]*//')"

if [ "$ENTITY" == "$TAG_NAME" -a "$ENTITY" == "$ATTRIBUTES" -a -z "$CONTENT" ]; then
 if [ -n "$ENTITY" ]; then
  if [[ ${ENTITY::1} == "/" ]]; then
   echo "$ENTITY"
  else
   tmp="$(echo $ENTITY | rev)"
  if [[ ${tmp::1} == "/" ]]; then
   echo "$ENTITY"
  else
   echo "$ENTITY"
   fi
  fi
 fi
else
```

```
 if [ -z "$CONTENT" ]; then
  ATTRIBUTES="$(echo $ATTRIBUTES |tr -d '\n[]' | sed -e 's/^[ \t]*//')"
  echo "$TAG_NAME * $ATTRIBUTES" # Atributy inside <...>
 else

  if [ "$TAG_NAME" == "$ATTRIBUTES" ]; then
   echo "$ENTITY * $CONTENT"
  else
   echo "$TAG_NAME * $ATTRIBUTES >> $CONTENT"
  fi
 fi
fi

done < /root/$1
```

Příloha C

Scénář invite.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<scenario name="Test Asterisk">

<send retrans="500">
<![CDATA[

INVITE sip:[service]@[remote_ip]:[remote_port];voicexml=menu2.vxml SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[call_number]
To: sut <sip:sipp@[remote_ip]:[remote_port]>
Call-ID: [call_id]
Cseq: 1 INVITE
Contact: sip:sipp@[local_ip]:[local_port]
Max-Forwards: 70
Subject: invite Test
Content-Type: application/sdp
Content-Length: [len]

]]>
</send>

<recv response="100" optional="true" >
</recv>

<recv response="180" optional="true" >
</recv>

<recv response="200" rtd="true">
</recv>

<send>
<![CDATA[
```

```
ACK sip:[service]@[remote_ip]:[remote_port] SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[call_number]
To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
Call-ID: [call_id]
CSeq: 1 ACK
Contact: sip:sipp@[local_ip]:[local_port]
Max-Forwards: 70
Subject: invite ACK
Content-Length: 0

]]>
</send>

<send>
<![CDATA[

INFO sip:[service]@[remote_ip]:[remote_port] SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[call_number]
To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
Call-ID: [call_id]
CSeq: 2 INFO
Max-Forwards: 70
Subject: info INFO
Content-Type: application/dtmf-relay
Content-Length: 22

Signal=2
Duration=250
]]>
</send>

<recv response="200" rtd="true">
</recv>

<pause milliseconds="20000">

<recv request="BYE">
</recv>

<!-- The 'crlf' option inserts a blank line in the statistics report. -->
<send>
<![CDATA[
```

```
BYE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[call_number]
To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
Call-ID: [call_id]
CSeq: 102 BYE
Contact: sip:sipp@[local_ip]:[local_port]
Max-Forwards: 70
Subject: Performance Test
Content-Length: 0

]]>
</send>

<recv response="200" crlf="true">
</recv>

<ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>

<CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>

</scenario>
```

Příloha D

Výpis z Asterisk CLI

```
Executing [service@service:1] Set("SIP/sipp-00000009", "vxml=voicexml=menu2.vxml")
 in new stack
-- Executing [service@service:2] Set("SIP/sipp-00000009", "vxml=menu2.vxml")
in new stack
-- Executing [service@service:3] Set("SIP/sipp-00000009", "parse=
-- ?xml * version="1.0" encoding="UTF-8"?
-- vxml * version="2.0" xmlns="http://www.w3.org/2001/vxml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd"
-- menu
-- property * name="inputmodes" value="dtmf"/
-- prompt * For sports press 1, For weather press 2, For Stargazer astrophysics press 3.
-- /prompt
-- choice * dtmf="1" next="http://www.sports.example.com/vxml/start.vxml"/
-- choice * dtmf="2" next="http://www.weather.example.com/intro.vxml"/
-- choice * dtmf="3" next="http://www.stargazer.example.com/astronews.vxml"/
-- /menu
-- /vxml
-- ") in new stack
-- Executing [service@service:4] Set("SIP/sipp-00000009", "context=property+prompt+")
 in new stack
-- Executing [service@service:5] MSet("SIP/sipp-00000009", "x=0") in new stack
-- Executing [service@service:6] GotoIf("SIP/sipp-00000009", "1?dosomething:continue")
 in new stack
-- Goto (service,service,7)
-- Executing [service@service:7] Set("SIP/sipp-00000009", "tmp=property") in new stack
-- Executing [service@service:8] Goto("SIP/sipp-00000009", "property,s,1") in new stack
-- Goto (property,s,1)
-- Executing [s@property:1] Set("SIP/sipp-00000009", "value=dtmf") in new stack
-- Executing [s@property:2] NoOp("SIP/sipp-00000009", "dtmf") in new stack
-- Executing [s@property:3] Goto("SIP/sipp-00000009", "service,service,cc") in new stack
```

```
-- Goto (service,service,9)
-- Executing [service@service:9] MSet("SIP/sipp-00000009", "x=1") in new stack
-- Executing [service@service:10] Goto("SIP/sipp-00000009", "next") in new stack
-- Goto (service,service,6)
-- Executing [service@service:6] GotoIf("SIP/sipp-00000009", "1?dosomething:continue")
 in new stack
-- Goto (service,service,7)
-- Executing [service@service:7] Set("SIP/sipp-00000009", "tmp=prompt") in new stack
-- Executing [service@service:8] Goto("SIP/sipp-00000009", "prompt,s,1") in new stack
-- Goto (prompt,s,1)
-- Executing [s@prompt:1] Verbose("SIP/sipp-00000009", "2,Performing Festival PROMPT")
 in new stack
== Performing Festival PROMPT
-- Executing [s@prompt:2] Set("SIP/sipp-00000009", "choice=choicechoicechoice")
 in new stack
-- Executing [s@prompt:3] Set("SIP/sipp-00000009", "text=
For sports press 1, For weather press 2, For Stargazer astrophysics press 3.")
 in new stack
-- Executing [s@prompt:4] GotoIf("SIP/sipp-00000009", "1?havechoice:nochoice")
 in new stack
-- Goto (prompt,s,5)
-- Executing [s@prompt:5] System("SIP/sipp-00000009", "echo "
For sports press 1, For weather press 2, For Stargazer astrophysics press 3."
| /usr/bin/text2wave -scale 1.5 -F 8000 -o /tmp/1546158007.18.wav")
in new stack
-- Executing [s@prompt:6] Read("SIP/sipp-00000009", "test,"/tmp/1546158007.18",1,,,15")
in new stack
-- Accepting a maximum of 1 digits.
-- <SIP/sipp-00000009> Playing '/tmp/1546158007.18.slin' (language 'en')
-- User entered '2'
-- Executing [s@prompt:7] MSet("SIP/sipp-00000009", "i=0") in new stack
-- Executing [s@prompt:8] Set("SIP/sipp-00000009", "numchoice=1+2+3+")
in new stack
[Dec 30 09:20:08] NOTICE[8220][C-00000009]: ast_expr2.y:761 compose_func_args: argbuf allo
[Dec 30 09:20:08] NOTICE[8220][C-00000009]: ast_expr2.y:780 compose_func_args: argbuf uses
-- Executing [s@prompt:9] GotoIf("SIP/sipp-00000009", "0?nothing:gotdigit")
in new stack
-- Goto (prompt,s,10)
-- Executing [s@prompt:10] Set("SIP/sipp-00000009", "iddd=1") in new stack
-- Executing [s@prompt:11] GotoIf("SIP/sipp-00000009", "1?doit:gotit")
in new stack
-- Goto (prompt,s,12)
-- Executing [s@prompt:12] NoOp("SIP/sipp-00000009", "NEXT ONE") in new stack
-- Executing [s@prompt:13] MSet("SIP/sipp-00000009", "i=1") in new stack
-- Executing [s@prompt:14] GotoIf("SIP/sipp-00000009", "1?nexxt:nothing")
in new stack
```

```
-- Goto (prompt,s,15)
-- Executing [s@prompt:15] Goto("SIP/sipp-00000009", "nextnum") in new stack
-- Goto (prompt,s,9)
[Dec 30 09:20:08] NOTICE[8220][C-00000009]: ast_expr2.y:761 compose_func_args: argbuf allo
[Dec 30 09:20:08] NOTICE[8220][C-00000009]: ast_expr2.y:780 compose_func_args: argbuf uses
-- Executing [s@prompt:9] GotoIf("SIP/sipp-00000009", "0?nothing:gotdigit")
in new stack
-- Goto (prompt,s,10)
-- Executing [s@prompt:10] Set("SIP/sipp-00000009", "iddd=2") in new stack
-- Executing [s@prompt:11] GotoIf("SIP/sipp-00000009", "0?doit:gotit")
 in new stack
-- Goto (prompt,s,18)
-- Executing [s@prompt:18] NoOp("SIP/sipp-00000009", "2 == 2 == 1")
 in new stack
-- Executing [s@prompt:19] Set("SIP/sipp-00000009", "tmp2=
http://www.weather.example.com/intro.vxml
-- ") in new stack
-- Executing [s@prompt:20] NoOp("SIP/sipp-00000009", "http://www.weather.example.com/intro
-- ") in new stack
-- Executing [s@prompt:21] Goto("SIP/sipp-00000009", "end") in new stack
-- Goto (prompt,s,24)
-- Executing [s@prompt:24] System("SIP/sipp-00000009", "rm -f /tmp/1546158007.18.wav") in
-- Executing [s@prompt:25] Goto("SIP/sipp-00000009", "service,service,cc")
in new stack
-- Goto (service,service,9)
-- Executing [service@service:9] MSet("SIP/sipp-00000009", "x=2")
 in new stack
-- Executing [service@service:10] Goto("SIP/sipp-00000009", "next")
in new stack
-- Goto (service,service,6)
-- Executing [service@service:6] GotoIf("SIP/sipp-00000009", "1?dosomething:continue") in
-- Goto (service,service,7)
-- Executing [service@service:7] Set("SIP/sipp-00000009", "tmp=")
 in new stack
-- Executing [service@service:8] Goto("SIP/sipp-00000009", ",s,1")
in new stack
-- Goto (service,s,1)
-- Executing [s@service:1] NoOp("SIP/sipp-00000009", ""KONEC"")
in new stack
-- Executing [s@service:2] Goto("SIP/sipp-00000009", "service,service,cc")
in new stack
-- Goto (service,service,9)
-- Executing [service@service:9] MSet("SIP/sipp-00000009", "x=3")
in new stack
-- Executing [service@service:10] Goto("SIP/sipp-00000009", "next")
 in new stack
```

```
-- Goto (service,service,6)
-- Executing [service@service:6] GotoIf("SIP/sipp-00000009", "0?dosomething:continue") in
-- Goto (service,service,11)
-- Executing [service@service:11] NoOp("SIP/sipp-00000009", "End of Loop")
in new stack
-- Executing [service@service:12] Hangup("SIP/sipp-00000009", "102")
in new stack
```

Příloha E

Ukázka VoiceXML aplikace v menu.vxml

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2001/vxml
http://www.w3.org/TR/voicexml20/vxml.xsd">
<menu>
<property name="inputmodes" value="dtmf"/>
<prompt>
For sports press 1, For weather press 2, For Stargazer astrophysics press 3.
</prompt>
<choice dtmf="1" next="http://www.sports.example.com/vxml/start.vxml"/>
<choice dtmf="2" next="http://www.weather.example.com/intro.vxml"/>
<choice dtmf="3" next="http://www.stargazer.example.com/astronews.vxml"/>
</menu>
</vxml>
```

Příloha F

Ukázka VoiceXML aplikace v example.vxml

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<form>
<block> <audio src="hello.wav"/> </block>
</form>
</vxml>
```

Příloha G

Obsah přiloženého CD

- install.txt - Instalační příkazy
- README.txt - Informace ohledně CD
- diep_luong.pdf - Text práce
- parse2.sh - Skript na parsování XML tagů
- sip_uri_opts.diff - Patch pro úpravu chan_sip.c
- Asterisk - Složka s konfiguračními soubory Asterisk /etc/asterisk
  - extension.conf - DialPlan
  - sip.conf - Konfigurace SIP účtu
  - voicexml.conf - Externí funkce k DialPlan
- SIPp - Složka s SIP scénářem
  - invite.xml - Upravení INVITE
- VoiceXML - Složka s VoiceXML dokumenty
  - example.vxml - Příklad přehrání audia
  - prompt.vxml - Příklad výpis textu a přehrání pomocí Festival
  - menu2.vxml - Příklad výpis textu a přehrání pomocí Festival a detekce DTMF vstupu