VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

SIP server s pokročilými funkcemi

SIP Server with Advanced Features

Rok 2016

Bc. Tomáš Škařupa

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Tomáš Škařupa**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T059 Mobilní technologie

Téma: SIP server s pokročilými funkcemi
SIP Server with Advanced Features

Jazyk vypracování: čeština

Zásady pro vypracování:

Jedním z trendů současných komunikačních řešení je green computing, ke kterému vede efektivní alokace výpočetních prostředků. Platforma openWRT je jedním z OS určených pro embedded řešení. Cílem diplomové práce je praktická realizace Asterisku v prostředí openWRT, jeho doplnění o bezpečnostní prvky jako je Fail2ban či IDS/IPS a pokročilé služby jako je presence a IM. Rovněž je požadováno stanovení výkonnostních limitů vytvořeného řešení pro dostupný HW, který by měl respektovat ekonomickou náročnost využití (např. výrobce TP-link).

1. Projekt Asterisk a jeho možnosti.
2. Open-source platforma openWRT.
3. Návrh a praktická realizace SIP serveru Asterisk v prostředí openWRT.
4. Řešení bezpečnosti v realizovaném návrhu.
5. Implementace pokročilých funkcí Asterisku na platformě openWRT.
6. Stanovení výkonnostích limitů a zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

M. Vozňák.*Voice over IP*. VŠB-TU Ostrava, vysokoškolská skripta, druhé vydání, 2009.
L. Surhone, M. Tennoe.*OpenWrt*. Betascript Publishing, 2011, ISBN 978-6135271591.
F. Rezac, M. Voznak, J. Safarik, P. Partila, K. Tomala. Security solution against denial of service attacks in BESIP system. In *Proc. SPIE 8755, Mobile Multimedia/Image Processing, Security, and Applications.*, Baltimore, 2013. DOI: 10.1117/12.2015423.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Miroslav Vozňák, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 29.04.2016

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *20. dubna 2016*

Škaruza
.....
podpis studenta

Poděkování

Rád bych poděkoval doc. Ing. Miroslavu Vozňákovi, Ph.D. za jeho vedení, odbornou pomoc a konzultaci při vytváření této diplomové práce.

Abstrakt

Tato práce se zaměřuje na zprovoznění SIP serveru na hardwaru s nízkými nároky na výpočetní výkon, zabezpečení serveru a stanovení výkonnostních limitů použitého hardwaru. V práci je popsán způsob konfigurace a instalace operačního systému OpenWrt pomocí tzv. křížové kompilace. Velká pozornost je věnována telefonní ústředně Asterisk a signalizačnímu protokolu SIP. Práce seznamuje čtenáře s XMPP serverem a způsobem jeho instalace a konfigurace v prostředí OpenWrt a jeho propojení s telefonní ústřednou Asterisk a následným vytvořením presence a IM. V rámci této diplomové práce je popsán způsob zabezpečení Asterisk severu a Prosody serveru pomocí fail2ban. V závěru práce je popsán způsob instalace celého systému na Raspberry Pi a stanovení výkonnostních limitů pomocí programu SIPp.

Klíčová slova

VoIP, OpenWrt, Asterisk, SIP, XMPP, Prosody, Raspberry Pi, SIPp

Abstract

This work focuses on how to get SIP server to work on hardware with lower requirements on computing power and server security; and setting performance limits of the used hardware. Moreover, it describes the ways how to configure and install operating system OpenWrt by the means of cross compilation. The biggest part is dedicated to Asterisk call center and signaling protocols SIP. Reader learns how to work with XMPP server and what are the ways of server installation and configuration in OpenWrt. Furthermore, connection of XMPP server with Asterisk call center, and creation of presence and IM are illustrated. In addition, this thesis explores the ways of securing Asterisk server and Prosody server by using fail2ban. And finally, the last part describes installation of the complete system on Raspberry Pi and setting performance limits via SIPp program.

Keywords

VoIP, OpenWrt, Asterisk, SIP, XMPP, Prosody, Raspberry Pi, SIPp

Obsah

Seznam použitých zkratek

Zkratka	Význam
ADPCM	Adaptive Differential Pulse-Code Modulation
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DOS	Denial of Service
DPCM	Differential Pulse-Code Modulation
FTP	File Transfer Protocol
GCC	GNU Compiler Collection
GSM	Global System for Mobile communications
HTTP	HyperText Transport Protocol
HW	Hardware
IAX	Inter-Asterisk eXchange
IETF	Internet Engineering Task Force
IDE	Integrated Development Environment
IDS	Intrusion Detection Systém
IM	Instant Messaging
IP	Internet Protocol
IPS	Intrusion Prevention Systém
IVR	Interactive Voice Response
JID	Jabber IDentifier
LTS	Long Term Support
NAT	Network Address Translation
NFS	Network File Systém
NIDS	Network Intrusion Detection Systems
PBX	Private Branch Exchange
PC	Personal Computer
PCM	Pulse-Code Modulation
PSTN	Public Switched Telephone Network
QoS	Quality of Service

RAM	Random Access Memory
RTP	Real-time Transport Protocol
RTCP	RTP Control Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UA	User Agent
UAS	User Agent Server
UAC	User Agent Client
URI	Uniform Resource Identifier
UDP	User Datagram Protocol
VoIP	Voice over Internet Protocol
XMPP	Extensible Messaging and Presence Protocol

Úvod

V dnešní době se stále častěji využívá pro telekomunikační služby protokol IP. Tato technologie se často označuje jako VoIP nebo internetová telefonie. VoIP je technologie, která umožňuje přenos dat, hlasu a videa. Tato rozšiřující se technologie vytváří konkurenci pro klasické provozovatele telefonních služeb. Dnes je běžné, že firmy v rámci své vnitřní sítě provozují nějaký komunikační systém založený na VoIP technologii. Nejčastěji se k tomuto účelu používá SIP protokol nebo IAX, případně speciální firemní protokoly. Protokol SIP má i tu výhodu, že dokáže většinou bez větších problému přecházet přes NAT. Důvodem, proč se firmy uchylují k implementaci této technologie je ušetření nákladů za provozování telefonních služeb.

Díky neustálému vývoji technologií dosahují dnešní embedded zařízení dostačeného výkonu a jsou schopny zvládnout i telefonní ústřednu. Raspberry Pi je malinký počítač, který se těší čím dál tím větší oblibě. Na zmíněné zařízení zareagovali i vývojáři operačního systému OpenWrt a vytvořili pro první dvě řady Raspberry Pi distribuci OpenWrt. Pro poslední, třetí, verzi Raspberry Pi zatím nebyla oficiální distribuce uvolněna.

Tato Diplomová práce se zaměřuje na zprovoznění SIP serveru na hardwaru s nízkými nároky na výpočetní výkon, zabezpečení SIP serveru, obohacení telefonní ústředny o pokročilou funkci a stanovení výkonnostních limitů použitého hardwaru.

První kapitola se zabývá projektem Asterisk a jeho možnostech využití v praxi. Tato kapitola obsahuje popis protokolů používaných ve VoP sítích. Velká pozornost je věnována signalizačnímu protokolu SIP. Ve druhé kapitole je popsána linuxová distribuce OpenWrt a jak si vytvořit tento systém dle představ pomocí křížové kompilace. V dalších kapitolách je řešena realizaci telefonní ústředny, za kterou byl zvolen Asterisk verze 11 a instalací jednotlivých modulů, které byly zapotřebí při řešení pokročilých funkcí. Ve čtvrté kapitole je vysvětlen způsob zabezpečení telefonní ústředny a to pomocí programu fail2ban nebo Snort. Poslední kapitoly patří implementaci pokročilých funkcí v prostředí Asterisku, propojení této telefonní ústředny s Prosody serverem, a implementaci celého systému na HW, za který byl zvolen Raspberry Pi 1B. V závěrečné kapitole je popsán testovací nástroj SIPp a StarTrinity. Pomocí nástroje SIPp byl stanoven výkonnostní limit pro Raspberry Pi 1B.

1 Projekt Asterisk a jeho možnosti

Asterisk je open-source softwarová ústředna BPX, která bývá nejčastěji instalována na standartních PC. Tato PBX ústředna nejčastěji běží na platformách Linux a Unix a slouží pro domácí uživatele, podniky, poskytovatele VoIP služeb. Systém je navržen tak, aby vytvořil rozhraní mezi telefonním hardwarem nebo softwarem a telefonní aplikací. Tento software se šíří jako licence GNU verze 2 nebo proprietární licence, která umožňuje využívat uzavřený zdrojový kód jako je např. G.729 kodek. Díky tomu, že je Asterisk napsán v jazyce C, lze ho uplatnit i v řadě embedded zařízeních jako jsou například routery. Asterisk byl vytvořen Markem Spencerem v roce 1999, který zveřejnil zdrojový jako open-source. O tři roky později založil vlastní firmu Digium, která se dlouhodobě zabývá vývojem Asterisku.

1.1 Možnosti Asterisku

Asterisk lze použít v těchto aplikacích:

- Různorodá VoIP gateway (MGCP, SIP, IAX, H.323)
  Gateway, která umožnuje propojeni klasických mobilních telefonů např. ze sítě GSM a VoIP telefonů. Propojení těchto dvou technologií nejčastěji používají firmy, aby snížili své náklady za provoz telefonů.
- Pobočková ústředna (PBX)
  Pobočková telefonní ústředna, PBX (Private branch exchange) je zařízení (počítač), které, umožňuje připojovat hovory, vytvořit připojení vzdálených telefonních poboček a telefonů, atd. Uživatelé se nejčastěji připojují k ústředně pomocí SIP hardwarového nebo softwarového telefony. Telefonní ústřednu je možné propojit s programem Skype, či Google Hangout.
- Interaktivní hlasový průvodce (IVR) server
  Jedná se systém využívaný v telekomunikačních službách, který je určen ke komunikaci se zákazníkem. Používá předdefinované hlasové segmenty. Tyto segmenty jsou určené ke zjištění základních údajů a zodpovězení jednoduchých dotazů např.: pro zjištění stavu Vašeho kreditu, stiskněte jedničku; pro možnost aktivace internetu dvojku; atd. Pokud je dotaz složitější, bývá hovor přesměrován na operátora.
- Softwarová ústředna (Softswitch)
  Pro vytvoření VoIP ústředny není potřeba žádný speciální HW, postačí obyčejný počítač, NUC, Raspberry Pi atd. Ústředna je realizovaná pomocí programu jakým je např. Asterisk. Nastavení ústředny se provádí buď přes webový prohlížeč nebo pomocí konfiguračních souborů
- Konferenční server
  Asterisk umožňuje vytvořit konferenční místnosti, kde může současně hovořit více lidí najednou.
- Šifrování telefonních nebo faxových volání
- Překlad čísel
- Řazení volání do front se vzdáleným zprostředkovatelem

Asterisk podporuje i pokročilé firemní služby

- Hudba na popředí čekajících ve frontě pro hovor
- Integrace text-to-speech modulů a rozpoznaní hlasu
- Propojení s PSTN sítí skrz digitální a analogové linky

V této kapitole bylo čerpáno z [1, 4, 5].

1.2 Protokoly

V této kapitole je uveden soupis nejběžnějších protokolů používaných ve VoIP sítích.

1.2.1 Protokol UDP a TCP

Oba protokoly spadají do transportní vrstvy v referenčním modelu ISO/OSI (4 vrstvy) a transportní vrstvy architektury TCP/IP (3 vrstva). Oba protokoly se používají k přenosu datových paketů mezi uzly až na místo určení.

1.2.1.1 ***TCP***

Protokol TCP se používá pro spolehlivý a spojovaný přenos dat. Spolehlivost se zajišťuje kontrolou doručení paketů (potvrzování). Pakety se na straně příjemce uspořádají tak, aby byly ve správném pořadí. Tento typ protokolu se používá pro: emailový server, ftp server, atd. [6]

1.2.1.2 ***UDP***

Protokol UDP se používá pro nespojované datagramové služby, nezaručuje ale doručení datagramů ani správné pořadí přenosu. Této vlastnosti se využívá v různých "real-time" přenosech, např. přenos videa nebo online hudby. Pokud se nějaké datagramy v sítí ztratí, koncový uživatel to pozná tak, že ve videu chybí kousek obrazu nebo na malý okamžik neuslyší zvuk. [6]

1.2.2 Protokol RTP

Jedná se o protokol pro přenos dat v reálném čase, zdrojem může být webová kamera nebo VoIP telefon. Tento protokol je standardem organizace IETF (Internet Engineering Task Force). Specifikace protokolu je volně dostupná na Internetu pod označením RFC 3550. Jeho hlavní úlohou je definování pořadových čísel paketů (sequence number), na základě kterých pak může aplikace na přijímací straně rozpoznat chybějící pakety nebo upravit jejich pořadí. RTP protokol nejčastěji používá transportní UDP protokol. Bezpečnou variantou tohoto protokolu je SRTP.

V případě ticha není přenášena žádná informace. Protistrana pouze vytváří šum pro navození přirozeného prostředí.

Protokol RTP dokáže označit obsah, tj. zdali se jedná o hlasovou nebo obrazovou zprávu a jakým způsobem byla komprimována (typ kodeku). Na přijímací straně se pak vyhodnocují daná data cílovou aplikací a data jsou převedena zpět do původní podoby. [7]

Protokol RTP je nešifrovaný a z toho důvodu, že u něj vzniká riziko odposlechu hovoru. Pro rekonstrukci hovoru se dá využit program Wireshark, který převede RTP tok na zvukovou stopu. V případě potřeby šifrování hovoru je možné využít protokol SRTP, který je odvozen od RTP.

1.2.3 **Protokol RTCP**

Tento protokol má za úkol sledovat kvalitu přenášených dat, které jsou přenášeny pomocí RTP. RTCP nepřenáší žádná vlastní uživatelská data, ale periodicky vysílá kontrolní data účastníkům účastnící se komunikace. Na základě získaných dat od účastníků vyhodnocuje následující parametry:

- zpoždění (Delay),
- kolísání zpoždění (Delay Variation),
- ztráta paketů (Packet Loss),
- počet přenesených bitů (Number of Transferred Bits),
- doba mezi vysláním paketů a příjmem odpovědi ze vzdálené strany.

1.2.4 **Protokol SDP**

Tento protokol bývá zapouzdřený v tělech některých SIP zpráv. Protokol nepřenáší uživatelská data, ale jen jejich popis. Bývá využíván ke komunikaci mezi účastnickými stranami. Obsahem komunikace může být domluva, jaké kódování multimediálních dat se použije, čísla portů pro RTP atd.[7]

1.2.5 **Protokol H.323**

Standart H.323 je signalizační protokol, který zastřešuje celou rodinu protokolů určených pro přenos hlasu, dat a videa. Tento protokol byl definovaný Mezinárodní telekomunikační Unií ITU a má celkem 7 verzí. První verze vznikla v roce 1996 a poslední verze vznikla v roce 2009. Samotný hovor je přenášen pomocí RTP protokolu.

Tento protokol v sobě obsahuje:

- H.225.0 – hovorová signalizace,
- Q.931 – nastavení volání a jeho ukončení,
- H.245 – protokol pro vyjednání parametrů multimediálních kanálů,
- H.235 – bezpečnostní a ověřovací mechanismy,
- RAS - řídí registraci, přístup a stav.

Při zasílání signalizačních informací pomocí H.225 se využívají jak protokoly UDP, tak i TCP. Standardně Gatekeeper naslouchá signalizaci H.225,0/RAS na portu UDP 1719 a případně i na portu 1718. Hovorová signalizace spojení H,225,0/Q.931 se přenáší na portu TCP 1720.

V této kapitole bylo čerpáno z [7].

1.2.6 Protokol IAX

Jedná se o komunikační protokol, který byl vytvořen v rámci projektu Asterisk PBX. Tento protokol vylepšoval dostupné VoIP protokoly. V dnešní době existuje verze 2 (IAX2). Protokol IAX se liší od ostatních protokolů především tak, že signalizační a mediální data jsem přenášena jedním datovým tokem. Data jsou posílána z jednoho portu a mají binární formát. Tento protokol využívá UDP protokol a díky tomu, že jsou signalizační a multimediální data multiplexována, snadněji přechází přes firewall a překlad adres NAT. [8]

1.3 Protokol SIP

Jedná se o signalizační protokol aplikační vrstvy, který vyvinula organizace IETF (Internet Engineering Task Force). Protokol SIP slouží k sestavení, modifikování a ukončení relace s jedním nebo více účastníky. V roce 1999 byl předložen navrhovaný standart protokolu SIP, který byl přijat jako RFC 2543. V tomto roce vznikla i pracovní skupina s názvem SIP, která převzala vývoj toho protokolu. Protokol má textovou strukturu podobnou např. protokolu HTTP (HyperText Transport Protocol). Textová struktura SIP napomáhá k jednoduchému ladění, ale je i díky tomu snadno rozšiřitelná. Komunikace mezi koncovými zařízeními probíhá pomocí protokolu SIP a je tvořena dialogy (jednotlivá spojení). Dialog je tvořen transakcemi, které jsou typu žádost a k ní náležící odpověď (odpovědi). SIP se stará jen o komunikaci a signalizaci multimediálních komponent, data jsou přenášena zcela nezávisle pomocí protokolů RTP. Tělo jednotlivých zpráv má formu <název> : <hodnota>, která popisuje přídavné informace. Protokol umí pracovat jak nad UDP tak i TCP protokolem a nejčastěji se používá v IP telefonii. [9]

SIP URI je uživatelovo SIP telefonní číslo, které připomíná e-mailovou adresu. Existují dvě podoby: SIP URI nebo SIPS URI. Formát zápisu pro obě verze je stejný, SIPS URI se však využívá pro zabezpečenou formu komunikace. URI obsahuje pole: user, password, host, port a případně doplňující informace týkající se URI parametrů. SIP URI má následující formát:

```
sip:user:password@host:port;uri-parameters?headers
```

User - uživatel

Jedná se o identifikátor konkrétního zdroje v síťové doméně. Tzv. "userinfo" URI může obsahovat položku user, password a @. Tato část je dobrovolná a nemusí se uvádět v případě, kdy vzdálený server nemá co do činění s rozlišením uživatelů, nebo pokud vzdálený server představuje onu službu, kterou chceme kontaktovat. Pokud je v adrese přítomný symbol @, nesmí chybět ani položka "user".

Password - heslo

Heslo je spojeno s konkrétním uživatelem a není z bezpečnostního důvodu doporučeno, aby bylo součásti SIP URI.

Hostname - host

Jedná se o jméno hostitelského systému, které poskytuje SIP službu. Hodnota může být zapsána buď ve tvaru úplného doménového jména nebo v číselné podobě - IPv4/IPv6 adresy.

Port

Tato položka specifikuje číslo portu, na který se má požadavek poslat. Pokud položka není specifikována, používá se implicitní hodnota 5060 pro TPC, UDP, SCTP; nebo 5061 pro SIPS, který je přenášen pomocí TCP

1.3.1 SIP zprávy

Zprávy protokolu SIP jsou dvojího druhu, jedná se o žádosti (metody) a odpovědi, někdy nazývané chybové hlášení. Každá zpráva obsahuje hlavičku zprávy (header) a může obsahovat tělo zprávy s popiskem médií (body, většinou SDP).

1.3.1.1 Metody (žádosti)

Žádosti jsou inicializují procedury, tzn., že sestavují, ukončují, modifikují spojení, případně oznamují požadavek v rámci další služby (IM).

Přehled nejčastějších žádostí:

- INVITE - žádost o navázání nového spojení nebo při potřebě změnit parametry již existujícího spojení,
- BYE - žádost o ukončení spojení,
- REGISTER - tato zpráva se používá k registraci současné adresy klienta u SIP serveru,
- ACK - potvrzení, které informuje protistranu, že přijal zprávu INVITE; patří k "three-way hand-shaking"; volaný periodicky zasílá zprávu 200 OK, dokud neobdrží ACK,
- CANCEL - používá se ke zrušení navozeného spojení, když není sestaven dialog (volaný ještě nepotvrdil INVITE),
- OPTIONS - speciální metoda, která se používá k zjištění vlastností SIP zařízení.

1.3.1.2 Odpovědi (response)

Odpovědi slouží k potvrzení, že byla žádost přijata a zodpovězena. Metodu ACK je jediná, na kterou se žádná odpověď nezasílá.

- 1xx - dočasná informativní odpověď - žádost je přijata a pokračuje se ve zpracování (100 TRYING -potvrzeni INVITE, 180 RINGING - telefon na protistraně vyzvání),
- 2xx - finální pozitivní odpověď - jedná se potvrzení v rámci úspěšné zpracování požadavku, např. 200 OK značí úspěšné potvrzení na předchozí událost,
- 3xx - značí přesměrování a je třeba vytvořit nový požadavek (generuje se při změně polohy uživatele),
- 4xx - značí chybu na straně klienta,
- 5xx - značí chybu na straně severu,
- 6xx - jedná se globální chybu.

Při vypracování této části kapitoly bylo čerpáno ze zdrojů [10,11].

1.3.2 Popis hlavičky SIP

Jednotlivé pakety se skládají z hlaviček a vlastních dat. Hlavička obsahuje informace o spojení (TTL, zdrojová cílová SIP adresa, použitý protokol, datum, atd.).

Výpis a popis některých polích SIP hlaviček

- From: adresa volajícího klienta, formát: <sip:jméno@doména:port>
- To: adresa volaného, formát: <sip:jméno@doména:port>
- Contact: aktuální skutečná adresa klienta, formát: <sip:jméno@IP_adresa>
- Call-ID: unikátní identifikace volání, formát: identifikátor@IP_adresa
- Cseq: identifikátor INVITE
- User-Agent: User Agent Server
- Date: datum
- Allow: pole podporovaných metod
- Content-Type: specifikace vnitřního protokolu
- Content-Length: délka těla v bajtech
- Marker: konec hlavičky

1.4 Základní prvky SIP architektury

Základními prvky SIP architektury jsou:

- User agent (UA)
- SIP proxy, registrar, redirect a location servery, často označováno jako SIP server.

User Agent (UA)

Jedná se o koncový prvek sítě - terminál, který je schopen zpracovat multimediální data. Terminál může být jak hardwarový, tak i softwarový SIPový telefon nebo výchozí brána do jiné sítě (PSTN). UA se dále dělí na User Agent Client (UAC) a User Agent Server (UAS). UAC má za úkol inicializaci spojení. UAS má na starosti generování odpovědí na příchozí žádosti. V koncovém SIP telefonu je implementován jak UAC tak i UAS. Úkolem serverů je zprostředkovat kontakt mezi volajícím a volanými tedy mezi UA. Celkem se rozlišují tři typy serverů: [12]

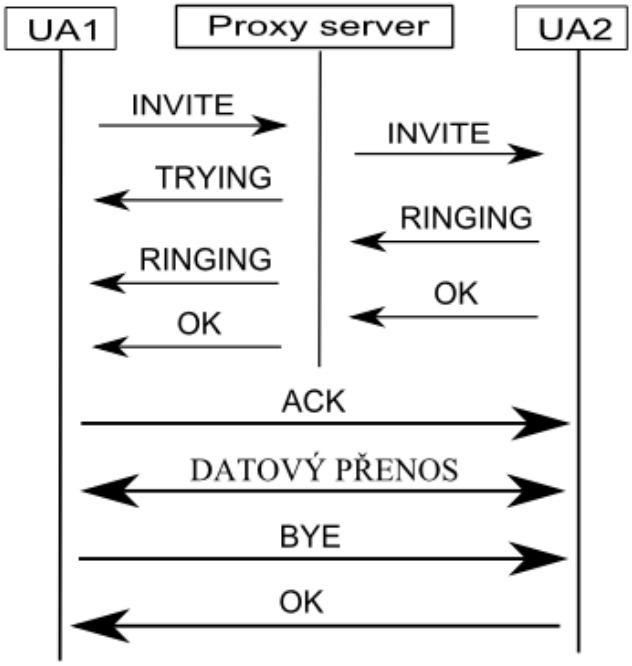
Proxy server

Tento server přijme žádost o spojení od UA případně od jiného proxy serveru. Po přijetí zprávy ji přepošle dalšímu serveru (pokud cílovou stanici nemá ve své správě) nebo jí předá cílovému UA ve své doméně. Jeho hlavním úkolem je, že zajišťuje směrování žádostí o spojení podle umístění adresáta. Existují dva základní proxy servery:

- Stateless (bezstavový),
- Stateful (má informace o stavech transakcí).

Ve stavu stateless, proxy server jen přeposílá zprávy, tzn., že nekontroluje smysluplnost zpráv a proto nezachytí replikující se zprávy. Proxy serveru ve stavu stateless trvá déle detekování nekonečné smyčky.

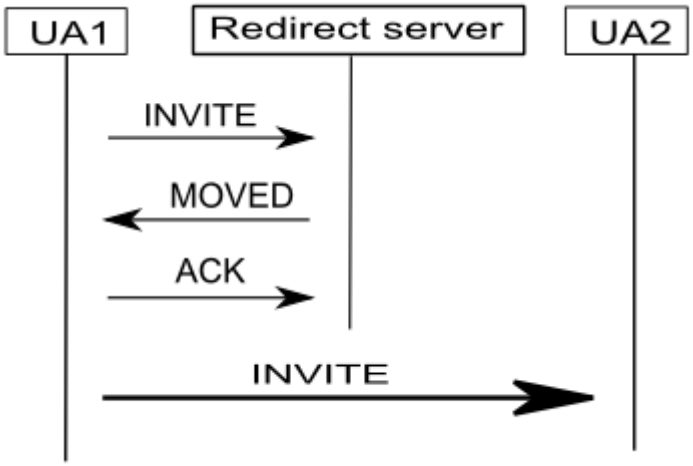
Stateful, tento typ proxy serveru při přijetí požadavku vytvoří záznam stavu a spravuje důležité informace, dokud nedojde k ukončení transakce nebo dialogu. [13,14]



Obrázek 1.1: *Proxy server*

Redirect server

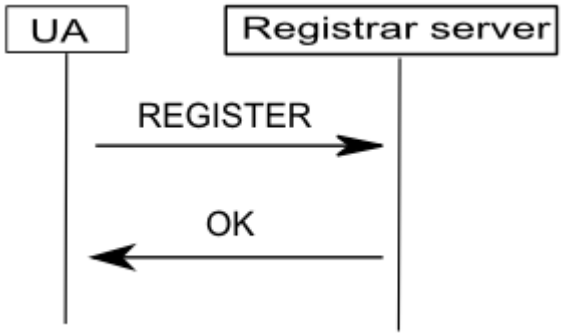
Tento server pracuje podobně jako Proxy server, tj. přijímá žádost od UA nebo jiných proxy serverů. Žádost o spojení avšak nepřeposílá dále ve směru k volanému, ale posílá tázajícímu informaci o tom, kam má žádost poslat, aby se dostala k cílovému UA. [13,14]



Obrázek 1.2: *Redirect server*

Registrar server

Tento typ serveru přijímá registrační žádosti od UA a aktualizuje podle nich databázi koncových zařízení. Díky tomu má informaci o jejich aktuální poloze (IP adresa, port a uživatelské jméno)



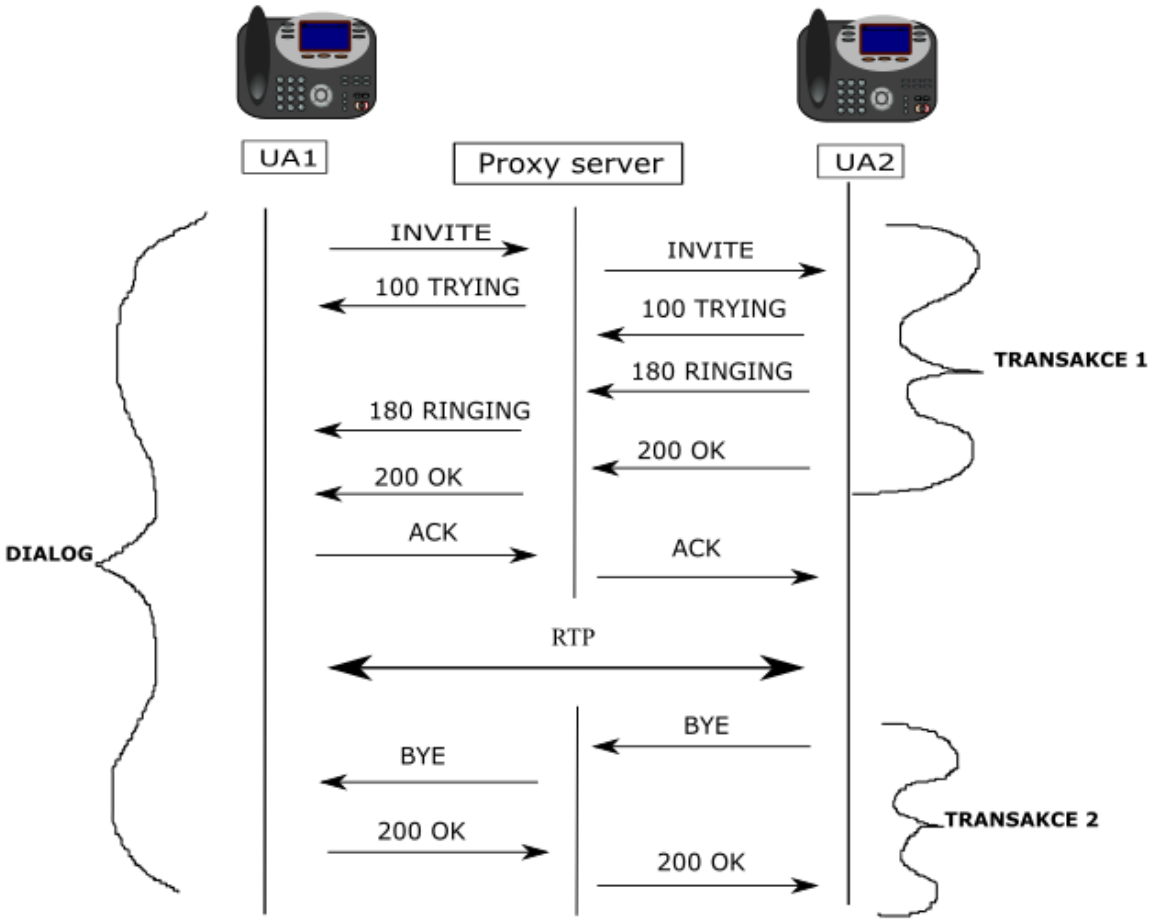
Obrázek 1.3: *Registrar server*

1.5 Signalizace k SIP proxy

Na obrázku 1.4 jsou vyobrazeny dvě transakce. Transakce je sekvence SIP zprávy, které se vyměňují mezi síťovými prvky. Každá transakce v sobě obsahuje jednu žádost a všechny odpovědi, které jsou vztaženy k dané žádosti. Může se jednat o žádnou, jednu nebo i více odpovědí. Pokud byla transakce zahájena žádosti INVITE, pak zpráva ACK je součásti transakce jen tehdy, pokud finální odpověď nebyla třídy 2XX. Pokud by na žádost INVITE přišla odpověď třídy např. 4xx, tak ACK je součástí transakce.

Na obrázku 1.4 je vyznačen jeden dialog. Dialog je soubor SIP zpráv peer-to-peer mezi dvěma koncovými body. Dialog se identifikuje pomocí CALL-ID, FROM (tag) a TO (tag). Zprávy, které mají

tyto tři identifikátory stejné, náleží jednomu dialogu. Dialogy jsou vytvořeny pomocí zpráv typu 2xx a 101-199, protože vrací značku (tag) v poli TO.



Obrázek 1.4: *Signalizace proxy server*

Spojení je navazováno pomocí procedury, která se nazývá „three-way handshake“. V okamžiku kdy chce volající, např. Alice, volat volanému, např. Bobovi, pošle žádost INVITE obsahující popis nabízeného spojení. Tahle žádost se zasílá na proxy server. Server zjistí, zda se daný uživatel nachází v jeho doméně (v SIP URI si zkontroluje položku hostname) a zda je uživatel připojen. To, že je uživatel (Bob) připojen zjistí tak, že od něj obdržel zprávu REGISTER. Pokud server zjistí, že se jedná o jinou doménu, pokusí se přeposlat požadavek INVITE na další server s danou doménou. Pokud se volající nachází v jeho doméně, tak mu přepošle zprávu INVITE. Dorazí-li zpráva v pořádku k příjemci (Bobovi), pošle Bob prozatímní odpověď RINGING, která značí, že telefon začal hrát vyzvánějící tón. Pokud se Bob rozhodne hovor přijmout, vyšle zpět odpověď OK s návratným kódem 200. Odpověď OK v sobě obsahuje informace o kodacích, adrese IP, číslech portů, na které bude Bob očekávat data od Alice. Jako třetí kokr v rámci „three-way handshake“ se posílá zpráva ACK, kterou zasílá volající strana (Alice) protistraně (Bobovi). Touto zprávou potvrzuje přijetí odpovědi od Boba. Proces navazování spojení je tímto kompletní a lze realizovat vlastní hovor (RTP). Pokud chce některý z účastníků hovor ukončit, pošle protistraně zprávu BYE.

1.6 VoIPTelefon

Jedná se o telefonní přístroj, který komunikuje prostřednictvím svého rozhraní s protokolem IP. Telefon tvoří koncového klienta ve VoIP síti. Koncoví klienti se rozdělují na softwarové a hardwarové.

1.6.1 Softwarový VoIP telefon

Jedna se o program, který se instaluje do počítače. V dnešní době lze nainstalovat VoIP softwarového klienta na velkou řadu operačních systémů. Hlavní výhodou softwarového klienta je to, že jich je většina zdarma a jsou volně šiřitelní. Nejznámější klienti jsou Yate, X-lite, Blink, SJphone.

V realizaci praktické části byl použit softwarový klient Blink a Yate. Oba tito klienti bylo nainstalování na operačním systému Ubunta.

Yate

Yate Client (Yet Another Telephony Engine) lze používat na platformách Windows, Mac OS a Linux. YateClient podporuje velké množství protokolů: XMPP, SIP, H.323, IAX, Google talk atd. Tento telefon lze stáhnout na oficiální stránce Yate.

Blink

Je komunikační klient používaný pro SIP protokol. Lze ho instalovat na platformy Linux, Windows a Mac OS. Program Blink není zatím v oficiální repositáři pro Ubuntu a je třeba ho nainstalovat dle návodu, který je uveden na jeho oficiálních stránkách.

1.6.2 Hardwarový VoIP telefon

Vzhledově se telefon nijak neliší od analogových nebo digitálních přístrojů určených pro telefonii. Telefon obsahuje numerickou klávesnici, programovatelná tlačítka, zobrazovací panel, mikrofon, atd. Programovací tlačítka často obsahují led diody, které slouží k signalizaci stavů. Tlačítka jsou nejčastěji naprogramována tak, že při jejich stisku se vytočí naprogramované číslo. Telefon pro VoIP na rozdíl od klasického telefonu, disponuje konektorem RJ45, který slouží připojení do sítě Ethernetu. Tento telefon se může napájet pomocí PoE nebo má v sobě další konektor pro připojení napájení. [15]

1.7 QoS

Jedná se o řadu technologií, která se zabývá řešením problémů okolo provozu v sítích. Cílem QoS je umožnit nastavení kvality přenosu pro data přenášena k cíli. QoS rozlišuje jednotlivé tipy přenosů a každému přenosu nastavuje jinou kvalitu. To znamená, že zajišťuje, aby se důležitý provoz doručil v pořádku v čase. QoS se stal populární díky přenosu hlasu ve VoIP a videa přes síť, kdy je pro ně třeba zjistiti určité vlastnosti /parametry přenosu.

1.7.1 Požadavky pro VoIP

Doporučení, která jsou vztažena pro VoIP technologii, aby se dosáhlo optimálního nasazení IP telefonie.

- Latence - koncové zpoždění (doba mezi vysláním paketu a jeho doručením) < 150 ms,
- Jitter - kolísání zpoždění (rozdíl v intervalech mezi přijímanými pakety) < 30 ms,
- Ztráta paketů - podíl přijatých a vyslaných paketů za čas < 1%,
- Šířka pásma - souvisí s propustností - 12 - 106 kbit/s v závislosti na vzorkování, kodeku a L2 režii.

2 Open-source platforma OpenWrt

OpenWrt je otevřený operační systém, který je v neustálém vývoji (GNU). Dle GNU licence se musí zveřejnit zdrojový kód systému, což umožnuje vytvořit deriváty daného systému. Jedná se o linuxovou distribuci, která je primárně určena pro směrovače na embedded zařízeních. Tento systém obsahuje kompletní souborový systém a manažera balíčků, pomocí kterého lze do systému doinstalovat celou řadu programů. Nechybí balíčky pro nastavení firewallu (iptables), pro vzdálený přístup k souborům (ftp, nfs,…), pro dálkovou správu systému (telnet, ssh).

2.1 Historie

Projekt OpenWrt byl zahájen v roce 2004. První verze tohoto systému byla určena pro směrovače Linksys WRT54G. Jednalo se o první stabilní verzi tohoto systému. V dnešní době existuje různé verze daného systému a lze na něm provozovat velkou spoustu aplikací.

Verze

- White Russian - tzv. Bílý Rus, pojmenovaný podle koktejlu, následně se i ostatní verze systému začali pojmenovávat podle známých koktejlů, systém vznikl roku 2007,
- Kamikaze - 2006-2010, byly vydány dvě verze tj. OpenWrt 7 (z roku 2007) a OpenWrt 8 (z roku 2008),
- BackFire - 2010-2011, první vydání bylo OpenWrt 10.03 ,
- Attitude Adjustment - byl vydán v roce 2013,
- Barrier Breaker - byl vydán v roce 2014,
- Chaos Calmer - pochází z roku 2015
- Bleading Edge - stejně jako Chaos Calmer pochází z roku 2015.

2.2 Cross Compilation OpenWrt

Křížová kompilace (cross compilation) je jednou z cest, jak vytvořit novou distribuci systému OpenWrt. Křížovou kompilací je myšlen překlad zdrojového kódu, ve formě čitelné člověkem, do binárního kódu. Na operačních systémech GNU/Linux s často používají překladače rodiny gcc (the GNU Compiler Collection). Při tvorbě distribuce se v prvním kroku definuje, pro jakou architekturu bude systém určen, velikost paměti systému a zároveň lze vybrat celou řadu knihoven a aplikací, které bude mít systém v sobě již nainstalován. Pro křížovou kompilaci je třeba mít alespoň 10Gb volné paměti v PC.

2.2.1 Postup

Nejdříve je nutné vytvořit složku

- `mkdir /home/openwrt`

Křížová kompilace se provádí v Linuxovém systému a je potřeba, aby měl v sobě nainstalováno celou řadu knihoven. Níže jsou vypsány ty nejdůležitější.

- `sudo apt-get install build-essential subversion git-core patch bzip2 flex bison autoconf gettext unzip libncurses5-dev ncurses-term zlib1g-dev gawk libz-dev libssl-dev`

Do vytvořené složky je potřeba stáhnout zdrojové kódy systému:

- `git clone git://git.openwrt.org/openwrt.git openwrtsource`

Pro následující kroky nesmí být uživatel přihlášen jako superuživatel tzv. root.

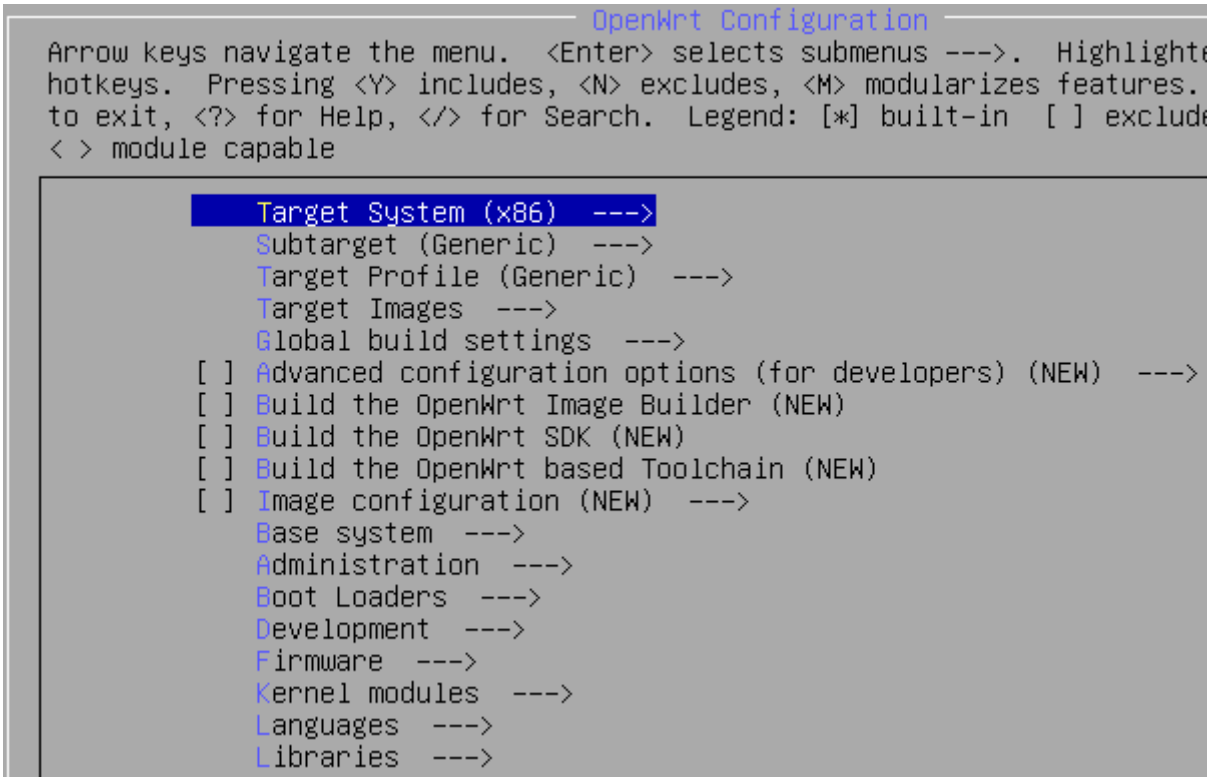
Ve složce je potřeba udělat update stažených balíčků:

- `./scripts/feeds update -a`
- `./scripts/feeds install -a`

Následně je možné provézt built OpenWrt pomocí příkazů:

- `make defconfig`
- `make prereq`
- `make menuconfig`

Pokud se vše podařilo, tak se po zadání posledního příkazu make menuconfig spustí aplikace pro vytvoření systému OpenWrt, ve které lze definovat všechny jeho části. Vše co se definuje v této grafické části se následně uloží do konfiguračního souboru s názvem ".configure".



Obrázek 2.1: *Openwrt konfigurace*

Pro spuštění samotné kompilace je třeba zadat příkaz.

- `Make`

Po zadání příkazu Make se spustí vytvoření definovaného systému.

Při tvorbě systému pro tuhle diplomovou práci byla zvolena paměť o velikosti 200 MB (minimální paměť pro implementaci všech funkcí uvedených v téhle DP je zhruba 60MB). OpenWrt má v sobě implementováno Asterisk server verze 11, XMPP server Prosody, editor nano, Fail2ban, webové rozhraní Luci, htop atd.

Po spuštění OpenWrt je třeba ověřit síťové rozhraní, které se provádí pomocí příkazu ifconfig (tento příkaz se využívá ke konfiguraci a správě síťových rozhraní v unixových operačních systémech) a zkontrolovat nastavení rozhraní eth0. Systém OpenWrt se nasazuje na embedded zařízení, kterými jsou routery. Tyto přístroje neobsahují primárně display a nejčastěji se k nim přistupuje přes ssh. Pro povolení ssh je potřeba editovat konfigurační soubor v /etc/config/firewall a povolit jeho přístup. Po spuštění je vhodné změnit přístupové heslo pro superuživatele (roota). Pro změnu se hesla se používá příkaz:

- `passwd "secure password"`

Pokud je třeba doinstalovat do systému OpenWrt další balíčky je možné použit příkazy:

- `opkg update`
- `opkg install „název balíčku“`

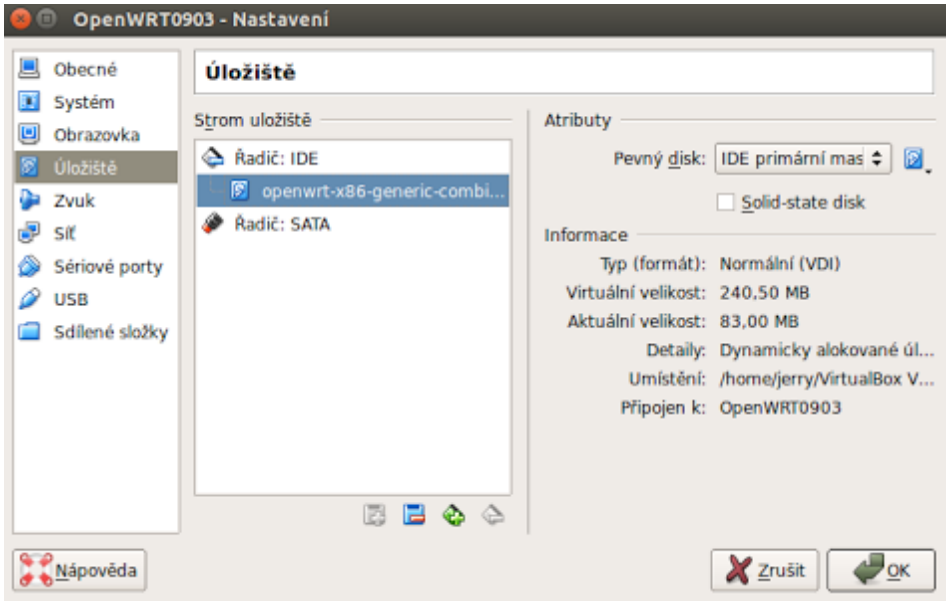
Při vypracování této kapitoly bylo čerpáno z [2, 16, 17].

2.3 Virtual Box

Nový systém z kapitoly 2.2 lze nahrát do embedded zařízení nebo ho lze vyzkoušet ve virtuálním zařízení např. VirtualBox.

VirtualBox je aplikace, která umožňuje virtualizovat operační systém na počítači. Díky této aplikaci je možné, aby v počítači bylo najednou spuštěno více operačních systému, aniž by je bylo nutné fyzicky instalovat do počítače. VirtualBox patři do rodiny opensourcových programů. Před nahráním systému do VirtualBoxu je třeba zkontrolovat v Biosu, zda je povolena virtualizace. [18]

Při nastavení virtuálního stroje je třeba dát pozor na sekci „uložiště“. V této časti je nutné zkontrolovat, zda byl přidán IDE řadič a k němu přidělen obraz systému OpenWrt.



Obrázek 2.2 *VirtualBox-uložiště*

3 Návrh a praktická realizace SIP serveru Asterisk v prostředí OpenWrt

3.1 Verze Asterisku

Program Asterisk je aktivně vyvíjen a díky tomu se pravidelně uvolňují nové verze systému. Každá verze má jen určitou dobu podpory, během které se odstraňují chyby ze systému. O tom, jak dlouho je jednotlivá verze podporována rozhoduje typ vydání. Existují dvě verze vydání. Long Term Support (LTS), která je podporována 4 roky od vydání. Během této podpory jsou vydávány opravy (patche). V rámci hlavní verze systému se uvolňují postupně další podverze. Kromě LTS se systém Asterisk vydává i ve verzi Standard, u které je plná podpora jeden rok a další rok se už jen vydávají bezpečností záplaty.[19]

Tabulka 3.1: Vývojové verze Asterisku

Release Series	Release Type	Release Date	Security Fix Only	EOL (End of Life)
1.2.X		21. 11. 2005	7. 8. 2007	21. 11. 2010
1.4.X	LTS	23. 12. 2006	21. 4. 2011	21. 4. 2012
1.6.0.X	Standard	1. 10. 2008	1. 5. 2010	1. 10. 2010
1.6.1.X	Standard	27. 4. 2009	1. 5. 2010	27. 4. 2011
1.6.2.X	Standard	18. 12. 2009	21. 4. 2011	21. 4. 2012
1.8.X	LTS	21. 10. 2010	21. 10. 2014	21. 10. 2015
10.X	Standard	15. 12. 2011	15. 12. 2012	15. 12. 2013
11.x	LTS	25. 10. 2012	25. 10. 2016	25. 10. 2017
12.x	Standard	20. 12. 2013	20. 12. 2014	20. 12. 2015
13.x	LTS	24. 10. 2014	24. 10. 2018	24. 10. 2019
14.x	Standard	2016-10 (tentative)	2017-10 (tentative)	2018-10 (tentative)
15.x	LTS	2017-10 (tentative)	2021-10 (tentative)	2022-10 (tentative)

V této práci jsem se rozhodl implementovat verzi 11. Důvodem bylo doporučení pana inženýra Šlachty, který se zabývá portováním systému Asterisk do prostředí OpenWrt. Novější Asterisk ve verzi 13 může mít problematické chování v prostředí OpenWrt.

3.2 Instalace Asterisku

Asterisk lze do prostředí OpenWrt nainstalovat pomocí několika způsobů. Lze ho implementovat do systému OpenWrt při konfiguraci systému, před spuštěním křížové kompilace. Nebo lze Asterisk stáhnout a nainstalovat pomocí balíčku z repositáře OpenWrt. V případě křížové kompilace je nejprve potřeba spustit konfiguraci systému pomocí příkazu:

- `make menuconfig`

V zobrazené konfigurační nabídce se Asterisk nachází v: Telefon->Network->Telephony-Asterisk11.

V případě, že se Asterisk instaluje pomoci repositáře OpenWrt, je třeba zadat příkazy:

- `opkg update`
- `opkg install asterisk11`

Po vybrání dané verze Asterisku lze definovat jaké moduly a kodeky bude ústředna obsahovat. Tyto moduly lze vybrat před spuštěním křížové kompilace tak, že po zatrhnutí Asterisku11 na cestě: Telefon->Network->Telephony-Asterisk11 je možné vejít do pod menu a vněm zatrhnout další moduly, kodeky, které nebyly označeny pří zatrhnutí programu Asterisk11. Moduly lze doinstalovat do Asterisku i z balíčkovacího systéme OpenWrt pomocí příkazu:

- `opkg install 'název modulu'`

V rámci této práce byl Asterisk11 doplněn v rámci křížové kompilace o následující moduly:

- Xmpp,
- Verbose,
- VoiceMail.

Příkazy pro Asterisk.

Zastavení běhu Asterisku se provádí v CLI pomocí příkazu:

- `core stop now`

Opuštění CLI Asterisku:

- `exit (+ press Enter)`

Opětovný návrat do CLI Asterisku:

- `Asterisk –r`

Spuštění Asterisku v debug módu:

- `Asterisk -cvvvvv`

3.3 Moduly

Asterisk je složen z komponent zvaných moduly. Tyto komponenty se starají o různé funkce ústředny. Přítomnost modulu v systému Asterisk lze zjistit pomocí CLI pomocí příkazu:

- `module show like "název_modulu"`

Následující moduly byly začleněny do Asterisku z důvodu jejich využití při řešení praktické části.

Voicemail modul

Daný modul umožňuje vytvořit hlasovou schránku pro jednotlivé účastníky. Pro konfiguraci hlasové schránky je třeba editovat soubor voicemail.conf, který se nachází v adresáři /etc/asterisk.

XMPP modul

Asterisk je schopen spolupracovat s XMPP protokolem a distribuovat informaci o stavu zařízení. Pro spolupráci s XMPP serverem je třeba, aby v Asterisku byl instalován res_jabber modul. Tento modul lze použít i k propojení s aplikací Hangout od společnosti Google.

Tento modul má být dostupný pro všechny verze OpenWrt, bohužel od verze 12 není k dispozici v balíčkovacím systému OpenWrt. Pokud během křížové kompilace dojde k chybě při instalování daného modulu, je možné, že při sestavování křížové kompilace byly špatně nastaveny proměnné v prostředí. Je třeba ověřit, zda je zkompilován balík iksemel.

- make package/iksemel/{clean,compile,install} V=99

Následně je potřeba znovu provést kompilaci balíku Asterisk

- make package/asterisk-11.x/{clean,compile,install} V=99

Následně pří zadání příkazu make, by měla kompilace doběhnout v pořádku až do konce a daný modul by měl být obsažen ve zvolené verzi Asterisku.

Verbose modul

Tento modul umožňuje zasílat zprávy do konsole v Asterisku. Tahle funkce pomáhá administrátorovi Asterisku sledovat události v ústředně.

Při vypracování této kapitoly bylo čerpáno z [20, 21].

3.4 Kodeky

Kodek je program, který zmenšuje objem dat. Kodeky ukládají data do zakódované formy k vůli snadnějšímu přenosu dat přes síť nebo uchování dat tak, aby zabírala méně místa.

Přehled podporovaných kodeků v Asterisku

- ADPCM, 32kbit/s,
- G.711 A-law, 64kbit/s (Evropa),
- G.711 μ-law, 64kbit/s (USA),
- G.722, 64kbit/s,
- G.726, 16/24/32/40kbit/s,
- GSM, 12-13kbit/s,
- LPC-10, 2.4kbit/s,
- Speex - (2.15-44.2 Kbps),
- G.729 – nutná licence (8Kbps),
- SILK – nutná licence (superwideband – 12 kHz),
- Siren7 – nutná licence, G.722.1 , 7 kHz,
- Siren14 – nutná licence, G.722.1 AnnexC, 14 kHz. [22]

Asterisk podporuje přenos videa pomocí kodeků:

- H.264,
- H.263 .

3.4.1.1 ***PCM***

Kódování PCM používá pulzně kódovanou modulaci. Tato modulace je složena ze tří kroků:

- vzorkování spojitého analogového signálu v rozsahu (0,3 - 3,4 kHz) pomocí vzorkovací frekvence 8 kHz,
- kvantování získaného vzorku,
- kódování.

Nejznámější představitelem daného typu kódování je kodek G.711. Jedná se o nejpoužívanější kodek PSTN sítí. Byl schválen ITU-U-T v roce 1988. Používají se dvě kódovací metody qlaw (Severní Amerika a Japonsko) a alaw (zbytek světa). Přenosová rychlost je 64kbit/s při vzorkovací frekvenci 8kHz a rozlišení 8 bitů. [23]

3.4.1.2 ***ADPCM***

Kódování ADPCM vychází z DPCM, ve kterém se nekódují navzorkovaná data, ale jejich rozdíl oproti odhadnutému průběhu signálu, který je možné částečně odhadnout. Odhadnutý průběh a navzorkovaný průběh signálu si jsou podobné, a proto má výsledný rozdíl mnohem menší dynamický rozsah a lze ho zakódovat pomocí menšího počtu bitů. ADPCM vylepšuje DPCM tak, že se přizpůsobuje konkrétní řeči, která se kóduje. Výsledkem je mnohem menší dynamický rozsah než v případě DPCM.

Nejznámější kodek s tímto typem kódováním je G.726. Jedná se o kodek, který v dnešní době nenabývá už moc používán. Tento kodek nahrazuje starší kodek G721. [23]

4 Řešení bezpečnosti v realizovaném návrhu

V dnešní době existuje nespočet robotů, kteří prohledávají internet a snaží se najít otevřený port. Jakmile je takový port nalezen, útočníci se pokusí uhodnout hesla v naději, že se dostanou do daného zařízení a k jeho citlivým údajům.

Nejčastěji jsou pod útokem servery, na kterých je třeba dělat pravidelné aktualizace, sledovat logovací soubory a v neposlední řadě mít všude v systému nastavena silná hesla.

Mezi nejznámější útoky patří DoS a jeho podtyp DDoS. DoS útok neslouží k odcizení citlivých údajů, ale k znepřístupnění určité služby na serveru. Díky tomu se daná služba stane nedostupnou pro klienty. Tento útok může mít více podob. DoS útok může být způsoben zasláním jediného speciálního paketu, který využívá zranitelnosti operačního systému nebo aplikace a zablokuje tím server. Nejčastěji má záplavový charakter. V tomto případě mohou být zdroje na severu narušeny nebo vyčerpány díky záplavě paketů. Pokud útok probíhá z jednoho místa, je útok snadno identifikován a izolován. Horší situace nastane v okamžiku, kdy je DoS útok distribuovaný (DDoS).

DDOS je určitý poddruh DoS útoku. Při tomto útoku je útok veden z velkého množství počítačů. Majitelé útočících počítačů ani nemusí vědět, že jejich počítače útočí na nějaký server. Tyto počítače jsou nejčastěji napadeny nějakým Malwarem nebo botem (programem) a nejčastěji se nazývají zombie. Díky tomu, že útok probíhá z více PC, je schopen mnohem více zahltit danou službu na serveru. [3, 24]

Jin Tang definoval, že minimální počet zpráv, které by se měly považovat za záplavový útok je 15 zpráv za sekundu, a s použitím navržené techniky lze detekovat takový útok s 88% pravděpodobností. [25]

Existují dva způsoby jak detekovat útok a anomálie v síti. Jeden způsob je založen na tom, že se proud dat porovnává s definovanými pravidly. Tento způsob detekuje vše, co je deklarováno jako neplatný provoz a zbytek považuje za platný provoz sítě. Druhý způsob je založen na detekci anomálií v síti, jedná se o porovnání právě probíhající sítové komunikace a komunikací, která byla zaznamenán v předchozí periodě. Tedy všechno co není známé, je považováno jako neplatné. [25]

V prostředí OpenWrt se lze nejčastěji setkat s ochranou implementovanou pomocí programu Snort nebo Fail2ban. V téhle práci jsou popsány oba způsoby.

4.1 Fail2ban

Fail2ban je program napsaný v skriptovacím jazyce Python. Spouští se při startu počítače a je aktivní po celou dobu běhu systému. Slouží k blokaci IP adres snažících se o neoprávněný přístup k systému. Skládá se z démona rcfail2ban a klienta fail2ban-client, který je schopen nastavovat pravidla a zobrazovat status blokovaných pokusů. Skenuje logovací soubory nejrůznějších programů např.: /var/log/asterisk/messages, /var/log/apache/error_log a logy mnoha jiných aplikací (např. sshd). V těchto souborech se snaží najít informace ohledně neúspěšných pokusech o přihlášení do systému. Pokud se takový záznam objeví, dojde k zablokování dané IP adresy pomocí nástroje např. iptables. Veškerá konfigurace se nachází v adresáři /etc/fail2ban. V tomto adresáři je umístěn soubor definující filtry filter.d, adresář s akcemi action.d a konfigurační soubory fail2ban.conf a jail.conf.

Filter.d obsahuje podmínky detekce. Pro nastavování podmínek se používají regulární výrazy. Tyto podmínky definují, co má v prohledávaných systémových logech hledat. Pokud v systémovém logu najde událost, která se shoduje s daným filtrem, spustí se další sled událostí v fail2band. [26]

Action.d je adresář, ve kterém je definováno, jaká akce se mají udělat v případě, že dojde ke shodě ve filtru, např. zakázaní IP adresy použitím iptables. [26]

Jail.conf je soubor, který se rozděluje do několika sekcí např. DEFAULT, SSH , Asterisk-iptables atd. V jednotlivých sekcích se nastavuje, jaký filtr se má použít ze složky filter.d, jaký je maximální počet pokusů o přihlášení do určité aplikace, jak dlouho bude mít dotyčný ban atd.[26]

Ukázka parametrů:

Default označuje výchozí nastavení, která budou použita, není-li pro danou chráněnou službu specifikována jiná hodnota

[DEFAULT]

- ignoreip : seznam IP adres na které nebudou pravidla platit např.:127.0.0.1 , 10.0.0.0/8,
- bantime : doba v sekundách, po kterou bude IP adresa zablokována,
- maxretry : udává počet pokusů, po které bude IP adresa zablokována.

Pod sekcí default se pak vyskytují další chráněné služby např. SSH atd., u kterých se nastavují další parametry

- enabled : nastavuje zapnutí/vypnutí služby pomoci true/false,
- port: určuje typ portu např. ssh,
- filter: definuje typ filtru např. sshd,
- logpath: určuje cestu k autorizačnímu logu např.: /var/log/auth.log.

Fail2ban se lze spustit pomocí příkazu:

- `fail2ban-client start`

4.1.1 Iptables

Pro funkci fail2ban je třeba mít nainstalovaný program Iptables. Tento program je obsažen v jádře Linuxu od verze 2.4 z roku 1999 a tak se nemusí většinou již instalovat.[27]

IPtables je rozdělen do 4 tabulek: filter, nat, mangle a raw. Po určení s jakou tabulkou se bude pracovat se používá přepínač –t, nebo –table + jméno tabulky. Pokud se nespecifikuje daná tabulka je defaultně vybrána tabulka filter.

Tabulky obsahují pravidla tzv. řetězce, které se dělí na odchozí, příchozí a přeposílanou komunikaci.

Tabulka Filter

Jedná se o základní tabulku pro filtrování, počítání, logování paketů. Tabulka obsahuje tři řetězce pravidel INPUT (pakety přicházející do systému), OUTPUT (pakety opouštějící systém), FORWARD (routované pakety).

4.1.1.1 ***Práce s Iptables***

Iptables obsahuje pravidla pro práci s jednotlivými pakety. Paket postupně putuje od jednoho pravidla k následujícímu, dokud některému z daných pravidel nevyhoví. Jakmile nějakému pravidlu paket vyhoví, provede se s ním jedna z příslušných akcí DROP/REJECT/ACCEPT. Pokud paket nevyhoví žádnému pravidlu, tak se akceptuje (ACCEPT) nebo zahodí (DROP). Pokud se jedná o uživatelem definovaný řetězec, paket je vrácen tam, odkud byl poslán (RETURN). Příkazy jsou case-senzitive. [28]

Definování nového příkazu vypadá následovně:

```
Iptables <tabulka> <příkaz> <řetězec> <specifikace_pravidla> <cíl>
```

Pokud nejsou definována žádná pravidla, tak výpis pravidel Iptables vypadá následovně. V takovém to případě Iptables propouští všechny pakety.

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Ukázka nastavení pravidel v Iptables.

```
iptables -P INPUT DROP
iptables –A INPUT  -i lo -j ACCEPT //povolení rozhraní loopback
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j
ACCEPT
iptables -A INPUT -p tcp --dport ssh -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

První příkaz určí, že pokud neexistuje nějaké pravidlo, tak se paket zahodí. Druhý příkaz povolí loopback. Další příkaz povolí průchodu paketu z navázaných spojeni. Příkaz 4 povolí ssh přístup na server. Posledním příkazem povolíme port 80, na kterém poslouchá webový server.[27, 28]

Konfigurace je dobré uložit do odboru pomocí příkazu:

```
iptables-save > /etc/iptables.rules
```

4.1.2 Fail2ban instalace

Nejdříve je nutné instalovat program python

- `Opkg update`
- `Opkg install python`

Samotný program lze stáhnout z několika zdrojů, v téhle práci uvádím dva zdroje, jedná se starší a nejnovější verzi:

- `Wget https://cloud.github.com/downloads/fail2ban/fail2ban/fail2ban_0.8.14.orig.tar.gz`
- `wget https://codeload.github.com/fail2ban/fail2ban/tar.gz/0.9.3`

Rozbalení složky:

- `tar zxvf fail2ban_0.X.X.tar.gz`

V rozbalené složce se následně zadá příkaz

- `python setup.py install`

K ověření, zda je program správně nainstalovaný lze použit příkaz:

- `fail2ban-client –h`

Pro odstranění IP, které dostala BAN lze použit příkaz:

- `fail2ban-client set "jmeno_jail_sekce" unbanip "IP_adresa"`

4.1.3 Konfigurace

Změna souboru /etc/asterisk/logger.conf

```
[general]
  dateformat = %F %T
[logfiles]
  console => notice,warning,error,debug
  messages => notice,warning,error
```

(dateformat => %F %T: tento příkaz určí formát času, který bude vypadat následovně: Year-Month-Day Hour:Minute:Second, např. [2008-10-01 13:40:04]). [29]

Do souboru /etc/fail2ban/jail.conf je zapotřebí přidat novou sekci asterisk-iptables. Ta nastavuje BAN IP adrese na 3 dny po 10 nepovedených pokusech o přihlášení. Ukázka kódu pro fail2ban verze 8.14.

```
[asterisk-iptables]
# if more than 10 attempts are made within 6 hours, ban for 3days
enabled  = true
filter   = asterisk
action   = iptables-allports[name=ASTERISK, protocol=all]
logpath  = /tmp/log/asterisk/messages
maxretry = 10
findtime = 21600
bantime = 259200
```

V případě nejnovější verze fail2ban (0.9.3) má vytvořená sekce asterisk lehce rozdílný tvar.

```
[asterisk-iptables]
enabled  = true
filter   = asterisk
port     = all #iax,sip,sip-tls
protocol = all
logpath  = /var/log/asterisk/messages
maxretry = 10
findtime = 21600
bantime = 259200
```

Systém OpenWrt neobsahuje složku /var a tak se veškeré logovací údaje ukládají do složky /tmp/log. V adresáři /tmp/log/asterisk/ se vytvoří soubor messages, do kterého se ukládají logovací údaje z CLI Asterisku.

Jednotlivé cesty, kam se ukládají logy z Asterisku lze zjistit z CLI Asterisku pomocí příkazu:

- `cli> logger show channels`

Zda fail2ban funguje lze otestovat tak, že po desátém neplatném pokusu o přihlášení k danému účtu na Asterisk server, se IP adresa, ze které probíhalo přihlašování, objeví jako blokována ve výpisu iptables. Výpis tabulky se provádí pomocí příkazu:

- `iptables -L.`

4.2 IDS /IPS systém

Detekce narušení (Intrusion Detection) je proces monitorování událostí v počítačovém systému nebo síti za účelem analyzovat, případně hledat známky narušení nastavené politiky. Tyto události mohou vzniknout od škodlivého softwaru, popřípadě útočníka, který se snaží získat přístup k softwaru.

- IDS - Intrusion Detection Systém jedná se o software, který detekuje narušení,
- IPS – Intrusion Prevention Systém - je IDS s rozšířením o možnosti ukončit činnost danému softwaru, který nepracuje jak má.

Jako implementaci IDS/IPS v systému OpenWrt se nejčastěji používá program SNORT

4.2.1 SNORT

Program Snort je open-source program, který slouží pro detekci narušení. Jedná se o jeden z nerozšířenějších IDS/IPS technologií na světě. Snort analyzuje síťovou komunikaci na základě pravidel a pokud v sítí nalezne nějakou hrozbu, tak provede příslušné akce jako např. zápis do logu. Tento program je nezávislý na platformě operačního systému a tak může být spuštěn jak na Windowsových, tak i na Linuxových, případně Unixových distribucích. Hardwarové požadavky se odvíjejí od velikosti monitorovací sítě a závisí i na množství dat protékající sítí. Tento program může být nakonfigurován v několika módech.

4.2.2 Jednotlivé režimy Snortu

Sniffer mode

Tento režim pouze čte pakety ze sítě a zobrazuje je uživateli na obrazovce. Pro spuštění Snortu v tomto režimu lze použít příkaz:

- `./snort -vd`

Paket logger mode

Tento režim je téměř stejný jako Sniffer mód, rozdíl je, že zachycené paketové data nebo hlavičky zapisuje (loguje) na disk, ukázka příkazu:

- `./snort -dev -l ./log`

Pokud je třeba ukládat data tak, aby se dali později analyzovat například programem tcpdump, stačí zadat ./snort -l ./log b. Pokud je potřeba posílat alerty na syslog, přidá se do příkazu -s.

Network Intrusion Detection System (NIDS)

Jedná se o nejzajímavější mód Snortu, který je zároveň nejnáročnější pro hardware. Tento režim zaznamenává jen tu aktivitu na síti, kterou vyhodnotí jako nebezpečnou. V tomto režimu má Snort dostupných 7 alert módů: full, fast, socket, syslog, console, cmg a none.

Tento režim se spouští pomocí příkazu:

```
snort -c /usr/local/snort/etc/snort.conf -l /var/log/snort
```

Parametr -c udává místo uložení konfiguračního souboru Snortu, parametr -l udává adresu místa, kam se zapisují události detekované Snortem.

Inline

V tomto módu se chová jako IPS a získává pakety z IPtables pomocí knihovny Libpcap a WinPcap, které pak následně vyhodnocuje pomocí definovaných pravidel.

4.2.3 Instalace

Nainstalovat Snort do OpenWrt lze několika způsoby. U prvního způsobu je možno Snort implementovat do OpenWRT při jeho konfiguraci před křížovou kompilací, ve druhém případě ho lze nainstalovat pomocí balíčků z repositáře Snort.

Snort se nachází v kompilačním balíčku OpenWrt na cestě:

- `Network->Firewall->snort`

Pro spuštění Snortu s jeho počáteční politikou pravidel lze využít příkaz:

```
Snort -c /etc/snort/snort.conf -l /tmp/log/snort --daq-dir
/usr/local/lib/daq
```

Při spuštění v NIDS režimu, zabere Snort skoro 200Mb paměti RAM. Je třeba editovat jeho konfigurační soubor a nechat v něm preprocesor určený pro SIP komunikace. V takovém případě Snort při svém startu zabere pouze 40 Mb paměti RAM.

4.3 Pravidla

Snort využívá jednoduchý popisující jazyk pro tvorbu pravidel. Při psaní víceřádkových pravidel je potřeba na konci řádku zadat zpětné lomítko. Pravidla se dělí do dvou logických oddílů, jedná s o hlavičku (header) a volbu (options). Hlavička obsahuje typ akce, protokol, zdrojovou a cílovou adresu a porty. Ve volbách jsou pak obsaženy výstražné části zprávy informující, které části paketu by měly být zkontrolovány.

Obecný tvar pravidla

```
action protokol zdroj_ip zdroj_port směr cíl_ip cil_port (options)
```

U příchozích paketů jsou porovnávány jejich zdrojové i cílové IP adresy a porty s pravidly v seznamu pravidel. Pokud nějaké pravidlo odpoví kontrolovanému paketu, pak se vyhodnocuje volba (options).

Seznam akcí pro pravidla:

- alert - generuje výstrahu použitím vybrané výstražné metody a pak zaznamená paket,
- log - zaznamenává (loguje) paket,
- pass - ignoruje paket,
- activate - generuje výstrahu a následně zavolá jiné dynamické pravidlo pro testování paketu,
- dynamic - tato akce je vyvolána jen dalšími pravidly při použití akce "activate",
- drop - blokuje a vytvoří záznam o paket,
- reject - blokuje paket, loguje ho a pak pošle TCP reset nebo ICMP zprávu o nedostupnosti portu, jestliže byl protokol UDP,
- sdrop - blokuje paket, ale neloguje ho.

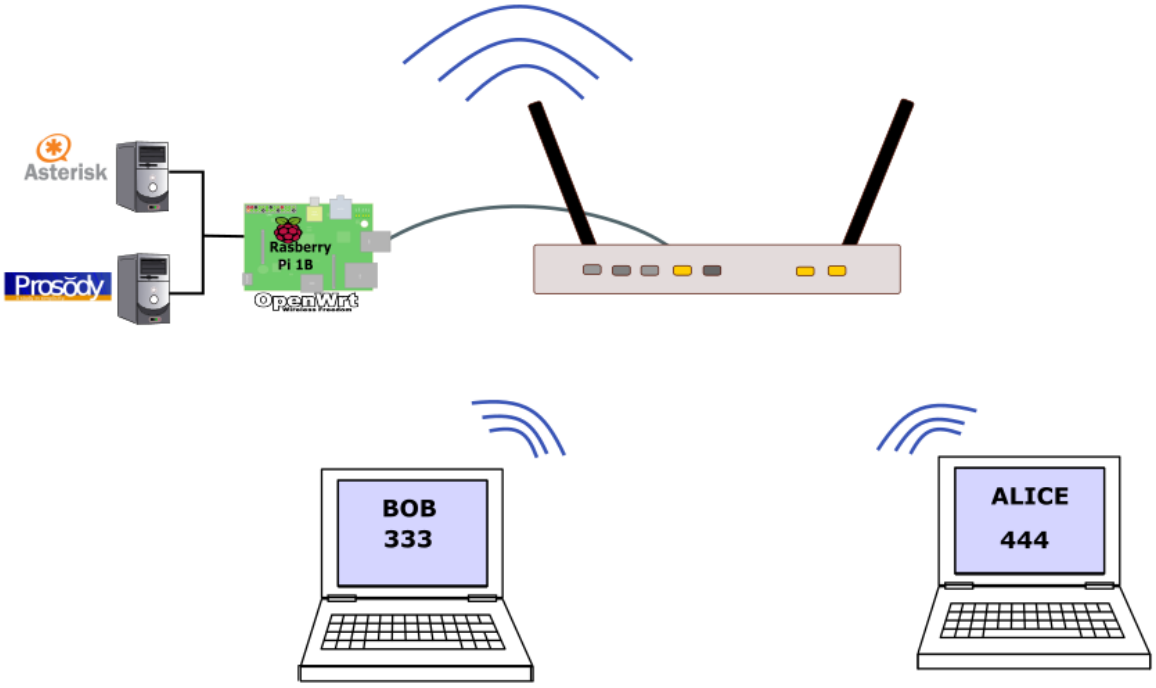
Ukázka pravidla:

```
alert ip any any -> any any (msg:"COMMUNITY SIP INVITE message
flooding"; content:"INVITE"; depth:6; threshold: type both, track
by_src, count 10, seconds 60; classtype:attempted-dos;
sid:100000158; rev:2;)
```

V této kapitole bylo čerpáno z [3, 30].

5 Implementace pokročilých funkcí v prostředí Asterisk

V rámci ukázky pokročilých funkcí v prostředí Asterisk byl zvolen Presence a IM. Pro jejich implementaci bylo zapotřebí nainstalovat do systému OpenWrt XMPP server. Celý systém byl umístěný na Raspberry Pi.



Obrázek 5.1: *Schéma zapojení*

5.1 Prosody

Jedná se o moderní komunikační XMPP server napsaný v jazyce Lua. Celá konfigurace se nachází v jednom souboru s názvem prosody.cfg.lua. Konfigurace souboru se dá rozdělit na dvě části. První část konfiguračního souboru obsahuje globální sekci a je výchozí pro všechny virtuální hostitele. V druhé části se definuje virtual host (doména, v rámci které se budou vytvářet účty pro uživatele) a komponenty (např. modul pro konferenci). Uživatelé jsou identifikováni pomocí JID (JabberID). Tvar JID je velice podobný e-mailové adrese.

Pomocí nástroje prosodyctl je možné přidat, odebrat nebo změnit heslo uživatele a to vše lze dělat za běhu systému.

- přidání uživatele: `prosodyctl adduser` uzivatel@domena.cz
- odebrání uživatele: `prosodyctl deluser` uzivatel@domena.cz
- změna hesla uživatele: `prosodyctl passwd` uzivatel@domena.cz

5.1.1 SSL/TLS

Prosody umožňuje šifrovat komunikaci pomocí certifikátu. Specifikace certifikátu se přidává k příslušnému virtual hostu a to sekce ssl.

Pro generování certifikátu lze použít dva příkazy:

Prosody verze 0,9 +

- `Prosodyctl cert generate example.com`

Prosody verze 0,8 -

- `openssl req -new -x509 -days 365 -nodes -out "example.com.crt" -newkey rsa:2048 -keyout "example.com.key"`

V téhle kapitole bylo čerpáno z [31, 32].

5.1.2 XMPP protokol

XMPP (eXtensible Messaging and Presence Protokol) je protokol, který byl v dřívější době známý spíše pod názvem Jabber. Jedná se o komunikační protokol standardizován Internet Engineering Task Force (IETF). Hlavním cílem protokolu je Instant Messaging (IM) a sdělování informací napříč sítí téměř v reálném čase. Při spojení s Asteriskem se používá k nastavení volání (signalizace). Protokol XMPP umožňuje zasílat zprávu v případě, když uživateli někdo volá, posílat zprávy zpět na ústřednu Asterisku nebo přesměrovat hovor do hlasové schránky nebo na jiné místo. [33]

Aby se dal propojit XMPP server s Asteriskem je třeba zkompilovat res_jabber modul. Tento modul obsahuje sadu funkcí, které jsou užitečné pro Asterisk.

5.1.3 Nastavení Prosody

V druhá části konfiguračního souboru prosody.cfg.lua, byl vytvořen Virtuální host s názvem domény "prosodyxmpp2". Pro danou doménu byl vytvořen certifikát, který umožňuje zabezpečenou komunikaci. Pro vytvoření certifikátu byl použit příkaz:

- `openssl req -new -x509 -days 365 -nodes -out "prosodyxmpp2.com.crt" -newkey rsa:2048 -keyout "prosodyxmpp2.com.key".`

Byly vytvořeny tři účty, jeden účet slouží pro propojení s Asterisk serverem a zbylé dva jsou určeny pro uživatele Boba a Alici. Příklad přidání uživatele:

- `prosodyctl adduser bob@prosodyxmpp2.com.`

Prosody server se spouští pomocí příkazu

- `/etc/init.d/prosody start`

5.1.4 Implementace Fail2ban

Prosody server podporuje Fail2ban verzi 0.9 a vyšší. V případě nižší verze Fail2ban, není tento program schopen v logovacím souboru Prosody serveru detekovat nepodařené pokusy o přihlášení. Aby byl Fail2ban verze 0.9+ schopen detekovat anomálie v logovacím souboru Prosody je třeba do Prosody serveru doplnit modul: mod_log_auth.lua. Tento modul se dá stáhnout z:

- `wget http://prosody-modules.googlecode.com/hg/mod_log_auth/mod_log_auth.lua`

Modul je třeba uložit do složky /usr/lib/prosody/modules. Po jeho stažení je třeba přidat daný název modulu do konfiguračního souboru Prosody (prosody.cfg.lua) a následně restartovat server.

Ve Fail2ban lze pak následně v souboru jail.conf vytvořit následující sekci:

```
[prosody]

enabled = true

port    = 5222

filter  = prosody

logpath = /var/log/prosody/prosody.log

maxretry = 10
```

Na závěr je potřeba vytvořit filtr s regulérním výrazem, který bude aplikován na logovacím souboru Prosody serveru.

```
[Definition]

failregex = Failed authentication attempt \(not-authorized\) from
IP: <HOST>

ignoreregex =
```

5.2 Nastavení Asterisk serveru

Pro nastavení Asterisku je třeba editovat několik konfiguračních souborů.

5.2.1 asterisk.conf

V tomto konfiguračním souboru se definují důležité parametry pro program Asterisk. Jsou zde uvedeny cesty k nejrůznějším adresářům, ke kterým přistupuje program např. umístění souboru, do kterého se ukládají jednotlivé logy, atd.

5.2.2 sip.conf

V daném souboru jsou definováni jednotliví uživatelé. Konfigurační soubor je rozdělen do několika částí. Na začátku je sekce [general]. V této části se definuje nastavení, které je platné pro všechny uživatele. Nejčastěji se zde nastavují kodeky, nat, číslo portu atd. Sekci [general] můžeme ponechat i prázdnou. Pod touhle sekcí se pak definují jednotliví uživatelé.

```
[general]             ;název sekce
allowtransfer=yes
bindport=5060         ;číslo portu na kterém naslouchá
realm=asterisk
bindaddr=0.0.0.0      ;IP adresa na které naslouchá  (0.0.0.0 značí
všechna síťová rozhraní )
disallow=all          ;zakázaní všech audio kodeků
allow=alaw            ;povolení kodeku s logaritmickou kompresí A-law


[444]                  ;nastavení klienta 444
type=friend            ;typ připojení
context=xmpp           ;odkaz na číslovací plán
secret=444             ;přihlašovací heslo
host=dynamic           ;povolení registrace účtu na všech IP adresách
callerid="alice"<444>  ;identifikátor volajícího
mailbox=444@VoiceMail  ;číslo hlasové schránky
```

5.2.3 **extension.conf**

V tomto konfiguračním souboru se definuje diaplán (číslovací plán). Diaplán je srdce systému Asterisk a definuje se v něm chování pro příchozí a odchozí volání. Diaplán je složen ze čtyř základních konceptů: context, extensions, priorities a applications. Jeho tvar je následující:

```
exten=>name,priority,application
```

Diaplán je rozdělen na sekce zvané contexts. Jedná se o základní organizační jednotku v rámci plánu volby. Jednotlivé contexty jsou od sebe izolovány. K vytvořenému contextu jsou přidruženy všechny příkazy, které jsou napsány pod jeho názvem.

Exten - odkazuje na číselný identifikátor.

Priorita - určuje pořadí v rámci vykonávání daných příkazu, nejvyšší priorita má číslo s hodnotou jedna. Pokud je vytočen daný extension provede se nejdříve příkaz s nejnižší hodnotou (nejvyšší prioritou).

Application - každá aplikace provádí svoji specifickou akci, např. přehraje tón, spojí hovor, zavěsí hovor atd.

5.3 Implementace funkce present v Asterisku za pomocí Prosody

Pro zjištění přítomnosti uživatele se využívá funkce JABBER_STATUS. V případě, že je uživatel available, vrátí se číslo s hodnotou 1, stav away vrací hodnotu 2, stav busy vrací 3 atd. V této ukázce se první tři stavy chápou jako stav, že je uživatel online, a zahájí se telefonní spojení. Vyšší hodnota značí, že volaný účastník není k dispozici. V tomto případě může uživatel zanechat hlasovou zprávu volajícímu, poslat mu zprávu na Hangout nebo ukončit sestavování hovoru. Uživatel se o jednotlivých možnostech dozví pomocí hlasového průvodce a zaslané zprávy.

Zvuková stopa pro hlasového průvodce byla vytvořena promocí programu WavePad Sound Editor. Stopa byla vytvořena ve formátu gsm a pojmenována jako IVR-DP. Vytvořený soubor byl nakopírován pomocí příkazu scp do složky /usr/lib/asterisk/sounds.

V následující podkapitole je rozepsán část diaplánu určeného pro Alici a Boba. Tento diaplán je umístěn v contextu s názvem xmpp.

```
exten => 444,1,Set(Status_Alice=${JABBER_STATUS(asterisk-
prosody,alice@prosodyxmpp2.com/Yate)});
```

Na začátku se zjistí, jaký je stav Alice na jejím XMPP klientu Yate. Do proměnné Status_Alice se v závislosti dle jejího stavu uloží číselná hodnota.

```
exten => 444,2,GotoIf($[${Status_Alice} < 4 ]?10:20)
```

Pokud se vrátí hodnota 1, 2 nebo 3, tak se provede příkaz s prioritou 10, pokud se vrátí číslo s hodnotou 4 a větší, tak se provede příkaz s prioritou 20.

```
exten => 444,10,JabberSend(asterisk-
prosody,alice@prosodyxmpp2.com,Prichazi hovor od:
${CALLERID(name)});
```

Alice je na svém XMPP klientu přítomna, a tak jí na její XMPP klient dojde informační zpráva. Zpráva je zaslaná pomocí funkce JabberSend. Obsahem zprávy je: Přichází hovor od: ${CALLERID(name)}. Funkce ${CALLERID(name)} vrací jméno volajícího, které bylo nastaveno v souboru sip.conf u nastavení klienta.

```
exten => 444,11,Dial(SIP/${EXTEN},30);bude vyzvanet 30s
```

Aplikace Dial () v číslovacím plánu sestaví telefonní spojení. Za proměnou ${EXTEN} se vloží volané číslo. Hodnota 30 značí, že telefon bude vyzvánět 30s, pokud příjemce daný hovor nevezme, tak se hovor zavěsí.

```
exten => 444,20,jabbersend(asterisk-
prosody,${CALLERID(name)}@prosodyxmpp2.com,  Volany ucastnik
${EXTEN} neni dostupny. Pro zanechani hlasove zpravy (voicemail)
stisknete 1, pro zaslani zpravy na Hangout stisknete 2, pro ukonceni
volani  stisknete 3.)
```

Pokud nebyl účastník Alice na svém XMPP klientu online, tak volajícímu účastníku přijde na jeho XMPP klient zpráva, která ho informuje, že může zanechat hlasovou zprávu, poslat zprávu na aplikaci Hangout nebo zavěsit hovor. Pokud chce volající (Bob) zanechat hlasovou zprávu, musí zadat

číslo 1 na svém XMPP klientu, pokud chce poslat informační zprávu na Hangout, musí zadat číslo 2, případně, pokud chce zavěsit hovor, musí zadat hodnotu 3. Pro získání dané hodnoty se využívá funkce JABBER_RECEIVE a hodnota se ukládá do proměnné Bob_volba.

```
same => n, Set(Bob_volba=${JABBER_RECEIVE(asterisk-
prosody,bob@prosodyxmpp2.com,30)})
```

Pokud si volající nic nevybere, tak se hovor sám ukončí za 30 s. Pro uložení volby volajícího je použita funkce JABBER_RECEIVE. Výsledek této funkce se uloží do proměnné Bob_volba.

```
exten => 444,21,Background(IVR-DP)
```

Uživatel se o jednotlivých možnostech dozví kromě zprávy i pomocí hlasového průvodce. Hlasový průvodce se přehraje pomocí funkce Background.

```
same =>n,GotoIf($["${Bob_volba}" = "1" ]?voicemailA,1);voicemail
```

Pokud volající účastník zadá hodnotu 1 na XMPP klientu, provede se extension s názvem voicemailA.

```
same =>n,GotoIf($["${Bob_volba}" ="2"] ?HangoutA,1)  ;hangout
```

Pokud volající účastník zadá hodnotu 2 na XMPP klientu, provede se extension s názvem HangoutA.

```
same =>n,GotoIf($["${Bob_volba}" ="3" ] ?konec,1)  ;konec
```

Pokud volající účastník zadá hodnotu 3 na XMPP klientu, provede se extension s názvem konec.

```
same => n, Goto(konec,50)
```

Pokud účastník nic nevybere během 30s, provede se extension konec.

```
exten => voicemailA,1,JabberSend(asterisk-
prosody,alice@prosodyxmpp2.com, Mate zanechanou hlasovou zpravu od:
${CALLERID(name)});
```

```
same => n, VoiceMail(444@VoiceMail)
```

Na XMPP klienta volaného (v našem případě Alice) dojde informační zpráva, že má zanechanou hlasovou zprávu od volajícího. V dalším kroku se spustí průvodce pro zanechání hlasové zprávy.

```
exten => 2000,1, VoiceMailMain(${CALLERID(num)}@VoiceMail)
```

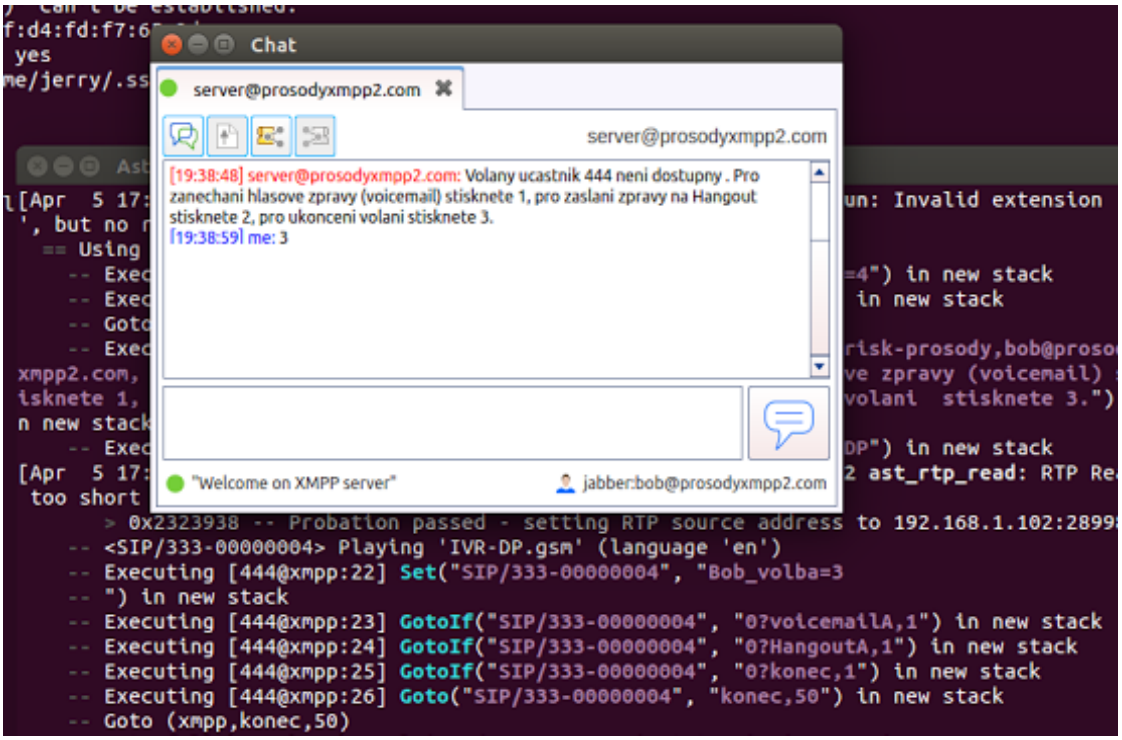
Každý uživatel si může svoji hlasovou zprávu vyzvednout po vytočení čísla 2000 a zadání hesla.

```
exten =>HangoutA,1,JABBERSend(alice-google,Bob333Hangout@gmail.com,
Mate zanechanou hlasovou zpravu od:  ${CALLERID(name)} )
```

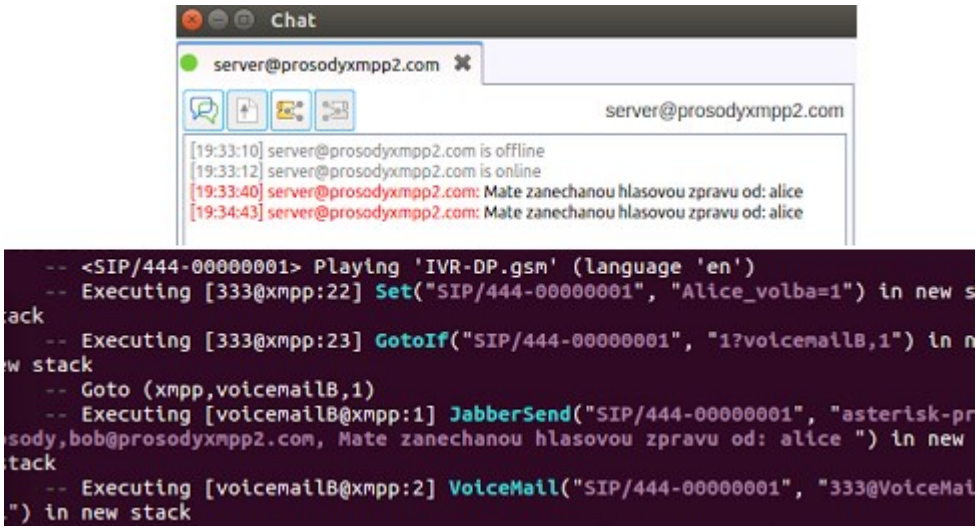
Tato část diaplánu umožňuje zaslat na Hangout požadovanou zprávu. Spojení na emailový účet Alice je definovaný v souboru xmpp.conf s kontextem alice-google.

```
exten => konec,1,Hangup()
```

Jedná se o diaplán s názvem konec. Hangup aplikace zruší aktivní spojení a uzavře kanál.



Obrázek 5.2 *Zpráva informující, že je uživatel nedostupný*



Obrázek 5.3 *Zpráva informující o zanechání hlasové zprávy*

5.3.1 XMPP.conf

Konfigurační soubor xmpp.conf slouží k propojení Asterisk serveru a XMPP serveru.

```
[asterisk-prosody]                    ;název spojení
type=client                           ;Typ Client nebo Component
```

```
serverhost=192.168.1.8                ;název serveru
uusername=server@prosodyxmpp2.com     ;JID
secret=server                         ;heslo pro pripojeni
priority=1                            ;priorita
port=5222                             ;port Prosody serveru 5222
usetls=no                             ;použití tls
usesasl=yes                           ;použití sasl
buddy=alice@prosodyxmpp2.com          ;výchozí uživatelé
buddy=bob@prosodyxmpp2.com
status=available                      ;status může být: chat,
available, away, …
statusmessage="Welcome on XMPP server" ;popis statusu
timeout=100                          ;;Timeout on the message stack.
```

V tomto souboru je možné propojit vytvořit spojení na účet Google a využívat tak Hangout chat.

```
[alice-google]
type=client
serverhost=talk.google.com
username=Bob333Hangout@gmail.com
secret=heslo_na_email
port=5222
usetls=yes
usesasl=yes
statusmessage="I am available"
timeout=100
```

5.3.2 Voicemail.conf

Tento konfigurační soubor se používá k nastavení hlasové schránky pro jednotlivé uživatele.

```
[VoiceMail]
333 => 333, Bob, bob@prosodyxmpp2.com      ;hlasová schránka pro
uživatele Boba
444 => 444, Alice, alice@prosody@xmpp2.com ;hlasová schránka pro
uživatele Alici
```

5.4 IM

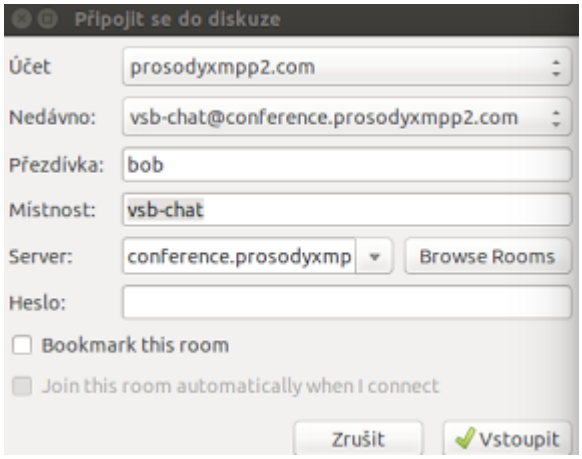
Server Prosody lze využít k vytvoření skupinového chatu, tzv. IM. Aby uživatel mohl vytvořit vlastní chatovací místnost na serveru Prosody, musí se vložit následující řádky do konfiguračního souboru prosody.cfg.lua.

```
Component "conference.prosodyxmpp2.com" "muc"
            restrict_room_creation = false
```

První řádek definuje název konferenční místnosti, proměnná restrict_room_creation nastavená na false umožnuje uživatelům si vytvářet vlastní chatovací místnost.

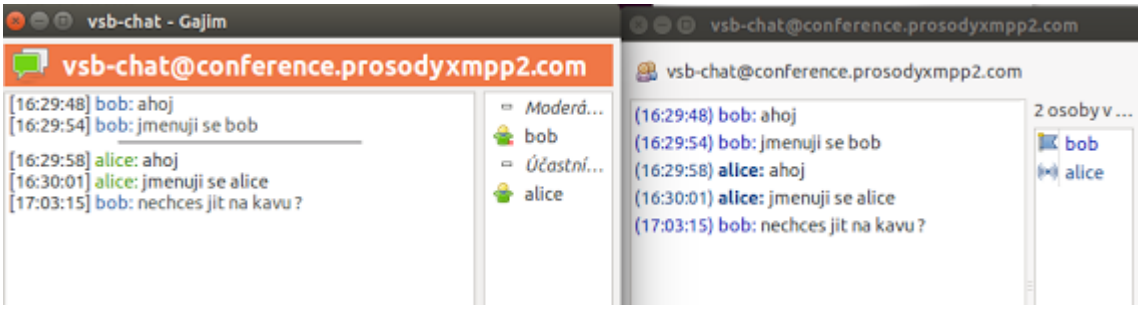
Pro vytvoření skupinového chatu v programu Pidgin se vybere v horní panelové nabídce možnost Kamaradi->Pridat chat. Objeví se nové okno, které je už z části předvyplněné, stačí jen doplnit název místnost např. VSB-chat a případně i heslo a následně kliknout na tlačítko přidat.

Na xmpp klientu Gadzim se lze k vytvořenému chatu připojit přes jeho nabídku: Vstoupit do diskuze->připojit se do diskuze. Objeví se přihlašovací formulář, ve kterém se zadá název chatu, ke kterému se chce uživatel přihlásit a klikne se na tlačítko vstoupit.



Obrázek 5.4 *Vytvoření IM*

Na obrázku 5.5 je zobrazena IM komunikace mezi uživatelem Alicí a Bobem.



Obrázek 5.5 *Komunikace v IM*

6 Stanovení výkonnostních limitů a zhodnocení dosažených výsledků

6.1 SIPP

Jedná se volně šiřitelný program z rodiny Open Source, který je určen především pro operační systém Linux a používá se jako testovací nástroj protokolu SIP. Program SIPp je schopen generovat a odesílat více hovorů na základě předem definovaných scénářů. Tyto scénáře jsou ve formátu xml. XML soubor se skládá z jednotlivých značek tzv. tagů. Tyto tagy mají logické pojmenování např.: tag recv je pro zprávy, které mají přijít, tag send slouží pro zprávy, které se mají odeslat. Jednotlivé část scénáře jsou v xml souboru obsaženy v poli CDATA. Mezi oba tagy se pak může psát a definovat spousta atributů, díky kterým lze popsat jakoukoliv situace, která může u SIP nastat. [34]

Velkou výhodou SIPp je schopnost vytvářet statistiky daného hovoru. Tyhle statistiky obsahují např.: délku hovoru, počet opakování odeslání nějaké zprávy atd.

6.1.1 Kompilace programu:

Program se stahuje jako komprimovaný balíček, který se pak rozbalí do uživatelem zvoleného adresáře. Samotná kompilace se dělá pomocí příkazu make. K příkazu make se u starších verzí programu SIPp daly přidat parametry. Těmito parametry mohly být pcapplay (díky ní bude SIPp podporovat media), ossl (SIPp podporuje TLS). U novější verze program SIPP se u k příkazu make parametry nezadávají a např.: pcaaplay a ossl se konfigurují předem pomocí příkazu "./configure –-with-openssl –with pcap".

Po kompilaci je možné spouštět program jen z dané složky. Je vhodné uložit tuto informaci o příkazu SIPp i do systému. Informaci lze přidat pomocí odkazu na binární soubor SIPp, který se při kompilaci vytvořil. Vytvoření odkazu lze provézt pomocí příkazu :

```
# ln –s /home/jerry/jerry/sip/sipp /usr/bin/sipp
```

6.1.2 Popis XML scénáře

Nástroj SIPp generuje SIP provoz v závislosti na použitém scénáři. Tvůrci programu nabízí řadu předdefinovaných scénářů, ty ale nedokážou vystihnout všechny možnosti, které program SIPp nabízí.

Ve scénářích lze generovat zprávy typu INVITE, ACK, BEY atd. Scénáře lze vytvořit jak z pohledu klienta, tak i z pohledu serveru. Díky tomu lze vytvořit komunikaci, která simuluje reálný systém.

Každý scénář musí začínat:

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

První řádek informuje, že se jedná o XML soubor a obsahuje informace, o jaké kódování se jedná. Druhým řádek definuje jméno a začátek scénáře. Jedná se dvou párový tag, z toho důvodu musí bát na konci ukončení tagu </scenario>.

Jak jíž bylo zmíněno výše, existují další tagy, které se vkládají mezi tag scenário např. send, reciave . Do těchto dvou tagů lze implementovat spoustu atributů, díky kterým lze popsat jakoukoliv situace, která může u SIPp nastat. V této práci uveden jen jejich výčet, více informací se dá nelézt na oficiálních stránkách program SIPp. [34]

- remote_ip : obsahuje IP adresu serveru (zadává se v příkazové řádce)
- trasport : obsahuje informaci o protokolu transportní služby
- local_ip : parametr je zadán pomocí příkazové řádky (přepínač „i“)a definuje IP adresu PC, na kterém je program Sipp spuštěn,
- local_port -port na kterém je SIPP spuštění
- respons : popisuje zprávu, která má být přijata,

SIPp se spouští v konzoli pomocí příkazu. Příkazu lze přidat několik parametrů:

- sf – odkaz na soubor se scénářem,
- inf – odkaz na soubor s informacemi o hovoru,
- m – maximum hovorů, kterého chceme dosáhnout,
- r – množství hovorů generovaných za jednu sekundu (nebo dobu uvedenou a
- přepínačem -rp),
- bg – SIPp se spustí na pozadí.
- l -maximální počet souběžných simultálních hovorů
- t - použije se transportní protokol
- timeout - čas po kterém se SIPp ukončí

6.2 StarTrinity

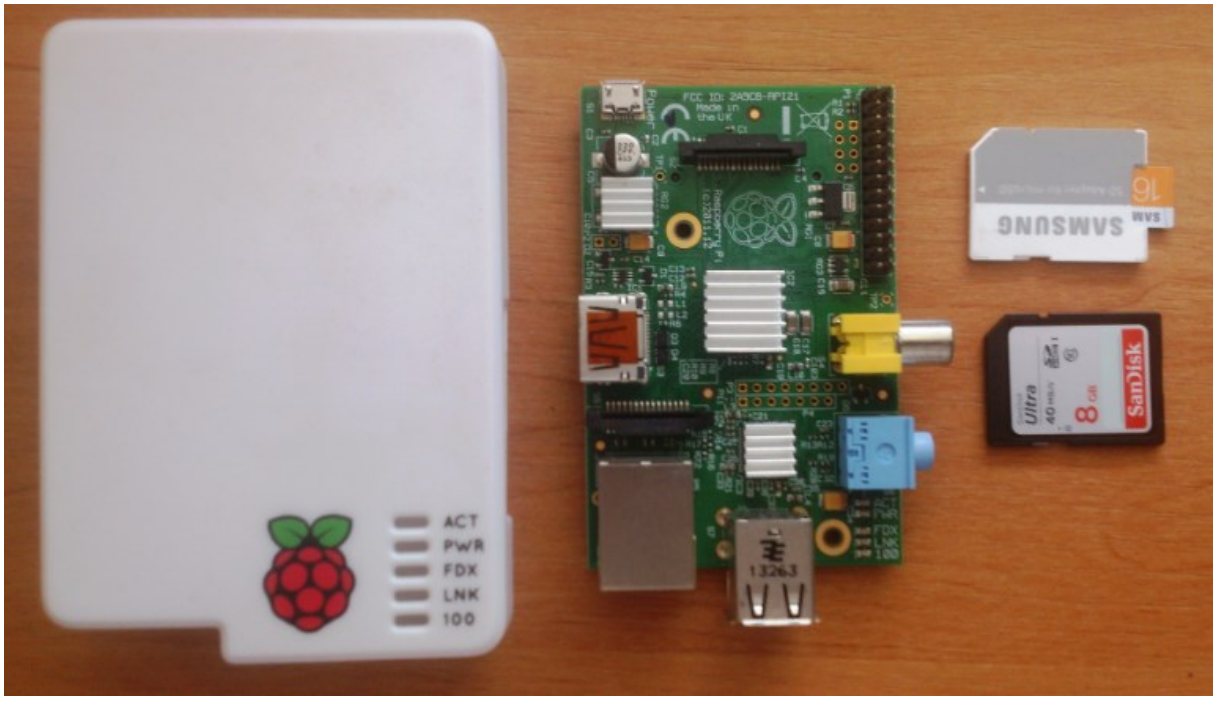
Jedná se o VoIP testovací nástroj, který umožňuje monitorovat VoIP síť, SIP software nebo hardware. Je schopen simulovat a pasivně monitorovat tisíce příchozích a odchozích simulačních hovorů s RTP obsahem. Tento nástroj dokáže analyzovat a posoudit kvalitu hovorů. Pro vyzkoušení nástroje je k dispozici tzv. freeware licence, která umožňuje současný běh maximálně 50 hovorů a celkem je možné odeslat 150 příchozích + odchozích hovorů. Po uskutečnění 150 hovorů je program zablokován a pro pokračování v testování je ho třeba restartovat. [35]

6.3 Raspberry Pi

Raspeberry Pi je miniaturní počítač o velikosti platební karty. Tento počítač je založen na SoC od společnosti Broadcom s platformě od ARM. Počítač nejčastěji obsahuje HDMI výstup, síťové port, USB porty. Tento počítač je napájený pomocí micro USB (5V). Díky jeho velikosti a nízké spotřebě není vybaven pevným diskem. Jako uložiště dat se používá paměťová karta. Nejnovější typ Rasberry Pi 3 je vybaven čtyř jádrovým procesorem s podporou 64bit instrukcí, 1GB operační paměti, wifi a bluetooth.

Pro testovací účely byl zvolen Raspberry Pi 1 Model B 512MB. Jedná se o jeden z prvních modelů Rasberry Pi. Parametry tohoto počítače jsou:

- Model : 1B
- SoC : BCM2835 (CPU, GPU, DSP, SDRAM, USB port)
- CPU : ARM1176JZF-S, 700MHz , jednojádrový,
- RAM : 512 MB
- USB: 2 x USB verze 2
- Ethernet : 10/100M
- WiFi : -
- Storage : SD Card
- Video : HDMI + RCA
- Audio : 3.5 TRS
- Power: 5V



Obrázek 6.1 *Testované Raspberry Pi 1B*

6.3.1 **Instalace OpenWrt na Rasberry Pi model 1B**

Pro zvolený model Rasberry lze stáhnout systém OpenWRT přímo na jejich oficiálních stránkách. Pro tento model je určena distribuce openwrt-brcm2708-bcm2708-sdcard-vfat-ext4.im. Tuto distribuci si lze vytvořit i pomocí křížové kompilace. Při spuštění kompilace pomocí příkazu make menuconfig se zvolí v sekci Target, platforma Broadcom BCM2708 /BCM2708. Po provedení kompilace se v adresáři bin, vytvoří adresář brcm2708. V tomto adresáři bude uložen soubor s názvem openwrt-brcm2708-bcm2708-sdcard-vfat-ext4.

Stažený nebo vytvořený soubor openwrt-brcm2708-bcm2708-sdcard-vfat-ext4 je potřeba zapsat na paměťové médium. Jako paměťové médium byla zvolena paměťová karta od společnosti SandDsk. K zápisu dat lze využít několik nástrojů, které jsou zdarma ke stažení na internetu.

Pod operačním systémem Windows lze k zápisu souboru na paměťovou kartu použít nástroj Win32DiskImager . Obsluha tohoto nástroje je jednoduchá, pouze se vybere místo, kde je uložen daný souboru a zařízení, do kterého se má soubor zapsat.

Pod operačním systémem Linux lze pro nahrání systému na paměťovou kartu použít následující příkazy:

- `dmesg`

Tento příkaz vypíše hlášení jádra operačního systému. Daný příkaz pomůže určit jméno fyzického oddílu, pod kterým je přístupná paměťová karta. Nejčastější název je sda,sdb,sdc atd. Následně se použije příkaz:

- `dd if=/home/username/Downloads/openwrt-brcm2708-bcm2708-sdcard-vfat-ext4.img of=/dev/sdX bs=2M conv=fsync`

První parametr tohoto příkazu určuje, odkud se budou data kopírovat (nejčastěji HDD), druhý parametr udává místo kam se budou data ukládat (z tohoto důvodu bylo potřeba zadat i příkaz dmesg)

Způsobů jak zapsat OpenWrt na paměťovou kartu je mnohem více, v této práci jsou ukázány jen tyto dva způsoby.

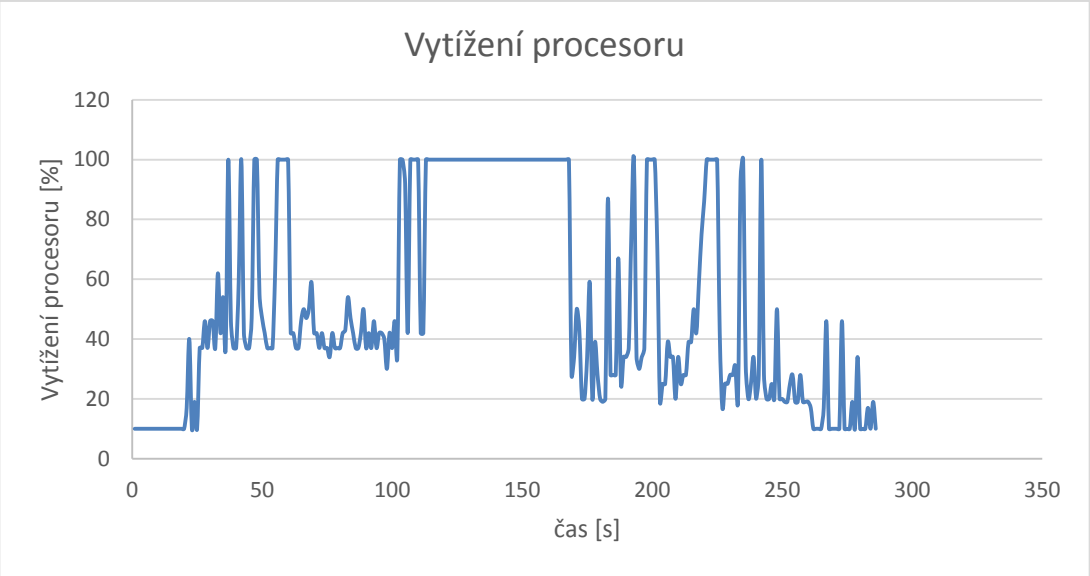
Opačným příkazem je možné zálohovat systém, který je uložen na SD kartě a díky tomu uživatel nepřijde o dlouhé hodiny strávené kompilováním a instalováním programů.

- `sudo dd if=/dev/disk1 of=~/SDCardBackup.dmg`

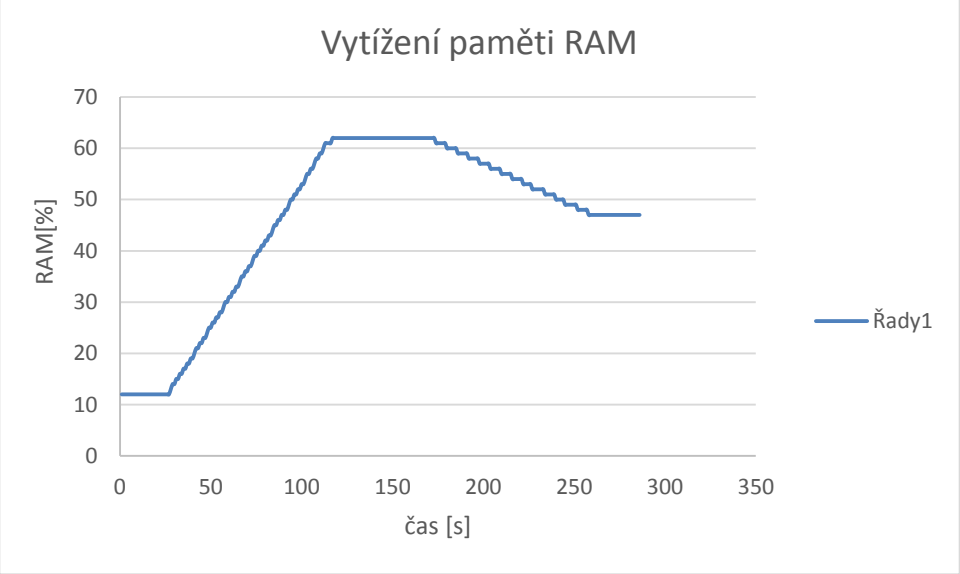
6.4 Zátěžové testy

6.4.1 Hovory bez RTP

Při daném testu bylo pomocí testovacího nástroje SIPp generováno každou sekundu 10 nových hovorů bez zaslání RTP. Každý hovor byl ukončen po 180 sekundách. Celkem proběhlo 3106 hovorů během 490s. Maximální počet souběžných hovorů bylo 1800. Pro tento byl použit účet s názvem sipp. Tento účet měl i se stejným jménem pojmenovaný context. Zatížení CPU a RAM si lze prohlédnout na obrázku 6.2 a 6.3.



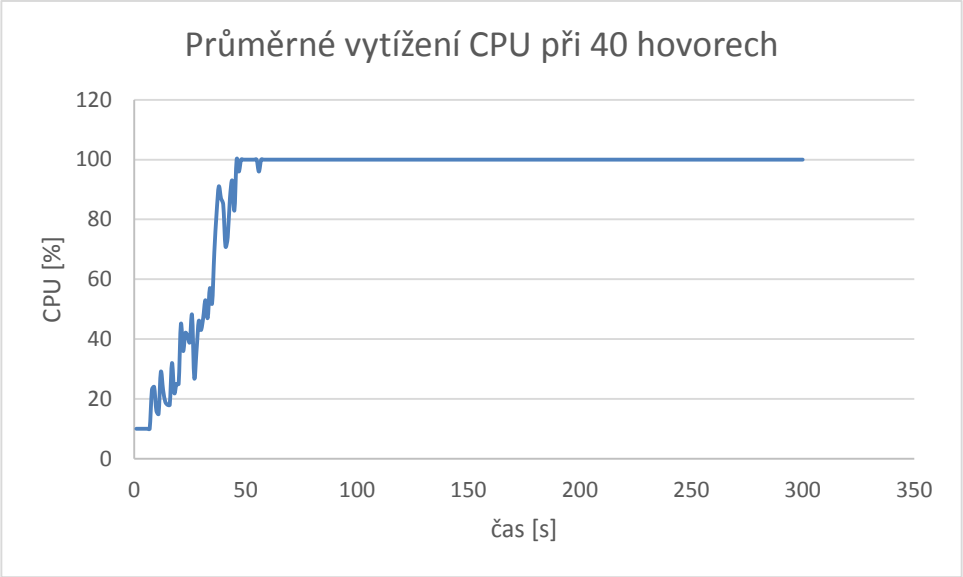
Obrázek 6.2 *SIPp - hovory bez RTP_ vytížení CPU*



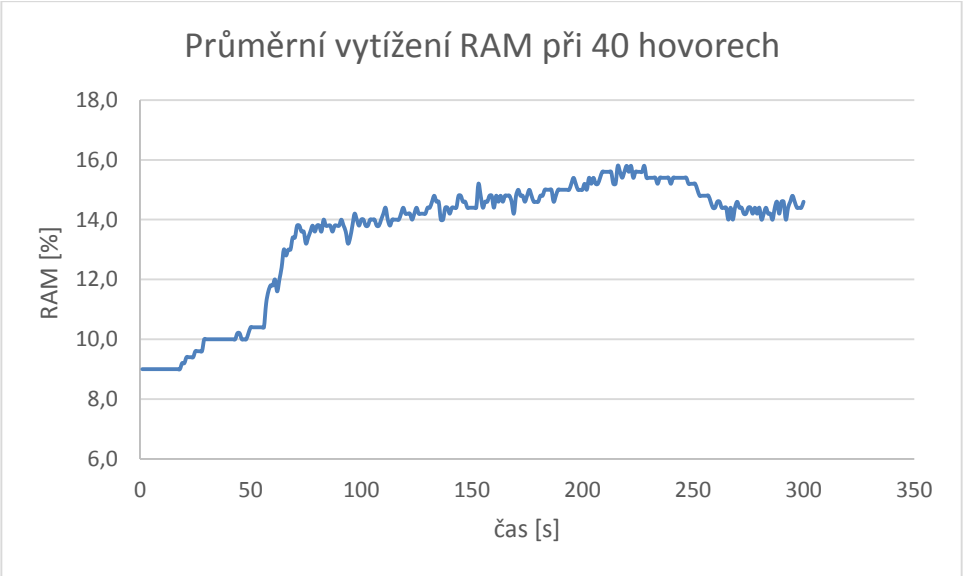
Obrázek 6.3 *SIPp - hovory bez RTP_ vytížení CPU*

6.4.2 **Hovory s RTP**

Při testování pomocí programu SIPp verze 3.4 bylo dosaženo maximálně 45 souběžných hovorů. Při vyšším počtu došlo k zaplnění paměti RAM na 98% a následnému pádu serveru Asterisk. Na obrázku 6.4 a 6.5 je zobrazeno vytížení CPU a RAM jen pro 40 souběžných hovorů. Je to z toho důvodu, že na OpenWrt běžel skript, který sbíral hodnoty ztížení CPU a RAM. V případě spuštění skriptu a programu SIPp simulujícího 45 hovorů došlo zaplnění paměti RAM a pádu Asterisk serveru. Pro vytvoření daných obrázků byla požita zprůměrována data z 5 simulačních hovorů. Pro zachycení síťové komunikace byl použit program Wireshark.

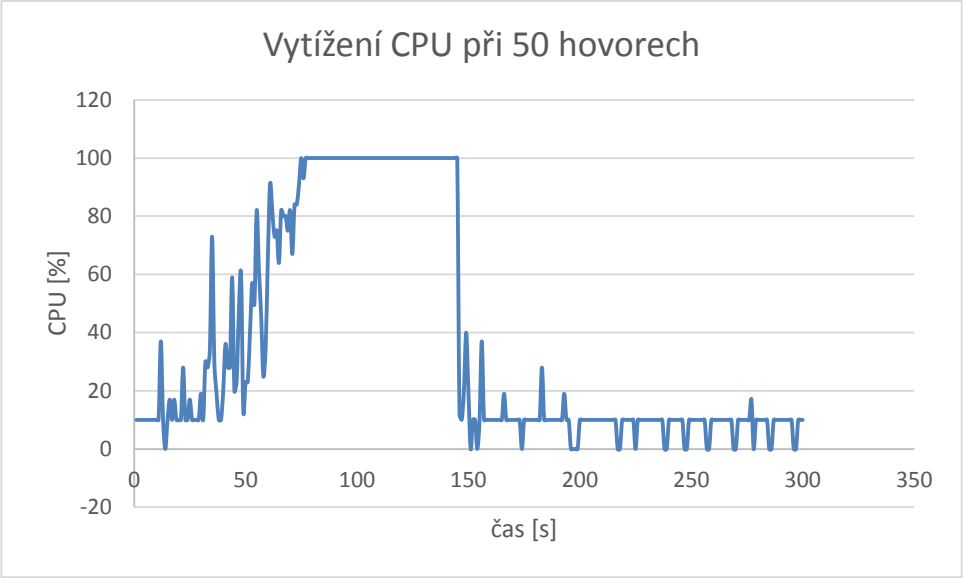


Obrázek 6.4 *CPU při simulace 40 hovorů*

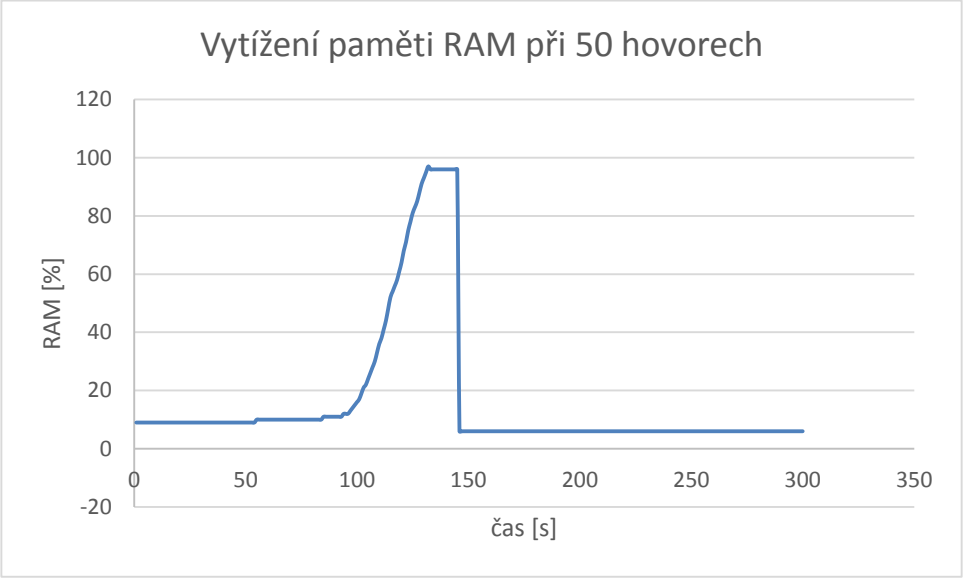


Obrázek 6.5 *RAM při simulace 40 hovorů*

Na obrázcích 6.6 a 6.7 je ukázána simulace pro 50 souběžných hovorů, při kterých došlo k pádu Asterisk serveru.

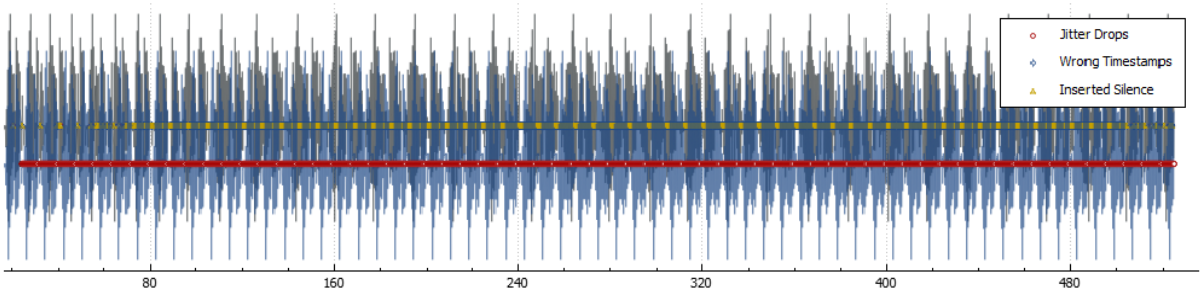


Obrázek 6.6 *CPU při simulace 50 hovorů*



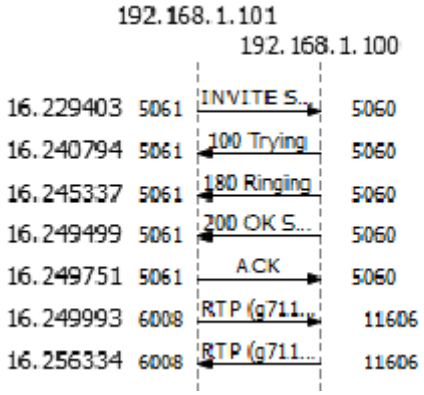
Obrázek 6.7 *RAM při simulace 50 hovorů*

Na obrázku 6.8 je grafické zobrazení RTP toku mezi Raspberry Pi (UAS) a jedním simulačním hovorem generovaného z SIPp.



Obrázek 6.8 *RTP tok*

Na obrázku 6.9 je zobrazena zasílání zpráv mezi Raspberry Pi (UAS) a jedním simulačním hovorem generovaného z SIPp.



Obrázek 6.9 *zprávy mezi UAS a UAC*

7 Zhodnocení

Na OpenWrt byl implementován SIP server Asterisk ve verzi 11. Tento program byl implementován do prostředí OpenWrt při jeho křížové kompilaci. Nejobtížnější byla implementace XMPP modulu, kterého bylo zapotřebí pro propojení s Prosody serverem a aplikací Hangout od Google. Systém OpenWrt je velice stabilní a má malé hardwarové nároky. Z tohoto důvodu byl nasazen na Raspberry Pi 1B.

Pro zabezpečení systému OpenWrt byl použit program Fail2ban. Tento program projevil velice nízké nároky na paměť RAM. Fail2ban byl použit pro zabezpečení Asterisk serveru a Prosody serveru. Fail2ban po svém nainstalování v sobě obsahoval soubor asterisk.conf, ve kterém byly obsaženy regulérní výrazy pro prohledávání logovacího souboru Asterisk. Pro Prosody server bylo zapotřebí vytvořit soubor prosody.conf a v něm definovat regulérní vyraz detekujíc pokus o přihlášení. Bylo i zapotřebí doplnit do Prosody server modul, který obohatil logovací soubor o informaci tykající se IP adresy případného účastníka. Při reálném testování program Fail2ban pokaždé zablokoval útočníkovu IP adresu po 10 nezdařených pokusech o přihlášení.

V rámci řešení pokročilých funkcí byl na OpenWrt instalován a nakonfigurován Asterisk server a Prosody server. Prosody server byl propojen s Asterisk serverem. Díky tomu mohl Asterisk získat informaci o přítomnosti jednotlivých účastníku v síti. Pokud byl příjemce nedostupný, volající byl vyzván jak hlasovým průvodcem, tak i informační zprávou, aby stiskl jedničku pro zanechání hlasové zprávy, dvojku pro zaslání zprávy na Hangout nebo trojku pro ukončení hovoru. Pokud volající nevybral ani jednu možnost, tak byl jeho hovor po určité době ukončen. Při implementaci IM byl využit server Prosody, který umožnil vytvořit chatovací místnost pro jednotlivé účastníky.

Po implementaci jednotlivých částí byla provedena série zátěžových testů pro stanovení výkonosti daného HW. Tyto testy byly provedeny pomocí programu SIPp. Dle testů bylo dokázáno, že Raspberry Pi 1B zvládne maximálně 45 souběžných hovorů. Na obrázcích 6.2 a 6.3 je zobrazeno vytížení CPU a RAM jen pro 40 účastníků. Je to z toho důvodu, že na Raspberry Pi běžel monitorovací skript. Tento skript zatěžoval CPU a v případě 45 souběžných hovorů došlo k zaplnění paměti RAM na 98% a následnému pádu serveru Asterisk.

Raspebrry 1B disponuje poměrně malým výkonem a v případě jeho nedostačujícího výkonu ho lze vyměnit za model Raspberry Pi 2, který v sobě ukrývá čtyř-jádrový 900MHz procesor a paměť RAM o velikosti 1Gb. Pro verzi 1 a 2 je k dispozici systém OpenWrt, který je neustále aktivně vyvíjen a umožňuje tak na Raspberry Pi implementovat řadu zajímavých aplikací. Tento malý počítač lze uplatnit především v oblasti Small Office/Home Office a Small Enterprise komunikacích (v oblastech do 50 účastníků).

Celkové náklady po vytvoření této ústředny jsou:

- Rasberry Pi B+ : 780 Kč
- Paměťová karta (microSD, class10, 8Gb): 200Kč

V případě použití vyšší verze Raspberry Pi 2 je cena o 200 Kč vyšší.

8 Závěr

V první kapitole teoretické části se zabývám popisem telefonní ústředny Asterisk a pro jaké aplikace je tahle telefonní ústředna určena. V kapitole jsou popsány nejrůznější protokoly, které se využívají ve VoIP. Konec této kapitoly je především zaměřen na SIP protokol. U tohoto protokolu jsou popsány jednotlivé typy zprávy a odpovědí a také jak probíhá navazování komunikace mezi jednotlivými účastníky.

Druhá kapitola je věnována opensourcové platformě OpenWrt. V úvodu této kapitoly je popsán systém OpenWrt a k čemu se dá tento systém využít. V další části této kapitoly je popis vytvoření vlastního systém pomocí křížové kompilace. Při kompilaci byl vytvořen nejnovější systém OpenWRT a to ve verze 15 s názvem Chaos Calmer. Tenhle systém měl nastavenou vnitřní paměť na 200MB a byl v něm implementován Asterisk11, Prosody, python, htop a mnoho dalších aplikací a knihoven.

Třetí kapitola je učena pro praktickou realizaci SIP serveru. Pro realizaci SIP serveru byl vybrán Asterisk verze 11, který má plnou podporu od vývojářů do roku 2017. Asterisk byl doplněn o XMPP modul v rámci propojení se serverem Prosody a aplikací Hangout od Google. Dalším důležitým modulem, který byl naistalován do Asterisku, byl Voicemail modul, díky kterému lze uživatelům zanechat hlasovou zprávu.

Co se týče zabezpečení systému, tak byl zvolen bezpečnostní systém Fail2ban. Tento systém má menší nároky na paměť RAM než program Snort. Ve Fail2ban byly nastavena pravidla pro zabezpečení serverů Asterisk a Prosody.

Pro implementaci pokročilých funkcí byl na OpenWRT nakonfigurován systém Prosody. Prosody server umožňuje systému Asterisk získat informace o přítomnosti účastníka v síti. Pokud byl příjemce nedostupný, volající byl vyzván jak hlasovým průvodcem, tak i informační zprávou, která mu byla zaslána na jeho XMPP klienta, aby stiskl jedničku pro zanechání hlasové zprávy, dvojku pro zaslání zprávy na Hnagout nebo trojku pro ukončení hovoru. Pokud volající nevybral ani jednu možnost, tak byl jeho hovor po určité době ukončen. Pokud uživatel stiskl jedničku, byl vyzván k zanechání hlasové zprávy. Příjemci byla zaslána na XMPP klienta informační zpráva, že mu byla zanechána hlasová zpráva a že si ji má vyzvednout při zavolání na číslo 2000. V další části této kapitoly je popsán způsob, jak vytvořit chatovací místnost.

Celý systém byl nainstalován na Raspberry Pi verze 1B. Implementací systému OpenWrt na tento hardware se zabývám v 6 kapitole. Tohle kapitola je věnována i zátěžovým testům. Zátěžové testy byly provedeny pomocí programu SIPp. Bylo provedeno několik testů. Při testování hovorů s RTP streamem bylo pomocí programu SIPp dosaženo maximálně 45 hovorů (bez zaznamenávání hodnot procesoru a paměti RAM). Při zaznamenávání vytížení procesoru a paměti RAM bylo docíleno maximálně 40 hovorů. Při vyšším počtu došlo k pádu serveru Asterisk vlivem plné paměti RAM. Pokud by dosavadní výkon hardwaru nestačil, je možné použít Rasberry Pi 2.0. Tento malý počítač lze uplatnit především v oblasti Small Office/Home Office a Small Enterprise komunikacích (v oblastech do 50 účastníků).

.

Použitá literatura

[1] M. Vozňák.[i]Voice over IP.[/i] VŠB-TU Ostrava, vysokoškolská skripta, druhé vydání, 2009.

[2] L. Surhone, M. Tennoe.[i]OpenWrt.[/i] Betascript Publishing, 2011, ISBN 978-6135271591.

[3] F. Rezac, M. Voznak, J. Safarik, P. Partila, K. Tomala. Security solution against denial of service attacks in BESIP system. In [i]Proc. SPIE 8755, Mobile Multimedia/Image Processing, Security, and Applications[/i]., Baltimore, 2013. DOI: 10.1117/12.2015423.

[4] ČERVENKA. Asterisk PBX [IpTelWiki]. In: CESNET [online]. 2010 [cit. 2016-04-01]. Dostupné z: https://sip.cesnet.cz/cs/swahw/asterisk

[5] DAVENPORT, Malcolm. License Information: Asterisk Project. In: Asterisk Project Wiki [online]. 2014 [cit. 2016-04-01]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/License+Information

[6] Protokol UDP (User Datagram Protocol). TechNeT: Resources and Tools for IT Professionals [online]. [cit. 2016-04-01]. Dostupné z: https://technet.microsoft.com/cs-cz/library/cc785220(v=ws.10).aspx

[7] PRAVDA, Ivan. Internetová telefonie (VoIP) a protokol SIP [online]. Praha [cit. 2016-04-01]. Dostupné z: http://data.cedupoint.cz/oppa_e-learning/2_KME/079.pdf. České vysoké učení technické v Praze.

[8] IAX IAX2 architecture: Inter-Asterisk eXchange protocol. Voip Think [online]. [cit. 2016-04-01]. Dostupné z: http://www.en.voipforo.com/IAX/IAX-architecture.php

[9] SIP: [IpTelWiki]. In: CESNET [online]. 2012 [cit. 2016-04-01]. Dostupné z: SIP: [IpTelWiki]. In: CESNET [online]. 2012 [cit. 2016-04-01]. Dostupné z: https://sip.cesnet.cz/cs/protokoly/sip

[10] SOUMAR, Michal. Signalizační protokol pro přenos hlasu přes datové sítě - SIP. In: Elektrorevue [online]. [cit. 2016-04-01]. Dostupné z: http://www.elektrorevue.cz/clanky/03003/index.html

[11] SIP URI. Voip-info.org [online]. [cit. 2016-04-01]. Dostupné z: http://www.voip-info.org/wiki/view/SIP+URI

[12] Úvod do VoIP. MB DATA [online]. [cit. 2016-04-01]. Dostupné z: http://www.mbdata.cz/uvoddovoip.htm

[13] Úvod do VoIP. MB DATA [online]. [cit. 2016-04-01]. Dostupné z: http://www.mbdata.cz/uvoddovoip.htm

[14] SOUMAR, Michal. Signalizační protokol pro přenos hlasu přes datové sítě - SIP. In: Elektrorevue [online]. [cit. 2016-04-01]. Dostupné z: http://www.elektrorevue.cz/clanky/03003/index.html

[15] Stolní IP telefony. JOYCE ČR [online]. 2016 [cit. 2016-04-01]. Dostupné z: http://www.joyce.cz/voip-produkty-stolni-ip/

[16] LOCHOVSKY, Milan. OpenWrt. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2015. Dostupné také z: https://cs.wikipedia.org/wiki/OpenWrt

[17] OpenWrt Version History. OpenWrt Wiki [online]. 2016 [cit. 2016-04-01]. Dostupné z: https://wiki.openwrt.org/about/history

[18] POLÁK, Michal. Začínáme s VirtualBoxem: Nastavení virtuálního počítače. In: AbcLinuxu [online]. Zvoleněves: Argonit, 2012 [cit. 2016-04-06]. Dostupné z: http://www.abclinuxu.cz/clanky/zaciname-s-virtualboxem-nastaveni-virtualniho-pocitace

[19] BRYANT, Russell. Asterisk Versions. In: Asterisk Project: Asterisk Project Wiki [online]. 2015 [cit. 2016-04-01]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Asterisk+Versions

[20] BRYANT, Russell. Asterisk Versions. In: Asterisk Project: Asterisk Project Wiki [online]. 2015 [cit. 2016-04-01]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Asterisk+Versions

[21] Asterisk: the definitive guide. Fourth edition. Sebastopol: O'Reilly, 2013. ISBN 9781449332426

[22] Asterisk codecs allow: Available values. In: VOIP Wiki [online]. California: Voip-info-LV, 2010 [cit. 2016-04-06]. Dostupné z: http://www.voip-info.org/wiki/view/Asterisk+codecs+allow

[23] VoIP, Kodeky přehled - 5.díl. Telefonní ústředny [online]. 2016 [cit. 2016-04-06]. Dostupné z: http://www.jktelekomunikace.evron.cz/cz/page/28363/voip-kodeky-prehled-5dil.html?detail=34822

[24] WEBER, Filip. DoS a DDoS útoky a ochrana proti nim (1). In: Svět sítí [online]. 2008 [cit. 2016-04-07]. Dostupné z: http://www.svetsiti.cz/clanek.asp?cid=DoS-a-DDoS-utoky-a-ochrana-proti-nim-1-742008

[25] Multimedia communications, services and security: 7th International Conference, MCSS 2014, Krakow, Poland, June 11-12, 2014. Proceedings [online]. 1st edition. New York: Springer, 2014 [cit. 2016-04-07]. ISBN 33-190-7568-3. Dostupné z: https://books.google.cz/books?id=vSS7BQAAQBAJ&pg=PA185&lpg=PA185&dq=snort+sip+rules+alert+ip+any+any&source=bl&ots=0KzBjFu_Pa&sig=0TZqjvcFmvYPKh_OuZcRSxdoovc&hl=cs&sa=X&ved=0ahUKEwi-yoGwp47KAhWomXIKHQzVAmQQ6AEILTAB#v=onepage&q=snort%20sip%20rules%20alert%20ip%20any%20any&f=false

[26] REMENEC, Juraj. Fail2Ban – zlyhaj a sedíš!. In: ABC Linuxu [online]. 2014 [cit. 2016-04-07]. Dostupné z: http://www.abclinuxu.cz/clanky/fail2ban-zlyhaj-a-sedis

[27] Iptables. In: Wiki Ubuntu [online]. 2015 [cit. 2016-04-07]. Dostupné z: http://wiki.ubuntu.cz/bezpe%C4%8Dnost/firewall/iptables

[28] DOČEKAL, Michal. Správa linuxového serveru: Linuxový firewall, základy iptables. LinuxEXPRES [online]. Brno: CCB [cit. 2016-04-07]. ISSN 1801-3996. Dostupné z: http://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-linuxovy-firewall-zaklady-iptables

[29] Logger.conf. MADSEN, Leif, Jim Van MEGGELEN a Russell BRYAN. Asterisk: The Definitive Guide [online]. 3. California: O'Reilly Media, 2011, s. 846 [cit. 2016-04-07]. Dostupné z: http://asteriskdocs.org/en/3rd_Edition/asterisk-book-html-chunk/Monitoring_id264504.html

[30] Writing Snort Rules. SNORT Users Manual 2.9.8.2: The Snort Project [online]. 2015 [cit. 2016-04-07]. Dostupné z: http://manual.snort.org/node27.html

[31] Prosody IM: Jabber/XMPP server [online]. 2016 [cit. 2016-04-07]. Dostupné z: https://prosody.im/

[32] DOČEKAL, Michal. Správa linuxového serveru: XMPP (Jabber) server Prosody. LinuxEXPRES [online]. Brno: CCB [cit. 2016-04-07]. ISSN 1801-3996. Dostupné z: http://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-xmpp-jabber-server-prosody

[33] Using XMPP (Jabber) with Asterisk. MADSEN, Leif, Jim Van MEGGELEN a Russell BRYAN. Asterisk: The Definitive Guide [online]. 3. California: O'Reilly Media, 2011, s. 846 [cit. 2016-04-06]. Dostupné z: http://www.asteriskdocs.org/en/3rd_Edition/asterisk-book-html-chunk/ExternalServices_id286789.html

[34] SIPp. SIPp [online]. 2014 [cit. 2016-04-08]. Dostupné z: http://sipp.sourceforge.net/doc/reference.html

[35] Startrinity [online]. 2016 [cit. 2016-04-08]. Dostupné z: http://startrinity.com/

Seznam příloh

Součásti DP je DVD.

Adresářová struktura přiloženého DVD:

- Obrázky - obsahuje vytvořené obrázky ve formátu svg
- Raspberry Pi - obsahuje konfigurační soubory pro jednotlivé servery, obraz paměťové karty, a systém OpenWRT pro Raspberry Pi 1B z křížové kompilace
- SIPp - zdrojové kódy k programu SIPp včetně scénáře
- Wireshark - obsahuje zachycené hovory

Příloha A: *Monitorovací skript*

```
#!/bin/bash
datum=`date +%Y-%m-%d`
vCPU_idle=" "
vCPU=" "
vRAM=" "
space="\n"
#while true;do
COUNTER=0

while [  $COUNTER -lt 300 ]; do
        echo CPU: `top -b -n1 | grep "CPU: " | grep "usr"| awk
'{print $2 + $4}'`
        vCPU=$vCPU$((`top -b -n1 | grep "CPU: " | grep "usr"| awk
'{print $2 + $4}'` ))$space


        echo RAM: `free -m |grep Mem | awk ' {printf
("%.0f\n",$3/$2*100)}'`
        vRAM=$vRAM$((`free -m |grep Mem | awk ' {printf
("%.0f\n",$3/$2*100)}'` ))$space


        echo CPU-idle: `top -b -n1 | grep "CPU: " | grep "idle"| awk
'{print 100 - $8}'`
        vCPU_idle=$vCPU_idle$((`top -b -n1 | grep "CPU: " | grep
"idle"| awk '{print 100 - $8}'` ))$space


  sleep 1
  let COUNTER=COUNTER+1
done


echo "zapis do souboru..."
echo -e $vCPU >> vCPU$1-$datum.txt
echo -e $vCPU_idle >> vCPU-idle$1-$datum.txt
echo -e $vRAM >> vRAM$1-$datum.txt
```

Příloha B: *XML scénář pro generování hovorů*

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">
<!--  Use with CSV file struct like: 101;192.168.1.4;[authentication
username=101 password=1234];
      (user part of uri, server address, auth tag in each line)-->
<scenario name="register_client">
 <send retrans="500">
   <![CDATA[
      REGISTER sip:[remote_ip] SIP/2.0
      Via: SIP/2.0/[transport]
[local_ip]:[local_port];branch=[branch]
      From: <sip:[field0]@[field1]>;tag=[call_number]
      To: <sip:[field0]@[field1]>
      Call-ID: [call_id]
      CSeq: [cseq] REGISTER
      Contact: sip:[field0]@[local_ip]:[local_port]
      Max-Forwards: 100
      Expires: 300
      User-Agent: SIPp/Win32
      Content-Length: 0
    ]]>
  </send>
  <recv response="100" optional="true">
  </recv>
  <recv response="401" auth="true">
  </recv>
  <send retrans="500">
    <![CDATA[

      REGISTER sip:[remote_ip] SIP/2.0
```

```
      Via: SIP/2.0/[transport]
[local_ip]:[local_port];branch=[branch]
      From: <sip:[field0]@[field1]>;tag=[call_number]
      To: <sip:[field0]@[field1]>
      Call-ID: [call_id]
      CSeq: [cseq] REGISTER
      Contact: sip:[field0]@[local_ip]:[local_port]
      [field2]
      Max-Forwards: 100
      Expires: 300
      User-Agent: SIPp/Win32
      Content-Length: 0
    ]]>
  </send>

  <!-- asterisk -->
  <recv response="100" optional="true">
  </recv>
   <!-- simple case - just jump over a line   -->
  <recv response="200" rtd="true" next="5">
  </recv>

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

  <label id="5"/>

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

      INVITE sip:[field3]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport]
[local_ip]:[local_port];branch=[branch]
```

```
    From: sipp <sip:[field0]@[field1]>;tag=[call_number]
    To: <sip:[field3]@[field1]:[remote_port]>
    Call-ID: [call_id]
    CSeq: [cseq] INVITE
    Contact: sip:[field0]@[local_ip]:[local_port]
    Max-Forwards: 100
    Content-Type: application/sdp
    Content-Length: [len]

    v=0
    o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
    s=-
    c=IN IP[local_ip_type] [local_ip]
    t=0 0
    m=audio [auto_media_port] RTP/AVP 8
    a=rtpmap:8 PCMA/8000
    a=rtpmap:101 telephone-event/8000
    a=fmtp:101 0-11,16
  ]]>
</send>

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

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


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

```
  <send>
    <![CDATA[

      ACK sip:[field3]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport]
[local_ip]:[local_port];branch=[branch]
      From: <sip:[field0]@[field1]>;tag=[call_number]
      To: <sip:[field3]@[field1]:[remote_port]>[peer_tag_param]
      Call-ID: [call_id]
      CSeq: [cseq] ACK
      Contact: sip:[field0]@[local_ip]:[local_port]
      Max-Forwards: 70
      Content-Length: 0
    ]]>
  </send>


 <label id="1" />
<nop>
    <action>
     <exec play_pcap_audio="./pcap/g711a.pcap"/>
    </action>
</nop>
<pause milliseconds="8000"/>
<nop next="1" />

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

      BYE sip:[field3]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport]
[local_ip]:[local_port];branch=[branch]
      From: <sip:[field0]@[field1]>;tag=[call_number]
```

```
      To: <sip:[field3]@[field1]:[remote_port]>[peer_tag_param]
      Call-ID: [call_id]
      CSeq: [cseq] BYE
      Contact: sip:sipp@[local_ip]:[local_port]
      Max-Forwards: 100
      Content-Length: 0

    ]]>
  </send>

  <!-- The 'crlf' option inserts a blank line in the statistics
report. -->
  <recv response="200" crlf="true">
  </recv>



  <!-- definition of the response time repartition table (unit is
ms)   -->
  <ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150,
200"/>


  <!-- definition of the call length repartition table (unit is ms)
-->
  <CallLengthRepartition value="10, 50, 100, 500, 1000, 5000,
10000"/>


</scenario>
```