Čistý list papíru

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická
Katedra telekomunikační techniky

VoIP ústředna s vysokou dostupností

VoIP PBX with high availability

Diplomová práce

Studijní program: Komunikace, multimedia a elektronika
Studijní obor: Sítě elektronických komunikací

Vedoucí práce: **Ing. Ján Kučerák**

Bc. Josef Kresl

Praha 2016

Čestné prohlášení

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

Datum:

…….
Podpis diplomanta

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra telekomunikační techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Josef Kresl**

Studijní program: Komunikace, multimédia a elektronika
Obor: Sítě elektronických komunikací

Název tématu: **VoIP ústředna s vysokou dostupností**

Pokyny pro vypracování:

Pomocí Open-Source softwarových prostředků navrhněte a relaizujte VoIP systém s vysokou dostupností. Otestujte jej a zhodnoťte vlastnosti svého systému v porovnání s komerčně dostupnými systémy na trhu.

Seznam odborné literatury:

[1] Schmidt, K.: *High Availability and Disaster Recovery - Concepts, Design, Implementation.* Springer-Verlag Berlin Heidelberg, 11/2006. 410 pages. ISBN: 978-3-540-24460-8.
[2] Asterisk High Availability Solutions. Dostupné na http://www.voip-info.org/wiki/view/ /Asterisk+High+Availability+Solutions [on-line]

Vedoucí: Ing. Ján Kučerák

Platnost zadání: do konce letního semestru 2016/2017

prof. Ing. Boris Šimák, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 21. 12. 2015

Poděkování

Na tomto místě bych rád poděkoval všem, kteří mi poskytli podklady pro vypracování této bakalářské práce. Zejména pak děkuji Ing. Jánu Kučerákovi za odborné vedení, konzultování a rady, které mi poskytoval v rámci individuálního projektu a v této diplomové práci. V neposlední řadě bych chtěl poděkovat svému oponentovi za věnovaný čas a svým rodičům a příbuzným za materiální a morální podporu, která mi byla poskytována v průběhu celého studia.

Anotace:

Tato diplomová práce se zabývá návrhem a realizací systému pro zajištění vysoké dostupnosti přenosu hlasu po datové síti, neboli VoIP ústředna s HA. V práci je obsažen obecný popis těchto systémových ústředen a také jejich základní principy a účel. V první části je rozebrána hlavní problematika zajištění vysoké dostupnosti pro tyto služby, návrh několika řešeních a možnosti jejich implementace a komunikace mezi jednotlivými vrstvami ústředen a konečným vlivem pro koncového uživatele dané služby. Hlavní část práce se zaměřuje na obecné přiblížení struktury a použitých funkcí, implementovaných v této práci jako je monitorování stavu systému, záloha a synchronizace dat. Dále je zde rozebrán software zajišťující monitorování a následné zajištění zálohy v případě nutnosti, nebo software pro synchronizaci a zálohu dat. Závěr práce je věnován především zjištěným výsledkům měření dostupnosti a zhodnocení rozdílů mezi jednotlivými řešeními, která byla vznikem této práce a praktické implementace a měření na jednotlivých ústřednách.

Klíčová slova: vysoká dostupnost, voip ústředna, záloha, synchronizace, sipp, asterisk, virtualizace, sip, monitorování systému, záloha dat

Abstract:

This thesis describes the design and implementation of high-availability voice over a data network, or VoIP PBX with HA. The work included a general description of the system of exchanges and their underlying principles and purpose. The first part discusses the main issues of high availability for these services, the proposal several solutions and possibilities for their implementation and communication exchanges between the individual layers and the final impact to the end user of the service. The main part focuses on the general structure and approach used functions implemented in this work as system monitoring, backup and data synchronization. There is also analyzed the software that provides monitoring and follow-up to ensure backup in case of necessity, or software for synchronization and data backup. The conclusion is devoted primarily to the results mentioned measure availability and appreciation of differences between solutions, which was the creation of this work and the practical implementation and measurement at individual exchanges.

Key words: high availability, VoIP PBX, backup, synchronization, SIP, Asterisk, virtualization, SIP, system monitoring, data backup

Obsah

Seznam zkratek

API - Application programming interface
CLI - Command Line Interface
CUBE - Cisco Unified Border Element
DNAT - Destination Network Address Translation
GNU - General Public License
GNU - General Public Licence
HA - High Availability
HSRP - Hot Standby Routing Protocol
HW - Hardware
IP - Internet Protocol
IVR - Interactive Voice Response
LAN - Local Area Network
LAN - Local area network
MTBF - Mean Time Between Failures
MTTR - Mean Time To Repair
NAT - Network Address Translation
NAT - Network Address Translation
NIC - Network interface card
OTV - Overlay Transport Virtualization
PBX - Private branch exchange
PSTN - Public Switched Telephone Network
RP - Raspberry Pi
SF - Software
SIP - Session Initiation Protocol
SIPp - Traffic generator for SIP
TCP - Transmission Control Protocol
TCP/IP - Transmission Control Protocol/Internet Protocol
UAC - User Agent Client
UAS - User Agent Server
UDP - User Datagram Protocol
VLAN - Virtual local area network
VoIP - Voice over IP
WAN - Wide Area Network
WAN - Wide Area Network
XML - Extensible Markup Language

1 Úvod

Hlavním úkolem této diplomové práce je za pomoci Open Source softwarových prostředků navrhnout a realizovat systém pro zajištění vysoké dostupnosti přenosu hlasu po datové síti, neboli VoIP ústředna s HA. V práci je obsažen obecný popis těchto systémových ústředen, jejich základní princip a také principy vysoké dostupnosti nebo zálohování a synchronizaci dat. Na základě shromážděných informací o různých řešeních vysoké dostupnosti, provedu testování a zhodnocení vlastností systémů. Základním hodnocením bude snaha porovnat komerční řešení proti open source řešením. Celou práci jsem se rozhodl rozdělit do několika částí, která budou tvořit ucelenější svazky informací týkající se dané problematiky. Základním úkolem této práce bude kompletní přiblížení řešené technologie a jejího vlivu na služby, kde se pokusím vytyčit výhody a nevýhody použitých řešení. Budu porovnávat mezi sebou použité systémy, shrnu a popíši jednotlivé kroky od prvotní konfigurace a instalace VoIP ústředen s vysokou dostupností až k výsledkům vzniklých na základě této práce.

V první části práce se budu věnovat objasnění pojmů jako je například (high avalibity = vysoká dostupnost) a hlavní myšlence důležitosti zajištění vysoké dostupnosti. Je zde také rozebrána hlavní problematika celé práce a zamyšlení se nad pojmy a strukturou softwarového, ale i hardwarového řešení celé problematiky. Návrh kompletního zařízení v sobě obsahuje několik podbodů, ke kterým jsem musel přistupovat jednotlivě a řešit je jako jeden návrh. S těmito body Vás seznámím teoreticky a následně uvedu některé příklady jejich konfigurace. Mezi nejdůležitější patří návrh správného sestavení síťové topologie pro zajištění vysoké dostupnosti, návrh několika řešeních a možnosti jejich implementace a komunikace mezi jednotlivými vrstvami ústředen a konečným vlivem pro koncového uživatele dané služby. Následně zde přiblížím způsob, jakým jsem celou problematiku řešil a mohl ji podrobit testování a hodnocení.

Další část práce se zaměřuje na obecné přiblížení a popsání struktury a použitých funkcí, implementovaných v této práci jako je monitorování stavu systému, záloha a synchronizace dat. Dále je zde popsána konfigurace a postupy konkrétního softwaru zajišťujícího monitorování a následné zajištění zálohy v případě nutnosti, nebo softwaru pro synchronizaci a zálohu dat. Z tohoto důvodu jsem většinu systémů implementoval do virtuálního prostředí, kde lze testovat veškeré možné konfigurace a aktualizace systému vysoké dostupnosti, bez složitějších komplikací a vlivu na reálný provoz. Pro zbylou část implementace a testování jsem použil raspberry pi zařízeních, které jsou velice kompatibilní a universální pro práci a testování. Na těchto zařízeních jsem zkoušel jak ověření samotné vysoké dostupní, tak ale hlavně jako celkové zařízení pro generování a přijímání datových přenosů a hovorů, realizovaných přes nastavenou VoIP ústřednu s vysokou dostupností.

Poslední část mojí práce je věnována především zjištěným výsledkům měření dostupnosti a zhodnocení rozdílů mezi jednotlivými řešeními, která byla vznikem této práce a praktické implementace a měření na jednotlivých ústřednách. Zde jsou jasně viditelné konkrétní systémy a jejich schopnosti a odolnost ve vytvořeném prostředí pro testování bench-markingu HA řešeních. Jako je například asterisk ústředna, kterou jsem použil pro testování sip provozu generovaného sipp systémem za účelem otestování infrastruktury, kterou jsem vytvořil. Výsledné testování vznikajících permutací použitých technologií na sdílení paměti dat a technologií pro aplikační failover úrovně jakou jsou

použité virtuální stroje, probíhalo na odlišných strojích a vybraných systémech, aby hodnocení nebylo zkreslené. Proto jsem také použil několik různých scénářů testování a opakovaných pokusů. Z tohoto důvodu v této části představím přesné řešení dílčích problematických částí a hlavně výsledné zhodnocení mnou zvolených a použitých systémů pro zajištění vysoké dostupnosti VoIP ústředny.

2 Vysoká dostupnost

2.1 Základní pojmy

Tato úvodní kapitola popisuje základní oblasti problematiky vysoké dostupnosti použité v rámci této práce pro vlastní srovnání. Kapitola je zaměřená na to, co to vysoká dostupnost (High Availabilty) je a jak celý koncept obecně vypadá. Jako první u popisu implementace vysoké dostupnosti je nutné si říci základní pojmy, kterými se můžeme často setkat. Dostupnost a vysoká dostupnost, se liší nejen v pojmu jako takovém, ale hlavně v popisu systému. Pojem dostupnost nám označuje systém, který poskytuje určitou kvalitu služby za daných podmínek. V oblasti VoIP ústředny znamená tento pojem období, kdy byla služba dostupná bez poruchy nebo výpadku. Například zajištění služby, že v pracovní době nedojde k výpadku. V případě nedostupnosti systému může dojít k několika situacím, jako jsou plánovaná a neplánovaná nedostupnost. Čas, během kterého není služba dostupná, se označuje jako doba výpadku. Při výpadku dostupnosti je jedním ze zásadních kritérií trvání výpadku. Vlastnost vysoká dostupnost charakterizuje systém, jímž zásadním předmětem je minimalizace doby výpadku. Tato vlastnost je nezbytná pro služby související s financemi, zdravotní péčí a jinými důležitými oblastmi, které vyžadují vysokou dostupnost. [1]

Dostupnost může být:

- úplnou, kdy došlý požadavek může být obsloužen kteroukoliv volnou OL (v telefonii se používá pojem dokonalý svazek vedení). [2]

- neúplnou, kdy určitý požadavek může být obsloužen volnou linkou jen z určité podskupiny K linek z celkového počtu N (nedokonalý svazek). Při neúplné dostupnosti může dojít ke ztrátě požadavku i tehdy, když jsou některá vedení svazku volná - jsou to ovšem vedení nedostupná požadavku. [2]

2.2 Co je vysoká dostupnost

Vysoká dostupnost (anglicky High availability neboli zkráceně HA) je pojem používaný ve výpočetní a telekomunikační technice a to jak v souvislosti s hardwarem, tak se softwarem. Vysoká dostupnost je charakteristika systému, jehož cílem je zajištění vyšší dohodnuté úrovně provozní výkonnosti, než za obvyklé období. Vysoká dostupnost je pojem vyjadřující spolehlivost systémů a služeb obvykle v oboru výpočetní techniky. K dispozici jsou tři principy návrhu systémů vysoké dostupnosti.

- Eliminace jednotlivých míst selhání. To znamená přidání redundance do systému tak, aby porucha nebyla součásti selhání celého systému.
- Spolehlivý cross-over. Ve víceprocesorových systémech, cross-over bod sám o sobě má tendenci se stát jediným bodem selhání. Vysoká dostupnost musí poskytovat spolehlivé řešení cross-over.
- Detekce poruch, jak k nim dojde. Jsou-li dodrženy předešlé principy, pak uživatel nikdy nezaznamená selhání.

2.3 Pojem dostupnost

Dostupností se rozumí veličina udávající to, že systém je k dispozici v případě, že je vyžádáno jeho použití. Dostupnost se vyjadřuje v procentech a lze ji považovat

za pravděpodobnost, že se systém poskytuje požadované funkce za určených podmínek a v daném časovém okamžiku. Dostupnost je dána jak spolehlivostí systému samotného, tak i s časem, kdy dojde k obnovení poskytování služby po poruše. Dostupnost je vyjádřena poměrem doby, kdy byl systém v daném časovém období dostupný a celkového časového období.

$$\alpha = \frac{střední\ doba\ mezi\ poruchami}{střední\ doba\ mezi\ poruchami + střední\ doba\ obnovy}$$

- Dostupnost na druhé straně představuje úroveň, do které je systém nebo součást funkční a k dispozici v případě, že je vyžadováno jeho použití.
- Střední doba mezi poruchami (MTBF, Mean Time Between Failures) označuje střední dobu v hodinách, po kterou se očekává, že zařízení nebo služba bude bezchybně pracovat.
- Střední doba obnovy (MTTR, Mean Time To Repair) je časový interval během kterého dojde k obnovení systému po poruše.
- Spolehlivost je schopnost systému, nebo je jeho součástí, vykonávat požadované funkce za daných podmínek po určené časové období. [2]

2.3.1 Běžná dostupnost

Mezi běžnou dostupnost zařízení se v podstatě řadí běžně dostupné aplikace a služby, které běží bez omezení, ale nemůžeme u nich zajistit přesnou garanci dostupnosti. Daná aplikace je dostupná klientům nebo uživatelům dokud nedojde k selhání některého z prvků systému podléhající té dané činnosti.

2.3.2 Vysoká dostupnost

Dostupnost se stává nebo už i stala v mnoha informačních a telekomunikačních oblastech základním pravidlem pro fungování firmy. Při výpadku se časový interval toleruje ve velmi malém rozsahu jak pro plánované, tak pro neplánované případy. Stálá dostupnost služby neboli vysoká dostupnost služby bez jakéhokoliv zastavení, je v reálném prostředí fungování systému nedosažitelné a proto různými prostředky a implementací dalších podpůrných systému nebo služeb se tento stav snažíme minimalizovat.
Velké množství dnešních firem si přeje mít úroveň dostupnosti aspoň 99.95 %, což představuje toleranci pětihodinového výpadku v průběhu jednoho roku. Další důležitou vlastností je doba trvání jednoho výpadku. Dnešní firmy považují za přijatelnou dobu neplánovaného výpadku maximálně 15 minut.

Dostupnost [%]	Prostoje za rok	Prostoje za měsíc	Prostoje za týden	Prostoje za den
90% ("jeden devíti")	36.5 dny	72 hodin	16,8 hodin	24 hodin
95%	18.25 dny	36 hodin	8,4 hodiny	1.2 hodiny
97%	10,96 dnů	21.6 hodin	5,04 hodin	43.2 minut
98%	7.30 dny	14,4 hodin	3.36 hodiny	28,8 min
99% ("dvě devítky")	3.65 dny	7.20 hodiny	1.68 hodiny	14,4 min
99,5%	1,83 dní	3.60 hodiny	50.4 minut	7.2 minut
99,8%	17,52 hodin	86.23 min	20.16 min	2,88 minuty
99,9% ("tři devítky")	8.76 hodiny	43,8 min	10.1 minut	1.44 minuty
99,95%	4.38 hodiny	21,56 minuty	5,04 minuty	43.2 sekund
99,99% ("čtyři devítky")	52,56 minuty	4.38 minuty	1.01 minuty	8.66 sekund
99,995%	26.28 min	2.16 minuty	30,24	4.32 sekund

			sekundy	
99,999% (pět "devítky")	5.26 minuty	25.9 sekund	6.05 sekund	864.3 milisekund
99,9999% ("šest devítek")	31,5 sekundy	2,59 sekundy	604.8 milisekund	86.4 milisekund
99,99999% ("sedm devítky")	3.15 sekund	262.97 milisekund	60.48 milisekund	8.64 milisekund
99.999999% ("osm devítky")	315.569 milisekund	26.297 milisekund	6.048 milisekund	0,864 milisekund
99.9999999% ("devět devítky")	31.5569 milisekund	2.6297 milisekund	0.6048 milisekund	0,0864 milisekund

Tabulka 1: Přehled procentuální dostupnosti v jednotkách času

Oficiální standardizovaná definice pojmu vysoká dostupnost není nikde uvedena. Z toho důvodu bylo harvardskou výzkumnou skupinou definováno několik takzvaných prostředí dostupnosti, třídy dostupnosti aby se dané parametry nebo pojmy mohly přesně přiřadit k jasné kategorii.[6]

- AE4 prostředí pro obchodní činnosti, které vyžadují nepřetržitý výpočetní běh systému a kde každá porucha nebo úprava je transparentní pro uživatele. To znamená, že nemůže dojít k žádnému přerušení práce, aby se nepřerušily žádné transakce, žádné ponížení výpočetního výkonu a trvalý provoz 24x7.[6]
- AE3 prostředí pro obchodní činnosti, které vyžadují nepřerušovaný výpočetní výkon služeb a to například v pracovních časových směnách, nebo během většiny hodin za den a po většinu dní v týdnu po celý rok. To znamená, že uživatel zůstává on-line v nejvíce vytěžovaných chvílích. V případě, kdy aktuální služba může potřebovat restart, mohou uživatelé zaznamenat snížení výkonu nebo větší odezvu služeb.[6]
- AE2 prostředí pro obchodní funkce, které umožňují minimální přerušení výpočetních služeb a to buď v hlavních časových dobách, nebo během většiny hodin za den a po většinu dní v týdnu po celý rok. To znamená, že spojení uživatele se systémem bude přerušeno, ale ve velmi krátkém intervalu s rychlým opětovným připojením. Také může dojít k dávkovému omezení služeb pro služby s vyšší prioritou a uživatel zaznamená snížení výpočetního výkonu systému.[6]
- AE1 prostředí pro obchodní funkce, které mohou být přerušeny, za podmínky nulové ztráty přenášených dat. V případě nekontrolovaného pádu služeb nebo náhlému vypnutí uživatelské práce, ale s garancí zpětné dostupnosti dat. Backup copy neboli zpětná záloha dat, která je k dispozici na záložních discích a záznamů v databázi nebo záznam souborů do systémového deníku, na základě kterých bude obnovena předešlá práce uživatele.[6]
- AE0 prostředí pro obchodní funkce, které mohou být přerušeny a kde zpětná dostupnost dat není podstatná. Pokud dojde k nekontrolovanému pádu služeb nebo náhlému vypnutí uživatelské práce, data mohou být ztracena nebo poškozena.
- Disaster Recovery neboli zotavení po havárii je schopnost systému paralelní dostupnosti, která je použitelná na některou z předešlých dostupností. Zajišťuje vzdálené zálohování informačního systému při nebezpečí nebo přírodních katastrofách, jako je například zemětřesení, požáry, povodně, hurikány, dlouhodobé výpadku napájení, vandalismus, nebo teroristický čin.[6]

Stupnice prostředí s dostupností vázaná na terminologii dle harvardské skupiny používanou po světě. Zde je vidět jasný přehled dostupnosti zařízení.

- AE0 Konvenční
- AE1 vysoce spolehlivých
- AE2 Vysoká dostupnost
- AE3 Odolné proti chybám
- AE4 Odolné proti poruchám

K dispozici máme také pro jasný přehled třídy dostupnosti používané v mezinárodním měřítku. Přehledné kategorizování procentuálních dostupností z tabulky číslo 1, kde jsou procentuální dostupnosti doplněny i o časové údaje.

- Dostupnost třída 1 (90%) – nebere se v potaz jako hodnotící kritérium
- Dostupnost třída 2 vysoce spolehlivé (99%)
- Dostupnost třída 3 vysoká dostupnost (99,9%)
- Dostupnost třída 4 odolné proti chybám (99,99%)
- Dostupnost třída 5 odolné proti poruchám (99.999%)
- Dostupnost třída 5 tolerantní proti jakémukoliv výpadku (99.999%)

Dostupnost třídy 1 s 90% je odstraněn z tohoto seznamu, 90% se považuje za konvenční, a tím spadá mimo rozsah působnosti a tříd tohoto seznamu. Systém je považován za "vysoce dostupný" v případě, že je zaručen výpadek menší než hodinu, což se odráží v tomto seznamu o 99,99% což je přibližně 53 minut za celý rok. [8]

2.3.3 Nedostupnost

Za nedostupnou je považována každá služba nebo aplikace, ke které se v daný časový moment nelze připojit nebo s ní pracovat předurčeným způsobem. V některých případech může být za nedostupnou považována i částečně omezené služby, která v daný požadovaný okamžik nevykonává svoji činnost známým způsobem. Obecně je však nedostupná každá služba, která nedělá to co má. Z tohoto důvodu došlo k rozdělení a to, že služba nemusí být plně nedosažitelná, ale každé její nežádoucí chování ji dělá nepoužitelnou a tím se stane nedostupnou. Doba neboli časový horizont, po který se služba nachází ve stavu nedostupnosti, se nazývá doba nedostupnosti. Pro tento případ je další dělení a to na plánované nebo neplánované výpadky.

2.3.4 Plánovaná nedostupnost

Plánovaná nedostupnost neboli plánované výpadky jsou stanoveny přesným harmonogram prací a lze se připravit předem. Typicky plánované úlohy jsou třeba výměna hardwarových komponentů serveru, aktualizace operačního systému, aplikace opravných nebo rozšiřujících balíčků aplikace, a mnoho jiných dějů. Některé z těchto činností vyžadují restart serveru, nebo v případě výměny hardware i vypnutí celého serveru. V tuto dobu jsou služby aplikace nebo aplikací běžících na konkrétním serveru nedostupné. Jednoduše řečeno, plánovaný výpadek je důsledkem nějaké plánované a předem připravené akce. [3]

2.3.5 Neplánovaná nedostupnost

Opakem je neplánovaný výpadek, je to nečekaná událost, na kterou se musí ihned reagovat a být na ni připraven. Mezi takovéto události patří například výpadek hardwarových komponentů serveru, na kterých běží vnitřní aplikace. Další událostí může být nestabilita v samotné aplikaci, která kvůli náhlé nevyřešené chybě spadla nebo se zacyklila. Výpadky tohoto typu můžou být způsobeny i vnějšími vlivy, jako je přerušením elektrického vedení, nebo nedostupností síťové konektivity. Neplánovaným výpadkům se nevyhneme, ale musíme se snažit jim předcházet důkladným sledováním statistik systému,

nebo vlastních aplikací, ale také celé infrastruktury, která je důležitá pro běh aplikace. Pro každý typ výpadku je třeba mít připravené nouzové postupy, jak v případě konkrétního výpadku postupovat. [3]

2.4 Realizace vysoké dostupnosti

Vysoké dostupnosti je možné dosáhnout několika způsoby [4]:

- Redundancí prvků
- Použitím hardwarových a softwarových přepínacích nástrojů (switching techniques)
- Rozvrhem plánovaných výpadků
- Eliminací interakce člověka se systémem
- Definováním automatického řízení pro případy výskytu chybových situací
- Rozsáhlým akceptačním testováním
- Definováním a trénováním reakcí operátora systému v případech neplánovaných výpadků, které nejsou ošetřeny automaticky.

Každá technologie vysoké dostupnosti má minimální požadavky, které je třeba před konfigurací specifického řešení splnit. Kromě splnění těchto požadavků je rovněž důležité určit, které prostředky mají být odolné. Tyto prostředky, jako jsou aplikace, data a zařízení, musí být vyhodnoceny a musí se určit, zda mají být vysoce dostupné. Vyžadují-li vysokou dostupnost, je důležité provést nezbytné změny prostředí před konfigurací řešení vysoké dostupnosti. Můžete například mít data umístěná v diskové oblasti, která by měla být vysoce dostupná. Před konfigurací řešení byste tato data měli přesunout do nezávislé diskové oblasti. K dosažení vysoké dostupnosti možná bude nutné provést i změny aplikací.

- Aplikace klastru

Jedním z klíčových prvků v klastrovaném prostředí je odolnost služeb. Jestliže chceme používat ve svém zařízení služby nebo aplikace s vysokou dostupností, musíme zajistit, aby tyto systémy měly specifické vlastnosti pro dostupnost. Odolnost našeho systému v klastru, je v možnosti nezávislého restartování zařízení a jeho redundance v jiném klastrovém uzlu, aniž by bylo nutno rekonfigurovat klienty. Navíc při selhání nebo přepnutí systému budou dostupná data přiřazena k odpovídající službě na redundantním klastru. To znamená, že když se služba a její data přepnou z primárního uzlu na záložní uzel, u uživatelů se projeví pouze minimální nebo žádné přerušení dané služby. Koncový uživatel služby nepotřebuje vědět, že se aplikace a data přesunula na záložní systém v jiném uzlu, ale zaručit vysokou dostupnost dat.

- Plánování odolnosti dat

Odolnost dat je schopnost systému, aby data služeb byla k dispozici uživatelům nebo aplikacím, která využívají daná data. Odolnosti dat dosáhneme použitím technologie přepínání disků, nebo zrcadlením mezi servery, nebo s logickou replikací dat mezi servery.

- Plánování odolnosti prostředí

Odolnost prostředí zajistí, že služby a aplikace zůstanou mezi prostředky definovanými v prostředí vysoké dostupnosti konzistentní. Je třeba identifikovat, které prostředky vyžadují ke své správné funkci konzistentní prostředí, a je třeba vytvořit administrativní

doménu klastru, která zajistí, že tyto prostředky zůstanou v řešení vysoké dostupnosti konzistentní. [6]

2.5 Síťová topologie

Kromě použitých systémů HA je důležitou součástí použití síťové topologie. Vytvořenou topologii pro samostatnou ústřednu musíme přizpůsobit pro dané použití. Systém HA může sdílet jedinou síťovou adresu rozhraní s asterisk ústřednou, nebo může mít zcela samostatné síťové rozhraní. Existuje mnoho způsobů, jak navrhnout a rozdělit síťovou topologii a rozhraní pro použití vysoké dostupnosti. Optimální konstrukce je závislá na okolnostech a podmínkách, pro které bude HA použito.

- Počet dostupných fyzických síťových karet na serveru
- Dostupnost VLAN v síti
- Použití jediné plovoucí VoIP IP mezi ústřednami
- Nastavení ústředen do jednoho nebo dvou datových center
- Dostupnost 2 vrstvy Tunely / VPN mezi datovými centry

V souvislosti s návrhem a přidělováním síťových rozhraní, musíte zvážit, jak HA a asterisk se připojí k místní síti sami navzájem. Pro celkový přehled různých řešení, je uvedeno několik nejvíce používaných návrhů zapojení.

2.5.1 Postup návrhu High Availability

Projektování sítě pro zajištění vysoké dostupnosti vyžaduje, abychom zvážili spolehlivost každého síťového hardwaru a softwarových komponent, volbu redundance, atributy protokolu, obvody a možnosti přenosu a energetických vlastností, které přispívají k celkové dostupnosti sítě. Při navrhování služeb vysoké dostupnosti pro podnikové sítě, si jako první musíme položit několik otázek. [3]

- Kde v síti by měl být záložní modul připojen?
- Jaký software pro zajištění spolehlivosti bude použit v síti?
- Jaké protokoly je třeba vzít v úvahu?
- Jaké budou rysy a požadavky na přenos dat a energetickou stabilitu?
- Jaké postupy a opatření budou zavedeny, aby se zabránilo výpadkům?

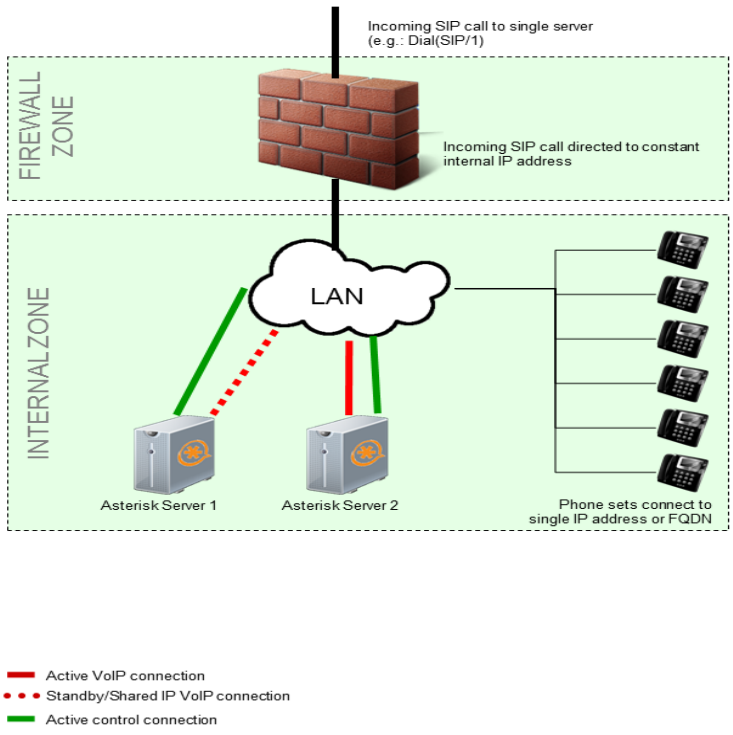
High Availability (HA), je obvykle dosaženo prostřednictvím shlukování, což znamená, že dva servery fungují jako jeden pro tu stejnou službu. Existuje mnoho způsobů, jak vytvořit dostupnost, z nichž každý má své vlastní výhody, rizika, náklady a kompromisy. Návrh vysoké dostupnosti se většinou skládá z jednoho primárního zařízení a jednoho nebo více záložních (sekundární) strojů pro dosažení dostupnosti a zálohy. Důležité však je, aby opravdu vznikla vysoká dostupnost nepoužívat žádné společné medium nebo přístupové cesty. Každé zařízení musí mít vlastní autonomní hardware, software a logická nebo řídící media.[4]

Dalším důležitým bodem je správná synchronizace dat, aby v případě výpadku byla použita aktuální data. Proto používaná zařízení musejí zůstat ve stále synchronizaci. Nicméně, synchronizace je jednou z největších výzev v celém návrhu. Je to další nejdůležitější kritérium při navrhování, výběru a realizace vysoké dostupnosti. K dispozici jsou dvě možnosti řešení tohoto problému. Nejjednodušší postup je nechat sdílené paměťové medium uspořádané do raid pole. Nicméně, tímto krokem porušíme jeden z

prvních cílů vzájemné autonomie systémů, kdy v případě vážného poškození raid pole dojde ke ztrátě všech dat. Druhá možnost, která nám zbyla, je použití nesdílených paměťových medií, která jsou mezi sebou vzájemně synchronizována. Pro vzájemnou synchronizaci dat mezi zařízeními existuje celá řada různých nástrojů a některé z nich zde jsou také ukázána a použita jejich konfigurace.[4]

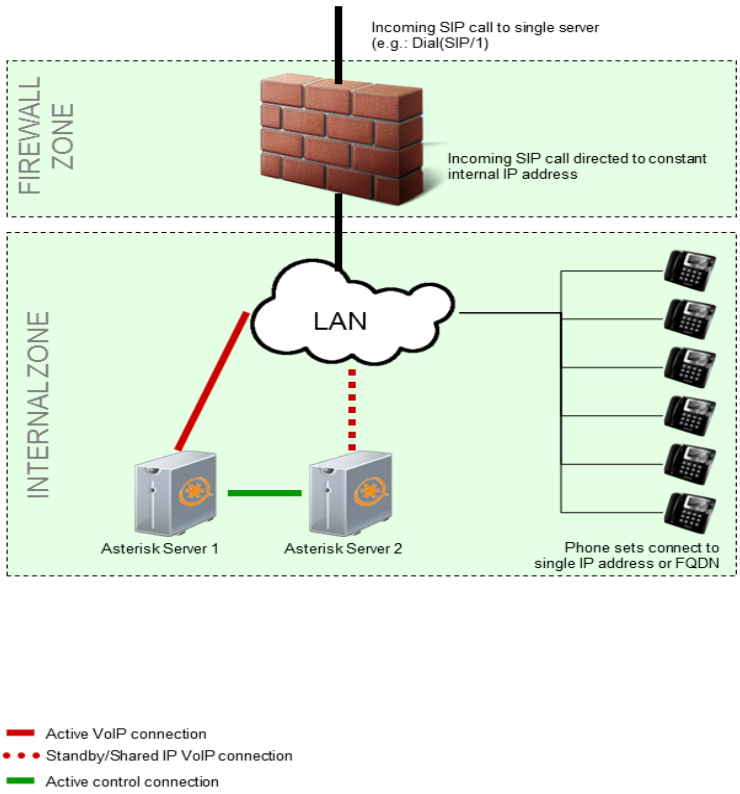
Aby HA bylo plně funkční, musí vědět, kdy primární zařízení je funkční nebo má převzít hlavní činnost sekundární zařízení. Nejjednoduší možnost selhání je monitorování zdali je zařízení ve stavu online a odpovídá na pravidelně vysílané echo dotazy. V případě žádné odpovědi se automaticky aktivuje HA. Nicméně, toto je zřídka případ v reálném provozu. Mnohem horší je situace, kdy primární zařízení pomalu zhoršuje svůj stav a omezuje poskytované služby. Nebo situace, kdy zařízení je ve stavu online a odpovídá, ale VoIP ústředna neprovede propojení hovoru nebo ztrácí pakety. Pro tyto případy je řešení mnohem složitější, jelikož je potřeba monitorovat všechny jednotlivé funkce a kontrolovat přišlá a odešlá data. Systémy, které se používají na kompletní monitorování všech služeb, jsou součástí patentovaných řešení a služeb povětšinou implementovany již do specializovaných komerčních zařízení pro VoIP ústředny. V našem případě jsou použity systémy na jednodušší bázi, kde nám dochází k monitorování stavu zařízení. [4]

A nyní si uvedeme několik příkladů nejběžnějšího zapojení síťové topologie pro VoIP ústřednu. V prvním případě máme dvě řídící IP adresy, jedna sdílená IP adresa pro VoIP, jednodatové úložiště. V této topologii, každé zařízení má svoji vlastní řídící IP a společnou sdílenou IP pro asterisk VoIP. Sdílená IP plave mezi zařízeními. Všechny NIC jsou na stejné fyzické síti LAN. Každé spojení může mít jednu nebo dvě fyzické NIC. V případě jedné NIC, přidá systém dostupnosti VoIP NIC jako virtuální IP, neboli virtuální NIC, které je aktivní u primárního zařízení. [5]



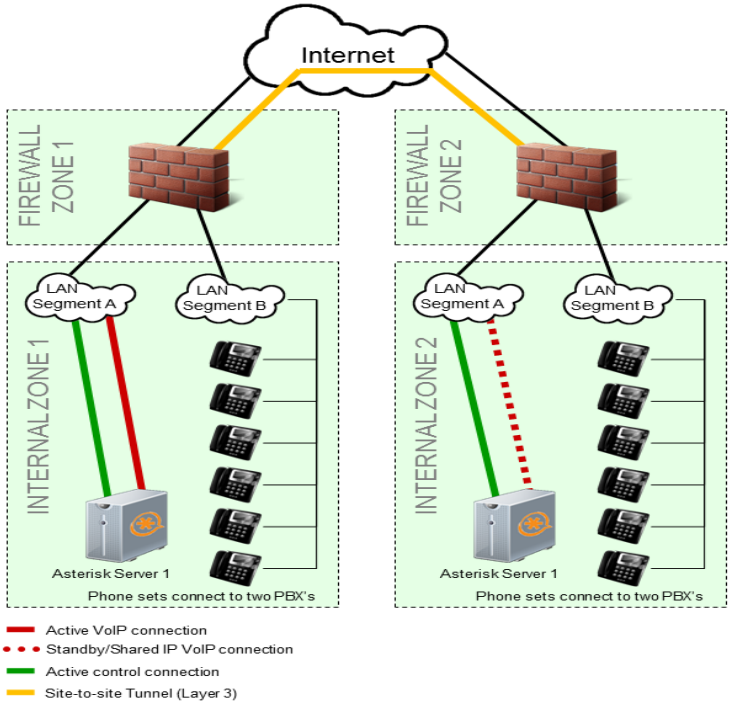
Obrázek 1: Příklad zapojení síťové topologie [5]

V této topologii, každé zařízení má svoji vlastní řídící IP pro systém HA a sdílejí jednu IP pro asterisk VoIP ústřednu. Sdílená IP plave mezi zařízeními. Tento způsob propojení a kontroly pomocí kříženého spojení je limitován přenosem a přístupem přes VoIP. Každé spojení musí mít dvě fyzické NIC. [5]



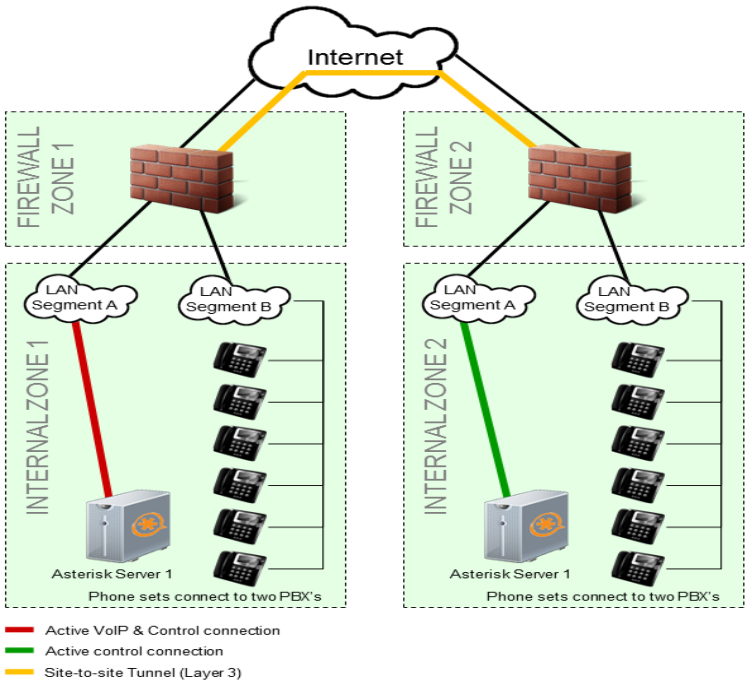
Obrázek 2: Příklad zapojení síťové topologie [5]

V této topologii každé zařízení má svoji vlastní řídící IP pro HA systém a sdílí jednu IP pro asterisk VoIP ústřednu. Spojení jsou umístěna do různých datových center, která jsou propojena VPN tunelem. HA systém kontroluje IP adresy v rozdílných VLAN podsítích. Tyto telefony musí být v jiné podsíti než je asterisk, aby bylo zajištěno správné směrování. Asterisk ústředna má svoji interní VoIP IP adresu stejnou, což umožňuje IP adresu transparentně přesouvat mezi jednotlivými zařízeními. Tato IP adresa může být stejná v případě přesměrování cesty v interní komunikaci, nebo bude jiná, když použiji DNAT přesměrování interního provozu. HA systém při startu nebo resetu provede akci, kdy aktualizuje NAT pravidlo pro spojovací trasy na každém firewallu řídícího routeru, který přesměrovává provoz na aktivních spojeních. [5]



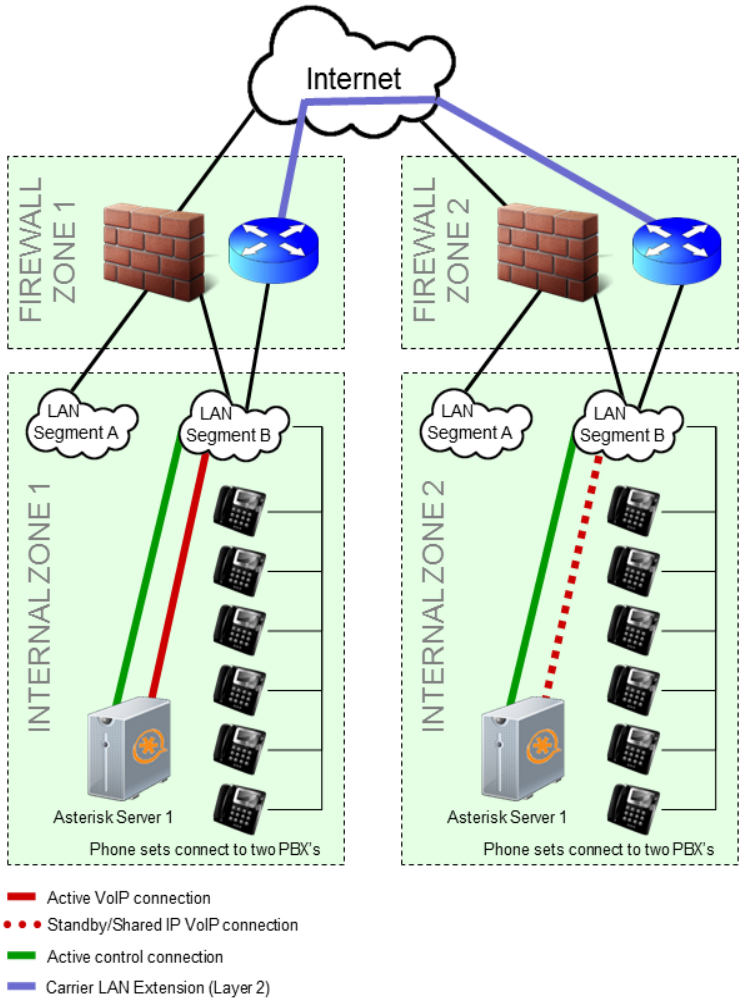
Obrázek 3: Příklad zapojení síťové topologie [5]

V této topologii má každý HA systém svoje vlastní spojení s řídící IP adresou pro HA systém dostupnosti a každý z nich má svoje vlastní spojení s VoIP IP adresou. Firewall zde působí jako registrační proxy server pro vnitřní VoIP telefony a v případě výpadku přehodí SIP provoz na záložní vnitřní spojení s asteriskem. HA systém spustí pouze asterisk ústřednu na aktivním spoji, takže proxy server bude detekovat SIP spojení a propojí příslušné spojení. Nabízí se ještě možnost, že mnoho telefonů poskytuje možnost registrace druhého SIP serveru. Tyto telefony mohou být nakonfigurovány s IP adresami z aktivního a záložního serveru. Telefony se budou pak vždy spojovat na správný server, protože HA systém deaktivuje pohotovostní spojení na asterisk ústřednu v proxy serveru. [5]



Obrázek 4: Příklad zapojení síťové topologie [5]

U této topologie má každé zařízení svoji vlastní řídící IP adresu pro HA systém a sdílí jednu IP adresu pro asterisk VoIP ústřednu. Spojení jsou také umístěna do různých datových center, ale jsou připojena přes rozšířené LAN sítě, kde daný rozšiřující router spravuje VPN spojení. VoIP IP adresa zůstane na stejné podsíti, což mi umožní virtuální IP adresu transparentně přesouvat mezi jednotlivými zařízeními. Kontrolované IP adresy HA systémem jsou v různých VLAN podsítí oproti VoIP IP adrese. Tento návrh předpokládá dostupnost OTV (v rozšířených datových centrech) přes rozšířenou službu druhé vrstvy LAN sítě, když to nebude možné, použijte předchozí topologie. [5]



Obrázek 5: Příklad zapojení síťové topologie [5]

3 Teoretický rozbor problematiky vysoká dostupnost

V této kapitole si přiblížíme teoreticky různé technologie nebo systémy, které jsou základem pro funkčnost VoIP ústředny samotné, tak ale i této práce. Postupně se budume snažit přiblížit v nejdůležitějších bodech informace o jednotlivých systémech a technologiích, tak aby v pozdějších kapitolách jsme se mohli věnovat pouze dané problematice.

3.1 VoIP

V posledních několika letech zaznamenalo telekomunikační odvětví velký pokrok. Hlavně v rozvoji datových sítí a zejména telefonní komunikace po internetové síti se setkala s velkou oblibou a to hlavně využitím v podnikatelském sektoru tak i u veřejnosti. Pro použití komunikace VoIP (Voice Over IP) jsou využívány stávající datové sítě vzniklé primárně pro internet síť. Použití těchto stávajících sítí snížilo náklady na realizaci propojení a zřízení telekomunikační sítě. Přispěly k ušetření nákladů v provozu, kdy nejsou využívány přepojované okruhy, ale datové sítě. VoIP technologie byla vytvořena jako alternativa k PSTN (Public Switched Telephone Network) zahrnující nynější telekomunikační síť. [8]

VoIP umožňuje přenos hlasu v sítích s přepojováním paketů založených na protokolu IP. VoIP tak tvoří alternativu ke klasické telefonii, založené na použití sítí s přepojováním okruhů přes veřejnou telefonní sít. Tato alternativa je velice perspektivní a to hned z několika důvodů. Datové sítě se v současnosti rozvíjejí a šíří mnohem rychleji než telefonní sítě. Přenos hovorů v datových sítích je ekonomicky výhodný, protože rozsáhlé a geograficky rozdělené společnosti mohou uskutečňovat hlasový provoz po vlastních datových sítích, jimiž většina společností disponuje. Připojení operátora poskytujícího na IP konektivitě i hlasové služby logicky vede k nižším cenám, neboť jsou redukovány náklady na komunikační infrastrukturu. Dalším aspektem IP telefonie jsou nové aplikace a doplňkové služby využívající informační technologie, právě v propojení s oblastí IT je ukrytý dosud nevyužívaný potenciál IP telefonie. Další důvod zavádění VoIP je snaha o sjednocení komunikačních standardů a vytvoření tak sítí s integrovanými službami, které jsou schopny nad jedinou infrastrukturou přenášet data, hlas nebo video. [9]

Nepředstavuje pouze alternativu k veřejným telefonním sítím, jedná se o kompletní řešení, které může být využito jak ve firemní, tak v soukromé sféře. VoIP pracuje na otevřených systémech, využívá stávající IP sítě, standardní prvky a známé operační systémy. Hlavní výhodou VoIP je výrazná redukce potřeby hardware i fyzického místa při zachování existujících technologií v daném systému a využití zavedených mechanismů sítí vytvořených na IP (Internet protocol) v celosvětově známé síti Internet. VoIP využívá sítě používající různé protokoly určené k přenosu digitálních dat. [9]

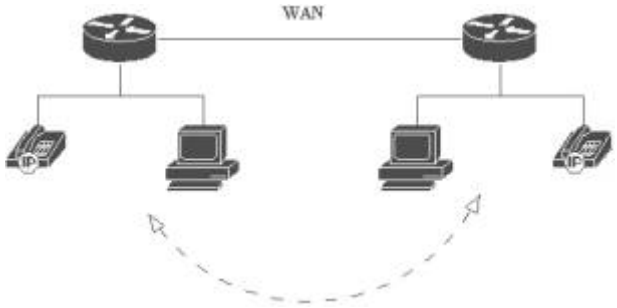
Pro připomenutí si zdůrazněme zásadní rozdíl mezi tím, co je VoIP a čemu se říká internetová telefonie, nebo IP telefonie.

- VOIP je technologie: chcete-li je to technická záležitost, říkající jak něco realizovat (konkrétně přenos hlasu po IP). Už nutně neříká, k čemu a jak se to využívá, resp. jaký efekt (užitek) z toho je. Kromě telefonování by se technologie VOIP daly využít například pro jednosměrné šíření mluveného slova na principu rozhlasového vysílání (pro hudbu už s zceale moc nehodí, protože je VOIP je optimalizován pro lidský hlas). [10]
- IP telefonie je služba: je to telefonování a to taková jeho varianta, která ke své realizaci využívá protokol IP (a tím i technologie VOIP). Neříká nic o tom, zda je

tento protokol přenášen v nějaké uzavřené privátní síti (a pak jde o telefonování "po" nějaké uzavřené privátní síti), či třeba po kabelové přípojce (ještě než tato vyústí do veřejného Internetu), nebo zda je tento protokol provozován po veřejném Internetu (tj. zda jde o telefonování po Internetu). [10]

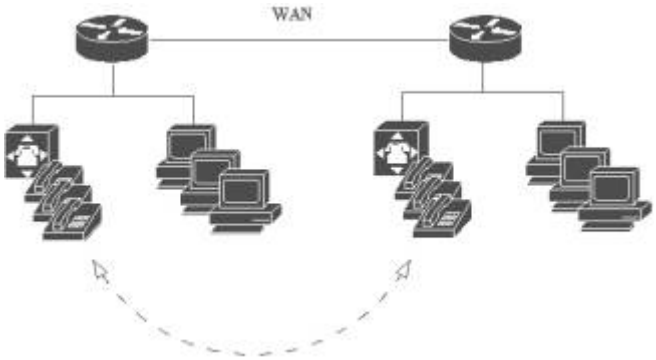
- Internetová telefonie je také službou: opět jde o službu charakteru telefonování, která k přenosu hlasu (v datové podobě) využívá veřejný Internet. Použitá technologie by obecně mohla být jakákoli, ale v praxi jde snad vždy o VOIP. [10]

V praxi a v běžné mluvě se rozdíl mezi VOIP a internetovou telefonií ztrácí a tyto dva termíny se pomalu stávají synonymy. Nezapomínejme ale na to, že tomu tak není, a že to jsou skutečně dvě principiálně odlišné věci. Jeden termín (VOIP) označuje technologii, přesněji skupinu technologií, druhý konkrétní službu (telefonování "po Internetu"). [10]



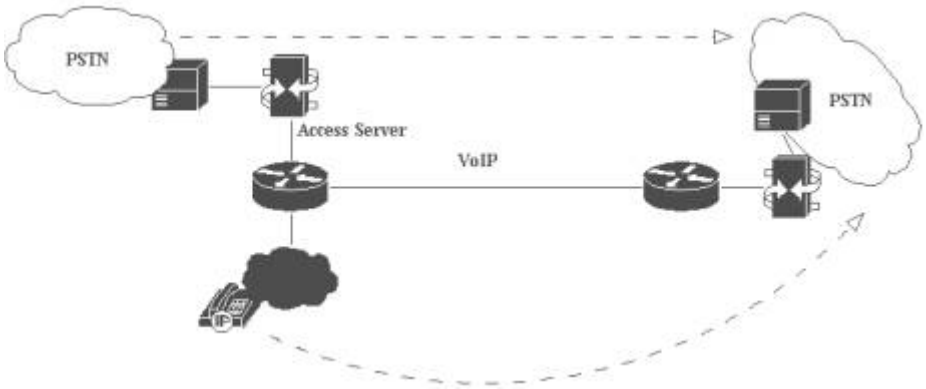
Obrázek 6: Využité IP telefonie [9]

Toto využití lze označit jako scénář IP-Phone to IP-Phone, vlastní IP telefon je realizován jako SW aplikace pro PC nebo hardware v podobě IP telefonu. Tento scénář je používán nejvíce pro vnitrofiremní komunikaci.



Obrázek 7: Využité IP telefonie [9]

Toto využití lze označit jako scénář PSTN to PSTN. Dochází k propojení sítí PSTN (Public Switched Telephone Network - klasická analogová telefonní síť) přes IP sít. PSTN je připojeno do IP sítě pomocí bran zajišťujících převod. Tento scénář je využíván také v rámci podniků mající PSTN síť, je také využíván operátory a je nabízen za cenově zvýhodněných volání.



Obrázek 8: Využité IP telefonie [9]

Scénář IP-Phone to PSTN (resp. PSTN to IP-Phone) je v podstatě analogií předchozího scénáře. Objevuje se ve veřejné síti a realizuje výstup z IP sítě v určitém bodě a nabízí přístup do PSTN jako službu VoIP to PSTN (nebo naopak).

Pro zprovoznění VoIP technologie potřebujeme protokoly, které zřídí signalizaci pro vznik a řízení hovoru. Součástí sítě jsou proxy servery, které nám registrují VoIP telefony a koordinují signalizaci mezi rozsáhlými sítěmi. V případě že proxy server zaregistruje volání na standardní telefonní číslo, převede pakety na klasické telefonní sítě signál na speciálních VoIP branách. Základem běžné komunikace je registrace účastníka u některé řídící jednotky, jako je VoIP operátor, který zprostředkovává sestavování hovorů. Nebo se nabízí další možnost, že daný zprostředkovatel vytvoří dvě spojení a bude fungovat jako man in the middle. Po prvotním sestavení hovoru je hovor průběžně rozdělován na malé fragmenty po několika milisekundách, které jsou následně zakódovány jedním z předem dohodnutých kodeků. Ty jsou následně přenášeny protokolem RTP (Real-time Transport Protocol) k příjemci tohoto hovoru. Podmínkou pro zaručení dobré funkčnosti je služba QoS (Quality of Service) na jednotlivých síťových prvcích, kde jsou postupně zpracovávány pakety komunikace. VoIP telefonie se právě kvůli těmto službám řídí jasným pravidlem komunikace na UDP (User Datagram Protocol) protokolu než na TCP (Transmission Control Protocol) protokolu, že je lepší paket zahodit, než se pokoušet daný paket doručit se zpožděním příjemci. [11]

3.1.1 VoIP protokoly

Protokoly VoIP jsou uzpůsobeny k realizaci přenosu hlasu za použití IP sítí. Realizována je telefonie a hlasová komunikace sítí Internetového Protokolu (IP), fungující na principu přenosu dat v podobě datagramů. Existují dva typy VoIP protokolů. První skupina, signalizační protokoly, starající se o inicializaci, správu, vedení a ukončování hovoru. Do skupiny signalizačních protokolu patří dva nejznámější z nich, H. 323 a SIP, které jsou používány v drtivé většině VoIP zařízeních umožňující přenos hlasu v IP síti. Druhou skupinou protokolů VoIP jsou protokoly komunikační. V souvislosti s přenosem hlasu je dobré zmínit protokol RTP (Real-time Transport Protocol), patřící mezi komunikační protokoly, který umožňuje právě přenos mediální informace. [8]
Příklady VoIP protokolů jsou: [9]

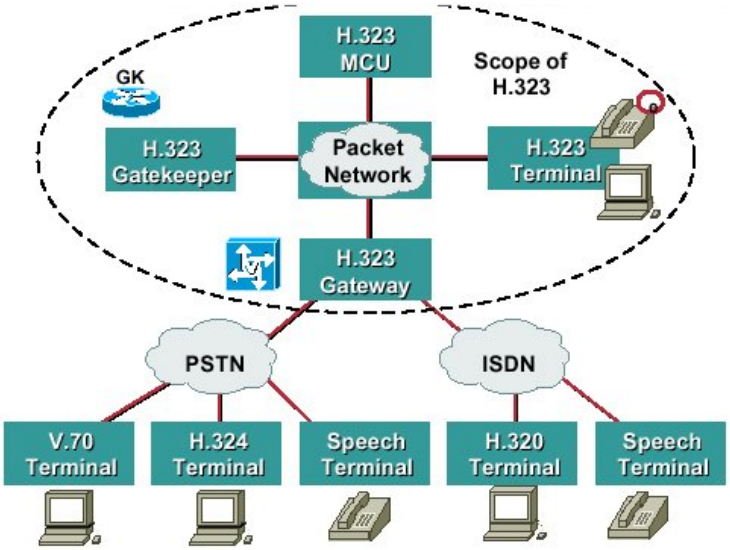
- H.323
- Brána Media Control Protocol (MGCP)
- Session Initiation Protocol (SIP)
- H.248 (Megaco)

- Real-time Transport Protocol (RTP)
- Real-time Transport Control Protocol (RTCP)
- Srtp (SRTP)
- Session Description Protocol (SDP)
- Inter-Asteriske Xchange (IAX)
- Jingle XMPP rozšíření VoIP
- Skype protokol

3.1.2 Protokol H. 323

H. 323 má 4 základní komponenty: [9]

- terminál
- gateway
- gatekeeper
- multipoint control unit



Obrázek 9: Zobrazení základních komponent H. 323 [9]

Terminál je základní a jedinou povinnou komponentou H. 323 sítě. Používá se pro obousměrnou komunikaci v reálném čase. Může to být PC nebo specializované zařízení, na kterém běží H. 323 stack. Závazně musí podporovat audio služby, video a data jsou volitelná. Je schopen komunikovat s jinými multimediálními terminály v jiných sítích. [9]

Gateway - GW (brána) zabezpečuje spojení H. 323 sítě s jinou sítí (např. ISDN), slouží jako překladač protokolů mezi různými sítěmi. Je volitelnou komponentou. [9]

Gatekeeper - GK (správce zóny) - Tuto komponentu si můžete představit jako mozek sítě. Ačkoli je to volitelná komponenta, má na starosti velice důležité služby jako autorizaci, autentifikaci, překlad telefonních čísel na IP adresy, účtování služeb, směrování hovorů apod. Jedná se vlastně o analogii inteligentní ústředny. Správci zóny v sítích nejsou nutní, ale pokud jsou použiti, koncové body musí používat jejich služby. Nejčastěji je GK realizován softwarem. GK může být integrován do kterékoli komponenty H. 323. [9]
Povinné služby definované doporučením H. 323 :

- Překlad adres - z telefonního čísla, dle ITU-T E. 164 (např. 233090025) na IP adresu terminálu (např. 147.32.201.127:1720).

- Řízení přístupu - přístup terminálu ke GK může být založen na autorizaci, momentální přenosové kapacitě a několika dalších kritériích. Pokud je řízení přístupu vypnuto, tak jsou přijímány všechny požadavky na spojení.
- Řízení přenosové kapacity - zahrnuje provádění požadavků na šířku přenosového pásma. Způsob poskytování šířky pásma a správa šířky pásma záleží na poskytovateli. GK může probíhajícímu hovoru nařídit snížení používané šířky pásma. Všechna tato rozhodnutí závisí na poskytovateli této služby a jsou mimo vliv standardu H. 323. Je-li řízení přenosové kapacity vypnuto, tak jsou přijímány všechny požadavky na změnu šířky pásma.
- Oblastní management - GK poskytuje doplňkové služby pro terminály, jednotky MCU a brány, které jsou registrovány uvnitř administrativní domény řízené gatekeeperem.

Multipoint Control Unit - MCU (jednotka pro řízení konferenčního spojení) zajišťuje komunikaci tří a více terminálů (konferenční hovory). Umožňuje převádět decentralizované konferenční spojení (data jsou určena skupině stanic) na spojení centralizované (data jsou určena právě jedné cílové stanici) a obráceně. [9]
Jednotka MCU se skládá z modulů MP a MC.

- MC (Multipoint Controller) - řídí sestavování konference, zjišťuje vlastnosti terminálů v konferenci, inicializuje a ukončuje kanály pro audio, video a datové přenosy.
- MP (Multipoint Processor) - zpracovává multimediální data přenášená v konferenci. MP je volitelný modul. Pokud jsou MP použity, tak mohou být umístěny samostatně v síti (i mimo jednotku MCU).

3.1.3 Protokol SIP

Protokol SIP je signalizačním protokolem pro sestavení, dohled a rozpad spojení mezi dvěma a více účastníky komunikace. Jde o textový protokol strukturou podobný například poštovnímu protokolu SMTP (Simple Mail Transfer Protocol) nebo protokolu HTTP (Hyper Text Transport Protocol). Protokol SIP nespecifikuje, jaký má být použit transportní protokol a není svázán s žádnými konkrétními komunikačními protokoly pro vlastní přenos multimediálních dat. Textová podstata protokolu napomáhá nejen jednoduchému ladění, ale především snadné rozšiřitelnosti. Protokol SIP je typu klient-server, takže komunikace probíhá výměnou dvou typů zpráv, požadavků a odpovědí. Klient i server jsou logickou částí jednoho prvku. SIP není limitován pouze na obsluhu telefonních spojení, ale lze ho také použít pro multimediální konference, instant messaging nebo online hry. [9, 12]
Komunikace prostřednictvím protokolu SIP probíhá následujícím způsobem.

1. lokace uživatele – určení koncového systému komunikace;
2. dostupnost uživatele – určení dostupnosti uživatele ke komunikaci;
3. přenosové parametry – určení parametrů s ohledem na přenosová média;
4. nastavení relace – nastavení parametrů relace mezi volaným a volajícím;
5. management relace - včetně převodu a ukončení relace, možnost změny parametru či služeb během spojení.

Hlavička SIP zprávy by měla obsahovat tyto údaje.

- To: Adresa volaného
- From: Adresa volajícího
- Via: Cesta, kudy prošel požadavek a kudy se bude vracet odpověď

- Call-Id: Identifikace volání
- Contact: Aktuální skutečná adresa klienta
- Record-Route: Databáze adres serverů dostávající veškerou komunikaci náležící k hovoru
- Request-URI (Uniform Resource Identifier): Aktuální adresát požadavku

User agents (UA) jsou koncovými zařízeními SIP sítě. Starají se o navazování spojení s ostatními UA. Nejčastěji se jedná o SIP telefony (ať už fyzické nebo ve formě aplikací běžících na PC) a brány do jiných sítí, typicky do PSTN. V rámci UA rozlišujeme ještě User Agent Client (UAC), což je část UA, která má na starosti iniciaci spojení a User Agent Server (UAS), která reaguje na příchozí žádosti a posílá odpovědi. V koncovém zařízení (SIP telefonu) je implementován jak UAC, tak i UAS. [9, 12]

Servery jsou v SIP architektuře zařízení, jejichž úkolem je zprostředkovat kontakt mezi volajícími a volanými (tedy mezi UA), což ale nevylučuje přímý kontakt koncových zařízení bez účasti serverů. Rozlišujeme tyto tři typy SIP serverů: [9, 12]

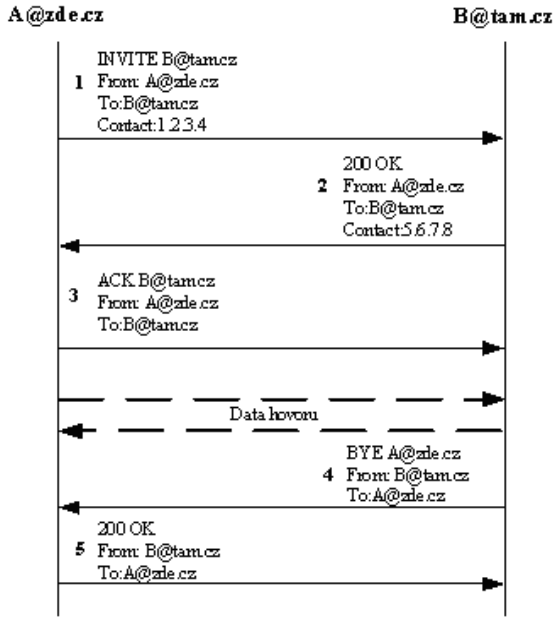
1. Proxy server. Tento server přijme žádost o spojení od UA nebo od jiného Proxy serveru a předá ji dalšímu proxy serveru (pokud volanou stanici nemá ve své správě) nebo přímo volanému UA pokud je tento v rámci jím spravované domény.
2. Redirect server. Stejně jako proxy přijímá žádosti o spojení od UA nebo proxy serverů, ale nepřeposílá je dále ve směru volaného, nýbrž posílá tázajícímu informaci, komu má žádost poslat, aby se dostala k volanému. Je pak na dotazujícím se, aby žádost na takto získanou lokalitu (další proxy server nebo volaný UA) poslal.
3. Registrar server přijímá registrační žádosti od UA a aktualizuje podle nich databázi koncových zařízení (location service), které jsou v rámci domény spravovány.

Jakkoliv jsou tyto servery definovány odděleně, v praxi se často jedná o jednu aplikaci, která přijímá registrace koncových uzlů a podle konfigurace se chová zároveň buď jako proxy nebo redirect server.
O komunikaci mezi komponenty SIP architektury jsem se zatím zmínil pouze v obecné rovině. Agenti a servery si posílají požadavky pomocí takzvaných metod. Jedná se o zprávy v textovém formátu a šesti základními metodami jsou

- INVITE - slouží k žádosti o sestavení spojení
- ACK - potvrzení INVITE finálním příjemcem zprávy (volaným)
- BYE - ukončení spojení
- CANCEL - ukončení nesestaveného spojení
- REGISTER - registrace UA
- OPTIONS - dotaz na možnosti a schopnosti serveru

Spojení je navazováno pomocí procedury, kdy klient A, který chce zavolat klientu B, vyšle požadavek INVITE (1). Tento požadavek obsahuje popis nabízeného spojení. Pokud dorazí zpráva v pořádku a byla pochopena, pošle klient B odpověď ve smyslu vyzváněcího tónu. Za předpokladu, že klient B se rozhodl hovor přijmout, vysílá zpět odpověď OK (2) s návratovým kódem 200. Odpověď 200 OK obsahuje IP adresu a číslo portu, na kterém bude klient B očekávat data od klienta A. Pro potvrzení navázání spojení zaslána zpráva ACK (3) klientem A protistraně, která potvrzuje přijetí odpovědi od klienta B. Nyní je spojení kompletní a může dojít k přenosu dat a rozhovoru. Až některý z účastníků hovor

ukončit, pošle protistraně požadavek BYE (4), který bude potvrzen odpovědí 200 OK (5).[9, 12]



Obrázek 10: Ukázka navazování spojení [12]

Odpovědi na SIP metody jsou zprávy uvedené číselným kódem. Systém kódů je převzat z HTTP protokolu. Např. SIPovská odpověď "404 Not Found" je velmi podobná té, které se objeví na web prohlížeči při přístupu na neexistující stránku. Číselné kódy odpovědí jsou členěny do šesti skupin: [9, 12]

- 1XX - informační zprávy (např. "100 Trying", "180 Ringing")
- 2XX - úspěšné ukončení žádosti ("200 OK")
- 3XX - přesměrování, dotaz je třeba směřovat jinam ("302 Moved Temporarily", "305 Use Proxy")
- 4XX - chyba, dotaz by se neměl ve stejné podobě opakovat ("403 Forbidden")
- 5XX - chyba na serveru ("500 Server Internal Error", "501 Not Implemented")
- 6XX - globální selhání ("606 Not Acceptable")

3.2 Asterisk

Oficiálně je Asterisk kombinací open source hybrid TDM a packet voice PBX, jedná se tedy o IVR (Interactive Voice Response) platformu s funkčností Automatic Call Distribution (ACD). IVR je zkratka pro interaktivní automatický odpovídač, který je jak z názvu vypovídá ovládán tónovou volbou (DTMF) nebo hlasem. Základní jednoduché IVR obsahuje systém hlasové pošty, který volajícímu nabídne například, uložení vzkazu v hlasové schránce nebo přepojení příchozího hovoru do cílové destinace. Komplikovanější IVR systémy umožňují volajícím procházet informačním seznamem a umožnit různé služby, jako sdělení zůstatku na účtu, informace o službách, vystavení objednávky a mnoho jiných služeb. Zkratka ACD znamená systém automatického rozdělování hovorů podle předem stanovených pravidel, například číslo volajícího, časové podmínky.

V porovnání s jinými systémy jde dle neoficiálních informací o jedno z nejlepších, flexibilních a rozšířených řešení v oblasti VoIP telefonie neboli integrovaného telekomunikačního softwaru. V celku jde tedy o kompletní open source softwarovou PBX (ústřednu), operující na jádrové platformě Unix, která poskytuje všechny vlastnosti, které

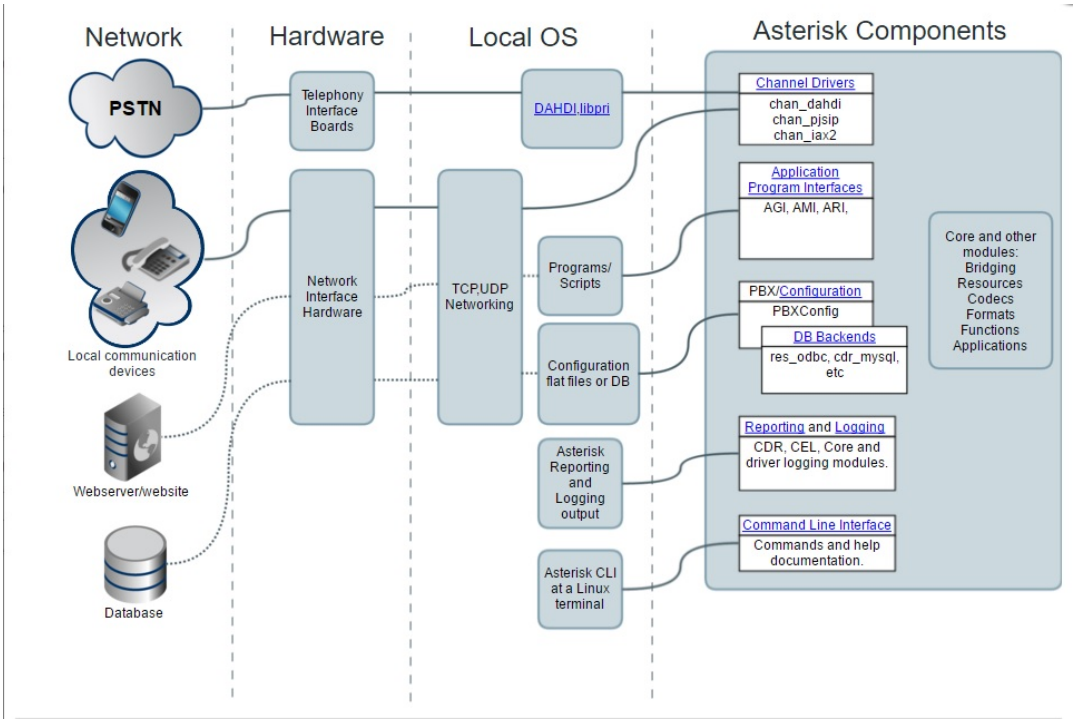
se od ústředny očekávají. Tato softwarová ústředna je obecnou distribucí open source, která splňuje podmínkami GNU (General Public Licence). Celý systém je uzpůsoben a navržen tak, aby vytvářel rozhraní pro telefonní hardware a software při použití libovolné telefonní aplikaci nebo přístroje.

Asterisk je plně konfigurovatelná ústředna, bez jakéhokoliv omezení patřící společnosti Digium, MarkSpencer. Asterisk lze zprovoznit na všech typech operačního systému Linux, Mac OS či BSD. Od svého prvotního vzniku bylo do světa vypuštěno několik verzí Asteriskod 1.2 až po poslední 13. Zprovoznění ústředny v plném provozu je možné po přidání několika softwarových balíčků řešící kompatibility protokolů nebo analogového přenosu dat, ale i bez těchto balíčků lze systém okamžitě použít, například pro místní LAN síť. Celková prvotní konfigurace je otázkou několika chvil. Po prvotní implemtaci systému, lze následně dokonfigurovat několik rozšiřujících funkce, hlasový záznamník, konferenční hovory, držení hovorů, předávání hovorů, hlasové pošty nebo také automatických odpovědí. [7, 14, 15]

Asterisk může být mimo jiné použit v těchto aplikacích:

- Pobočková ústředna včetně rozhraní do PSTN
- VoIP gateway pro různé protokoly (MGCP, SIP, IAX, H. 323) a rozhraní
- Voice mail služby s adresářem
- Interaktivní hlasový průvodce (IVR)
- Softswitch jako čistě softvérové řešení komunikačního serveru
- Konferenční server (funkce Meet me, konferenční místnosti)
- Pro šifrování spojení
- Pro překlad čísel
- Pro systém předplacených volání
- Systém pro směrování cestou nejnižších nákladů (LCR)
- Centrum volání (Call Center)
- Vzdálené kanceláře pro existující PBX
- TDM přes Ethernet

Aby bylo možné asterisk plně používat, je dobré mít představu vztahů funkcí a jejich závislostí pro pozdější konfiguraci. Níže na obrázku 11 je zjednodušené schéma určené k ilustraci vztahů některých hlavních složek v systému asterisk. [7, 14, 15]



Obrázek 11: Grafické znázornění vztahu funkcí v asterisk [14]

3.2.1 Architektura Asterisk

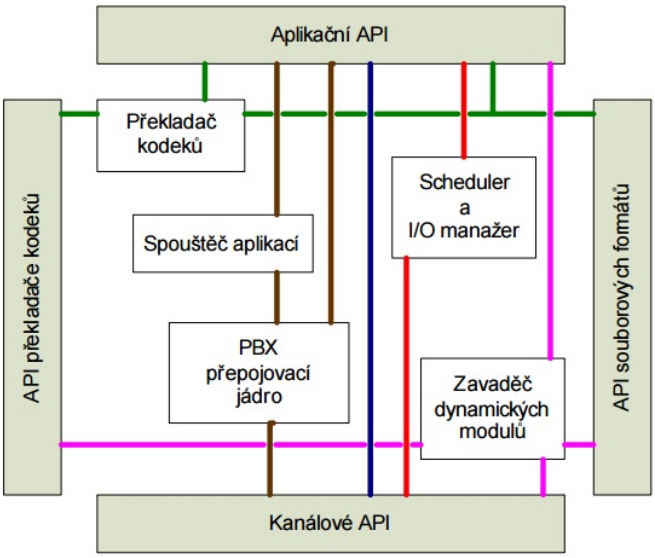
Architektura Asterisku je navržena jako jednotné prostředí pro propojení daných účastníků s možností rozšiřitelnosti o další specifické technologie. Z obrázku 12 je patrné, že se architektura Asterisk skládá z tzv. jádra a okolních API (Application programming interface). Asterisk má jádro, které může mít vliv na různé řídící moduly. Tyto moduly ovládají přilehlé kanály, které odkazují na jednotlivé funkcionality implementované do systému asterisk. Také se jimi připojují externí zařízení nebo programové balíčky, což usnadní celkovou komunikaci jádra a těchto balíčků.

Jádro asterisk systému je centrálním softwarovou řídící jednotkou. Jádro je základním prvkem, který poskytuje propojení jednotlivých modulů, funkcí a konfiguračních souborů. Při spuštění jádro prvotně načte konfigurační soubory a podle nich nastaví jednotlivé moduly a funkcionality pro chod asterisk ústředny. Okolo systému centrálního jádra PBX jsou definovány specifické API. Jádro ovládá vnitřní propojení PBX, myšleno specifické protokoly, kodeky a HW rozhraní telefonních aplikací. To dovolí v Asterisku použít každou vhodnou technologii a HW za účelem vykonávání základních funkcí - propojování HW a aplikací. [7, 14, 15]

Jádro Asterisku vnitřně ovládá tyto položky:

- PBX spojování (PBX Switching), hlavním cílem Asterisku je samozřejmě propojování v PBX, spojování mezi různými uživateli a automatizovanými úlohami. Přepojovací jádro transparentně spojuje příchozí volání na různých HW a SW rozhraních.
- Spouštěč aplikací (Application Launcher) spouští aplikace zajišťující služby jako jsou například hlasová pošta, přehrání souboru a výpis adresáře.
- Překladač kodeků (Codec Translator) používá moduly kodeků pro kódování a dekódování různých zvukových kompresních formátů používaných v telefonním prostředí. Množství dostupných kodeků je vhodné pro různorodé potřeby a docílení stavu rovnováhy mezi zvukovou kvalitou a použitou šířkou pásma.

- Scheduler a I/O manažer slouží k ovládání nízko úrovňových úloh a systémového řízení pro optimální výkon podle stavu zatížení.



Obrázek 12: Grafické znázornění jádra a API modulů v asterisk [7]

3.3 SIPp

SIPp je open source generátor SIP provozu pro telefonní VoIP ústředny. SIPp generátor umožnuje simulovat chování pro UAC i pro UAS a generovat signalizaci přenosu medií. Nejvíce se využívá pro generování provozu SIP zpráv a simulování zátěžových modelových situací, kterými se ověřuje zabezpečení ústředny a jejího provozu v různých situacích, které by mohly nastat. První zdrojový kód SIPp programu napsal Richard Gayraud a následně jej upravil Olivier Jacques. V poslední době je program stále vylepšován a to díky jeho velké oblibě, jeho všestrannosti a hlavně jednoduchosti v konfiguraci. Celý program se ovládá přes jednoduchou příkazovou řádku, kde lze ovládat všechny jeho funkce díky jasným příkazům. Postupem jeho vylepšování širokou komunitou uživatelů byla implementována možnost funkce importování XML souboru, kterým lze vytvořit komplikovanější komunikační schémata a CSV soubory obsahující rozšiřující data použitelné při psaní skriptů. Při spuštění programu se jako první načte CSV soubor, který obsahuje seznam uživatelů. S těmito údaji o uživatelích, pak odesílá požadavky přes testovaný systém do dalšího SIPp agenta, který běží na jiné stanici nebo může komunikovat i sám se sebou. Načtené údaje o uživatelích slouží agentům neboli SIPp generátoru pro vzájemnou komunikaci, při které jsou schopni distribuovaně měřit dobu, kterou scénář trval dle metodiky TRT. Načtení uživatelé jsou udržováni v několika skupinách, podle toho jestli jsou zaregistrováni nebo jsou součástí hovoru. Skupiny, do kterých jsou řazeny, jsou plně přizpůsobitelné v rámci použitého testovacího scénáře. Význam skupin má vliv na provozovaný scénář, který zajišťuje, aby se vše vykonávalo ve vhodném stavu, například aby se nezahazovaly hovory neregistrovaných uživatelů nebo přidržení hovoru v případě nedostupnosti koncové stanice. SIPp zaznamenává průběh spuštěných scénářů a jejich výsledků do nově vzniklých CSV souborů, které dále lze jakkoliv analyzovat. Program je vytvořen v programovacím jazyce C++, díky tomu si uživatelé mohou program upravit nebo vylepšovat dle potřeby. Po přidání programu do systému je součástí i základní schéma pro rychlé použití, ale ne vždy může být zcela

funkční, vše záleží na použitém systému nebo jiných nastaveních. V případě této situace je možnost upravit daný XML dokument a přizpůsobit jej přesně daným podmínkám.[16]

-r	Nastaví rychlost volání (volání za sekundu)
- rate_increase	Zvýší počet hovorů za -fd sekund Příklad: -rate_increase 10 -fd 10s= nárůst hovorů o 10 každých 10 sekund
-m	Zastavit test a ukončí program, když jsou hovory zpracované
-timeout	Ukončí SIPp program za daný čas. Příklad: -timeout 20s ukončí po 20 sekundách
-l	Nastaví maximální počet souběžných hovorů.Po dosažení tohoto limitu, sníží počet hovorů, dokud se nerovná maximálnímu počtu.
-s	Nastaví uživatelské jméno kontaktu. Výchozí hodnota je "service".
-ap	Nastavit heslo pro autentizaci
-Inf	Vloží hodnoty z externího souboru ve formátu CSV do scénářů při volání.
-d	Určuje délku "pauzu". Výchozí hodnota je 0 a výchozí jednotkou milisekund.
-sd	Uloží výchozí použitý scénář (použitelné pro UAC a UAS)
-sf	Načte vytvořený alternativní soubor xmlse scénářem.
-sn	Použití výchozího scénáře pro UAC a UAS

Tabulka 2: Přehled nejčastějších příkazů v sipp konfiguraci

3.4 Open source řešení vysoké dostupnosti

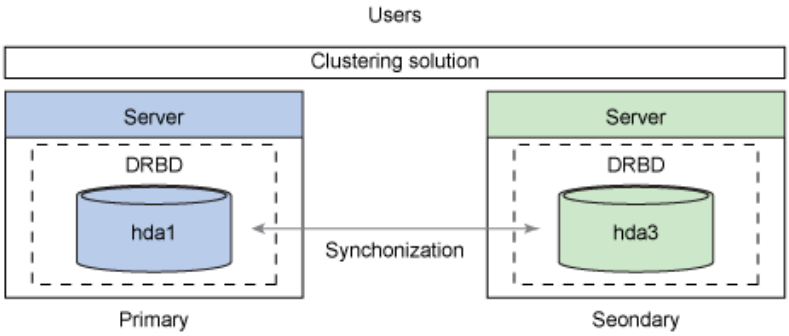
Pro řešení problematiky vysoké dostupnosti existuje celá velká řada různých řešení a to buď už ve formě boxů a před konfigurovanými zařízeními, která se propojí v síťové infrastruktuře, nebo čistě softwarová řešení, která je nutno nakonfigurovat dle naší situace. Dále tyto možnosti řešení vysoké dostupnosti je nutno rozdělit na komerční nebo nekomerční open source řešení. Zde je vybráno několik nejčastěji zmiňovaných systémů a řešení vysoké dostupnosti, která jsem rozdělil, právě dle komerčnosti. V této podkapitole jsou uvedena čistě nekomerční open source řešení, která jsou používana pro zajištění vysoké dostupnosti.

3.4.1 DRBD

DRBD systém je vlastně (Distributed Replicated Block Device) linuxová softwarová komponenta, která usnadňují výměnu a sdílení uložených systémových dat díky síťovému zrcadlení. DRBD umožňuje udržet konzistenci dat mezi několika systémy v síti, také zajišťuje vysokou dostupnost linuxových systémů a dat. Tento linuxový software je šířený pod svobodnou licencí GNU. DRBD lze použít se všemi běžnými linuxovými distribucemi pro synchronní replikaci uložených dat mezi pasivními systémy a aktivním systémem. Data lze číst a zapisovat na oba systémy současně. Systém může pracovat ve spojení s řídícím a monitorovacím programem heartbeat. DRBD tvoří logické vrstvy blokového zařízení, která jsou stejně veliká. Do tohoto logického oddílu lze zařadit jakékoliv adresáře nebo složky, jako jsou například telefonní záznamy nebo jiná důležitá data. Zápis je prováděn do primárního uzlu, odkud je dále přenášen do blokových zařízení nižší úrovně a rozšířen dále do sekundárních uzlů. Sekundární uzel následně přenáší data na odpovídající bloková zařízení nižší úrovně, jako tomu je u primárního uzlu. V případě selhání primárního uzlu je proces řízení clusterů automaticky přehodnocen na sekundární uzel a ten se stane v danou chvíli primárním uzlem. Tento přechod může vyžadovat následné ověření integrity souborového systému. Po návratu primárního uzlu do online stavu, se

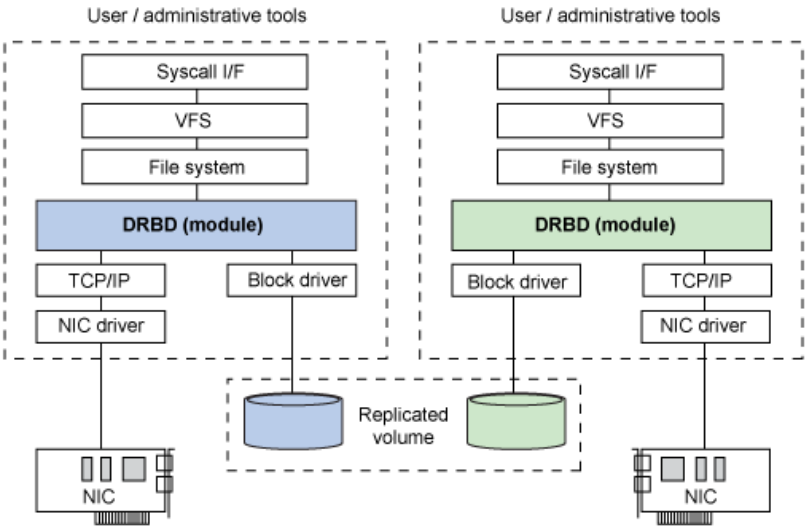
uzel stane opět primárním a navýší svoji úroveň, ale ne vždy nebo se stane sekundárním uzlem, ten je následně resynchronizován z tehdejšího sekundárního, nyní primárního uzlu. Synchronizace DRBD uzlů se provede jen u těch bloků, které byly změněny v době výpadku. [19, 20]

Na obrázku 13. je přehled o DRBD v rozložení dvou samostatných serverů, které poskytují nezávislé paměťové zdroje. Jeden ze serverů je definována jako primární a druhý jako sekundární. Uživatelé přistupují k DRBD blokovému zařízení jako k tradičnímu místnímu blokovému zařízení nebo jako Storage Area Network nebo síťově připojený disk.



Obrázek 13: Grafické znázornění rozložení DRBD serverů [19]

DRBD je rozdělen do dvou samostatných částí: modulu jádra, který implementuje chování a sadu pro správu uživatelského prostoru aplikací používaných ke správě DRBD disků, znázornění na obrázku 14. Modul jádra obsahuje ovladač pro virtuální blokové zařízení, které je replikováno mezi lokální disky a vzdálené disk v síti. Systém DRBD implementuje rozhraní blokového ovladače pro správu a konfiguraci disku, ale i síťový ovladač, jehož koncový bod je definován IP adresou a číslem portu. [20]



Obrázek 14: Grafické znázornění architektůry v linuxu [19]

3.4.2 Systém Ucarp

CARP je v podstatě (Common Address Redundancy Protocol) a je to síťový multicast protokol, který umožňuje několika počítačům ve stejné síti sdílet jednu nebo více

virtuálních IP adres. Hlavním účelem tohoto protokolu je poskytnout a zajistit převzetí služeb při selhání hlavního komunikačního bodu. Někdy se dá i použít pro load balancicg služeb. Princip je v seskupení několika fyzikálních systémů společně pod jednou virtuální adresou, z nichž je jeden prvek neboli systém jako primární a reaguje na všechny pakety učené pro danou podruženou skupinu systému. V případě selhání nebo přechodu master zařízení do stavu offline začnou ostatní systémy ve skupině poskytovat virtuální adresu namísto hlavního. Pokud je ve skupině více prvků, tak jsou zařízení jasně označena stupněm v pořadí zálohy. Po obnovení předchozího zařízení se s něj stává záložní s vyšší prioritou. Aby byla zajištěna daná stabilita, musí každé zařízení zařazené do skupiny mít vlastní soukromou adresu. Na základě této adresy jednotlivá zařízení komunikují mezi sebou a ověřují svojí dostupnost. V případě neověření dostupnosti přebírá záložní systém virtuální adresu. [21]

3.4.3 Systém gluster-fs

Glusterfs je škálovatelný distribuovaný síťový souborový systém určený pro velká datová úložiště a je dostupný snad v každé distribuci linuxu. Lze dynamicky zvětšovat, nebo zmenšovat jeho kapacitu, základním prvkem jsou zde souborové systémy vzájemně komunikujících strojů. Pomocí glusteru je možné vytvářet různé disky rozprostřené přes různé stroje. Základním datovým prvkem pro systém glusterfs je cihla (brick), jedná se o nejmenší jednotku, podobně jako je partition. Jedeno zařízení může exportovat více bricků, disk neboli volume se vždy rozkládá přes jeden nebo více bricků. Gluster podporuje následující rozložení, která je možné zprovoznit.

- Distributed volume – volume je rozloženo přes skupinu bricků. Pro každý soubor platí, že jeho data existují právě na jednom bricku. Toto rozložení použijeme v okamžiku, kdy chceme výhradně škálovat.
- Replicated volume – volume je replikováno ve více briccích a každý soubor se nachází ve všech briccích. Toto rozložení použijeme, je-li prioritou High-Availability.
- Stripped volume – volume je rozloženo přes skupinu bricků, přičemž rozdíl s Distributed volume, může být jeden soubor uložen napříč bricky. Toto rozložení použijeme, je-li třeba ukládat soubory větší, než jsou velikosti disků na strojích.
- Distributed Stripped volume – je kombinace, ve které je kladen důraz na výkon a na obrovské soubory.
- Distributed Replicated volume – kombinace, jedná se v podstatě o RAID 10.
- Geo-replication – asynchronní jednosměrná replikace. Cílový stroj nemusí být připojen do clusteru.

Glusterfs obsahuje klienta i server komponentu. Servery jsou obvykle nasazovány jako skladovací cihly, kde na každém serveru běží glusterfs démon, který běží na pozadí a posílá data na synchronizaci klientům. Glusterfs klient se připojuje k serveru s vlastním protokolem přes TCP/IP a vytváří kompozitní virtuální svazky ze serveru. [22, 23]

3.4.4 Systém heartbeat

Heartbeat je softwarový démon, který poskytuje periodický signál generovaný hardwarem nebo softwarem pro indikaci normálního provozu. Obvykle je tento signál odeslán mezi zařízeními v pravidelných intervalech a to v řádu sekund. To umožňuje klientům vědět o přítomnosti nebo nepřítomnosti druhého zařízení. Pokud protistrana neobdrží po určitou dobu obvykle několik signálových intervalů, předpokládá se, že došlo k selhání. Pro správnou funkčnost se často kombinuje se správou clusterů, která zajišťuje

spouštění a zastavování služeb jako jsou webové servery, virtuální IP adresa. V systému Heartbeat uzly komunikují prostřednictvím výměny paketů, zvané heartbeat signál, povětšinou dvakrát za sekundu. Je-li Heartbeat používán v systému s více uzly, jeden stroj je určen jako primární uzel a druhý jako sekundární uzel. V případě, že primární uzel selže nebo vyžaduje resetování, sekundární uzel může převzít hlavní roli pro poskytování služeb přiřazených k primárnímu uzlu. Heartbeat je obyčejně dodáván se softwarovou komponentou s názvem distribuovaného replikovaného blokového zařízení (DRBD), což usnadňuje výměnu sdílených úložných dat v síťovém zrcadlení. [24, 25]

3.4.5 Systém pacemaker

Pacemaker je software pro vysokou dostupnost šířený pod open source licencí. Tento software byl součástí Linux-HA projektu a byl ze začátku vyvíjen podobně jako předchozí systém, pak ale byl oddělen do samostatného projektu. Pacemaker dosahuje maximální vysoké dostupnost svých služeb clusterů pomocí detekce a zotavení ze záložních uzlů a to na základě detekce poruch a stavu zařízení. Pro správnou funkčnost se často kombinuje se správou clusterů, která zajišťuje spouštění a zastavování služeb jako jsou webové servery, virtuální IP adresy. Je-li pacemaker používán v systému s více uzly, jeden stroj je určen jako primární uzel a druhý jako sekundární uzel. V případě, že primární uzel selže nebo vyžaduje resetování, sekundární uzel může převzít hlavní roli pro poskytování služeb přiřazených k primárnímu uzlu. [26]

3.4.6 Elastix

Elastix je distribuce svobodného softwaru, který integruje různé komunikační technologie v jednom balíčku, jako jsou PBX, fax, e-mail, kalendář, databáze. Všechny tyto balíčky jsou integrovány a ovládány přes webové rozhraní a zahrnuje funkce, jako je například call centra s prediktivním vytáčením. Funkčnost Elastix je založen na open source projektech, jako je Asterisk , FreePBX , HylaFax , Openfire a Postfix. Díky těmto různým balíčkům a přidaným komponentám lze na jednom stroji nakonfigurovat celou řadu služeb. Díky možnostem přidávání balíčku je zde také implementován balíček pro vysokou dostupnost. V základním sestavení bez žádné další podpory nebo zařízení je zde použita synchronizace dat pomocí DRBD systému a monitoring stavu zařízení, který zajišťuje heartbeat systém. Elastix poskytuje mnoho jiných řešení, jako jsou box-to-box řešení, která v sobě obsahují další propracovanější systémy, ale tento způsob řešení již spadá mimo kategorii open source. [33]

3.4.7 Flip1405

Flip1405 je další z mnoha možných řešení pro zajištění vysoké dostupnosti. Tento systém vznikl jako open source skript, který byl a je, dle potřeb, stále upravován pro různá možná řešení. Flip1405 byl prve při svém vzniku používán jen pro monitorování stavu zařízeních a poskytování přenosné virtuální IP adresy, kdy na poskytovanou virtuální adresu jsou přiřazeny všechny poskytované služby zařízením. V případě zjištění, že primární zařízení je mimo provoz, automaticky převezme veškerý provoz a virtuální adresu záložní zařízení. Až po několika úpravách byla do skriptu přidána služba pro synchronizaci dat. Tuto synchronizaci mezi zařízeními zajišťuje rsync, kdy jsou zálohovány pouze soubory, které byly změněny nebo aktualizovány.

3.5 Komerční řešení vysoké dostupnosti

Po kapitole nekomerčních systémů pro zajištění vysoké dostupnosti, je také důležité přiblížit, několik řešení z komerční časti. V této podkapitole je uvedeno několik

komerčních řešení, ať už od známých firem nebo i těch méně známých, ale zaměřených na stejnou problematiku. Dle dostupných informací o různých systémech jsem udělal průřez vlastností jednotlivých systémů pro další možné porovnání vlastností nebo schopností jednotlivých systémů a zařízení.

3.5.1 FreePBX

FreePBX HA je vyvinuta ke komerčním účelům a řeší vysokou dostupnost pro VoIP ústředny. FreePBXv sobě obsahuje DRDB cluster pro správu dat a pacemaker pro monitorování stavu a dostupnosti zařízení. To umožňuje automatické zrcadlení veškerých potřebných dat a zajištění vysoké dostupnosti v případě výpadku mezi dvěma freePBX systémy. Všechny telefony a jiná zařízení jsou registrovány na plovoucí virtuální IP adrese, takže v případě výpadku a zajištění vysoké dostupnosti mezi systémy budou adresy transparentní a nedojde k žádným změnám. Po zotavení původního primárního serveru PBX, lze celu komunikaci přepnout zpět do primárního uzlu a to během několika sekund. Nastavení a konfigurace freePBX systému se provádí v freePBX GUI rozhraní, které lze použít i jako on-line nástroj pro správu a v něm vše řídit nebo snadno přepínat mezi uzly.[30]

3.5.2 Haast

Haast je softwarový balíček, který vytváří high availability řešení. Haast může detekovat celou řadu poruch na jednom asterisk serveru a automaticky převede kontrolu jinému serveru, což vede k telefonnímu prostředí s minimálními časy výpadku. Jedná se o softwarový balíček s možností přepnutí během několika sekund. Vestavěné inteligentní řízení sítě umožňuje sdílení jediné IP adresy, která je sdílena mezi jednotlivými vrstevníky, takže telefonní klienti se automaticky připojí k aktivnímu asterisk serveru bez zaznamenání změny. Vestavěná replikace a synchronizace mezi servery, také snižuje nároky na údržbu služeb a podpůrné činnosti. Haast obsahuje širokou škálu senzorů pro detekci selhání vrstevníka, jeho hardwaru, sítě nebo připojení. Tyto senzory přispívají k celkovému zdraví a zajištění vysoké dostupnosti, která umožňuje automaticky přijmout opravná opatření, upozornění správce, nebo v nejhorším případě řádné spuštění do pohotovostního spojení. Haast nepotřebuje žádné jiné systémy pro high availability nebo kontrolu stavu systému, ani nevyžaduje sdílení na úrovni diskových zařízení. Ve skutečnosti Haast nesdílí žádný hardware, ani logické zařízení mezi vrstevníky, čím se minimalizuje možnost selhání.

Systém Haast je postaven na čtyřech základních technologiích: snímač zdroje, synchronizace zdroje, řadič clusteru a vzájemného propojení. Senzor zdroje je odpovědný za sledování zdravotního stavu v místním počítači neboli spoji, přepínáním různých rozhraní nebo subsystémů pro určení, jestli je místní spojení aktivní nebo je neaktivní a je vyžadováno přepnutí služeb. Stav místního spojení je sledován jako zdravotní stav, kdy tento stav dosáhne kritické úrovně, dojde k upozornění administrátora a vybraní následné akce. Toto celé proběhne automaticky.

Synchronizace zdroje je zodpovědná za replikaci změn v souborech, adresáře, tabulky a databáze z aktivního spojení do pohotovostního neboli záložního zařízení. Synchronizace zdroje se automaticky iniciuje v určených intervalech a provede úpravu nebo synchronizaci dat do pohotovostních spojení. Synchronizace probíhá vždy z aktivního do pohotovostního spojení, a to pouze v případě kdy zdravotní stav obou zařízeních je správný.

Řadič clusteru je zodpovědný za spouštění a zastavování asterisk ústředny na místním spojení. Reaguje s čidlem zdroje, se synchronizací zdroje a vzájemným propojením. Řadič clusteru také provádí spuštění nebo restartování asterisk ústředny, což umožňuje značné přizpůsobení a flexibilitu při činnostech s asterisk ústřednou.

Vzájemné propojení je zodpovědné za veškerou komunikaci Haast systému mezi vrstevníky, včetně zdravotního stavu. Koordinuje převzetí služeb při selhání, posílání zpráv, sledování vzdálených senzorů. V případě, že spojení přejde z nějakého důvodu do režimu offline, pak přežívající spojení předpokládá, že druhé spojení selhalo a pokusí se převzít veškeré operace jako aktivní spojení. [27]

- Asterisk kompatibilita
- Senzory monitorování stavu zařízení
- Synchronizace spojení
- Sdílené IP adresa
- Autonomní Peers
- Nezávislost na vzdálených spojích
- Rychlá obnova při výpadku
- Inteligentní záloha
- Nezávislost na programech a příslušenstvých
- Šifrované spojení komunikace

3.5.3 Sark-HA

Nabízí high availability pro asterisk ústřednu prostřednictvím řešení tzv.out-of-the box. Sarkha funguje na základě osvědčeného standartu založeného na linux HA technologii clusteru, který provozuje dva asterisk servery. Jeden aktivní, nebo primární server a druhý server pasivní, který je připraven k použití v případě selhání primárního serveru. V případě poruchy asterisk nebo linux aktivního serveru, záložní server automaticky převezme kontrolu nad všemi prostředky a pokračuje v provozu jako primární do té doby, dokud není primární server schopen opět připraven převzít roli primárního serveru. Čas potřebný k převzetí vysoké dostupnosti se může lišit v závislosti na tom, jak jsou nastaveny parametry monitorování stavu primárního serveru, je ale docela běžné, že velké systémy jakou jsou například ústředny obsahující 250 - 300 telefonů, mohou trvat méně než 30 sekund k celkovému převzetí služeb. To je doba potřebná k přesunu dat a hovorů z primárního serveru a umístit odchozí hovory na pohotovostní záložní server.

Sark 200 je kompletní PBX řešení v podobě mini serveru, který obsahuje ARM procesor s nízkou spotřebou v malé velikosti. Reálný čas přepnutí systému v případě výpadku u tohoto typu, trvá méně než 20 sekund. Servery jsou zapojeny v synchronizaci s použitím rsync metody zálohy dat mezi ústřednami. Sail je rychlá, lehká asterisk ústředna s webovým prohlížečem pro konfiguraci komponent systému. Lze jej nasadit k použití do malých a středních podniků. [28]

3.5.4 Q-Suite

Q-Suite je výkonný software pro ústředny, který poskytuje komplexní řešení s rychlým přepnutím systémů. Vyvažování zátěže umožňuje Q-Suite systému používat více asterisk serverů v clusteru a zvládnout vysokou dostupnost souběžných hovoru vstupujících do ústředny. Tato funkce umožňuje Q-Suite škálovat hovory na více serverů asterisk a tím zvládnout velké výkyvy zatížení, nebo hodně vytížených kontaktního center s velkým množstvím operací, kde nabízí pokročilé řazení hovorů a jejich následné směrování. Q-Suite poskytuje softwarový mechanismus pro monitorování stavu zařízení a vytváření spojení vysoké dostupnosti. Toto vylepšuje SIP proxy, které je součástí řešení a dále posiluje schopnost Q-Suite poskytovat vysokou dostupnost s velkou kapacitou hovorů a pro přenos souběžného volání pro kontaktní centra ústředen, které chtějí využít asterisk jako jejich telefonní platformu. Vestavěné nástroje a rozsáhlé sady funkcí dělají Q-Suite ideálním kandidátem pro kontaktní centra založená na asterisk systému. Q-Suite s

vyvažováním zátěže zvládne velmi vysoký počet souběžných hovoru. Q-Suite ovládá a řídí směrování a distribuci, pomocí funkcionalit založených na směrování hovorů. [29]

3.5.5 SERVERware

SERVERware je dalším řešením pro zajištění vysoké dostupnosti, v tomto případě se jedná o nejekonomičtější způsob, jak začít poskytovat IP služby z jednoho serveru. Toto dané řešení obsahuje všechny potřebné součásti, aby byly zajištěny poskytovatelům nabízené služby. SERVERware edition může být nasazen také na místní hardwarové zařízení pro zajištění redundance vysoké dostupnosti v případě, kdyby mělo primární zařízení selhat. Zajišťuje použití zařízení pro zrcadlení dat a poskytuje rychlé převzetí služeb v době výpadku. Síť systému může obsahovat až 256 serverů vytvářejících rozsáhlou podsíť virtuálních privátních serverů, ze kterých je zajišťována platforma poskytovaných služeb IP adresace. To umožňuje poskytovatelům vybraných služeb nabízet komplexní, flexibilní a škálovatelné IP služby, jako je pošta, web, hostované PBX ústředny. [31]

3.5.6 Cisco CUBE

Cisco Unified Border Element (CUBE) poskytuje vysokou dostupnost (HA) a to díky řešení box-to-box zařízeních. Jedná se o směrovače řady cisco na bázi Hot Standby Routing Protocol (HSRP). Technologie HSRP poskytuje vysokou dostupnost sítě tím, že směruje IP provoz všech počítačů v sítí bez ohledu na dostupnost každého jednotlivého směrovače v dané síti. HSRP se používá v sítích skupin směrovačů pro výběr aktivního směrovače a záložního směrovače. HSRP monitoruje vnitřní a vnější rozhraní, pokud zaznamená, že rozhraní spadne, celý přístroj je považován za nedostupný a mimo provoz, čímž se záložní zařízení ve skupině stane aktivní a přebírá odpovědnost hlavního routeru. Box-to-box redundance zařízeních se používá protokol HSRP, aby se utvořila HSRP dvojice aktivního a pohotovostního směrovače. Aktivní a pohotovostní dvojice zařízení sdílejí stejnou virtuální IP adresu a neustále si mezi sebou vyměňují stavové zprávy. Tyto zprávy obsahují informace o CUBE relacích a kontrolních bodech tvořené aktivními a pohotovostními dvojicemi směrovačů. To umožňuje záložnímu routeru, aby mohl okamžitě převzít všechnu odpovědnost zpracování hovorů v případě, že hlavní router by byl mimo provoz a buď plánovaně, nebo z neplánovaných důvodů. Box-to-box redundance zařízeních implementuje CUBE vysokou dostupnost a podporuje zachování médií přes HSRP protokol přepnutím SIP hovorů. Tato technologie je podporováno od cisco operačního softwaru verze 15.1.2T. Zachování signalizace hovoru během výpadku je podporován v nejnovější verzi cisco operačního softwaru. [32]

4 Použité topologie a systémy

Konfigurace jednotlivých systémů pro vysokou dostupnost VoIP ústředny je realizována hlavně ve virtuálních prostředích virtual boxu. Zde jsme postupně vytvořili jednotlivé servery s různými systémy z kapitoly 3.4. Ke každému serveru, který jsme následně nainstalovali, jsme také museli vytvořit kopii tohoto serveru, která slouží jako záložní server v případě výpadku hlavního serveru. Na všech serverech, kde se použilo některé z řešení vysoké dostupnosti, se nainstalovala i asterisk VoIP ústředna, kteá se nakonfigurovala dle daných požadavků. K těmto serverům se ještě nainstaloval server pro generování SIP zpráv a monitorování provozu sítě. Pro jednotlivé servery se použil jako operační systém centos a ubuntu. Operační systém centos je použit u telefonních ústředen a to hlavně kvůli implementovaným balíčkům a menší náročnosti pro vytvořené virtuální prostředí. Pro generování a monitorování sítě se použil operační systém ubuntu, který je přehlednější a graficky orientovaný. Ke generování SIP zpráv se použil SIPp generátor, jehož konfigurace bude příblížena později a k odchytávání paketů v síti se použil tcpdump. Rozšiřující možností pro následné testování vysoké dostupnosti se zvolil vnější SIPp generátor implementovaný do raspberry pi zařízení, do které se nainstaloval operační systém raspbian. V případě nedostatečného výkonu raspberry pi, bude SIPp implementován do nově vzniklého virtuálního systému. Toto jsme zvolili pro účely testování vysoké dostupnosti a možných rozdílů v použitých SIP zprávách. K vnějšímu monitorování sítě se pak použil program na odchytávání paketů wireshark.

4.1 Virtualbox

VirtualBox je softwarový virtualizační nástroj distribuovaný společností oracle. Díky tomuto softwaru jsem mohl bez problémů nainstalovat rozdílné operační systémy pod již běžící operační systém na jednom stroji, bez fyzického zásahu do tohoto stroje. Ovládání virtualboxu je velice intuitivní a bez problémů se s ním naučí pracovat kdokoliv. Virtuální stroj se nám vždy spustí v novém okně pod již běžícím systémem. Přidání virtuálního stroje do virtualboxu je v doprovodu průvodce, který obsahuje všechny důležité nastavení, jako je vytvoření disku, určení velikosti operační paměti a vybrání systému, který budeme chtít virtualizovat. Dále, není důležité nějak hlouběji představovat virtualbox a nebo zacházení s ním. V našem případě jsme do virtualboxu přidali zařízení pro instalaci ústředen s odlišnými řešeními vysoké dostupnosti a virtuální stroj pro generování SIP zpráv. Virtualizovaný stroj nijak nerozeznáme od běžného systému, máme přístup ke všem periferiím přiřazených k virtuálnímu stroji při jeho prvotní konfiguraci. [36]

4.1.1 Virtualizace

Virtualizace umožňuje, aby na jednom fyzickém zařízení, neboli na jednom hardware běželo více oddělených zařízení s vlastním operačním systémem. Fyzický stroj každému takovému virtuálnímu stroji emuluje virtuální hardware jako je procesor, paměť, disk, síťová karta, mechaniky, periferní zařízení a další. Virtualizace umožňuje souběžný běh několika virtuálních strojů vedle sebe paralelně na jednom fyzickém zařízení. Zároveň tyto operační systémy běží na od sebe izolovaných, takže se navzájem nijak neovlivňují nebo neruší. [37]

4.2 Tcpdump

Tcpdump je paketový analyzátor, který pracuje v příkazové řádce. Používá se hlavně na analýzu a odchytávání komunikace v počítačových sítích s různými protokoly, zejména

TCP/IP, k níž je počítač připojen nebo jeho síťové rozhraní. Tcpdump je šířen pod BSD licencí, což se jedná o svobodný software k volnému užívání. Program tcpdump lze také použít pro analyzování chování počítačové sítí, jejího výkonu anebo aplikací, které generují nebo přijímají nějaké síťové pakety. Nejčastěji, ho ale lze použít pro analýzu síťové topologie, pro kontrolu zda je správně nastaveno směrování, a tím rychle a snadno vyřešit některé vzniklé problémy. Lze jej také použít pro mnohem specializovanější účely zachytávání komunikace jiných uživatelů sítě. V našem případě jsme jej použili pro zachytávání komunikace SIP zpráv. Pro úplnost jen některé základní parametry, se kterými se ihned zorientujeme v jeho použití. [35]

```
$ - sudo tcpdump -i eth0
```
Začne na výstup vypisovat jednotlivé přijaté pakety, u kterých se vždy pokusí zjistit doménové jméno zdrojové a cílové adresy.

```
$ - sudo tcpdump -i eth0–n
```
Vypisuje každý paket, který prochází přes zadaný síťový port.

```
$ - sudo tcpdump -i eth0 -n dst host 192.168.1.135 port 22
```
Vypisovat pouze adresy, které mají danou cílovou IP adresu na daném portu. Cílová adresa je značena dst nebo zdrojová adresa značená src.

```
$ - sudo tcpdump -i eth0 tcp -w soubor.pcap
```
V tomto případě se použilo odchytávání paketů na rozhraní eth0 a následné uložení do souboru.

4.3 Wireshark

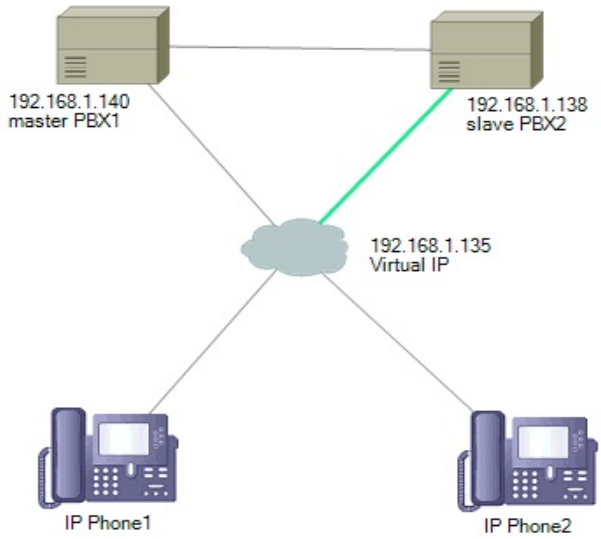
Wireshark je protokolový analyzér a paketový sniffer. Wireshark je dosti funkcích a použití hodně podobný paketovému analyzeru tcpdump, ale obsahuje mnohem větší počet analyzerů komunikačních protokolů a formátů, díky čemuž dokáže odchytit mnohem větší část komunikace. Další rozdílností je grafické uživatelské rozhraní a mnoho možností uspořádání a filtrování zobrazených informací. Jeho nejčastější použití je analýza a ladění problémů v počítačových sítích, ladění a vývoj softwaru, vývoj komunikačních protokolů a studium síťové komunikace, kde je jedním z nejdůležitějších pomocníků. V případě použití wiresharku se síťovou kartu přepne do promiskuitního režimu a díky čemu zachytává veškerou komunikaci na připojeném médiu. Tento software je multiplatformní a používající widgety pro implementaci uživatelského rozhraní a knihovnu pcap na zachytávání paketů. Data pro analýzu mohou být zachycena přímo ze síťového ethernet kabelu nebo načtena ze souboru. Zachytávat komunikaci můžeme z mnoha typů sítí, jako je ethernet, IEEE 802.11, PPP, a loopback. Výsledná zachycená data jsou následně buď uložena do souboru, kde ještě před uložením můžeme data programově upravit nebo překonvertovat, anebo data rovnou procházena a zkoumána v grafickém prostředí kde pro přehlednost aplikujeme odpovídající filtry. V případě této práce lze odchycená VoIP volání detailně prozkoumat a překontrolovat zdali dané směrování opravdu odpovídá naším představám.

Pro začátek a rychlou orientaci v programu wireshark je nutno říci, že Wireshark pracuje pouze pasivně, což znamená, že neodesílá žádná data z daného počítače. Po spuštění programu se objeví přede mnou přívětivé a přehledné prostředí s menu, ve kterém si vybereme položku (interface list), kde jsou vidět dostupné síťové prvky celého počítače, které můžu odchytávat. Vybereme si odpovídající síťové rozhraní a potvrzujícím i zároveň startovacím tlačítkem jednoduše spustíme odchytávání síťové

komunikace. Pro jasnou představu nyní spustíme vytáčení hovoru na místní VoIP ústřednu. Okamžitě se začne ukazovat, jak probíhá celá síťová komunikace s VoIP ústřednou v případě jednoho hovoru. V horní části okna programu se vypisují probíhající pakety. Pakety se standardně řadí podle času, který je ve stejném okně jako druhý z leva. Dále si zde můžeme všimnout IP adresy zdroje (Source), odkud pochází daný paket komunikace a IP adresy příjemce (Destination), kam tento paket komunikace je směrována. V další části je typ protokolu a základní informace o daném paketu. Pod tímto oknem je další větší okno, ve kterém se zobrazují podrobnější informace o daném paketu, který se vybral v předchozí nabídce. A v nejspodnějším okně celého programu jsou zobrazeny data, která jsou přímo poslána po síti. [34]

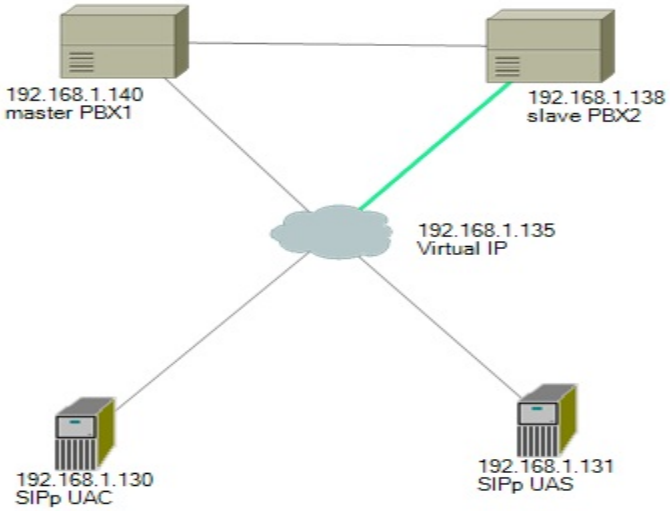
4.4 Použitá topologie

Při realizaci této práce je jako první bod návrh síťové topologie tak, aby byla zaručena funkčnost všech používaných systémů. Při návrhu síťové infrastruktury můžeme použít nepřeberné množství a typů zapojení, jak jsem přiblížil v kapitole 2.5.1, kde jsou použity jedny z nejčastějších realizací zapojení. Při návrhu jako takovém se v tomto případě vycházel hlavně z požadavků, co od dané sítě očekáváme a následně možnosti, kterými celé propojení můžeme realizovat. Po seznámení se s okolnostmi se zrealizovalo jedno ze základních propojení místních systémů. Zjednodušená prvotní ukázka sítě je na obrázku 15, kde je vidět použitá IP adresace a princip realizace.



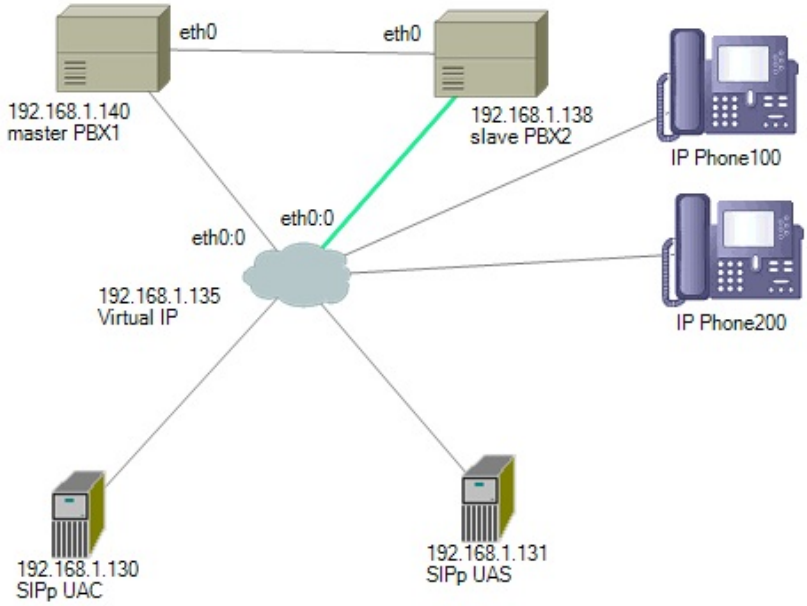
Obrázek 15: Základní síťová adresace vysoké dostupnosti

Na tomto obrázku číslo 15 je znázorněna základní síťová adresace mezi VoIP ústřednami a telefonními klienty, kde je využíváno sdílené virtuální IP adresy, která je použitá jako výchozí IP adresa VoIP ústředny. K této virtuální IP adrese se následně připojují a registrují uživatelé ústředny. Jsou použity dvě VoIP ústředny, z čehož jedna je jako hlavní a druhá je záložní, v případě selhání hlavní ústředny, aby byla zajištěna vysoká dostupnost. Jako hlavní je použita ústředna s adresou 192.168.1.140, na které je nainstalován a nakonfigurován asterisk systém a systém pro řešení vysoké dostupnosti, které jsem v průběhu moji práce měnil. V případě selhání hlavní ústředny je zde záložní s adresou 192.168.1.138, která komunikuje v pravidelných intervalech s hlavní ústřednou, aby si ověřila, že nedošlo k selhání a také kontroluje sdílená data, aby na obou ústřednách byla aktuální data. Virtuální adresu primárně poskytuje hlavní VoIP ústředna, a v případě selhání ji začne poskytovat záložní ústředna.



Obrázek 16: Síťová adresace s použitými SIPp generátory

Zde na obrázku číslo 16 je síťová adresace doplněna o SIPp generátory SIP zpráv. Tyto generátory SIP zpráv zajišťují stálé vytížení sítě, které simuluje reálný provoz ústředny. Generátor s IP adresou 192.168.1.130 je značen jako UAC strana, která dané SIP zprávy generuje. Výchozím bodem pro odesílání zpráv je použita virtuální IP adresa asterisk ústředny. Druhý generátor s IP adresou 192.168.1.131 označený jako UAS, která reaguje na příchozí žádosti od UAC strany a posílá ji odpovědi. Zdrojovou IP adresou pro příjem žádostí je opět použita virtuální adresa ústředny.



Obrázek 17: Síťová adresace kompletní topologie

Kompletní síťová adresa a návrh topologie sítě je vidět na obrázku číslo 17. Je zde dobře vidět znázornění přípojného bodu ústředny pro použité telefonní klienty i generující SIP zařízení. U použitých VoIP ústředen byla implementována pouze jedna síťová karta eth0, která má přidělenou pevnou IP adresu pro konfiguraci, ale také pro monitorování stavu zařízení a plovoucí adresu, značenou jako eth0:0 pro zajištění připojení telefonních klientů k ústředně.

5 Konfigurace a implementace jednotlivých řešení

5.1 Konfigurace síťové adresace

Pro bezproblémovou funkčnost systému je důležité nastavit statickou IP adresaci používaných zařízení. Zde je ukázáno jak nastavit statickou IP adresaci na systému centos. Pro lepší přehled je v tabulce číslo 3 přehled použité síťové adresace.

Jméno	Hostname	IP adresa	Netmask	Gateway	DNS1	DNS2
PBX_1	centos1	192.168.1.140	255.255.255.0	192.168.1.1	192.168.1.1	8.8.8.8
PBX_2	centos2	192.168.1.138	255.255.255.0	192.168.1.1	192.168.1.1	8.8.8.8
SIPp_1	UAC	192.168.1.130	255.255.255.0	192.168.1.1	192.168.1.1	8.8.8.8
SIPp_1	UAS	192.168.1.131	255.255.255.0	192.168.1.1	192.168.1.1	8.8.8.8
asterisk	asterisk	192.168.1.135	255.255.255.0	192.168.1.1	192.168.1.1	8.8.8.8

Tabulka 3: Přehled použité síťové adresace

Dále následuje přehled pro konfiguraci statického směrování. Celý výpis je v příloze. [Příloha 1: Konfigurace statického směrování.]

```
$ - nano /etc/sysconfig/network-scripts/ifcfg-eth0
```

Pro editaci a následnou konfiguraci rozhraní eth0, se použil nano editor, který je součástí každého unixového systému. Po vykonání daného příkazu se nám otevře editor, kam vložíme a opravíme následující původní konfiguraci. Následující konfigurace nám nastaví rozhraní eth0 dle potřebných požadavků.

```
$ - nano /etc/sysconfig/network

NETWORKING=yes
NETWORKING_IPV6=no
HOSTNAME=centos1
```

V tomto konfiguračním souboru zakážeme možnost použití IPv6 a nastavíme hostjméno systému.

```
$ - nano /etc/resolv.conf

nameserver 192.168.1.1
nameserver 8.8.8.8
```

nastavíme DNS servery, dle tabulky číslo 3, které jsou přidány v souboru resolv.conf.

V tomto případě je název hostitele centos1 a centos2, které nastavímv / etc / hosts:

```
$ - nano /etc/hosts

127.0.0.1  centos1
::1 localhost.localdomain localhost6 localhost
192.168.1.140   centos1
192.168.1.138   centos2
192.168.1.135   asterisk
```

Po těchto předchozích změnách v konfiguraci musíme provést restart síťové služby, aby se nám aktualizovalo nebo konfigurovalo, co jsme aplikovali. Pro restartování nám stačí, abychom provedli následující příkaz.

```
$ - /etc/init.d/network restart
```
Po zadání tohoto příkazu nám systém vypíše, zdali restartování bylo provedeno úspěšně. Pokud vše proběhlo správně, tak se konfigurace povedla a jen si ověříme, zdali tomu opravdu tak je. Pro zkontrolování síťové konfigurace nám postačí vypsat aktuální nastavení.

```
$ - ifconfig

eth0  Link encap:Ethernet  HWadr 08:00:27:AE:B8:05
      inet adr:192.168.1.140  Všesměr:192.168.1.255
      Maska:255.255.255.0
      inet6-adr: fe80::a00:27ff:feae:b805/64 Rozsah:Linka
      AKTIVOVÁNO VŠESMĚROVÉ_VYSÍLÁNÍ BĚŽÍ MULTICAST  MTU:1500
      Metrika:1
      RX packets:269 errors:0 dropped:0 overruns:0 frame:0  TX
      packets:369 errors:0 dropped:0 overruns:0 carrier:0
      kolizí:0 délka odchozí fronty:1000
      RX bytes:27696 (27.0 KiB)  TX bytes:33541 (32.7 KiB)
```

Jak je vidět vše je správně nakonfigurováno. Nyní si ještě ověřím hostjméno systému.
```
$ - hostname
centos1
```

5.2 Konfigurace ústředny Asterisk

Po vytvoření virtuálního stroje, v následujícím případě virtuální telefonní ústředny a nastavení veškerých prvotních konfigurací jako, byla statická IP adresa a nebo firewall, se nyní budeme věnovat instalaci a konfiguraci jednoho z nejdůležitějších systémů. Při instalaci asterisk VoIP ústředny jsme se rozhodli k nainstalování asterisk verze 11 a to z toho důvodu, zvolení stabilní a prověřené verze tohoto systému. Tuto verzi jsme si stáhli z oficiálního zdroje asterisk. Stažení asterisk systému se provede následujícím příkazem.

```
$ - wget http://downloads.asterisk.org/pub/telephony/ asterisk/
asterisk-11-current.tar.gz
```
Příkaz pro stažení asterisk systému.

```
$ - wget http://downloads.asterisk.org/pub/telephony/libpri/
libpri-1.4-current.tar.gz
$ - wget http://downloads.asterisk.org/pub/telephony/dahdi-
linux/dahdi-linux-current.tar.gz
```
Můžeme stáhnout i rozšiřující knihovnu pro asterisk. Libpri je knihovna, která umožňuje asterisk ústředně komunikovat s ISDN přípojkami. Vhodnou součástí je i následující knihovna.
DAHDI knihovna umožňuje asterisk ústředně komunikovat s analogovými a digitálními telefony a telefonními linkami, včetně připojení k veřejné telefonní síti nebo PSTN. V případě této práce se rozšiřující knihovny nepoužil, jelikož nebyla potřeba využití jejich vlastností.

```
$ - nano /etc/selinux/config
$ - cp asterisk-11-current.tar.gz /usr local/src
$ - cd /usr/local/src
```
Předtím než se pustíme do instalace, nastavíme pomocí editoru nano v souboru selinux položku selinux=disabled. Po stažení asterisk systému jej překopírujeme do systémové

složky, kam se následně pomocí příkazu cd přesuneme, abychom mohli pokračovat v konfiguraci.

```
$ - tar -zxvf asterisk-11-current.tar.gz
```

Přesun do systémové složky, kam jsme předtím přesunuli i stažený balíček, který nyní rozbalíme. Extrahujeme zdrojový kód z každého archivu pomocí příkazu tar. Tyto parametry – zxvf jsou nedílnou součástí příkazu tar a řeknou, co přesně chceme dělat s komprimovaným souborem. Volba „z“ říká systému, rozbalit soubor před pokračováním, volba „x“ říká extrahovat soubory z archivu, volba „v“ způsobí, že během extrahování souboru bude uživatel informovaný, co se přesně dělá. Vypíše jméno každého souboru v archívu a poslední volba „f“ informuje příkaz tar, že se extrahoval soubor ze souborového archivu.

```
$ - cd /usr/local/src/asterisk-11.X.Y
```

Nyní je čas zkompilovat a nainstalovat asterisk. Přesuneme se do nově vzniklého adresáře po extrahování, kde jsou zdrojové kódy asterisk ústředny.

```
$ - ./configure
```

Provede se kontrola, abychom jsme se ujistili, že je vše v pořádku a operační systém obsahuje všechny požadované balíčky. V tomto případě jsme museli ručně přidat chybějící balíčky, na které nás upozornil chybovou hláškou.

```
$ - cd /contrib/scripts
$ - ./install_prereq install
```

Jelikož chybových hlášek bylo více, tak se použil přiložený install_prereq skript, který dané problémy vyřešil a opravil. Tento skript je součástí asterisk složky, nalézt jej je možné v podadresáři contrib/scripts. Po doinstalování zbývajících balíčků a opravení chybových hlášek, opět provedeme příkaz configure. Nyní již asterisk nevypsal žádnou chybu a vypsal jen, co mám dále udělat.

```
$ - make menuSelect
```

Následující krok po úspěšné kontrole je sestavení asterisku. Než, ale provedeme sestavení, tak je dobré pomocí tohoto příkazu sestavit menu, ve kterém jsou vypsány jednotlivé moduly. V tomto menu jsou automaticky zvoleny základní moduly, ale lze plno modulů pomocí tohoto menu dopřidat nebo změnit. Překontrolovaný výběr modulů uložím a mohu již provést finální sestavení.

```
$ - make
```

Sestavování asterisku bude nějakou chvilku určitě trvat. Během sestavování lze vidět podrobný výpis činnosti asterisku. Po doběhnutí sestavení provedeme konečnou instalaci, což nás asterisk k tomu i sám vyzve.

```
$ - make install
```

Konečná instalace je už velice jednoduchá a je hned nainstalována. Doporučuje se po úspěšné instalaci přidat ještě ukázkové soubory. Ty nám pak pomohou s případnou konfigurací.

```
$ - make samples
```

Přidání ukázkových dat do konečné instalace asterisk ústředny.

```
$ - make config
```

Než dokončíme instalaci asterisku přidáme v posledním kroku inicializační skript. Tento skript se spustí při spuštění asterisk ústředny. Bude sledovat běžící službu asterisk a v případě, náhlé chyby restartuje nebo zastaví asterisk.

```
$ - make install-logrotate
```
A ještě přidáme skript, který bude zaznamenávat generované logy systému. Zajití automatické mazaní nepotřebných nebo už starých logů a zlepší vyhledávání logů.

```
$ - /etc/init.d/asterisk start
```
Spustíme službu asterisk.

```
$ - /etc/init.d/asterisk status
```
Pokud bylo vše správně provedené, tak si nyní ověříme stav asterisk ústředny. Výsledkem byla odpověď, že asterisk běží na pozadí operačního systému jako démon.

```
$ - asterisk –rvvvvvvvvvvvvvvvvvvvvvvvv
```
Jelikož asterisk běží jako démon na pozadí, tak nic nevypisuje. Proto, abychom viděli podrobné vypisování asterisku, stačí zadat pouze tento příkaz. Čím delší bude část příkazu – rvvvv…., tím podrobnější výpis mi asterisk bude vypisovat. Nyní můžeme vidět, co přesně asterisk dělá.

5.2.1 Konfigurace asterisk klientů

Aby asterisk mohl sestavovat hovory a sloužit jako VoIP ústředna, je nutné provést konfiguraci a přidání klientů do konfiguračních souborů asterisk. Musíme provést první základní konfiguraci asterisku a následně přidat klienty, kteří budou obsluhováni touto ústřednou. Po přidání klientů do konfiguračních souborů je nutné ještě sestavit, jak daný hovor bude vypadat, aby asterisk věděl jak takového klienta obsloužit. Pro tyto účely budeme editovat dva konfigurační soubory asterisku. Soubor, ve kterém budou zaznamenání daní klienti, se jmenuje sip.conf a druhým souborem je extensions.conf, kde bude popsáno jak hovor daného klienta obsloužit. Možností takovéto konfigurace je nepřeberné množství různých možností nastavení anebo přizpůsobeních pro danou situaci.

```
$ - cd /etc/asterisk
```
Konfigurační soubory jsou umístěny v této složce. Jelikož jsme nainstalovali ukázková data, tak je v této složce hodně konfiguračních souborů co vůbec nevyužijeme. To není na škodu, aspoň je lze použít jako vzorové soubory.

```
$ - nano /etc/asterisk/sip.conf
$ - nano /etc/asterisk/extensions.conf
```
Editaci souborů pro konfiguraci klientů provedeme pomocí nano editoru. Začneme se souborem sip.conf kam budu vkládat vkládat nastavení pro klienty. Sekce general se vztahuje pro celý asterisk systém. Další sekce se týkají již jednotlivých klientů. Celý výpis souboru sip.conf je v příloze. [Příloha 2: Editace souboru sip.conf]

Konfigurace jednoho z klientů použitých v asterisk. Více z konfigurace v příloze: [Příloha 2: Editace souboru sip.conf]

```
[105]
host=dynamic
type=friend
username=105
```

```
context=malinovnik
secret=malina
```

- type=friend zajistí, že telefon bude moci volat i přijímat hovory od asterisk ústředny. Ústředna si u něj také bude pamatovat jeho současnou IP adresu.
- secret= představuje heslo. Tímto heslem se bude telefon registrovat na ústřednu. Bezpečné jsou kombinace písmen a číslic. V obou případech zadejte vlastní smyšlená hesla.
- userid zajistí, že se bude zobrazovat na displeji volaného telefonu.
- host=dynamic zajistí, že se telefon může k ústředně přihlásit z libovolné IP adresy.
- context=malinovnik je jméno sekce, která se bude provádět, když budu s telefonem realizovat odchozí hovor. Kontext, mimo jiné, určuje číslovací plán ústředny a budeme se mu věnovat v souboru extensions.conf.

Vypsání konfigurace ze souboru extensions.conf. V tomto souboru jsou definovány konfigurační soubory, neboli dialplan ústředny. Dial plan říká ústředně, co má dělat s voláním. Pro krátký přehled co daný výpis dělá. V případě uživatele označeného jako 105 v souboru sip.conf se řeklo ústředně, uživatel 105 pod telefonním číslem 105 bude-li někdo volat na něj, počká jednu sekundu a následně zvedne hovor. Tento případ konfigurace je pro užití SIPp generátorů. [Příloha 3: Výpis konfigurace extensions.conf]

5.3 Konfigurace SIPp

Konfigurace této kapitoly patří k těm obtížnějším. Instalace jako taková má průběh snadný, ale následná konfigurace zařízeních UAC a UAS je už mnohem náročnější. Proto se pokusíme co nejlépe tuto část popsat, aby konfigurace a následné odladění na VoIP ústředně proběhlo co nejlépe. Jako první si musíme nainstalovat SIPp generátor. Po odzkoušení jednotlivých verzí se doporučuje nainstalovat tu nejnovější, kterou lze najít. V tomto případě se použije verzi 3.5.1, kterou jsme dohledali jako nejaktuálnější a nejspolehlivější. V této kapitole se budeme pohybovat v operačním systému ubuntu, proto jsou použity i jiné příkazy.

```
$ - sudo apt-get install libpcap-dev
```

Před instalací SIpp provedeme instalci pcap knihovny, kterou budeme následně potřebovat. Po úspěšné instalaci lze přejít rovnou k práci s SIPp.

```
$ - wget https://github.com/SIPp/sipp/releases/download/
v3.5.1/sipp-3.5.1.tar.gz
```

Jak bylo již zmiňováno, pro instalaci bude použita tato verze 3.5.1, kterou si nyní stáhneme z tohoto zdroje.

```
$ - tar -zxvf sipp-3.5.1.tar.gz
$ - cd sipp-3.5.1
```

Po úspěšném stažení provedeme hned extrahování souborů z archívu s použitím stejného příkazu a definic pro rozbalování jako jsme použili v kapitole 5.2. A přesuneme se do nově vzniklé složky, kde budeme dále pracovat.

```
$ - ./configure --with-pcap
$ - make
```

Nyní použijeme předem nainstalovanou knihovnu pcap. Tu přidáme jako doplněk pro konfiguraci SIPp systému. Konfigurace nám bude vypisovat informace a kontrolovat jestli

náš operační systém obsahuje vše potřebné pro konečnou instalaci. Pokud se vyskytne nějaký nedostatek nebo chyba provedeme její opravení nebo doinstalovaní chybějícími knihovnami. V tomto případě proběhlo vše v pořádku a lze pokračovat ke konečné instalaci. Všechny následující operace nebo příkazy se provádějí ve složce sipp-3.5.1 co se vytvořila.

```
$ - ./sipp -sn uas
$ - ./sipp -sn uac 127.0.0.1
```

Abychom jsme si otestovali, jestli se instalace podařila, tak provedeme jednoduchý příkaz na vyzkoušení funkčnosti. Každý příkaz vložím do příkazového řádku v nově otevřeném terminálu pro každý příkaz.

Vše funguje jak má, takže se můžeme pustit do příkazů, které bude testovat ústřednu. Testování jsme provedli dvojí, první se použilo dotazování se ústředny na její předdefinované služby, jako je funkce echo nebo přehrání fráze. Druhé je pomocí volání klienta a použití ústředny k propojení hovoru mezi uživateli, ale o tomto až později na nadcházející kapitole.

```
$ - ./sipp -sn uac -s 100 -d 7s -r 5 -rp 1s -l 10 -i 192.168.1.131
192.168.1.135 -trace_stat
$ - ./sipp -sn uac -s 200 -d 7s -r 5 -rp 1s -l 10 -i 192.168.1.131
192.168.1.135 -trace_stat
$ - ./sipp -sn uac -s 100 -d 7s -r 10 -rp 1s -i 192.168.1.131
192.168.1.135 -trace_stat
```

Tímto příkazem spustím generování SIP zpráv a dotazování se ústředny. Po spuštění okamžitě vidím přehled poslaných zpráv, čekajících, úspěšně nebo neúspěšně obsloužených zpráv.

- sn: Použití výchozího scénáře, v mém případě je to scénář pro uac.
- s: Nastaví uživatelské jméno, neboli klienta ústředny, který bude dotazován.
- d: Ovládá délku hovorů. Přesněji řečeno, řídí dobu trvání hovoru. Výchozí hodnotou je 0 a výchozí jednotkou milisekund.
- r: Nastaví rychlost volání za sekundu.
- rp: Určí, kolik hovorů za určitou dobu bude generováno, neboli n volá každých m milisekund. V mém případě -r 5 -rp 1s => 5 hovorů každou 1 sekundu.
- i: Nastavení lokální IP adresy a vzdálené pro směrování.
- l: Nastaví maximální počet souběžných hovorů. Po dosažení tohoto limitu, budou hovory snižovány do daného maxima.
- trace_stat: Uloží všechny statistiky do csv souboru.

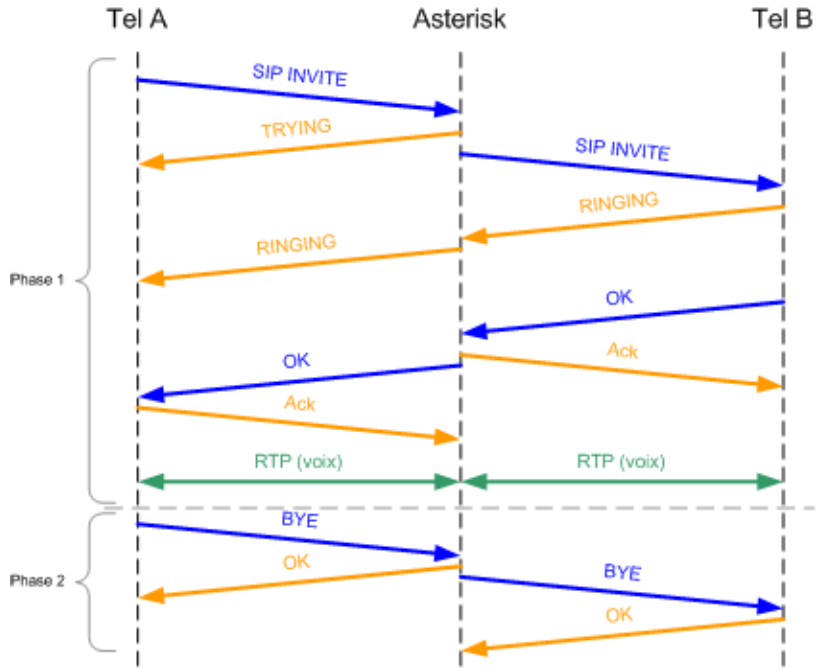
5.3.1 Konfigurace scénáře

Vytvoření scénáře patří k těm složitějším částem této celé konfigurace. Scénář musí být sestaven přesně podle požadavků, jaké od něj očekáváme. Zároveň musí být vytvořen podle pravidel, aby byl přesně funkční. Důležité je i dodržování přesné syntaxe a kódování. Pokud se rozhodneme vytvořit scénář, tak musíme vytvořit scénář jak pro uac stranu, tak ale i pro uas stranu. Jelikož naším záměrem je použití scénáře pro propojení hovoru pomocí ústředny. Aby protistrana mohla odpovídat, je důležité nakonfigurovat scénáře pro obě strany. Před posláním požadavků je nutné, aby proběhla registrace klientů v asterisk ústředně. Bez registrace neví ústředna, jací klienti jsou připojeni a kde je má hledat. To má za následek vytvoření dalšího scénáře, který zajistí registraci klientů. Tento registrační

scénář už může být stejný pro obě strany uas a uac. Po vytvoření scénářů spustíme jako první scénář s registrací klientu, který provede registraci a až poté spustíme scénáře pro testování propojování hovorů.

Konfigurace scénáře pro SIPp je sestavována podle jmenovitého protokolu SIP. Ten má za úkol sestavení, dohled a rozpad spojení mezi dvěma a více účastníky komunikace. Přesně podle toho také musí vypadat výsledný scénář. Bude tvořen jednotlivými metodami komunikace SIP protokolu. Více o protokolu SIP anebo metod navazování komunikace je v kapitole 3.1.3.

- INVITE - slouží k žádosti o sestavení spojení
- ACK - potvrzení INVITE finálním příjemcem zprávy (volaným)
- BYE - ukončení spojení
- CANCEL - ukončení nesestaveného spojení
- REGISTER - registrace UA



Obrázek 18: Navazování spojení SIP protokolu

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<scenario name="Basic Sipstone UAC">
</scenario>
```

Každý scénář musí začínat a končit přesně definovanou syntaxí a je ve formátu xml. Základní části scénářů pro navázání komunikace, tak první musí proběhnout zmíněná registrace účastníků. [Příloha 4: Scénář registrace klientů]

Po zaregistrování účastníků se bude moci provést inivite. Následovat bude výpis invite metody, který obsahuje v sobě hlavičku SIP zprávy. Jak je v příloze invite scénáře. [Příloha 5: Invite pro UAC]

Zde je ukázána metoda invite, která v sobě obsahuje hlavičku SIP zprávy pro navázání spojení s protistranou. Uvnitř příkazů send, musí být vložena SIP zpráva mezi "<! [CDATA" a "]]>" definující značky pro zprávu. Vše mezi těmito značkami bude odesláno ke vzdálené protistraně. V tomto případě se použilo statické směrování a také každý klient má svoje heslo zabezpečující. Proto se také použil externí csv soubor ve kterém jsou

vypsané potřebné údaje pro správný průběh invite metody. Kvůli csv souboru se musel následně upravit i scénář. [Příloha 6: Scénář pro UAC s csv daty]

Po invite zprávě následuje odpověď od protistrany, v tomto případě od UAS strany. V druhém scénáři pro UAS stranu musí být odpověď ringing. Po zaslání dotazu inivite a odpovědi ringing je hned stranou UAS odeslána i odpověď na dotaz invite a to potvrzení 200OK. Tyto dvě zprávy odesílá strana UAS. Zase je tato odpověď upravena pro možnost použitého externího csv souboru. [Příloha 7: Scénář pro UAS stranu]

Pokud byly úspěšně obdrženy odpovědi od UAS strany ve tvaru ringing a 200OK, je poslána další ACK metoda od UAC strany. Opět tato metoda musí být upravena pro použití externího souboru csv. [Příloha 6: Scénář pro UAC s csv daty]

S ACK zprávou jsou odeslány i RTP data, aby se co nejvíce přiblížila reálnému hovoru. Proto posíláme pcap audio data, abychom touto zprávou odesílali RTP data. Data jsou přenášena po dobu osmi sekund. Tato dobu trvání přenosu je zvolena dle průměrné doby trvání hovoru. [Příloha 6: Scénář pro UAC s csv daty]
Na zaslaná data strana UAS zareaguje pouze přijetím. [Příloha 7: Scénář pro UAS stranu]

Po odeslání dat následuje rozpad spojení mezi uživateli. Pro rozpojení pošle UAC strana modul pro rozpojení a rozloučení se s protistranou. Modul bye je opět upraven pro implementaci externího csv souboru. Byl vyslán modul pro rozpojení spojení stranou UAC. [Příloha 6: Scénář pro UAC s csv daty]
Na tento modul je vyslána pouze potvrzující odpověď stranou UAS. Po tomto potvrzení je navázané spojení rozpojeno. [Příloha 7: Scénář pro UAS stranu]

Kompletní výpisy scénářů jsou přiloženy v příloze. Po nakonfigurování scénářů zbývá už jen jejich spuštění v příkazové řádce. Nově vzniklé scénáře jsou uloženy do složky systému SIPp, která vznikla již na začátku při instalaci systému. Nyní, když už je vše uloženo na správném místě a nakonfigurováno může se provést spuštění. Spuštění se provede podobně, jako spuštění vzorového scénáře a zasílání dotazů na ústřednu v předchozí kapitole. Jen zde jsou pozměněna některá specifika, aby se mohly implementovat vytvořené xml scénáře a csv soubor s daty. První část příkazu je pro UAC stranu a spodní část příkazů je pro stranu UAS. První se na obou stanicích sputí příkaz s regist souborem a následně druhý příkaz.

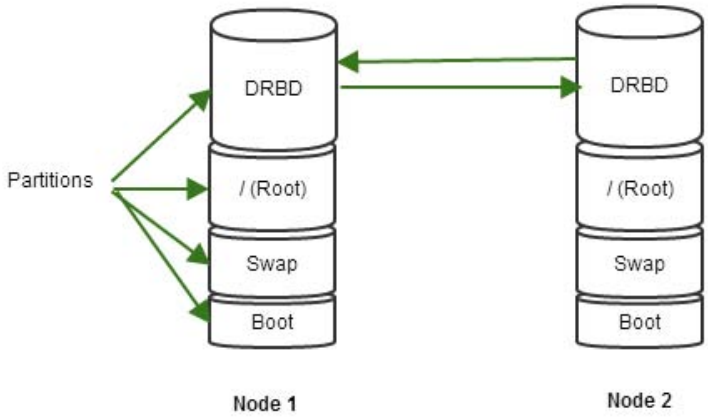
```
$ - ./sipp -sf registr.xml -inf hovor.csv -r 5 -rp 1s -i
192.168.1.130 192.168.1.135 -trace_stat –rate_max 15
$ - ./sipp -sf call_uac.xml -inf hovor.csv -r 5 -rp 1s -i
192.168.1.130 192.168.1.135 -trace_stat

$ - ./sipp -sf registr.xml -inf hovor_uas.csv -r 5 -rp 1s -i
192.168.1.130 192.168.1.135 -trace_stat –rate_max 15
$ - ./sipp -sf call_uas.xml -inf hovor_uas.csv -r 5 -rp 1s -i
192.168.1.130 192.168.1.135 -trace_stat
```

5.4 Konfigurace DRBD systému

Pro správnou funkčnost drbd systému je důležité, aby konfigurace probíhala na dvou stejných paměťových jednotkách se stejným rozdělením oddílů disku. Z tohoto důvodu je nejlepší provádět konfiguraci na nový systém. Při realizaci jsme zvolili raději novou instalaci systému, aby nedošlo k náhodné chybě. Rozdělení paměťového média jsme

provedli ručně na obou systémech stejně. Příklad rozdělení disku je vidět na obrázku číslo 19 pod textem.



Obrázek 19: Rozdělení paměťového média pro drbd [20]

Po rozdělení disku je nainstalován operační systém, v tomto případě je použita distribuce systému elastix, založena na systému centos. Balíček elastix, který je použit, v sobě obsahuje již nainstalovanou asterisk ústřednu. Ta se nakonfigurovala podle kapitoly 5.2. Následně se v nově vzniklém systému nakonfigurovala statická adresace, podle postupu, který byl popsán výše. Do takto připraveného systému jsou jen dokonfigurovány hlavní systémy pro zajištění vysoké dostupnosti a sdílení dat. Ve výsledku jsou připravené dva totožné zdroje, z čeho jeden je hlavní a druhý je záložní. Jelikož nyní je používán centos, rozdílný operační systém proti předchozí kapitole. Budou se také lišit použité příkazy pro konfiguraci.

```
$ - yum list installed net-snmp*
```
Vypíše instalované balíky s názvem začínajícím na net snmp a zobrazí se seznam instalovaných balíčků. Tímto se může překontrolovat, jestli již jsou instalovány následující balíček snmp. V tomto případě se musel tento balíček doinstalovat.

```
$ - yum install net-snmp-utils
```
Chybějící balíček, co je potřeba instalovat je net-SNMP utils. Tento krok se dokončí snadno pomocí příkazu instalace. Instalace proběhla úspěšně a pokračuje se vytvořením nové partition.

```
$ - fdisk /dev/sda
```
Zde budeme postupovat podle následujících kroků, k vytvoření a přidání nového oddílu. Pro porovnání si necháme pomocí příkazu (p) vypsat nynější rozdělení. Po úspěšném přidání se tento výpis rozšíří o nově vzniklý oddíl. Abychom tuto partition vytvořili, postupujeme následovně.

- Přidání nové partition (n)
- Vybereme primární (p)
- V tomto případě jsme určili číslo oddílu (4)
- Potvrzení klávesou enter
- Pro změnu ID systémového oddílu stisknu "t"
- Zvolíme číslo bloku "4"
- Vyberte si HEX 83, napíše se jen 83 a potvrdíme enter
- Uložení změn provedeme "w"

- Zkontroluji změny ve výpise (p)

Restartujeme oba servery, jak hlavní tak i náhradní, jakmile dokončíme přidání oddílu a jeho uložení. Po restartu zkontrolujeme znovu tabulku výpisu na obou serverech, jestli nedošlo ke změně a jsou na obou serverech stejné rozdělení.

```
$ - mkfs.ext3 /dev/sda4
```
Zformátování nové partition.

```
$ - dd if=/dev/zero bs=1M count=500 of=/dev/sda4; sync
```
Vyčistíme souborový systém na novém oddílu, abychom jej měli připravený a čistý.

```
$ - yum install drbd83 kmod-drbd83
```
Stáhneme a nainstalujeme drbd systém pro synchronizaci dat. Po nainstalování by bylo dobré ověřit, jestli síťová konfigurace je nastavena správně. Pokud je vše v pořádku, lze provést důležitou konfiguraci drbd systému v jeho konfiguračním souboru.

```
$ - nano /etc/drbd.conf
```
Drbd systém se nainstaloval a vznikl konfigurační soubor pro nastavení celého drbd systému. Tento konfigurační soubor je nutné editovat a nastavit dle potřeby. Pro editaci použijeme nano editor a vložíme konfiguraci, viz. níže.

```
global { usage-count no; }
resource replica {
protocol C;

# Zde si lze zvolit časování kontroly dle vlastní potřeby
startup { wfc-timeout 10; degr-wfc-timeout 30; }
disk { on-io-error detach; }
net {

# Tyto řádky pomáhají serverům rozhodnout, který server bude hlavní a záložní v případě že budou oba hlavními
    after-sb-0pri discard-younger-primary;
    after-sb-1pri discard-secondary;
    after-sb-2pri call-pri-lost-after-sb;
    cram-hmac-alg "sha1";
    shared-secret "Wind2Hear2See!";
# Heslo si každý může zvolit svoje
}
syncer { rate 10M; }
# Nastavíme hostname hlavního zařízení
on elastix1 {
    device /dev/drbd0;
    disk /dev/sda4;
# Vložení IP adresy hlavního zařízení, přiřazenou pod zvolený hostname, v případě dvou NIC použiji tu pro synchronizaci dat
    address 192.168.1.140:7788;
    meta-disk internal;
}
# Nastavíme hostname záložního zařízení
on elastix2 {
    device /dev/drbd0;
```

```
    disk /dev/sda4;
```
Vložení IP adresy záložního zařízení, přiřazenou pod zvolený hostname, v případě dvou NIC použiji tu pro synchronizaci dat
```
    address 192.168.1.138:7788;
    meta-disk internal;
}
}
```

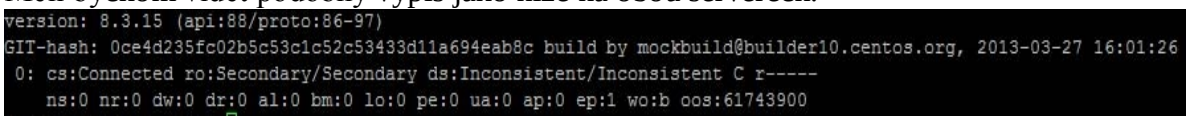
Konfiguraci uložíme, a abychom jsme si ušetřili práci, překopírujeme konfigurační soubor drbd systému na druhé záložní zařízení.
```
$ - scp /etc/drbd.conf elastix2@192.168.1.138:/etc/
```

Inicializují se meta datové oblasti drbd na obou serverech před jejich spuštěním.
```
$ - drbdadm create-md replica
```

Spuštění drbd systém, na obou zařízeních.
```
$ - service drbd start
```

Nyní se zkontroluje, jestli je spuštěna služba na obou zařízeních, v současné době jsou oba označení jako sekundární.
```
$ - cat /proc/drbd
```

Měli bychom vidět podobný výpis jako níže na obou serverech.



Obrázek 20: Výpis drbd systému

Jelikož je potřeba, aby na hlavním zařízení se drbd systém hlásil jako primární, proto na hlavním serveru spustíme příkaz na přepsání a synchronizaci dat, tím se určí že drbd systém bude primární.
```
$ - drbdadm -- --overwrite-data-of-peer primary replica
```

Na záložním zařízení si spustíme příkaz pro sledování synchronizace. Můžeme vidět, že se už statusy změnili dle potřeby a probíhající synchronizaci. Tu necháme doběhnout až dokonce.
```
$ - watch -n 1 cat /proc/drbd
```

Nyní se nastaví souborový systém na hlavním serveru.
```
$ - mkfs.ext3 /dev/drbd0
$ - mkdir /replica
$ - mount /dev/drbd0 /replica

$ - drbdadm role replica
```
Odpověď systému na hlavním serveru by měla být primary/secondary. Nastavení je v pořádku, jak je požadováno.

5.4.1 Implementace asterisku do drbd

Pokud už nebudeme chtít aktualizovat, nebo nějak jinak upravovat asterisk ústřednu, můžeme provést implementaci do drbd oddílu. V tomto oddílu budou data mezi oběma zařízeními synchronizována. V případě výpadku budou mít stále aktuální data. Nyní

provedeme přesun souborů a adresářů, které chceme replikovat na DRBD oddíl. Pokud bychom toto neudělali, byl by asterisk nadále spuštěn v kořenovém oddílu a jeho soubory by nemohli být replikovány mezi ústřednami. Replikace docílím přesunutím souborů a adresářů do drbd oddílu do adresáře replica. Po přesunutí smažeme původní kořenové soubory i adresáře a namísto nich vytvoříme jen symbolický odkaz, který bude odkazovat do složky drbd systému replica. Vytvoření odkazů zastoupí původní soubory a asterisk nepozná žádnou změnu.

Přesuneme se do složky replica na drbd oddíle a postupně spustíme následující sady příkazů. První provede archvaci původního souboru a adresářů a poté tento archív extrahujeme v místní složce replica. Použití příkazu, jak bylo již přiblíženo v předchozích kapitolách a proto nebude žádný problém tento příkaz aplikovat. Po přesunutí souborů bude následovat další sada příkazu pro smazání původních kořenových souborů. A jako poslední se použije sadu příkazů k vytvoření odkazů na přesunutí souborů. Druhá sada příkazů je pro smazání původních kořenových souborů. Poslední sada příkazů, je k vytvoření odkazů na přesunuté soubory z kořenových adresářů. [Příloha 8: Implementace dat do DRBD]

Vše jsme přesunuli, smazali a nahradili odkazy, tak nyní postupně povypínáme služby, kterých se týkala předchozí sada příkazů.

```
$ - service mysqld stop
$ - service asterisk stop
$ - service httpd stop
$ - service elastix-portknock stop
$ - service elastix-updaterd stop
```

Podobnou konfiguraci musíme nyní provést i na záložním zařízení. Proto provedeme odpojení stroje od oddílové složky replica a přepnu server manuálně do stavu secondary.

```
$ - umount /replica
$ - drbdadm secondary replica
```

Na druhém záložním serveru provedeme opak toho na prvním serveru. Provedeme manuálně přiřazení primary módu a připojení oddílové složky.

```
$ - drbdadm primary replica
$ - mount /dev/drbd0 / replica
$ - cd /replica
```

Jsme přepnutý na druhém zařízení, a pokud všechny předchozí příkazy až do tohoto fungovali správně, měl bychom ve složce replica na záložním stroji vidět soubory, které jsme přesouvali na hlavním serveru. Takže nyní musíme odstranit soubory z kořenových adresářů, jako jsme to udělali na hlavním a vytvořit symbolické odkazy. Příkazy se použijí úplně stejné jako předtím. První smazání soubor. No a po smazání soubrů, musíme vytvořit symbolické odkazy. [Příloha 8: Implementace dat do DRBD]

A aby to bylo úplně stejné, zastavíme i služby, kterých se provedené změny týkali, stejnými příkazy jako před chvílí u hlavního zařízení.

Než bude úplně vše hotové, vrátíme původní nastavení primary a sekundary serverů. Hlavní zařízení chceme mít zase jako primary a proto se manuálně přepnou servery do původních stavů. Na záložním serveru provedeme odpojení oddílového adresáře od zařízení a nastavím jej na pozici secondary.

```
$ - umount /replica
```

```
$ - drbdadm secondary replica
```

Teď se už jen připojí adresářový oddíl drbd systému k hlavnímu zařízení a nastavím jej na hodnotu primary.

```
$ - drbdadm primary replica
$ - mount /dev/drbd0 /replica
```

Abychom si byli úplně jistý, tak zkontrolujeme statusy pomocí příkazu. Odpověď by měla být primary/secondary.

```
$ - drbdadm role replica
```

Nyní jsou zařízení nastavena na automatickou synchronizaci dat v případě výpadku.

5.5 Konfigurace heartbeat systému

Pro konfiguraci monitorovacího systému jako je heartbeat, je velice důležité nastavení a případná konfigurace síťové adresace na zařízeních. Tento systém bude monitorovat stav hlavního serveru. V případě výpadku hlavního zařízení převezme činnost na záložní server. Heartbeat bude také kontrolovat a spouštět automaticky i další služby, které jsou běžnou součástí, jako je databáze, VoIP ústředna anebo webové rozhraní. Zakážeme jednotlivé služby na obou serverech, aby se nespouštěli sami při stratu systému a následně je i vypneme. Takže se nebudou sami spouštět a jejich spuštění bude mít na starost heartbeat systém, ten bude řídit spuštění služby na základě toho, které zařízení neboli systém bude hlavní.

```
$ - chkconfig asterisk off
$ - chkconfig mysqld off
$ - chkconfig httpd off
$ - chkconfig elastix-portknock off
$ - chkconfig elastix-updaterd off

$ - service mysqld stop
$ - service asterisk stop
$ - service httpd stop
$ - service elastix-portknock stop
$ - service elastix-updaterd stop
```

Služby jsme povypínali a nyní se budeme věnovat konfiguraci. Začneme konfigurací souboru ha.cf, použiji editor nano abychom mohli provést konfiguraci a vložit příkazy.

```
$ - nano /etc/ha.d/ha.cf
```

Konfigurační obsah souboru ha.cf, důležité je nezapomenout nastavit správně jména host systémů.

```
debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility local0
keepalive 2
deadtime 30
warntime 10
initdead 120
udpport 694
bcast eth0
auto_failback on
node elastix1
node elastix2
```

Dalším souborem k editaci je authkeys, heslo je na výběru každého, kdo by ho chtěl změnit.

```
$ - nano /etc/ha.d/authkeys
```

Obsah souboru, kde je nejdůležitější heslo.

```
auth 1
1 sha1 Quick2TheDraw!
```

Ještě je dobré změnit práva tohoto souboru, aby nenastali komplikace.

```
$ - sudo chmod 600 /etc/ha.d/authkeys
```

Posledním souborem, který je potřeba editovat a upravit tak konfiguraci, je haresources. Zde systému heartbeat určíme, jaké ze zařízení bude hlavní, a také se nastavuje virtuální IP adresu. To je velice důležité, jelikož na tuto virtuální IP adresu budou přistupovat veškerá zařízení a bude ji používat asterisk ústředna.

```
$ - nano /etc/ha.d/haresources

elastix1 drbddisk::replica
Filesystem::/dev/drbd0::/replica::ext3
IPaddr::192.168.1.135/24/eth0:0 mysqld asterisk httpd elastix-
portknock elastix-updaterd
```

Po provedené konfiguraci překopírujeme všechny soubory, které jsme editovali na druhý záložní server. Nesmí se však zapomenout na upravení práv souboru authkeys, jako tomu bylo na hlavním zařízení.

```
$ - scp /etc/ha.d/ha.cf /etc/ha.d/authkeys /etc/ha.d/haresources
elastix2@192.168.1.138:/etc/ha.d/
```

Veškerá konfigurace je úspěšně nastavená a lze spustit heartbeat systém na obou dvou zařízeních.

```
$ - service heartbeat start
```

Po spuštění se ještě přidá systém heartbeat do automatického spouštění při startu zařízení. Systém bude nyní automaticky spouštěn a monitorovat stav zařízení aby zajistill vysokou dostupnost.

```
$ - chkconfig --add heartbeat
$ - chkconfig heartbeat on
```

5.6 Konfigurace gluster-fs systému

Tuto konfiguraci systému glusterfs je nejlépe začít z čisté instalace. Pro správnou funkčnost systému je důležité, aby konfigurace probíhala na dvou stejných paměťových jednotkách se stejným rozdělením oddílů disku. Z tohoto důvodu je nejlepší provádět konfiguraci na nový systém. Při realizaci jsme zvolili raději novou instalaci systému centos, aby nedošlo k náhodné chybě. Rozdělení paměťového média jsme provedli ručně na obou systémech stejně. Následně po nové instalaci operačního softwaru jsme nainstalovali asterisk systém a nakonfigurovali podle kapitoly 5.2. Jakmile bylo vše připravené, pokračovali jsme na stěžejní část a to na instalaci systému glusterfs. Po několika minutách, jsme ale narazili na komplikace, se kterými jsme se trápili dosti dlouhou dobu. Proto zásadním krokem před instalací tohoto systému je konfigurace firewallu operačního systému centos. Veškeré komplikace co vznikaly, byly právě

zapříčeněné firewallem. Nastavení výjimky není naštěstí nikterak komplikované. Nastavení firewallu je nutné provést na obou zařízeních. První jej provedeme na hlavním zařízení.

```
$ - iptables -I INPUT --s 192.168.1.140 -m tcp -p all --dport 111
-j ACCEPT
$ - iptables -I INPUT -s 192.168.1.140 -m tcp -p all --dport
24007:24008 -j ACCEPT
$ - iptables -I INPUT -s 192.168.1.140 -m tcp -p all --dport
49152:49170 -j ACCEPT
$ - iptables -I INPUT -s 192.168.1.140 -m tcp -p all --dport
24007:24008 -j ACCEPT
```

Nastavení firewallu na záložním zařízení.

```
$ - iptables -I INPUT --s 192.168.1.138 -m tcp -p all --dport 111
-j ACCEPT
$ - iptables -I INPUT -s 192.168.1.138 -m tcp -p all --dport
24007:24008 -j ACCEPT
$ - iptables -I INPUT -s 192.168.1.138 -m tcp -p all --dport
49152:49170 -j ACCEPT
$ - iptables -I INPUT -s 192.168.1.138 -m tcp -p all --dport
24007:24008 -j ACCEPT
```

```
$ - iptables -L –n
```

Uložení přidaných změn. Přepínač (–n) je zde proto, že je potřeba pouze IP adresa, nikoli doménová jména. Po uložení provedeme restartování, aby se projevily přidané změny.

```
$ - service iptables restart
```

A pro jistotu vyprázníme filtr, aby někde nezůstalo něco nechtěné.

```
$ - iptables –F
```

V případě, že by komplikace přetrvávaly i nadále a přidané vyjímky se nijak neprojevily a mohla se dokončit konfigurace. Je možnost celý firewall vypnout těmito příkazy do doby, než by se dohledala jasná příčina chybovosti nastavení ve firewallu.

```
$ - service iptables stop
$ - chkconfig iptables off
```

Teď provedeme vytvoření nového oddílu na obou zařízeních.

```
$ - fdisk /dev/sdb
```

Zde se bude postupovat podle následujících kroků, k vytvoření a přidání nového oddílu. Pro porovnání si necháme pomocí příkazu (p) vypsat nynější rozdělení. Po úspěšném přidání se tento výpis rozšíří o nově vzniklý oddíl. Abychom tuto partition vytvořili, postupujeme následovně.

- Přidání nové partition (n)
- Vybereme primární (p)
- V tomto případě jsme určili číslo oddílu (4)
- Potvrzení klávesou enter
- Pro změnu ID systémového oddílu stisknout "t"
- Zvolíme číslo bloku "4"
- Vyberte si HEX 8e, napíše se jen 8e a potvrdíme enter
- Uložení změn provedeme "w"
- Zkontrolujeme změny ve výpise (p)

Restartujeme oba servery, jak hlavní tak i náhradní, jakmile dokončíme přidání oddílu a jeho uložení. Po restartu zkontrolujeme znovu tabulku výpisu na obou serverech, jestli nedošlo ke změně a mám na obou serverech stejné rozdělení.

```
$ - mkfs.ext3 /dev/sdb4
```
Zformátování nové partition.

```
$ - dd if=/dev/zero bs=1M count=512 of=/dev/sdb4; sync
```
Vyčistíme souborový systém na novém oddílu, abychom jej měli připravený a čistý.

Balíček systému glusterfs lze stáhnout z tohoto zdroje. Pokud by nastala situace špatného repozitáře pro stážení, je tu uveden i postup přidání repozitáře a opětovného stažení.
```
$ - wget -P /etc/yum.repos.d http://download.gluster.org/
pub/gluster/glusterfs/LATEST/RHEL/glusterfs-epel.repo
```

Postup vložení nového repozitáře pro stažení glusterfs systému. Nejprve importujeme klíč GPG pro softwarový balík.
```
$ - rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY*
```

Provede se přidání repoztáře.
```
$ - rpm --import https://fedoraproject.org/static/ 0608B895.txt
$ - cd /tmp wget http://dl.fedoraproject.org/pub/epel
/6/x86_64/epel-release-6-7.noarch.rpm rpm -ivh epel-release-6-
7.noarch.rpm
$ - yum install yum-priorities
$ - nano /etc/yum.repos.d/epel.repo
```

A v tomto souboru epel.repo se provede editace a přidání řádků priority [EPEL] oddílu.
```
[epel]
name=Extra Packages for Enterprise Linux 6 - $basearch
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-
6&arch=$basearch
failovermethod=priority
enabled=1
priority=10
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6
```

Výše zmíněná situace o špatném repozitáři může nastat. V tomto případě byly mírné problémy, hlavně nalezení správného zdroje pro stažení. Proto je lepší o této možnosti vědět a předejít tak komplikacím.

Po úspěšném stažení se provede instalace systému a jeho následné spuštění.
```
$ - yum install glusterfs-server
$ - /etc/init.d/glusterd start
$ - chkconfig glusterd on
```

Na hlavním serveru se provede příkaz pro vytvoření důvěrného pool spojení mezi mými zařízeními. Z hlavního serveru připojíme záložní. Můžeme pro definování zařízení použít IP adresu nebo i host jméno, ale pro přesnost jsme použili IP adresu. Pokud vše proběhlo úspěšně, budem o tom obratem infomováni.

```
$ - gluster peer probe 192.168.1.138
```

Nebo si úspěšnot jednoduše ověříme následujícím dotazem. Systém odpoví výpisem o počtu připojení a jeho stavu.

```
$ - gluster peer status
```

Výpis úspěšného spojení.

```
[root@server1 ~]# gluster peer status
Number of Peers: 1
Hostname: 192.168.1.138
Uuid: 7cd93007-fccb-4fcb-8063-133e6ba81cd9
State: Peer in Cluster (Connected)
```

Dále vytvoříme sdílené spojení, neboli sdílenou složku a jak glusterfs toto nazývá volume spojení. Toto spojení bude pojmenováné v mém případě asterisk a k němu definované dvě repliky hlavního a záložního zařízení. Připojené repliky můžeme definovat host jménem nebo ip adresou.

```
$ - gluster volume create asterisk replica 2 transport tcp
192.168.1.140:/etc/asterisk 192.168.1.138:/etc/asterisk
```

Spuštění nově sdílené složky.

```
$ - gluster volume start asterisk
```

Na záložním stroji se provede restartování služby glusterfs

```
$ - /etc/init.d/glusterfsd restart
```

O správnosti konfigurace se přesvědčíme následovně.

```
$ - gluster volume info
```

Ve výpise vidíme stav a připojené replikační složky.

```
Volume Name: asterisk
Type: Replicate
Status: Started
Number of Bricks: 2
Transport-type: tcp
Bricks:
Brick1: 192.168.1.140:/etc/asterisk
Brick2: 192.168.1.138:/etc/asterisk
```

Takto můžeme vytvořit replikační složky pro všechny potřebné služny nebo data, která chceme, aby byla sdílena a synchronizována mezi zařízeními. Lze ještě například nakonfigurovat pouze jednosměrné spojení, kam se data z vybraných replik budou pouze nahrávat a tím i zálohovat.

```
$ - gluster volume set asterisk auth.allow 192.168.X.X
```

Tento příkaz pro jednosměrnou replikaci jsme vyzkoušeli a funkčnost byla bez problů, ale v tomto případě jsme to nijak dále nevyužili. Jelikož jsme měli repliky mezi hlavním a záložním serverem, pro které jsme zajistili vysokou dostupnust obousměrnou replikací.

5.7 Konfigurace carp systému

Carp neboli ucarp umožnuje rychlé a jednoduché řešení pro monitorovací systém vysoké dostupnosti. Dokáže virtualizovat i několik virtuálních IP adresech najednou, ale použijeme pouze jednu virtuální IP adresu. Carp poskytuje virtuální adresu přes hlavní

server, v případě selhání hlavního zařízení přejde automaticky poskytování virtuální adresy na záložní zařízení. Až se opět obnoví funkce hlavního serveru, tak záložní zařízení předá poskytování virtuální adresy zpět hlavnímu. Instalace systému ucarp se provede následně.

```
$ - sudo yum install ucarp -y
```

Úspěšná instalace proběhla a přesuneme se do nově vznilé složky, kde budeme pokračovat v konfiguraci.

```
$ - cd /etc/ucarp/
```

Pro případ selhání nebo chyby si vytvoříme kopoji konfiguračního souboru. Kopie souboru bude umístěna ve stejné složce jako vzorový soubor.

```
$ - cp vip-001.conf.example vip-001.conf
```

Kopii konfiguračního souboru již máme a pokračujeme v editaci původního konfiguračního souboru. Tato konfigurace se týká hlavního serveru, pro záložní server předvedu konfiguraci později.

```
$ - nano vip-001.conf
```

Obsah pro konfigurační soubor hlavního serveru.

```
#Master
ID=001
# Network Interface
BIND_INTERFACE=“eth0“
#Real master IP
SOURCE_ADDRESS=“192.168.1.140“
# Virtual IP
VIP_ADDRESS=“192.168.1.135“
# Carp Password
PASSWORD=“carp“
OPTIONS=“-k 1 –P –shutdown –preempt“
```

Jako source adresu jsme použili statické IP adresy zařízení, pro každy server jsme tedy napsali příslušnou statickou IP adresu. Kolonka vip adresy je u obou zařízeních stejná a to je IP adresa virtuální, která se bude v případě výpadku vysílat ze záložního zařízení. Zabezpečení je také na obou zařízeních stejné, aby se navzájem dohodly a potvrdily svoje zabezpečení. Důležitou kolonkou je kolonka options, tady nastavuji chování jednotlivých zařízení, jak se mají zachovat v případě výpadku nebo, které bude jako hlavní. Tato kolonka je na každém zařízení jiná, podle toho jak zařízení budu chtít nakonfigurovat. Znak –k nastaví prioritu, čím menší číslo použiji, tím větší prioritu přiřadím serveru. Druhým znakem –P, neboli taky (–preempt) nastavím, aby se stal hlavním zařízením. V případě, kdy po selhání hlavního zařízení se opět stává aktivním, a nebudeme chtít, aby došlo k výpadku při přepnutí virtuální IP adresy ze záložního na hlavní server, lze tuto možnost zakázat. Zakázání automatického přepnutí zpět na hlavní server provedeme tak, že buď nepřiřadíme rozšiřující znaky (-k a –P), a nebo u volby –k nastavíme stejné hodnoty. V tomto případě jsme nastavili systémy na zpětné přepnutí ze záložního zařízení.

Pro případ selhání nebo chyby si také u záložního serveru vytvoříme kopii konfiguračního souboru. Kopie souboru bude umístěna ve stejné složce jako vzorový soubor. Kopii konfiguračního souboru již máme a pokračujeme v editaci původního konfiguračního souboru. Tato konfigurace se týká záložního serveru. Obsah pro konfigurační soubor záložního serveru.

```
#Slave
ID=001
# Network Interface
BIND_INTERFACE=“eth0“
#Real slave IP
SOURCE_ADDRESS=“192.168.1.138“
# Virtual IP
VIP_ADDRESS=“192.168.1.135“
# Carp Password
PASSWORD=“carp“
# Other Options, see documentation for more information
OPTIONS=“-k 10 –shutdow“
```

Následně provedeme jen zapnutí systému carp na obou zařízeních a jeho nastavení do funkce automatického spouštění při nabýhání operačního systému, také na obou zařízeních.

```
$ - service ucarp start
$ - chkconfig ucarp on
```

5.8 Konfigurace pacemaker systému

Tento systém je dalším z řady řešení vysoké dostupnosti. Pacemaker byl několikrát pozměňován, proto je dobré si dát pozor na to, jaky operačný systém je používán. V tomto případě je používán systém centos verze 6.7. Jelikož se jedná o vyšší verzy než je hraniční 6.4, tak součástí použitého pacemakeru bude i systém cman. Než se tu pustíme do konfigurace, tak provedeme kontrolu nastavené IP adresace, jestli vše odpovídá. Provedeme rovnou instalaci. Jak je vidět instaluje se více balíků, které jsou součástí balíku pacemaker.

```
$ - yum install pacemaker cman pcs ccs resource-agents
```

Pro případ selhání nebo chyby si vytvoříme kopoji konfiguračního souboru. Kopie souboru bude umístěna ve stejné složce jako vzorový soubor.

```
$ - cp /etc/corosync/corosync.conf.example /etc/
corosync/corosync.conf
```

Nastavíme a nakonfigurujeme cluster na primárním stroji.

```
$ - ccs -f /etc/cluster/cluster.conf --createcluster newcluster
$ - ccs -f /etc/cluster/cluster.conf --addnode 192.168.1.140
$ - ccs -f /etc/cluster/cluster.conf --addnode 192.168.1.138
$ - ccs -f /etc/cluster/cluster.conf --addfencedev pcmk
agent=fence_pcmk
$ - ccs -f /etc/cluster/cluster.conf --addmethod pcmk-redirect
192.168.1.140
$ - ccs -f /etc/cluster/cluster.conf --addmethod pcmk-redirect
192.168.1.138
$ - ccs -f /etc/cluster/cluster.conf --addfenceinst pcmk
192.168.1.140 pcmk-redirect port=192.168.1.140
$ - ccs -f /etc/cluster/cluster.conf --addfenceinst pcmk
192.168.1.138 pcmk-redirect port=192.168.1.138
```

Po provedené konfiguraci překopírujeme všechny soubory, které jsme editovali na druhý záložní server.

```
$ - scp /etc/cluster/cluster.conf elastix2@192.168.1.138
/etc/cluster/
```

Před spuštěním je nutné vypnout kontrolu quirum, ta se vypne na obou zařízeních.
```
$ - echo "CMAN_QUORUM_TIMEOUT=0" >> /etc/sysconfig/cman
```

A provedeme spuštění služeb na obou strojích a ještě se ujistíme a nastavíme automatické spuštění služby při resetovní.
```
$ - chkconfig cman on
$ - chkconfig pacemaker on
$ - service cman start
$ - service pacemaker start
```

Po spuštění provedeme konfiguraci clusteru nan hlavním serveru a vytvoření virtuální IP adresy.
```
$ - pcs property set stonith-enabled=false
```

Jeden z nejběžnějších způsobů, jak nasadit pacemaker je v konfiguraci dvou uzlů. Proto quirum jako koncept nemá smysl v tomto scénáři, uplatnění je dobré v případě když je k dispozici více uzlů.
```
$ - pcs property set no-quorum-policy=ignore
```
Vytvoření plovoucí IP na hlavním zařízení, mění IP adresu a jméno serveru podle potřeby:
```
$ - pcs resource create asterisk ocf:heartbeat:IPaddr2
ip=192.168.1.135 cidr_netmask=24 op monitor interval=3s
```
Vyvořil jsem virtuální IP adresu, která se bude hlásit jako asterisk. Tato adresa bude vysílána ze zařízení podle stavu hlavního serveru. Příkaz op monitoru interval = 3s nastavuje službě interval časový, aby věděl, po jaké době má zkontrolovat stav služeb mezi uzly.

Nastavení preferencí virtuální adresy k hlavnímu zařízení.
```
$ - pcs constraint location asterisk prefers
192.168.1.140=INFINITY
```

Ověření statusu, jestli vše funguje.
```
$ - pcs status
```
nebo
```
$ - crm_mon -1
```

Pro zobrazení úplné konfigurace je možnost použít výpis.
```
$ - pcs config
```

Nyní můžeme nasimulovat umělý výpadek služeb. Ten se spustí následujícím příkazem a zastaví danou službu bez vědomí clusteru. Pro lepší informovanost si spustíme výpis konfigurace anebo, v interaktivním režimu pomocí (crm_mon) příkazu a do nastaveného času ověřování služeb bychom měli vidět zprávu o selhání a přesunu virtuální adresy na záložní zařízení.
```
$ - crm_resource --resource asterisk --force-stop
```

6 Výsledné zhodnocení použitých systémů

Jak název napovídá, zde se budeme věnovat výslednému zhodnocení použitých systémů vysoké dostupnosti. Celkem se povedlo použít, neboli spíše vyzkoušet, čtyři systémy. V podkapitolách jsou uvedeny naměřené hodnoty použitých sytémů a to včetně ukázky hodnocení a postupu během hodnocení. Hlavním bodem pro hodnocení byla doba výpadku a počet ztracených hovorů během výpadku hlavní VoIP ústředny. Jelikož právě tyto údaje jsou dle uvážení ty nejzásadnější. Bude také zhodnocena náročnost konfigurace použitých systémů. Celé hodnocení na jednotlivých systémech se provedlo dvěma způsoby. První způsob hodnocení doby nečinnosti a počtu ztracených hovorů byl pomocí zasílání SIP zpráv na předdefinovanou funkcionalitu echo na VoIP ústředně. Zde se stanovila pro vygenerované dotazy dobu trvání při zaslání na ústřednu, abychom docílili reálnější podoby měření. Druhým způsobem hodnocení bylo vytvoření telefonního spojení mezi dvěma účastník, abychom dané měření co nejvíce přiblížili reálnému zatížení, použili jsme během spojení RTP protokol pro přenos datových zpráv. Jako datovou zprávu jsme použili audio soubor, který se během spojení přenášel. Doba přenosu se po dlouhém uvažování a zkoumání průměrné doby jednoduchého hovoru stanovila na osm sekund. Tuto dobu jsme určili jako dostatečně dlouhou pro dané testování a přenos dat. Měření jsme také rozdělili do čtyř až pěti skupin dle počtu probíhajících hovorů, které se následně samostatně několikrát měřilo, abychom mohli stanovit průměrnou hodnotu doby nečinosti a ztracených hovorů u jednotlivých systémů vysoké dostupnosti. Toto rozdělení do skupin bylo také za účelem předem odzkoušených počtu spojení vůči stabilitě pro použitý hardware. Sice veškerá infrastruktura byla vytvořena ve virtuálním prostředí, ale po několika zkoušeních a měřeních jsme zjistili ztráty způsobené použitým hardwarem pro virtualizaci systémů. Ke zmíněným ztrátám přibyly i prodlevy reakcí na vnější úkonony a obsluhu systémů. Proto u použitých skupin jsme omezelili počty generovaných spojení nebo jejich maximální probíhající počet. Dobu nečinnosti a počet ztracených hovorů jsme měřili pomocí záznamu statistik během měření a také pomocí odchytávání provozu v síti mezi klienty a VoIP ústřednou. Na základě těchto získaných dat se následně zjistily počty úspěšných a neúspěšných hovorů, anebo čas nečinností systémů v síti. Z takto získaných infomací na základě několika opakovaných meření pro každou skupinu a každý systém, jsme určili průměrné výsledky použitých systémů a za daných podmínek.

Než se ale pustíme do hodnocení, je nutné provést registraci účastníků ve VoIP ústředně. Jelikož v případě použití SIPp generování zpráv nedochází k automatické registraci v síti. Proto je nutné zaregistrování pomocí upraveného scénáře, aby k těmto problémům nedošlo v pruběhu měření, jsme stanovili dobu platnosti registrace na celý den od zaregistrování v ústředně. Tuto registraci můžeme provést jak pro hlavní, tak i pro záložní server.

6.1 Testování systému DRBD a Heartbeat

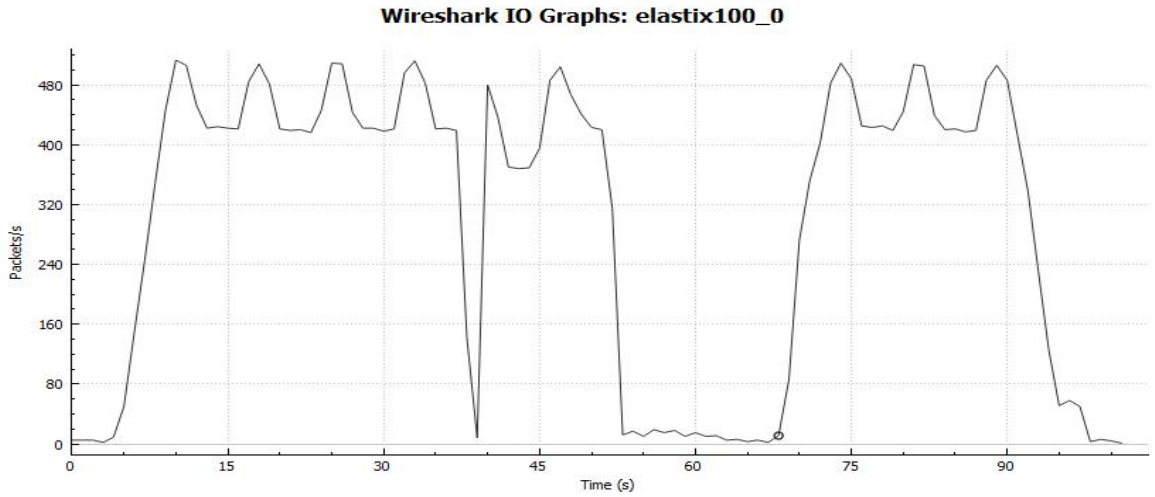
Takto testované systémy zajišťující vysokou dostupnost, jsme použili v upraveném operačním systému balíčeku elastix, který jak jsme výše napsali. Obsahuje již asterisk ústřednu a potřebné knihovny pro testované systémy. Proto je pojmenování grafu dle tohoto balíčku. V první fázi budou uvedeny výsledky hodnocení pro první způsob měření. A to je hodnocení doby nečinosti a počtu ztracených hovorů pomocí zasílání SIP zpráv na předdefinovanou funkcionalitu echo na VoIP ústředně. V druhé fázi jsou zde zvěřejněny výsledky hodnocení za pomoci vytvoření telefonního spojení mezi dvěma účastníky a ústřednou s použitím RTP protokolu. Jsou zde uvedeny tři typy, na kterých se hodnotil čas nečinnosti a ztracené hovory. Jako první je měření s generovánými dvěma hovory každou

sekundu a maximálním počtem deseti obsluhovanými hovory. Pak následuje měření s generovánými čtyřmi hovory, každou sekundu a maximálním počtem deseti obsluhovanými hovory. Jako poslední je měření s generovánými čtyřmi hovory každou sekundu a maximálním počtem třiceti obsluhovanými hovory. Do každé skupiny jsme přidali tabulku s výsledky testovacích hovorů, které se prováděli, abychom mohli lépe určit dobu nečinnosti a ztráty hovorů při daném zatížení ústředny.

První typ měření s generovánými dvěma hovory každou sekundu a maximálním počtem deseti obsluhovanými hovory. Delká hovoru je osm sekund. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu.

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	Successfl Call	Failed Call	Response Time	Call Length
0:17:53	0:00:00	2	0	0	0	0	00:000	00:000
0:18:53	0:01:00	2	1,249	75	56	9	00:054	06:530
0:19:26	0:01:33	2	1,026	96	87	9	01:654	08:301

Tabulka 4: Výsledky hodnoceného hovoru dvě volání každou sekundu s maximem deset



Obrázek 21: Graf vytížení sítě v okamžiku přerušení spojení pro dva hovory každou sekundu s maximem deset.

Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
Elastix100_0	52,913	68,431	15,518	9
2	50,77	65,894	15,124	9
3	49,248	63,993	14,745	8
4	49,87	62,769	12,899	10
5	50,1	64,642	14,542	12
6	53,41	68,534	15,124	9
7	55,89	71,744	15,854	8
8	50,73	62,857	12,127	9
9	49,8	60,82	11,02	10
10	46,2	60,254	14,054	9

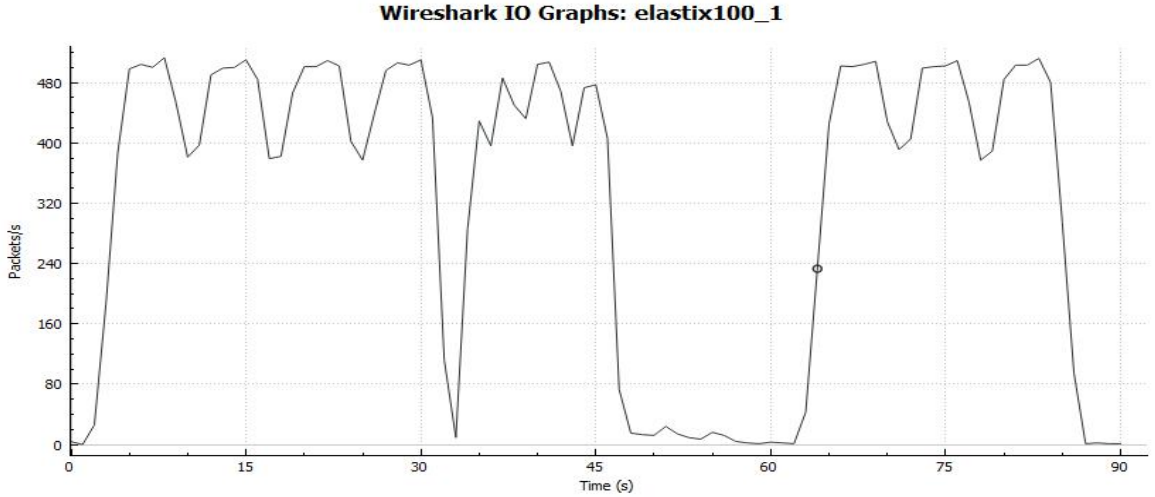
průměrná hodnota	50,89	64,99	14,10	9

Tabulka 5: Výsledky testovacích hovorů průměrného hodnocení pro dva hovory každou sekundu s maximem deset

Dalším druhým typem měření s generovánými čtyřmi hovory každou sekundu a maximálním počtem deseti obsluhovanými hovory. Delká hovoru je osm sekund. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu.

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	Successfl Call	Failed Call	Response Time	Call Length
0:26:52	0:00:00	4	0	0	0	0	00:000	00:000
0:27:52	0:01:00	4	1,249	75	54	11	00:055	06:520
0:28:17	0:01:24	4	1,120	95	84	11	01:688	08:308

Tabulka 6: Výsledky hodnoceného hovoru čtyř volání každou sekundu s maximem deset.



Obrázek 22: Graf vytížení sítě v okamžiku přerušení spojení pro čtyři hovory každou sekundu s maximem deset.

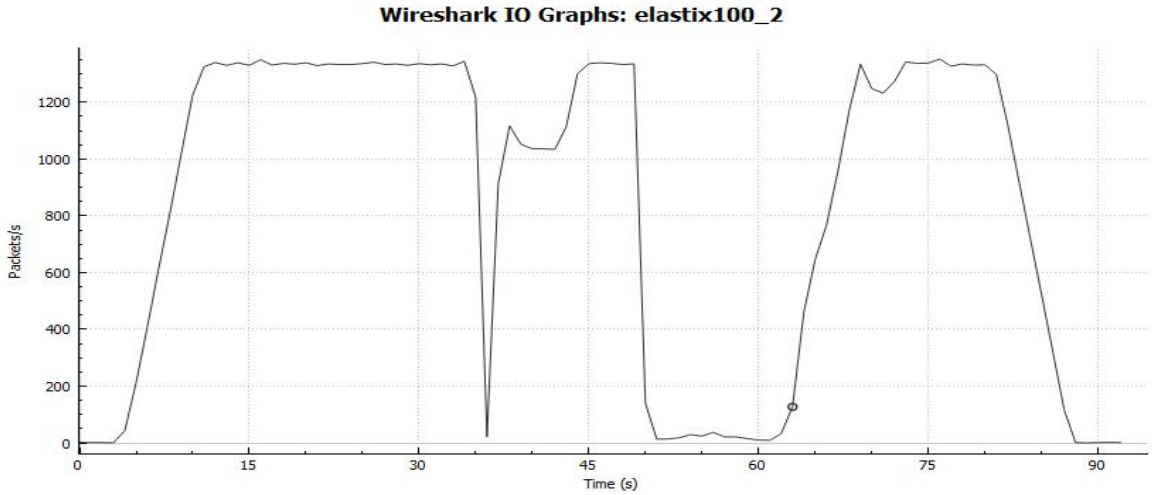
Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory [s]
Elastix100_1	47,288	63,06	15,772	11
2	50,21	65,1	14,89	9
3	48,77	64,227	15,457	10
4	50,63	65,5	14,87	9
5	49,75	68,49	18,74	11
6	50,72	70,24	19,52	12
7	47,22	64,55	17,33	11
8	49,57	63,78	14,21	9
9	45,802	61,612	15,81	10
10	50,74	65,96	15,22	11
průměrná hodnota	49,07	65,25	16,18	10

Tabulka 7: Výsledky průměrného hodnocení pro čtyři hovory každou sekundu s maximem deset.

A třetím typem měření s generovánými čtyřmi hovory každou sekundu a maximálním počtem třiceti obsluhovanými hovory. Delká hovoru je osm sekund. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu.

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	Successf l Cal	Failed Call	Response Time	Call Length
0:17:53	0:00:00	2	0	0	0	0	00:000	00:000
0:18:53	0:01:00	2	1,249	75	56	9	00:054	06:530
0:19:26	0:01:33	2	1,026	96	87	9	01:654	08:301

Tabulka 8: Výsledky hodnoceného hovoru čtyř volání každou sekundu s maximem třicet.



Obrázek 23: Graf vytížení sítě v okamžiku přerušení spojení pro čtyři hovory každou sekundu s maximem třicet.

Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
Elastix100_2	50,16	62,159	11,999	9
2	48,34	60,81	12,47	8
3	47,21	58,45	11,24	9
4	50,11	61,31	11,2	11
5	56,78	72,58	15,8	13
6	49,89	61,59	11,7	9
7	52,47	65,71	13,24	8
8	61,213	77,923	16,71	9
9	51,2	61,62	10,42	11
10	50,7	62,08	11,38	9
průměrná hodnota	51,81	64,42	12,62	10

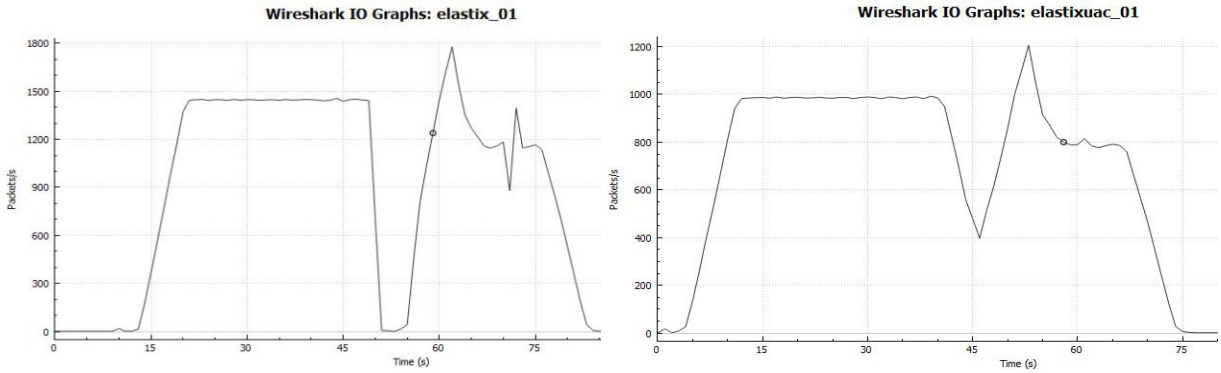
Tabulka 9: Výsledky průměrného hodnocení pro dva hovory každou sekundu s maximem třicet.

V druhé fázi jsou zde zvěřejněny výsledky hodnocení za pomoci vytvoření telefonního spojení mezi ústřednou a dvěma účastníky s použitím RTP protokolu pro přenos dat. Do každé skupiny se přidala tabulku s výsledky testovacích hovorů, které jsme prováděli, abychom mohli lépe určit dobu nečinnosti a ztráty hovorů při daném zatížení ústředny. U těchto měření je důležité povšimnout si sloupce aktuální hovory (current call). Jsou to hovory, které byly přerušeny ve stavu jejich spojení a přenosu dat v okmažiku ukončení testování. Tyto hovory se nezapočítávají do ztacených hovorů, jelikož nastali až po přepnutí. Další částí, které je dobré si povšinout, je porovnání graf výpisu stavu sítě na straně UAC (vpravo) a UAS (vlevo). Je zde patrný rozdíl v poklesu generovaných hovrů na straně UAC proti přijímací straně UAS, kde pokles není tak patrný. Vyhodnocení neaktivního času a ztracených hovrů je z generující UAC strany. Další rozdílností je délka hovoru, pro toto testování jsme si vytvrořili vlastní testovací scénaře, kde máme dobu přenosu dat osm sekund, poté jsem nastavil dobu čekání mezi vyzváněním a přijmutím hovoru jednu sekundu, abychom test přiblížili reálným podmínkám. Při ukončení hovoru je nastavena další pauza čtyři sekundy. Proto se celková doba hovoru prodloužila na třináct sekund, viz tabulka 10.

První typ měření je ukázka spojení mezi dvěma klienty a VoIP ústřednou s generovánými dvěma hovory každou půl sekundu. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu.

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	Current Call	Successfu l Call	Failed Call	Call Length
17:06:02	0:00:00	10	0	0	0	0	0	00:000
17:07:02	0:01:00	10	3.548	213	79	134	0	13:030
17:07:41	0:01:39	10	2.320	231	32	195	4	13:500

Tabulka 10: Výsledky hodnoceného hovoru dvou volání každou půl sekundu.



Obrázek 24: Graf vytížení sítě v okamžiku přerušení spojení mezi dvěma účastníky pro dva hovory každou půl sekundu. Vlevo strana UAS a vpravo UAC.

Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
Elastix_01	50,406	54,93	4,524	4
2	51,02	56,23	5,21	5
3	50,77	55,64	4,87	8
4	48,36	53,35	4,99	2
5	47,54	53,19	5,65	3

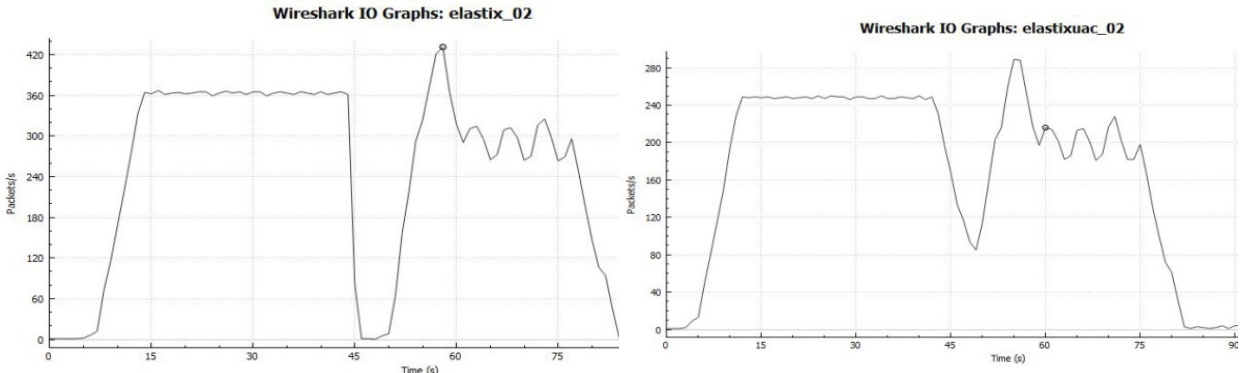
6	49,27	56,68	7,41	1
7	52,14	57,01	4,87	3
8	47,28	51,53	4,25	9
9	50,63	55,81	5,18	4
10	49,55	55,33	5,78	3
průměrná hodnota	49,70	54,97	5,27	4

Tabulka 11: Výsledky průměrného hodnocení pro dva hovory každou půl sekundu.

Dalším druhým typem měření je ukázka spojení mezi dvěma klienty a VoIP ústřednou s generováným jedním hovorem každou sekundu. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	Successfl Call	Current Call	Failed Call	Call Length
17:16:53	0:00:00	10	0	0	0	0	0	00:000
17:17:53	0:00:59	10	0.883	53	32	21	0	13:030
17:18:36	0:01:42	10	0.616	63	54	8	1	13:383

Tabulka 12: Výsledky hodnoceného jednoho volání každou sekundu.



Obrázek 25: Graf vytížení sítě v okamžiku přerušení spojení mezi dvěma účastníky pro jeden hovor každou sekundu. Vlevo strana UAS a vpravo UAC.

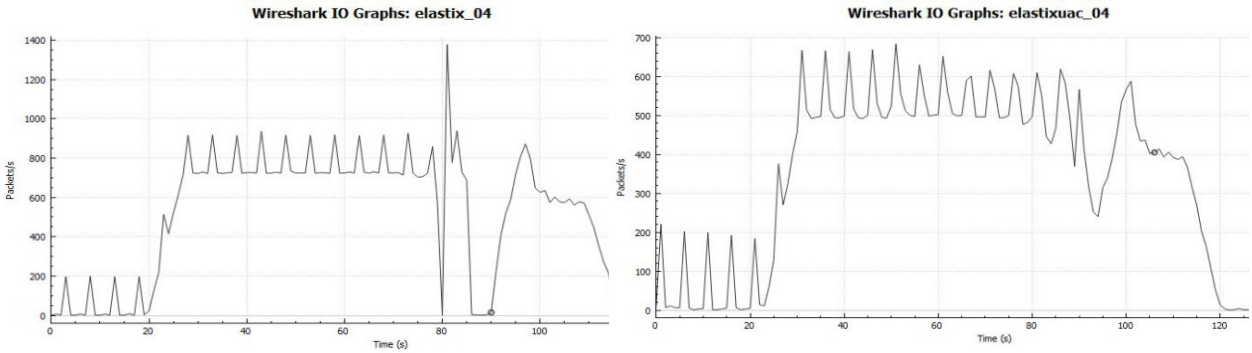
Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
Elastix_02	44,78	50,27	5,49	1
2	48,14	53,34	5,2	1
3	49,21	54,99	5,78	1
4	49,1	53,46	4,36	6
5	42,38	48,62	6,24	5
6	44,74	51,09	6,35	1
7	45,81	51,08	5,27	3
8	42,63	48,85	6,22	3
9	42,98	48,26	5,28	3
10	44,37	49,8	5,43	1
průměrná hodnota	45,41	50,98	5,56	3

Tabulka 13: Výsledky průměrného hodnocení pro jeden hovor každou sekundu.

Předposledním typem měření je ukázka spojení mezi dvěma klienty a VoIP ústřednou s generovánými čtyřmi hovory každé dvě sekundy. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu.

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	Successful Call	Curren t Call	Failed Call	Call Length
17:37:33	0:00:00	10	0	0	0	0	0	00:000
17:38:33	0:00:59	10	1,783	107	81	26	0	13:035
17:39:33	0:01:59	10	1,416	170	152	18	2	13:326

Tabulka 14: Výsledky hodnoceného hovoru čtyř volání každé dvě sekundy.



Obrázek 26: Graf vytížení sítě v okamžiku přerušení spojení mezi dvěma účastníky pro čtyři hovory každé dvě sekundy. Vlevo strana UAS a vpravo UAC.

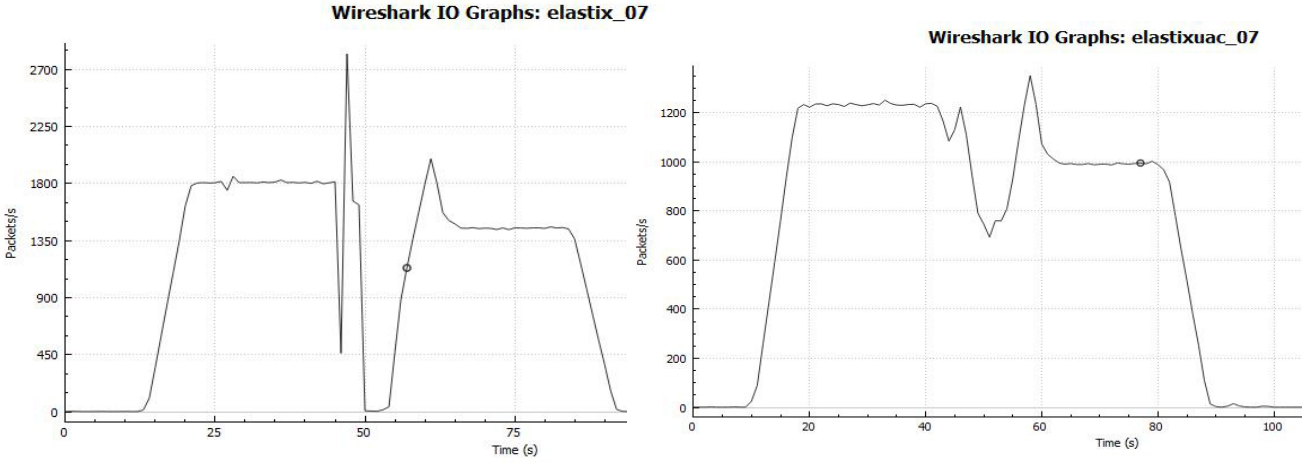
Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
Elastix_04	85,69	90,17	4,48	2
2	90,34	95,29	4,95	2
3	86,19	91,95	5,76	1
4	79,87	85,27	5,4	2
5	81,44	86,17	4,73	2
6	83,85	88,07	4,22	5
7	88,79	93,6	4,81	1
8	81,96	86,33	4,37	3
9	83,41	87,59	4,18	3
10	86,47	91,6	5,13	5
průměrná hodnota	84,80	89,60	4,80	3

Tabulka 15: Výsledky průměrného hodnocení pro čtyři hovory každé dvě sekundy.

Posledním čtvrtým typem měření, je ukázka spojení mezi dvěma klienty a VoIP ústřednou s generovánými pěti hovory každou sekundu. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu.

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	Current Call	Successfu l Call	Failed Call	Call Length
18:23:39	0:00:00	10	0	0	0	0	0	00:000
18:24:39	0:01:00	10	4.349	261	98	163	0	13:091
18:25:39	0:02:00	10	2.649	318	40	273	5	13:418
18:25:50	0:02:11	10	2.411	318	40	273	5	13:418

Tabulka 16: Výsledky hodnoceného hovoru pěti volání každou sekundu.



Obrázek 27: Graf vytížení sítě v okamžiku přerušení spojení mezi dvěma účastníky pro pět hovorů každou sekundu. Vlevo strana UAS a vpravo UAC.

Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
Elastix_07	49,73	53,88	4,15	5
2	51,7	55,97	4,27	19
3	48,2	51,78	3,58	13
4	49,21	52,42	3,81	18
5	48,64	54,12	5,48	6
6	51,78	56,99	5,21	5
7	55,2	59,87	4,67	18
8	48,7	53,21	4,51	14
9	49,48	54,71	5,23	10
10	48,71	53,88	5,17	9
průměrná hodnota	50,15	54,68	4,61	12

Tabulka 17: Výsledky průměrného hodnocení pro pět hovorů každou sekundu.

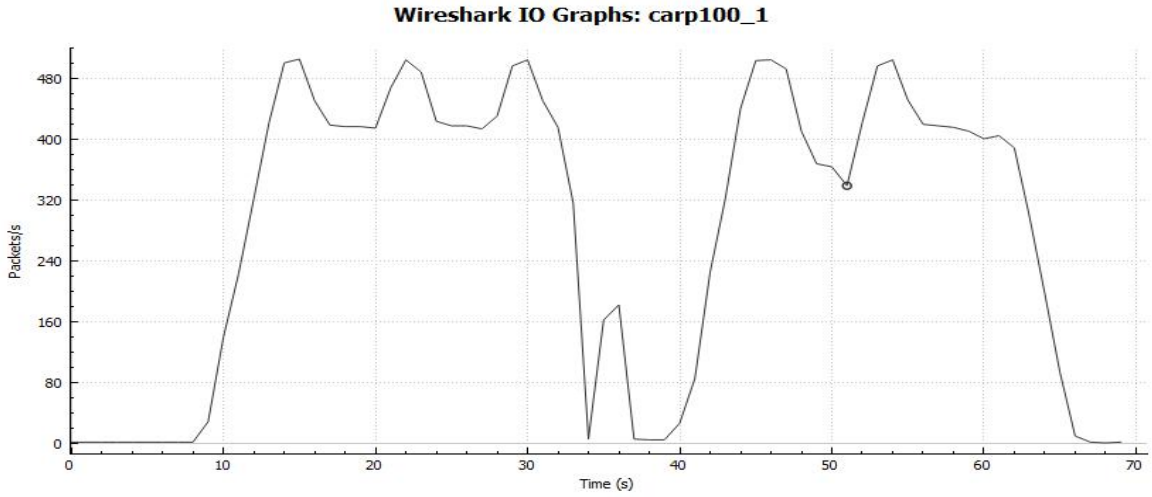
6.2 Testování systému Carp a Gluster-fs

Dalšimi systémy zajišťujícími vyskou dostupnou, které jsme použili, jsou carp a gluster. Hodnocení se provedlo dle obdobných kroků jako předchozí podkapitole. V první fázi budou uvedeny výsledky hodnocení pro první způsob měření. To je hodnocení doby nečinosti a počtu ztracených hovorů pomocí zasílání SIP zpráv na předdefinovanou funkcionalitu echo na VoIP ústředně. V druhé fázi jsou zde zvěřejněny výsledky hodnocení za pomoci vytvoření telefonního spojení mezi dvěma účastníky a ústřednou s použitím RTP protokolu. Pro první fázi jsou zde uvedeny tři typy, na kterých se hodnotil

čas nečinnosti a ztracené hovory. Jako první je měření s generovánými dvěma hovory každou sekundu a maximálním počtem deseti obsluhovanými hovory. Pak následuje měření s generovánými čtyřmi hovory každou sekundu a maximálním počtem deseti obsluhovanými hovory. A jako poslední je měření s generovánými čtyřmi hovory každou sekundu a maximálním počtem třiceti obsluhovanými hovory. Do každé skupiny jsme přidali tabulku s výsledky testovacích hovorů, které jsme prováděli, abychom mohli lépe určit dobu nečinnosti a ztráty hovorů při daném zatížení ústředny. První typ měření s generovánými dvěma hovory každou sekundu a maximálním počtem deseti obsluhovanými hovory. Delká hovoru je osm sekund. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu.

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	SuccessfulCall	Failed Call	Response Time	Call Length
16:50:54	0:00:00	2	0	0	0	0	00:000	00:000
16:51:52	0:00:57	2	1.094	63	56	7	00:296	07:684
16:51:52	0:00:57	2	1.094	63	56	7	00:296	07:684

Tabulka 18. Výsledky hodnoceného hovoru dvě volání každou sekundu s maximem deset



Obrázek 28: Graf vytížení sítě v okamžiku přerušení spojení pro dva hovory každou sekundu s maximem deset.

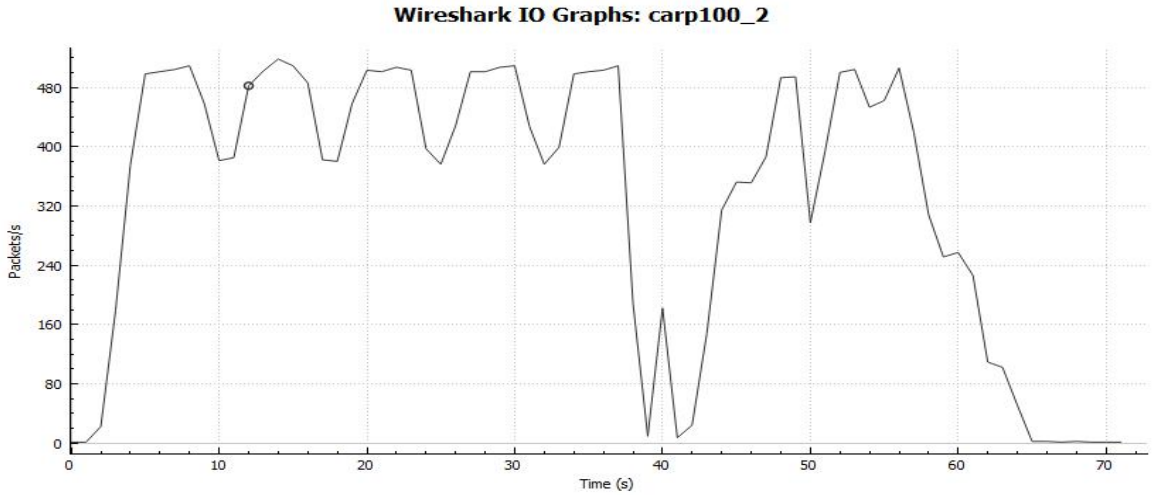
Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
carp100_01	36,62	40,62	4	7
2	39,32	43,72	4,4	7
3	37,55	42,42	4,87	6
4	40,23	44,16	3,93	8
5	41,2	46,63	5,43	7
6	39,42	44,31	4,89	9
7	35,72	40,84	5,12	7
8	36,19	41,56	5,37	8
9	37,59	41,48	3,89	9
10	37,01	41,38	4,37	9
průměrná hodnota	38,09	42,71	4,63	8

Tabulka 19. Výsledky průměrného hodnocení pro dva hovory každou sekundu s maximem deset.

Dalším druhým typem měření s generovánými čtyřmi hovory každou sekundu a maximálním počtem deseti obsluhovanými hovory. Delká hovoru je osm sekund. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu.

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	Successfu l Call	Failed Call	Response Time	Call Length
17:15:45	0:00:00	4	0	0	0	0	00:000	00:000
17:16:45	0:01:00	4	1,283	77	68	6	00:127	07:095
17:16:47	0:01:02	4	1,231	77	71	6	00:127	07:092
17:16:47	0:01:02	4	1,231	77	71	6	00:127	07:092

Tabulka 20: Výsledky hodnoceného hovoru čtyř volání každou sekundu s maximem deset.



Obrázek 29: Graf vytížení sítě v okamžiku přerušení spojení pro čtyři hovory každou sekundu s maximem deset.

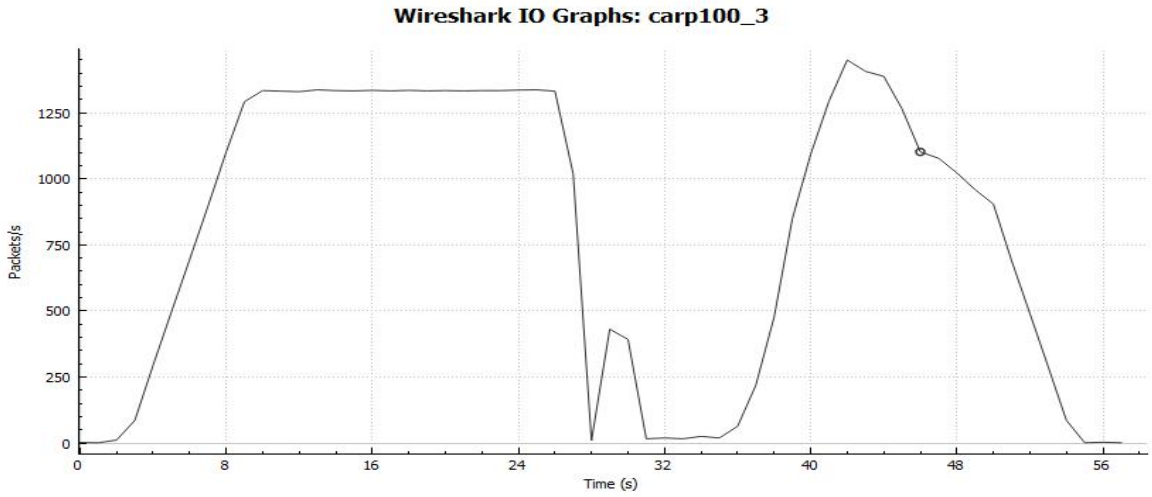
Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
carp100_02	40,81	42,82	2,01	6
2	43,64	46,92	3,28	6
3	41,46	45,6	4,14	7
4	39,78	43,21	3,43	9
5	45,61	48,5	2,89	6
6	40,18	42,61	2,43	6
7	42,71	45,21	2,5	3
8	43,65	47,14	3,49	5
9	40,59	44,2	3,61	6
10	41,83	44,78	2,95	8
průměrná hodnota	42,03	45,10	3,07	6

Tabulka 21: Výsledky průměrného hodnocení pro čtyři volání každou sekundu s maximem deset.

A třetím typem měření s generovánými čtyřmi hovory každou sekundu a maximálním počtem třiceti obsluhovanými hovory. Delká hovoru je osm sekund. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu.

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	SuccessfulCall	Failed Call	Response Time	Call Length
17:20:21	0:00:00	4	0	0	0	0	00:000	00:000
17:21:13	0:00:52	4	2,90	152	129	23	00:409	08:177
17:21:13	0:00:52	4	2,90	152	129	23	00:409	08:177

Tabulka 22: Výsledky hodnoceného hovoru čtyř volání každou sekundu s maximem třicet.



Obrázek 30: Graf vytížení sítě v okamžiku přerušení spojení pro čtyři hovory každou sekundu s maximem třicet.

Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
Elastix_04	30,49	36,24	5,75	23
2	32,24	38,54	6,30	18
3	34,54	41,77	7,23	23
4	33,69	39,08	5,39	23
5	35,12	40,31	5,19	19
6	33,67	40,10	6,43	20
7	34,91	40,79	5,88	20
8	34,64	39,56	4,92	19
9	33,36	39,08	5,72	21
10	36,12	41,93	5,81	18
průměrná hodnota	33,88	39,74	5,86	20

Tabulka 23: Výsledky průměrného hodnocení pro čtyř volání každou sekundu s maximem třicet.

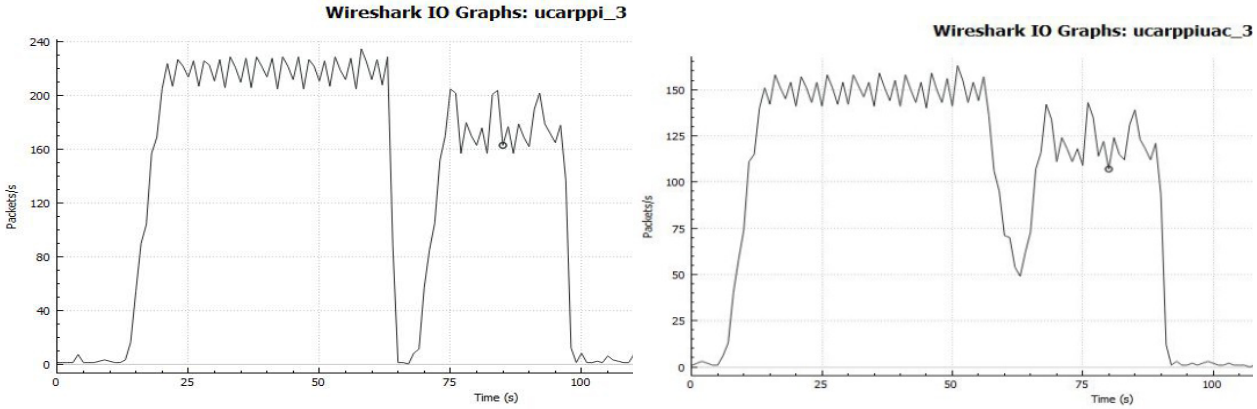
V druhé fázi jsou zde zvěřejněny výsledky hodnocení za pomoci vytvoření telefonního spojení mezi ústřednou a dvěma účastníky s použitím RTP protokolu pro přenos dat. Do každé skupiny se přidala tabulku s výsledky testovacích hovorů, které jsme prováděli, abychom mohli lépe určit dobu nečinnosti a ztráty hovorů při daném zatížení ústředny. U těchto měření je důležité povšimnout si sloupce aktuální hovory (current call). Jsou to

hovory, které byly přerušeny ve stavu jejich spojení a přenosu dat v okmažiku ukončení testování. Tyto hovory se nezapočítávají do ztacených hovorů, jelikož nastali až po přepnutí. Další částí, které je dobré si povšinout, je porovnání graf výpisu stavu sítě na straně UAC (vpravo) a UAS (vlevo). Je zde patrný rozdíl v poklesu generovaných hovrů na straně UAC proti přijímací straně UAS, kde pokles není tak patrný. Vyhodnocení neaktivního času a ztracených hovrů je z generující UAC strany. Další rozdílností je délka hovoru, pro toto testování jsme si vytvrořili vlastní testovací scénaře, kde máme dobu přenosu dat osm sekund, poté jsem nastavil dobu čekání mezi vyzváněním a přijmutím hovoru jednu sekundu, abychom test přiblížili reálným podmínkám. Při ukončení hovoru je nastavena další pauza čtyři sekundy. Proto se celková doba hovoru prodloužila na třináct sekund, viz tabulka 23.

První typ měření je ukázka spojení mezi dvěma klienty a VoIP ústřednou s generovánými dvěma hovory každou půl sekundu. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu.

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	Current Call	Successfu l Call	Failed Call	Call Length
15:43:10	0:00:00	10	0	0	0	0	0	00:000
15:44:10	0:00:59	10	0.53	32	6	26	0	13:032
15:45:10	0:02:00	10	0.37	45	4	40	1	13:192
15:45:46	0:02:35	10	0.29	45	4	40	1	13:192

Tabulka 24: Výsledky hodnoceného hovoru dvou volání každou půl sekundu.



Obrázek 31: Graf vytížení sítě v okamžiku přerušení spojení mezi dvěma účastníky pro dva hovory každou půl sekundu. Vlevo strana UAS a vpravo UAC.

Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
ucarppi_3	64,72	68,91	4,19	1
2	55,21	59,29	4,08	2
3	67,23	71,4	4,17	1
4	67,01	70,97	3,96	1
5	58,82	63,19	4,37	1
6	59,22	63,2	3,98	2
7	64,67	68,8	4,13	3
8	53,97	58,05	4,08	1
9	69,34	73,21	3,87	1

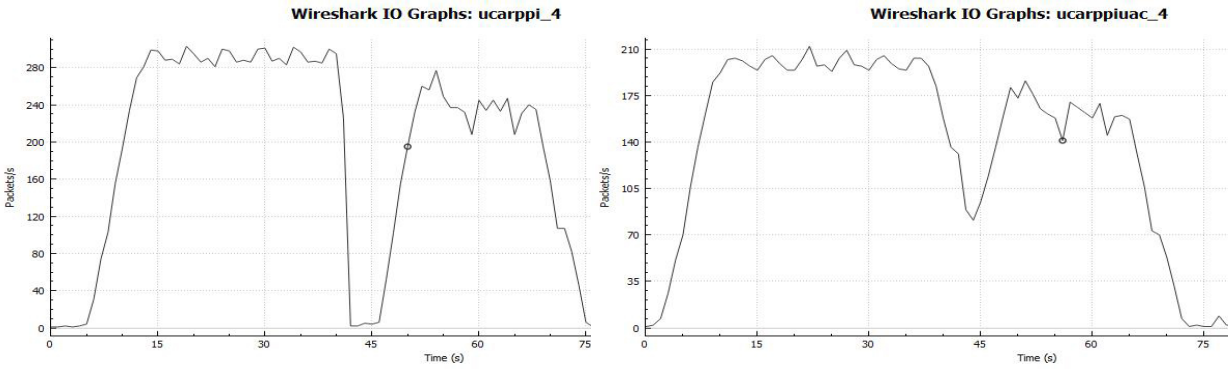
10	68,03	71,82	3,79	1
průměrná hodnota	62,82	66,88	4,06	1

Tabulka 25: Výsledky průměrného hodnocení pro dva hovory každou půl sekundu.

Dalším druhým typem měření je ukázka spojení mezi dvěma klienty a VoIP ústřednou s generováným jedním hovorem každou sekundu. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	Current Call	Successful Call	Failed Call	Call Length
15:50:59	0:00:00	10	0	0	0	0	0	00:000
15:51:59	0:01:00	10	0,68	41	17	24	0	13:030
15:52:47	0:01:47	10	0,41	45	7	37	1	13:544

Tabulka 26: Výsledky hodnoceného jednoho hovoru každou sekundu.



Obrázek 32: Graf vytížení sítě v okamžiku přerušení spojení mezi dvěma účastníky pro jeden hovor každou sekundu. Vlevo strana UAS a vpravo UAC.

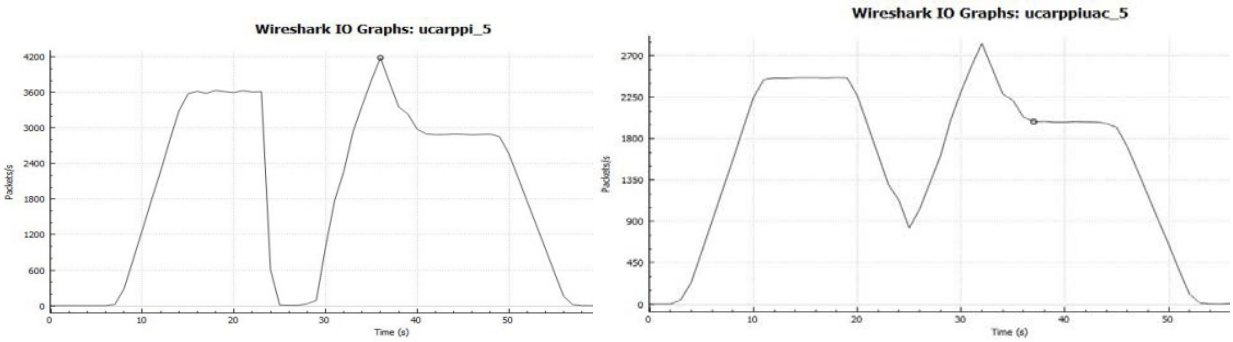
Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
ucarppi_4	41,75	46,25	4,5	1
2	45,67	49,12	3,45	2
3	39,45	44,12	4,67	2
4	49,35	53,3	3,95	2
5	42,56	47,79	5,23	1
6	44,12	49,28	5,16	1
7	42,67	47,32	4,65	1
8	39,54	43,83	4,29	2
9	40,55	45,23	4,68	2
10	40,86	44,98	4,12	1
průměrná hodnota	42,65	47,12	4,47	2

Tabulka 27: Výsledky průměrného hodnocení pro jeden hovor každou sekundu.

Posledním typem měření je ukázka spojení mezi dvěma klienty a VoIP ústřednou s generováními pěti hovory každou sekundu. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu.

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	Curent Call	Successfu l Call	Failed Call	Call Length
15:57:33	0:00:00	10	0	0	0	0	0	00:000
15:58:33	0:00:59	10	4,89	360	80	270	10	13:728
15:58:45	0:01:11	10	5,06	360	80	270	10	13:728

Tabulka 28: Výsledky hodnoceného hovoru pěti volání každou sekundu



Obrázek 33: Graf vytížení sítě v okamžiku přerušení spojení mezi dvěma účastníky pro pět hovorů každou sekundu. Vlevo strana UAS a vpravo UAC.

Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
ucarppi_5	24,18	28,87	4,69	10
2	30,42	36,92	4,78	9
3	33,47	38,86	4,92	9
4	30,26	34,54	4,28	10
5	32,59	37,27	4,68	8
6	33,51	37,56	4,05	8
7	37,58	42,95	4,89	7
8	37,19	42,22	4,77	9
9	35,61	40,52	4,91	7
10	35,82	40,45	4,63	9
průměrná hodnota	33,06	38,02	4,66	9

Tabulka 29: Výsledky průměrného hodnocení pro pět hovorů každou sekundu.

6.3 Testování systému Flip1405

Flip1405 je další z mnoha možných řešení pro zajištění vysoké dostupnosti. Tento systém vznikl jako open source skript, který byl a je stále upravován pro různá možná řešení, dle potřeb. Flip1405 byl prve při svém vzniku používán jen pro monitorování stavu zařízení a poskytování přenosné virtuální IP adresy, kdy na poskytovanou virtuální adresu jsou přiřazeny všechny poskytované služby zařízením. V případě zjištění, že primární zařízení je mimo provoz, automaticky převezme veškerý provoz a virtuální adresu záložní zařízení. Tyto jeho funkcionality pracují a poskytuje virtuální IP adresu mezi dvěma zařízeními, jak jsme odzkoušeli a zjistili. Bohužel, jsme jej nemohli otestovat ve schopnostech zajištení vysoké dostupnosti VoIP ústředny, jelikož se jedná pouze o řešení pro poskytování dané virtuální IP adresy. Pro správnou funkčnost zajištění vysoké dostupnosti je zapotřebí, aby poskytoval správu jednotlivých systémů v případě selhání hlavního zařízení. Tuto volbu bohužel neposkytuje, a tudíž v případě selhání hlavního

serveru převede virtuální IP adresu na záložní zařízení, ale neprovede rychlou inicializaci nebo restart systémů. Tato inicializace je důležitá pro přenačtení IP adresy poskytovanou jinou síťovou kartou. Konfigurace a správa tohoto systému není nikterak složitá a patří k těm nejrychlejším v případě rychlého nasazení. Pro komplexnost je posléze nutná konfigurace zabezpečeného shh spojení mezi zařízeními, jak pro přenost dat systémemu rsync a i vzájemnou komunikaci.

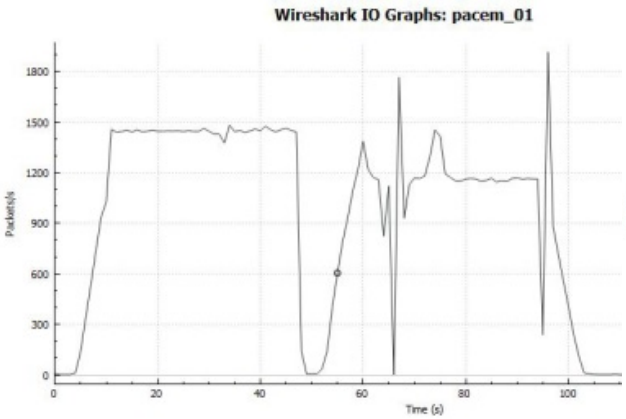
6.4 Testování systému Pacemaker a DRBD

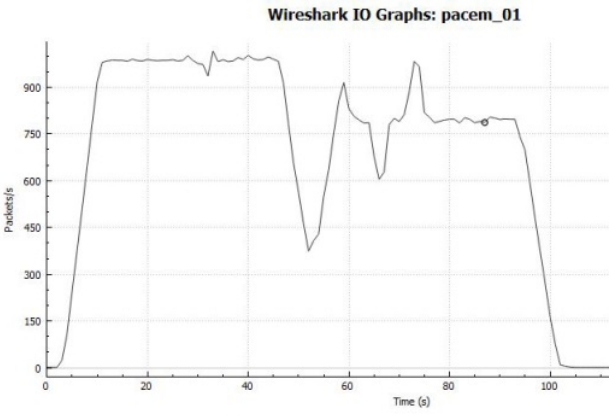
Posledním systémem zajišťujícím vyskou dostupnou, který se použil, je pacemaker. Je to obdoba systému heartbeat pro monitorování stavu systému a poskytování virtuální IP adresy. Jeho podobnost se systémem heartbeat je způsobena začínajícím společným vývojem. Abychom zjistili, zda jsou si opravdu tak podobní, provedli jsme několik testování jako v předchozích podkapitolách. Hodnocení se provedlo hlavně pro druhou fázi testování a to pro hodnocení za pomoci vytvoření telefonního spojení mezi dvěma účastníky a ústřednou s použitím RTP protokolu, které jsou zde uvedeny. Do každé skupiny hodnocení jsme opět přidali tabulku s výsledky testovacích hovorů, které jsme prováděli, abychom mohli lépe určit dobu nečinnosti a ztráty hovorů při daném zatížení ústředny. Jak se již uvedlo, u těchto měření je důležité povšimnout si sloupce aktuální hovory (current call). Jsou to hovory, které byly přerušeny ve stavu jejich spojení a přenosu dat v okmažiku ukončení testování. Tyto hovory se nezapočítávají do ztacených hovorů, jelikož nastali až po přepnutí. Další částí, které je dobré si povšinout, je porovnání graf výpisu stavu sítě na straně UAC (vpravo) a UAS (vlevo). Je zde patrný rozdíl v poklesu generovaných hovrů na straně UAC proti přijímací straně UAS, kde pokles není tak patrný. Vyhodnocení neaktivního času a ztracených hovrů je z generující UAC strany. Další rozdílností je délka hovoru, pro toto testování jsme si vytvrořili vlastní testovací scénaře, kde máme dobu přenosu dat osm sekund, poté jsem nastavil dobu čekání mezi vyzváněním a přijmutím hovoru jednu sekundu, abychom test přiblížili reálným podmínkám. Při ukončení hovoru je nastavena další pauza čtyři sekundy, proto se celková doba hovoru prodloužila na třináct sekund, viz tabulka 29.

První typ měření je ukázka spojení mezi dvěma klienty a VoIP ústřednou s generovánými dvěma hovory každou půl sekundu. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu.

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	Current Call	SuccessfulCall	Failed Call	Call Length
18:29:38	0:00:00	10	0	0	0	0	0	00:000
18:30:38	0:01:00	10	3.65	219	73	141	5	12:654
18:31:38	0:02:00	10	2.69	323	32	282	9	13:221
18:31:58	0:02:19	10	2.30	323	32	282	9	13:221

Tabulka 30: Výsledky hodnoceného hovoru dvou volání každou půl sekundu.





Obrázek 34: Graf vytížení sítě v okamžiku přerušení spojení mezi dvěma účastníky pro dva hovory každou půl sekundu. Vlevo strana UAS a vpravo UAC.

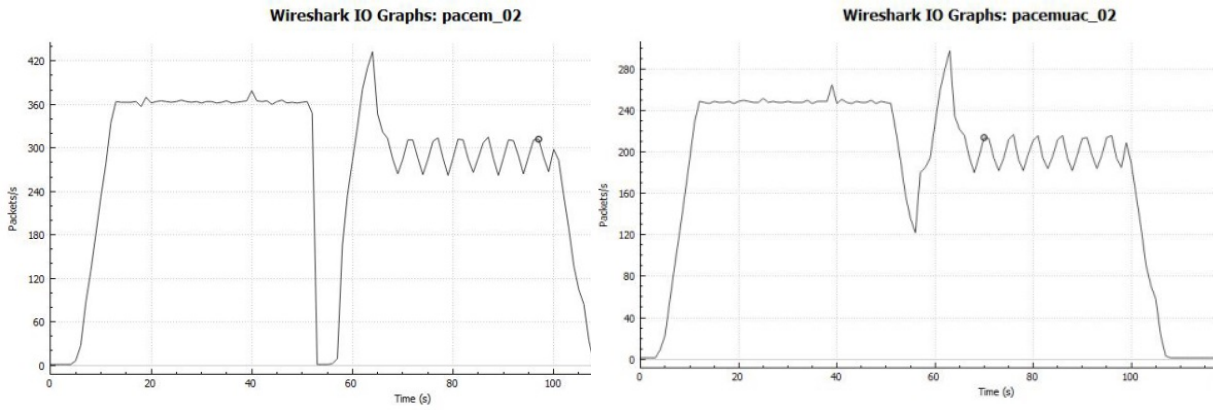
Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
pacem_01	48,06	52,03	3,97	9
2	47,75	51,99	4,24	12
3	52,33	56,71	4,38	11
4	50,59	54,48	3,89	9
5	49,59	53,34	3,75	7
6	50,73	55,6	4,87	9
7	52,4	55,98	3,58	9
8	52,6	56,48	3,88	13
9	49,57	53,91	4,34	10
10	48,51	52,59	4,08	7
průměrná hodnota	50,21	54,31	4,10	10

Tabulka 31: Výsledky průměrného hodnocení pro dva hovory každou půl sekundu.

Dalším druhým typem měření je ukázka spojení mezi dvěma klienty a VoIP ústřednou s generováným jedním hovorem každou sekundu. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu.

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	Current Call	SuccessfulCall	Failed Call	Call Length
18:39:29	0:00:00	10	0	0	0	0	0	00:000
18:40:29	0:00:59	10	0.93	56	17	39	0	13:029
18:41:29	0:02:00	10	0.71	86	8	77	1	13:278
18:41:47	0:02:18	10	0.62	86	8	77	1	13:278

Tabulka 32: Výsledky hodnoceného jednoho volání každou sekundu.



Obrázek 35: Graf vytížení sítě v okamžiku přeřušení spojení mezi dvěma účastníky pro jeden hovor každou sekundu. Vlevo strana UAS a vpravo UAC.

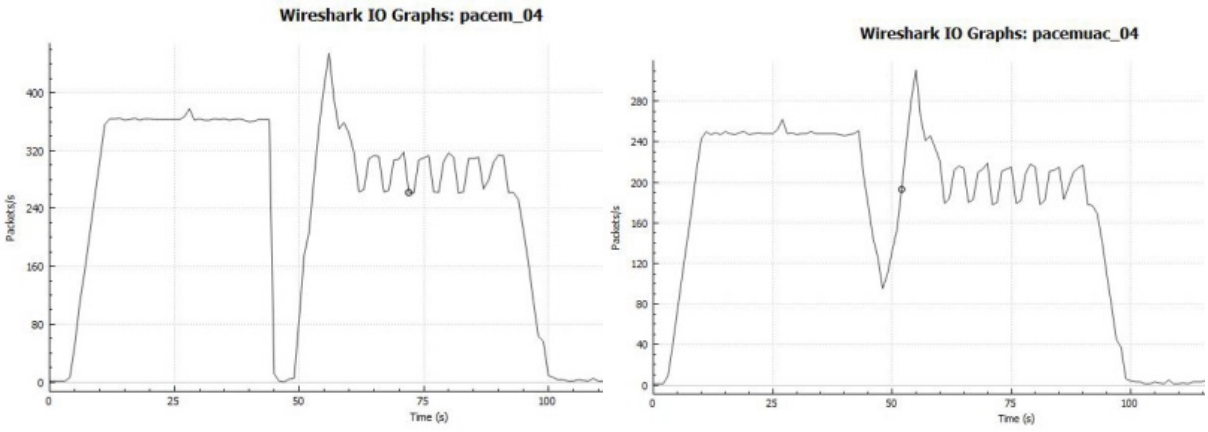
Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
pacem_02	52,53	57	4,47	1
2	55,45	60,32	4,87	2
3	53,48	58,71	5,23	1
4	49,89	55,37	5,48	1
5	52,57	57,75	5,18	1
6	53,73	58,49	4,76	2
7	52,19	56,72	4,53	3
8	54,69	59,31	4,62	2
9	52,35	57,1	4,75	1
10	53,71	58,58	4,87	1
průměrná hodnota	53,06	57,94	4,88	2

Tabulka 33: Výsledky průměrného hodnocení pro jeden hovor každou sekundu.

Předposledním typem měření je ukázka spojení mezi dvěma klienty a VoIP ústřednou s generovánými čtyřmi hovory každé dvě sekundu. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu.

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	Current Call	SuccessfulCall	Failed Call	Call Length
18:46:40	0:00:00	10	0	0	0	0	0	00:000
18:47:40	0:00:59	10	0.90	54	22	32	0	13:033
18:48:40	0:01:59	10	0.65	79	8	70	1	13:306
18:49:07	0:02:27	10	0.53	79	8	70	1	13:306

Tabulka 34: Výsledky hodnoceného hovoru čtyř volání každé dvě sekundy.



Obrázek 36: Graf vytížení sítě v okamžiku přerušení spojení mezi dvěma účastníky pro čtyři hovory každé dvě sekundy. Vlevo strana UAS a vpravo UAC.

Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
pacem_04	44,07	49,05	4,98	1
2	45,67	50,71	5,04	2
3	44,78	49,62	4,84	3
4	47,39	52,45	5,06	2

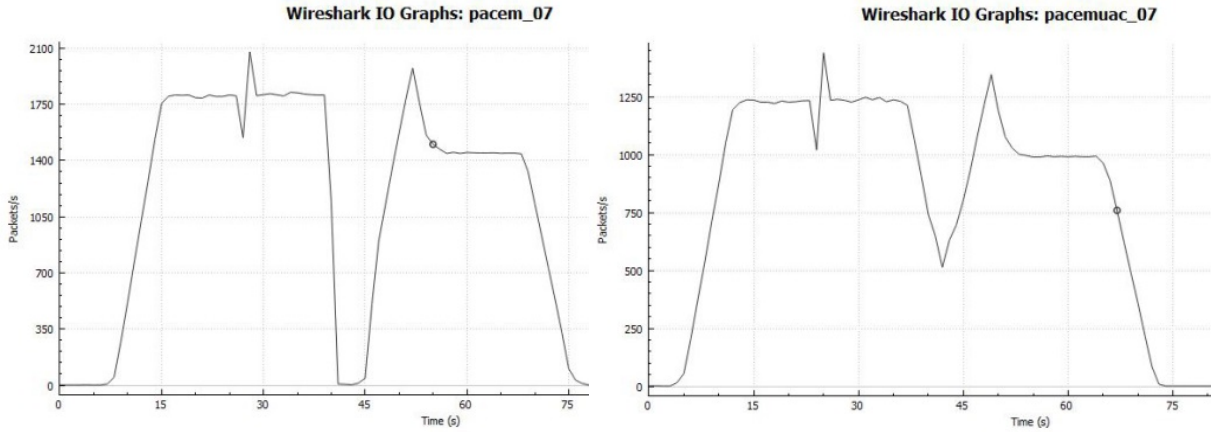
5	42,58	48,01	5,43	3
6	43,59	48,58	4,99	2
7	44,72	50,15	5,43	1
8	45,26	50,49	5,23	1
9	46,23	51,41	5,18	3
10	44,72	49,93	5,21	2
průměrná hodnota	44,90	50,04	5,14	2

Tabulka 35: Výsledky průměrného hodnocení pro čtyři hovory každé dvě sekundy.

Posledním čtvrtým typem měření, je ukázka spojení mezi dvěma klienty a VoIP ústřednou s generovánými pěti hovory každou sekundu. Časové údaje v tabulce jsou v sekundách a údaje pro call rate jsou hovory za sekundu.

Current Time	Elapsed Time	Target Rate	Call Rate	Total Call Created	Current Call	SuccessfulCall	Failed Call	Call Length
18:59:16	0:00:00	10	0	0	0	0	0	00:000
19:00:16	0:01:00	10	4.26	256	99	157	0	13:042
19:01:02	0:01:45	10	2.55	271	41	225	5	13:506

Tabulka 36: Výsledky hodnoceného hovoru pěti volání každou sekundu.



Obrázek 37: Graf vytížení sítě v okamžiku přerušení spojení mezi dvěma účastníky pro pět hovorů každou sekundu. Vlevo strana UAS a vpravo UAC.

Jméno/ID měření	Začátek nečinnosti [s]	Konec nečinnosti [s]	Doba nečinnosti [s]	Ztracené hovory
pacem_07	40,61	44,96	4,35	5
2	41,34	46,21	4,87	6
3	39,84	45,09	5,25	10
4	39,94	44,48	4,54	8
5	41,58	46,25	4,87	9
6	40,87	45,15	4,68	6
7	42,51	47,07	4,56	7
8	41,59	46,1	4,69	6
9	42,61	47,18	4,59	10
10	43,73	48,46	4,73	8
průměrná hodnota	41,46	46,10	4,71	8

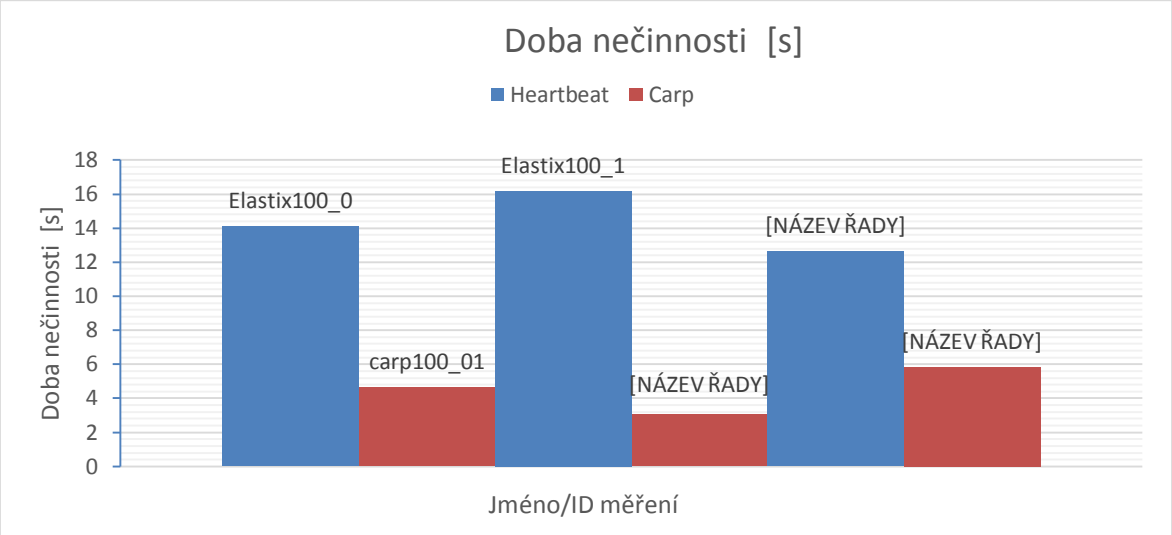
Tabulka 37: Výsledky průměrného hodnocení pro pět hovorů každou sekundu.

6.5 Výsledné zhodnocení

Výsledné zhodnocení a porovnání použitých systémů se provedlo pro první a druhou fázi testování a to pro hodnocení doby nečinosti a počtu ztracených hovorů pomocí zasílání SIP zpráv na předdefinovanou funkcionalitu echo na VoIP ústředně. V druhé fázi jsou zde zvěřejněny výsledky hodnocení za pomoci vytvoření telefonního spojení mezi dvěma účastníky a ústřednou s použitím RTP protokolu. Výsledné zhodnocení je a porovnání je provedeno pro průměrné hodnoty, které byly provedeny pro jednotlivé typy měření podle rozdělení do jednotlivých fází hodnocení. Jako první jsou uvedeny výsledky pro první fázi testování, kde jsou mezi sebou porovnány systémy heartbeat a ucarp. Zařazení systému pacemaker do porovnání v první fázi testování je zavádějící, jelikož jeho průměrné výsledky byly shodné se systémem heartbeat. V následující tabulce je souhrné porovnání průměrných hodnot podle typu měření jednotlivých systémů. Hodnoty v procentech jsou pro dokreslení celkového přehledu, avšak musíme vzít v úvahu rozdílné počty celkových hovorů nebo celkové doby měření. Proto je hlavním kritériem ve výsledném zhodnocení doba nečinosti a ztracené hovory.

Jméno/ID měření	Celková délka měření [s]	Doba nečinnosti [s]	Doba nečinnosti [%]	Celkový počet hovorů	Ztracené hovory	Ztracené hovory [%]
Elastix100_0	93	14,1	15,16	96	9	9,38
Elastix100_1	84	16,18	19,26	95	10	10,53
Elastix100_2	93	12,62	13,57	96	10	10,42
carp100_01	57	4,63	8,12	63	8	12,70
carp100_02	62	3,07	4,95	77	6	7,79
carp100_04	52	5,86	11,27	152	20	13,16

Tabulka 38: Výsledné porovnání první fáze.



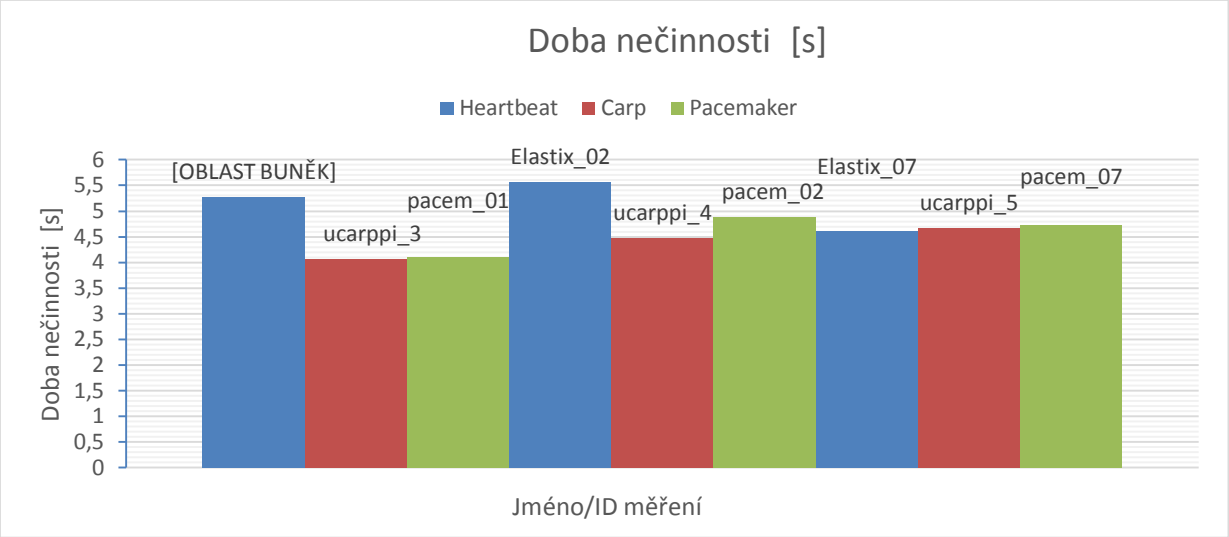
Obrázek 38: Graf doby nečinnosti

V druhé části výsledného zhodnocení a porovnání jsou uvedeny výsledky pro druhou fázi testování, kde jsou porovnány systémy heartbeat, ucarp a i pacemaker. V této fázi testování, již byly výsledky testování systému pacemaker rozdílné a proto je vý výsledném

zhodnocení. V následující tabulce je souhrné porovnání průměrných hodnot podle typu měření jednotlivých systémů. Hodnoty v procentech jsou pro dokreslení celkového přehledu, avšak musíme vzít v úvahu rozdílné počty celkových hovorů nebo celkové doby měření. Proto je hlavním kritériem ve výsledném zhodnocení doba nečinosti a ztracené hovory.

Jméno/ID měření	Celková délka měření [s]	Doba nečinnosti [s]	Doba nečinnosti [%]	Celkový počet hovorů	Ztaracené hovory	Ztaracené hovory [%]
Elastix_01	99	5,27	5,32	199	4	2,01
Elastix_02	102	5,56	5,45	55	3	5,45
Elastix_04	119	4,8	4,03	154	3	1,95
Elastix_07	131	4,61	3,52	278	12	4,32
ucarppi_3	155	4,06	2,62	41	1	2,44
ucarppi_4	107	4,47	4,18	38	2	5,26
ucarppi_5	71	4,66	6,56	280	9	3,21
pacem_01	139	4,1	2,95	231	10	4,33
pacem_02	138	4,88	3,54	78	2	2,56
pacem_04	147	5,14	3,50	71	2	2,82
pacem_07	105	4,71	4,49	230	8	3,48

Tabulka 39: Výsledné porovnání druhé fáze.



Obrázek 39: Graf doby nečinnosti

7 Závěr

Tématem diplomové práce byla pomocí open source softwarových prostředků realizace VoIP systému s vysokou dostupností a její následné zhodnocení vlastností pro porovnání s komerčně dostupnými systémy. V úvodu práce jsem zpracoval rešerši se zaměřením na tuto problematiku zajištění vysoké dostupnosti, ve které hodnotím a porovnávám získané studie zabývající se tímto tématem. Před samotnou realizací systému vysoké dostupnosti jsem se snažil utřídit myšlenky a pojmy ohledně vysoké dostupnosti, které mi samostatně bez spravného kontextu přišly zmatečné a zavádějící. Jeden z důležitých bodů bylo také uvědomění si principů a spojení teoretických vědomostí s funkčním řešením. Nicméně zpracovávání získaných infomací použitých v práci mi pomohly si uvědomit hlavní body, kterými se dále zabývat. Následně jsem začal střádat infomace o možných řešeních, která by se dala použít. Jak jsem zjistil, možností řešení a systémů je v této problematice nepřeberné množství. Ať už se podívám na již předpřipravená komplexní řešení, nebo na samostatné softwarové prostředky, kdy při jejich správném spojení vznikne podobný výsledek. Proto jsem vybral několik open source systémů, které se v této oblasti používají již delší dobu anebo byla přímo pro toto vyvíjena. Problém nastal v situaci, kdy jsem získával infomace o komerčních řešeních, abych mohl dle čeho hodnotit. Jelikož u komerčních systémů jsou přesné infomace o jejich výkonosti skryty.

Po setřídění informací mě čekala následná realizace podle zadání moji práce. Realizaci systémů zajišťující vysokou dostupnost jsem provedl ve virtuálním prostředí, kde jsem byl limitován pouze výkonností hardwaru pro zajištění virtualizace. Konfigurace jednotlivých systémů byla dalším ze základních problematik, jelikož velká většina těchto systémů nebyla aktualizována a proto bylo nutné postupovat po jednotlivých bodech a řešit souvisle vznikající komplikace. Na základě dodržování tohoto postupu jsem zprovoznil čtyři systémy pro zajištění vysoké dostupnosti VoIP systémů. Během realizace jsem se také seznámil se systémem asterisk, zajišťujícím VoIP služby. Po konfiguraci open source systémů, VoIP ústředny a také celkové infrastruktury jsem mohl začít hodnocením použitých systémů.

Zhodnocení je jedním ze základních a důležitých úkonů v celé diplomové práci. Hodnocení jsem nejprve chtěl provádět s pomocí externího zařízení v podobě raspberry pi, ale po několika testováních a průběžných rekonfigurací stále docházelo k výpadkům v generovaných hovorech a zamrzání systému. Po zjištění těchto problémů jsem hodnocení provedl dvěma způsoby. První pomocí generování zpráv mezi ústřednou a účastníkem a druhý způsob byl pomocí vytvoření spojení přes ústřednu a dva učastníky. Abych dosáhl přesnějších hodnot, provedl jsem daná měření několikrát po sobě. Na základě těchto měření, jsem dospěl ke konečným výledkům moji práce. Ze čtyř systémů se jeden systém flip1405 pro hodnocení nemohl použít, jelikož jeho funkcionalita neumožnila správů služeb zavyslích na vysoké dostupnosti a poskytovaní virtuální adrese. Ze zbylých systémů se dva systémy heartbeat a pacemaker podobají, proto jsem zhodnotil i jejich implemntaci a rychlost použití. Podle mého hodnocení bych jako nejlepší systém označil carp, je rychlý, v případě výpadku a umožňuje rychlou a snadnou konfiguraci a lze mu určit správu jakýchkoliv služeb v případě vysoké dostupnosti. Na druhé místo bych zařadil heartbeat, v některých situcích byl rychlejší než zbývající pacemaker, který už není ani tolik využívaný s malou podporou konfigurace. Další výhodou heartbeat systému je jeho možnost propojení s drbd systémem a jejich vzájemná stabilita a komplexnost. V porovnání vůči komerčním systémům je použití open source systémů mnohem lepší

volbou, hlavně v rychlosti řešení vysoké dostupnosti, v případě použitého testovacího prostředí, ve kterém jsem prováděl hodnocení. Nevýhodou oproti komerčním systémům je jejich nepřipravená a složitější implementace a v případě selhání a využití funkcionalit vysoké dostupnosti dojde k přerušení VoIP služeb a přerušení probíhajích hovorů, oproti hot stanby řešením v komerční části systémů. Mezi jedno z kritérií je i zvážení pořizovacích nákladů, což v případě řešení pro nevytížené služby s možností krátkého výpadku služeb, je jasnou volbou open source řešení.

Cíle stanovené v úvodu mojí diplomové práce jsem se snažil splnit co nejpřesněji a nejlépe, jak bylo možné. Plnění jednotlivých bodů, která mi vznikla během zpracovávní, byla pro mne velice užitečná a naučná. Donutilo mě to se dívat na dané problematiky jiným pohledem, než jsem předtím praktikoval. Problematika, kterou jsem se zabýval, byla pro mě zajímavá a pestrá, hlavně poučná. Jelikož jsem si mohl ověřit jednotlivá řešení v praxi a tak zjistit, výhody open source softwarů a potvrdit je. Po možných dalších úvahách v možnostech rozšíření moji práce, je implementace do reálné sítě a ověření si tak, zdali vyhodnocení odpovídá i reálnému provozu.

8 Seznam literatury

[1] Vysoká dostupnost v relačních databázových systémech [online]. Brno, 2014 [cit. 2016-05-17]. Dostupné z: http://is.muni.cz/th/395769/fi_m/diplomova_prace_DolgikhM_25_05_2014.pdf. Masarykova univerzita. Vedoucí práce Doc. RNDr. Vlastislav Dohnal, Ph.D.

[2] Provozní zatížení v telekomunikacích. ČVUT PRAHA. Studijní skripta. české vysoké učení technické v Praze. Vedoucí práce Doc. Ing. František Křížovský, CSc.

[3] Self-Study: Designing High-Availability Services [online]. [cit. 2016-05-16]. Dostupné z: http://www.ciscopress.com/articles/article.asp?p=375501&seqNum=2

[4] Asterisk High Availability Design [online]. voip-info [cit. 2016-05-16]. Dostupné z: http://www.voip-info.org/wiki/view/Asterisk+High+Availability+Design

[5] HAAst [online]. www.telium.ca [cit. 2016-05-16]. Dostupné z: http://www.telium.ca/pages/products/haast/

[6] AVAILABILITY ENVIRONMENT CLASSIFICATIONS [online]. hrgresearch [cit. 2016-05-16]. Dostupné z: http://www.hrgresearch.com/pdf/AEC%20Defintions.pdf

[7] TELEFONNÍ ÚSTŘEDNY ASTERISK [online]. CESNET, 2008 [cit. 2016-05-17]. Dostupné z: http://www.ip-telefon.cz/data/downld/44.pdf. Vedoucí práce Ing. Miroslav VOZŇÁK, Ph.D.

[8] TESTOVÁNÍ ODOLNOSTI SIP PBX [online]. Praha, 2015 [cit. 2016-05-17]. Dostupné z: https://dspace.cvut.cz/handle/10467/61183. Diplomová práce. čvut. Vedoucí práce Ing. Kučerák Ján.

[9] Úvod do VoIP [online]. mbdata [cit. 2016-05-16]. Dostupné z: http://www.mbdata.cz/uvoddovoip.htm

[10] Jak funguje VOIP [online]. earchiv: Jiří Peterka [cit. 2016-05-16]. Dostupné z: http://www.earchiv.cz/b06/b0401003.php3

[11] Extrakce a zpracování hlasu z VOIP paketů [online]. Praha, 2015 [cit. 2016-05-17]. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/62991/F8-BP-2015-Kucera-Josef-thesis.pdf?sequence=1. Bakalářská práce. čvut. Vedoucí práce Mgr. Jan Starý, Ph.D.

[12] SIP [online]. cesnet: cesnet [cit. 2016-05-16]. Dostupné z: https://sip.cesnet.cz/cs/protokoly/sip

[14] Asterisk Architecture [online]. asterisk: asterisk [cit. 2016-05-16]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Asterisk+Architecture%2C+The+Big+Picture

[15] Asterisk PBX [online]. cesnet: cesnet [cit. 2016-05-16]. Dostupné z: https://sip.cesnet.cz/cs/swahw/asterisk

[16] SIPp [online]. SIPp: SIPp [cit. 2016-05-16]. Dostupné z: http://sipp.sourceforge.net/

[17] High availability with the Distributed Replicated Block Device [online]. ibm: ibm [cit. 2016-05-16]. Dostupné z: http://www.ibm.com/developerworks/library/l-drbd/

[18] Distributed Replicated Block Device [online]. abclinuxu: abclinuxu, 2011 [cit. 2016-05-16]. Dostupné z: http://www.abclinuxu.cz/blog/Atol/2011/6/atol-distributed-replicated-block-device

[19] High availability with the Distributed Replicated Block Device [online]. ibm: ibm [cit. 2016-05-16]. Dostupné z: http://www.ibm.com/developerworks/library/l-drbd/
[20] What is DRBD [online]. DRBD: DRBD, 2011 [cit. 2016-05-16]. Dostupné z: http://www.drbd.org/en/
[21] CARP your way to high availability [online]. linux: linux [cit. 2016-05-16]. Dostupné z: https://www.linux.com/news/carp-your-way-high-availability
[22] Installing GlusterFS - a Quick Start Guide [online]. gluster: gluster [cit. 2016-05-16]. Dostupné z: http://gluster.readthedocs.io/en/latest/Install-Guide/Overview/
[23] GlusterFS [online]. support.dce.felk.cvut.cz: cvut [cit. 2016-05-16]. Dostupné z: https://support.dce.felk.cvut.cz/mediawiki/index.php/GlusterFS
[24] Heartbeat [online]. linux-ha.: linux-ha. [cit. 2016-05-16]. Dostupné z: http://linux-ha.org/wiki/Heartbeat
[25] Heartbeat [online]. search enterpri selinux: search enterpri selinux [cit. 2016-05-16]. Dostupné z: http://searchenterpriselinux.techtarget.com/definition/Heartbeat
[26] Pacemaker [online]. clusterlabs: clusterlabs [cit. 2016-05-16]. Dostupné z: http://clusterlabs.org/wiki/Pacemaker
[27] HAAst [online]. telium [cit. 2016-05-16]. Dostupné z: http://www.telium.ca/pages/products/haast/
[28] High Availability for SARK [online]. sailpbx: sailpbx [cit. 2016-05-16]. Dostupné z: http://www.sailpbx.com/mediawiki/index.php/High_Availability_for_SARK
[29] System and method for fail-safe call survival for IP contact centers [online]. q-suite: q-suite [cit. 2016-05-16]. Dostupné z: http://www.q-suite.com/HA-call_survival-IP_contact_centers
[30] FreePBX High Availability [online]. freepbx: freepbx [cit. 2016-05-16]. Dostupné z: http://wiki.freepbx.org/display/FCM/FreePBX+High+Availability
[31] High Availability Features [online]. bicom systems: bicom systems [cit. 2016-05-16]. Dostupné z: http://www.bicomsystems.com/products/serverware
[32] Cisco Unified Border Element High Availability [online]. cisco: cisco [cit. 2016-05-16]. Dostupné z: http://www.cisco.com/c/en/us/support/docs/voice-unified-communications/unified-border-element/112095-cube-hsrp-config-00.html
[33] Why choose an Elastix Appliance? [online]. Elastix: Elastix [cit. 2016-05-16]. Dostupné z: http://blogs.elastix.org/en/2015/06/01/why-choose-an-elastix-appliance/
[34] Learn Wireshark [online]. Wireshark: Wireshark [cit. 2016-05-16]. Dostupné z: https://www.wireshark.org/#learnWS
[35] DESCRIPTION TCPdump [online]. tcpdump: tcpdump [cit. 2016-05-16]. Dostupné z: http://www.tcpdump.org/manpages/tcpdump.1.html
[36] Oracle VM Communities [online]. oracle: oracle [cit. 2016-05-16]. Dostupné z: http://www.oracle.com/technetwork/server-storage/virtualbox/learnmore/index.html
[37] Definition - What does Virtualization mean [online]. techopedia: techopedia [cit. 2016-05-16]. Dostupné z: https://www.techopedia.com/definition/719/virtualization
[38] How To Heartbeat [online]. imanudin: imanudin [cit. 2016-05-16]. Dostupné z: https://imanudin.net/2015/03/18/how-to-configure-online-failoverfailback-on-centos-6-using-heartbeat/

9 Seznam obrázků

10 Seznam tabulek

11 Seznam příloh

12 Přílohy

12.1 Příloha 9: Konfigurace statického směrování

```
DEVICE="eth0"
NM_CONTROLLED="yes"
ONBOOT=yes
HWADDR=20:89:84:c8:12:8a
TYPE=Ethernet
BOOTPROTO=static
NAME="System eth0"
UUID=5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03
IPADDR= 2001:db8::c0ca:1eaf
NETMASK=255.255.255.0
IPADDR=192.168.1.140
GATEWAY=192.168.1.1
DNS2=8.8.8.8
DNS1=192.168.1.1
```

12.2 Příloha 2: Editace souboru sip.conf

```
[general]
bindport = 5060
context = malinovnik
disallow=all
allow=ulaw
allow=alaw
dtmfmode=auto
allowguest=yes
bindaddr=192.168.1.135
videosupport=yes
srvlookup=yes
directrtpsetup=no
directmedia=no
nat=yes
recordhistory=yes

[105]
host=dynamic
type=friend
username=105
context=malinovnik
secret=malina
```

12.3 Příloha 3: Výpis konfigurace extensions.conf

```
[general]
static=yes
writeprotect=yes

[malinovnik]
exten => 105,1,Dial(SIP/105)
exten => 105,2,Hangup()
exten => 106,1,Dial(SIP/106)
exten => 106,2,Hangup()
```

```
;prehrani nadyktovaneho vzkazu po zvednuti
exten => 100,1,Answer
exten => 100,n,Playback(demo-echotest)
exten => 100,n,Echo()
exten => 100,n,Hangup()
;hlaska po zvednuti sluchatka
exten => 200,1,Answer()
exten => 200,2,Playback(hello-world)
exten => 200,3,Hangup()
```

12.4 Příloha 4: Scénář registrace klientů

```
<scenario name="registration">
<send retrans="500">
<![CDATA[
REGISTER sip:[field1] SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
Max-Forwards: 70
From: "sipp" <sip:[field0]@[field1]>;tag=[call_number]
To: "sipp" <sip:[field0]@[field1]>
Call-ID: reg///[call_id]
CSeq: 2 REGISTER
Contact: <sip:sipp@[local_ip]:[local_port]>
Expires: 3600
Content-Length: 0
User-Agent: SIPp
[field2]
]]>
</send>
</scenario>
```

12.5 Příloha 5: Invite pro UAC

```
  <send retrans="500">
    <![CDATA[
     INVITE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
     Via: SIP/2.0/[transport]
     [local_ip]:[local_port];branch=[branch]
     From: sipp
     <sip:sipp@[local_ip]:[local_port]>;tag=[pid]SIPpTag09[call_nu
     mber]
     To: sut <sip:[service]@[remote_ip]:[remote_port]>
     Call-ID: [call_id]
     CSeq: 1 INVITE
     Contact: sip:sipp@[local_ip]:[local_port]
     Max-Forwards: 70
     Subject: Performance Test
     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 8 101
     a=rtpmap:8 PCMA/8000
     a=rtpmap:101 telephone-event/8000
     a=fmtp:101 0-11,16
    ]]>
</scenario>
```

12.6 Příloha 6: Scénář pro UAC s csv daty

```
  <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:
     <sip:[field0]@[local_ip]:[local_port]>;tag=[call_number]
     To: <sip:[field3]@[remote_ip]:[remote_port]>
     Call-ID: [call_id]
     CSeq: 2 INVITE
     Contact: sip:[field0]@[local_ip]:[local_port]
     [field2]
     Max-Forwards: 70
     Subject: Performance Test
     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 8 101
     a=rtpmap:8 PCMA/8000
     a=rtpmap:101 telephone-event/8000
     a=fmtp:101 0-11,16
    ]]>
  </send>

  <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]@[local_ip]:[local_port]>;tag=[call_number]
     To: <sip:[field3]@[remote_ip]:[remote_port]>[peer_tag_param]
     Call-ID: [call_id]
     CSeq: 3 ACK
     Contact: sip:[field0]@[local_ip]:[local_port]
     Max-Forwards: 70
     Subject: Performance Test
     Content-Length: 0
    ]]>
  </send>

  <nop>
```

```
    <action>
      <exec play_pcap_audio="pcap/g711a.pcap"/>
    </action>
  </nop>
  <pause milliseconds="8000"/>

  <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]@[local_ip]:[local_port]>;tag=[call_number]
     To: <sip:[field3]@[remote_ip]:[remote_port]>[peer_tag_param]
     Call-ID: [call_id]
     CSeq: 4 BYE
     Contact: sip:[field0]@[local_ip]:[local_port]
     Max-Forwards: 70
     Subject: Performance Test
     Content-Length: 0
    ]]>
  </send>
```

12.7 Příloha 7: Scénář pro UAS stranu

```
  <send>
    <![CDATA[
     SIP/2.0 180 Ringing
     [last_Via:]
     [last_From:]
     [last_To:];tag=[call_number]
     [last_Call-ID:]
     [last_CSeq:]
     Contact: <sip:[local_ip]:[local_port];transport=[transport]>
     Content-Length: 0
    ]]>
  </send>

<pause milliseconds="1000"/>

  <send retrans="500">
    <![CDATA[
     SIP/2.0 200 OK
     [last_Via:]
     [last_From:]
     [last_To:];tag=[call_number]
     [last_Call-ID:]
     [last_CSeq:]
     Contact: <sip:[local_ip]:[local_port];transport=[transport]>
     Content-Type: application/sdp
     Content-Length: [len]
     v=0
     o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
     s=-
     c=IN IP[media_ip_type] [media_ip]
```

```
     t=0 0
     m=audio [auto_media_port] RTP/AVP 8 101
     a=rtpmap:0 PCMU/8000
    ]]>
  </send>

  <recv request="ACK"
        optional="true"
        rtd="true"
        crlf="true">
  </recv>

  <send>
    <![CDATA[
     SIP/2.0 200 OK
     [last_Via:]
     [last_From:]
     [last_To:];tag=[call_number]
     [last_Call-ID:]
     [last_CSeq:]
     Contact: <sip:[local_ip]:[local_port];transport=[transport]>
      Content-Length: 0
    ]]>
  </send>
</scenario>
```

12.8 Příloha 8: Implementace dat do DRBD

```
$ - cd /replica
$ - tar -zcvf etc-asterisk.tgz /etc/asterisk/
$ - tar -zxvf etc-asterisk.tgz
$ - tar -zcvf var-lib-asterisk.tgz /var/lib/asterisk/
$ - tar -zxvf var-lib-asterisk.tgz
$ - tar -zcvf usr-lib-asterisk.tgz /usr/lib/asterisk/
$ - tar -zxvf usr-lib-asterisk.tgz
$ - tar -zcvf var-www.tgz /var/www/
$ - tar -zxvf var-www.tgz
$ - tar -zcvf var-spool-asterisk.tgz /var/spool/asterisk/
$ - tar -zxvf var-spool-asterisk.tgz
$ - tar -zcvf var-lib-mysql.tgz /var/lib/mysql/
$ - tar -zxvf var-lib-mysql.tgz
$ - tar -zcvf var-log-asterisk.tgz /var/log/asterisk/
$ - tar -zxvf var-log-asterisk.tgz
$ - tar -zcvf tftpboot.tgz /tftpboot/
$ - tar -zxvf tftpboot.tgz

$ - rm -rf /etc/asterisk/
$ - rm -rf /var/lib/asterisk/
$ - rm -rf /usr/lib/asterisk/
$ - rm -rf /var/spool/asterisk/
$ - rm -rf /var/lib/mysql/
$ - rm -rf /var/log/asterisk/
$ - rm -rf /tftpboot/
$ - rm -rf /var/www
```

```
$ - ln -s /repdata/etc/asterisk/ /etc/asterisk
$ - ln -s /repdata/var/lib/asterisk/ /var/lib/asterisk
$ - ln -s /repdata/usr/lib/asterisk/ /usr/lib/asterisk
$ - ln -s /repdata/var/spool/asterisk/ /var/spool/asterisk
$ - ln -s /repdata/var/lib/mysql/ /var/lib/mysql
$ - ln -s /repdata/var/log/asterisk/ /var/log/asterisk
$ - ln -s /repdata/var/www /var/www
$ - ln -s /repdata/tftpboot /tftpboot
```