VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

POROVNÁNÍ VÝKONNOSTI SIP IMPLEMENTACÍ V OPEN SOURCE PBX ASTERISK A KAMAILIO

COMPARISON OF SIP IMPLEMENTATIONS USED IN ASTERISK AND KAMAILIO OPEN SOURCE PBX'S

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jakub Rajj**

AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Pavel Šilhavý, Ph.D.**

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Jakub Rajj ***ID:*** 186451

Ročník: 2 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Porovnání výkonnosti SIP implementací v Open source PBX Asterisk a Kamailio

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte možnosti Open Source pobočkových ústředen Asterisk a Kamailio. Proveďte porovnání jejich funkcionalit. Realizujte výkonnostní testování pro porovnání alespoň tří LTS verzí PBX Asterisk vždy pro obě implementace protokolu SIP, tj. jak nativní, tak PJSIP ve scénáři SIP Proxy i B2BUA a výkonnostní testování PBX Kamailio. Zaměřte se jak na dosažitelný počet hovorů, tak na maximální počty transakcí, stabilitu a chybovost. Během testů zaznamenávejte vytížení CPU, velikost obsazené paměti, počty otvíraných souborů a další parametry. Testy realizujte alespoň na dvou odlišných distribucích OS Linux bez limitace otevřených souborů (ulimit). Výsledky testování zpracujte do přehledných tabulek a grafů. Přílohou práce budou grafické průběhy sledovaných parametrů během testů.

DOPORUČENÁ LITERATURA:

[1] Meggelen, J.V., Bryant, R., Madsen, L. Asterisk: The Definitive Guide: Open Source Telephony for the Enterprise - 5th Edition. O'Reilly Media, Inc., 2019. ISBN 978-1-492-03160-4.

[2] Goncalves, F.E., Iancu, B.A. Building Telephony Systems with OpenSIPS – 2nd Edition. Packet Publishing Ltd., 2016. ISBN 978-1-78528-061-0.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 24.5.2022

Vedoucí práce: doc. Ing. Pavel Šilhavý, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

Abstrakt

Diplomová práce se zabývá porovnáním a testováním pobočkových ústředen Asterisk a Kamailio. Dále se zabývá porovnáním stacku SIP a PJSIP. Na začátku práce je umístěna teoretická část, popisující pobočkové ústředny, protokol SIP a PJSIP. Dále jsou popsány testované pobočkové ústředny Asterisk a Kamailio a jsou porovnány jejich funkce. Jednotlivé ústředny jsou nainstalovány, nakonfigurovány a pomocí testeru Spirent TestCenter C1 jsou prováděny testy. Výsledky testu byly zpracovány do grafů a v závěru jsou diskutovány výstupy těchto testů.

Klíčová slova

VoIP, SIP, PJSIP, Asterisk, Kamailio

Abstract

The diploma thesis deals with the comparison and testing of private branch exchanges Asterisk and Kamailo. It also deals with the comparison of stack SIP and PJSIP. At the beginning of the thesis, there is a theoretical part describing private branch exchanges, SIP protocol and PJSIP. In the next section, the tested PBXs Asterisk and Kamailio are described and there is comparison of it´s functions. The private branch exchanges are installed, configured and tests are performed using the Spirent Testcenter C1. The results of the tests were processed into graphs and at the end there is a discussion about the results.

Keywords

PBX, SIP, PJSIP, Asterisk, Kamailio

Bibliografická citace:

RAJJ, Jakub. Porovnání výkonnosti SIP implementací v Open source PBX Asterisk a Kamailio. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2022. 139 s. Vedoucí diplomové práce byl doc. Ing. Pavel Šilhavý, Ph.D.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Porovnání výkonnosti SIP implementací v Open source PBX Asterisk a Kamailio jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: …… ……

podpis autora

Poděkování

Chtěl bych velmi poděkovat panu doc. Ing. Pavlovi Šilhavému, Ph.D., za odborné vedení, za poskytnuté pracoviště, veškerou pomoc, trpělivost a čas, který mi byl věnován při realizaci diplomové práce.

V Brně dne: …… ……

podpis autora

Obsah

Seznam symbolů a zkratek

Zkratky:

IP	Internet protocol
FDM	Frequency Division Multiplex
SS7	Signaling System Number 7
TDM	Time Division Multiplex
VoIP	Voice over IP
PBX	Private Branch Exchange
QoS	Quality of Services
RTP	Real Time Protocol
SRTP	Secure Real Time Protocol
SIP	Session Initiation Protocol
IAX2	Inter–Asterisk eXchange 2
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
SSRC	Synchronization Source Identifier
RTCP	Real Time Control Protocol
SR	Sender Report
RR	Receiver Report
SDES	Source description
CNAME	Canonical end–point Identifier
SRTCP	Secure Real Time Control Protocol
AES	Advanced Encryption Standard
ECB	Electronic codebook
CBC	Cipher block chaining
CFB	Cipher feedback
OFB	Output feedback
CTR	Counter
MGCP	Media Gateway Control Protocol
SCCP	Skinny Client Control Protocol
MCu	Multicontrol unit

UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
URI	Uniform Resource Identifier
HTTP	Hypertext Transfer Protocol
SDP	Session Description Protocol
IVR	Interactive Voice Response
B2BUA	Back–to–Back User Agent
AGI	Asterisk Gateway Interface
AMI	Asterisk Manager interface
CEL	Channel Event Logging
CDR	Call Detail Record
CTI	Computer telephony Integration
CEL	Call Event Log
MWI	Message warning indication
CentOS	Community ENTerprise Operating Systém
RHEL	Red Hat Enterprise Linux

Seznam obrázků

Seznam tabulek

ÚVOD

Telefonní ústředny si od svého vzniku až do současné doby prošly obrovským technickým vývojem. Od počátku tohoto vývoje, kdy jedny z prvních ústředen používaly manuální přepojování, přes několik generací ústředen s již automatickým přepojováním, uběhla dlouhá doba. Každá z těchto generací přišla s využitím nových elektronických součástek, ale v současné sobě se využívá především ta poslední, tedy pátá generace pobočkových ústředen. Ta je pojata úplně odlišným způsobem než předchozí generace a charakteristickým pojmem je pro ni pojem softswitch. S příchodem této generace se začaly využívat i nové přenosové protokoly a signalizační protokoly.

Diplomová práce se zabývá právě touto generací pobočkových ústředen, konkrétně dvěma zástupci: pobočkovou ústřednou Asterisk a proxy serverem Kamailio. Ty patří mezi velké množství open source pobočkových ústředen, které se v současné době používají. Práce se zaměřuje na popis jednotlivých ústředen, otestování a porovnání vlastností obou těchto open source pobočkových ústředen. Mezi jejich vlastnosti může patřit například zatížení procesoru a s tím související maximální množství navázaných spojení a transakcí.

První kapitola práce se zabývá úvodem do problematiky pobočkových ústředen. Je zde nastíněn jejich generační vývoj a krátce jsou popsány jednotlivé generace. Následuje popis páté generace, což je hlavní část této kapitoly. Pasáž o páté generaci obsahuje opět popis této generace, ale také popis nových pojmů, které se v souvislosti s ní objevily. Součástí kapitoly jsou dále informace o protokolech pro přenos dat a o signalizačních protokolech. Jsou zde popsány protokoly, které jsou nezbytné pro správné fungovaní IP telefonie.

Druhá a třetí kapitola se zabývá open source pobočkovými ústřednami Asterisk a Kamailio. Nejprve je uveden krátký teoretický úvod do problematiky a popis obou ústředen. Následně už se přechází k praktické implementaci, instalaci, konfiguraci a celkovému zprovoznění obou ústředen.

Ve čtvrté kapitole jsou porovnávány funkcionality projektů Asterisk i Kamailio. Funkcionality jsou zde v podobě seznamu a je k nim uvedeno, zda je funkcionalita zakomponována u některého z projektů.

Pátá kapitola se zabývá popisem pracoviště, na kterém je prováděno měření. Dále se zabývá testerem Spirent TestCenter C1, kde jsou popsána jednotlivá použitá nastavení. V závěru jsou popsány použité testovací scénáře, z nichž byly získány výsledky.

V poslední kapitole jsou výsledky všech měření. Sem byly zahrnuty výsledky měření provedených na různých konfiguracích virtuálního počítače, výsledky měření pro dva režimy komunikace u ústředny Asterisk, výsledky pro implementaci nativního stacku SIP i implementaci stacku PJSIP u ústředny Asterisku a výsledky měření u SIP proxy serveru Kamailio. V závěru kapitoly jsou výsledky porovnány a zhodnoceny.

1. ÚVOD DO PROBLEMATIKY

Telefonní ústředny jsou spojovacím systémem, který byl a i nadále zůstává důležitým prvkem komunikačních systémů. K těmto spojovacím systémům přichází požadavky k vytvoření spojení s jinými účastníky, v případě telefonních ústředen je to požadavek na vytvoření telefonního spojení s jiným účastníkem. Účelem telefonních ústředen je pak najít správného koncového účastníka, nebo jinou vhodnou ústřednu, pro kterou je požadovaný účastník dostupný a vytvořit mezi nimi komunikační kanál. Slouží tedy ke spojování telefonních hovorů, ale má i další funkce, jako například tarifikace hovorů a další. V historicky prvních ústřednách byly hovory přepojovány manuálně a to tak, že existovala přepojovací místnost a v ní jedna nebo více operátorek, které přijímaly požadavky a manuálně spojovaly okruhy pomocí konektorů, aby požadavek na spojení odeslaly dál směrem k cíli. Toto bylo historické pojetí a jednalo se zde o přepínání okruhů. Okruhy byly aktivní po celou dobu spojení a označovaly se také jako spojované systémy. To se vyskytovalo ještě u čtvrté generace telefonních ústředen.

Novější telefonní ústředny již upustily od tohoto manuálního principu a začalo se využívat automatického přepojování, protože se současnými požadavky by nebylo možné je všechny obsloužit. Od páté generace telefonních ústředen se přešlo k nespojovanému charakteru komunikace a začalo se využívat přepínání paketů. I nadále ovšem existují systémy, které mohou využívat oba způsoby přepojování. Jak spojení se spojovaným charakterem, tak i nespojovaným. Tyto systémy se nazývají hybridní systémy.

1.1 Generační vývoj ústředen

Telefonní ústředny prošly od svého vzniku značným technickým vývojem. Jednotlivé etapy vývoje se označují jako vývojové generace [1]. Těchto generací je doposud pět a ke každé následuje krátký popis klíčových prvků, které jsou pro danou generaci typické a případně i typická ústředna pro danou generaci [2]. První komerční telefonní ústředna vznikla v lednu roku 1987 v New Havenu, v Connecticutu. Tato ústředna spadala do nulté generace, i když tento pojem není tak často využívaný. Spadaly by sem všechny telefonní ústředny, využívající manuální přepínání okruhů formou

obsluhy spojovacího pole operátorkami. Patří sem jedny z prvních telefonních ústředen.

Do první generace se řadí ústředny, které k přepojování okruhů ve spojovacím poli využívají elektromagnetické krokové voliče. Pro přenos více hovorů v jednom přenosovém médiu se využívalo fantomového vedení. Řízení bylo decentralizované a využívala se k němu reléová logika. Typickým zástupcem ústředen z této generace je ústředna Tesla P51.

V druhé generaci využívaly ústředny spojová pole, která sestávala z křížových spínačů. Pro přenos více hovorů v jednom přenosovém médiu se začala využívat technika frekvenčního dělení (Frequency Division Multiplex – FDM). Zůstala zachována reléová logika řízení hovorů. Typickým modelem pro tuto generaci, který se dlouhou dobu využíval i u nás, je ústředna TESLA PK 202.

U třetí generace se využívají poloelektronické systémy s diskrétními elektrotechnickými prvky, jako jsou diody a tranzistory, a k tomu se využívá křížové spojovací pole. I zde se zachovalo využití FDM pro přenos více hovorů přenosovým médiem mezi ústřednami. Pro řízení se využívají procesorové obvody. Třetí generace je specifická pro přenášení analogového signálu přes spojové pole. Typickým modelem této generace je Tesla UE201.

Se čtvrtou generací došlo k digitalizaci přenášeného signálu. Spojování je ve spojovacím poli prováděno digitálně. Pro přenos dat se využívá časového multiplexu (Time Division Multiplex – TDM). Pro signalizaci se začal využívat signalizační systém číslo 7 (Signaling System Number 7 – SS7). SS7, který je uveden v dokumentu ITU–T Q.700 [3], je specifický pro čtvrtou generaci ústředen. Pomocí ústředen čtvrté generace došlo k vybudování velké části pevné i mobilní sítě na našem území a vznikla tak plně digitální síť. Typickým zástupcem čtvrté generace je Alcatel S12.

Pro pátou generaci je charakteristický pojem internetová telefonie a VoIP (Voice over IP) [4]. Mezi těmito pojmy je rozdíl, který se často opomíjí. Pojmem VoIP je myšlena technologie, umožňující přenos hlasu po datové síti, kdežto pojem internetová telefonie představuje službu. Jak bylo zmíněno výše, u páté generace se upustilo od spojování okruhů a přešlo se k přepojování paketů. To s sebou přineslo mnoho výhod, ale také určité nevýhody. Technologie VoIP již nevyžaduje mít

v ústřednách páté generace spojové pole. Díky absenci spojového pole začaly vznikat další alternativy k doposud existujícím proprietárním ústřednám. Vzniká tak nový software, nahrazující funkce doposud běžných telefonních ústředen – takzvané softwarové ústředny (anglicky softswitch). Využití softwarových ústředen vedlo ke značnému snížení ekonomických nákladů, protože komunikace probíhá formou paketů právě přes datovou síť. Novou možností, kterou přineslo využití IP paketů, je jednodušší práce s hlasovými soubory v reálném čase [4]. Hlas je zpracováván tak, že se s určitou frekvencí odebírají vzorky, kterým se přiřadí hodnota. Hodnoty jsou následně zakódovány a rozděleny na bloky, které se přenáší v IP paketech. Díky tomu už nemusí být rezervovány zdroje pro daný hovor, síť je méně zatěžována. Dalším důvodem, proč se začaly využívat open source pobočkové ústředny (private branch exchange – PBX), jsou také vyšší náklady pro jednotlivé licence proprietárních ústředen. Na druhou stranu dostal zákazník ucelený a fungující systém, do kterého nemá možnost moc zasahovat či si upravovat kód, ale v případě problémů má podporu ze strany výrobce. Nevýhodou je, stejně jako u všech real time služeb, že i pro internetovou telefonii je potřeba zajistit určitou kvalitu služeb QOS (Quality of Services). Služby pracující v reálném čase jsou velmi náchylné na zpoždění, které by mělo být maximálně 200 ms. Pokud je zpoždění konstantní, lze se mu přizpůsobit [4]. Obtížnější situace nastává, když se hodnota zpoždění mění. Toto zpoždění se nazývá kolísavé zpoždění – jitter. To bývá často způsobeno cestou paketů k cíli po různých cestách mezi směrovači, nebo různě dlouhou dobou zpracování paketů. Pro ochranu proti zpoždění a jitteru se využívají buffery a jitter buffery, které dokáží do určité míry redukovat vliv zpoždění. Příliš vysoká hodnota zpoždění a kolísavého zpoždění vede k růstu ztrátovosti a úplné nedostupnosti služby. Pro přenos dat se využívá Real Time Protocol (RTP), případně Secure Real Time Protocol (SRTP). Pro signalizace se využívají některé protokoly H.323, Session Initiation Protocol (SIP) nebo protocol Inter–Asterisk eXchange 2 (IAX2).

1.2 VoIP Protokoly

Protokoly využívané u VoIP lze rozdělit na protokoly pro přenos dat a pro řízení a signalizaci. Těchto protokolů je velké množství. Tato práce bude zaměřena

na převážně nativní implementaci SIP a novou implementaci PJSIP, a proto jim bude v teoretické části věnován větší prostor.

Pro přenos se na transportní vrstvě využívá protokol UDP, ale v některých případech se může využívat i TCP. UDP zaručuje rychlý přenos dat. Oproti TCP má menší hlavičku, a tedy i menší nároky na síť. Na rozdíl od TCP protokol nezaručuje, že budou všechny pakety doručeny ke svému cíli a nezaručuje ani jejich správné pořadí. Tohoto se využívá právě u služeb pracujících v reálném čase [4]. U těchto služeb by opakovaný přenos ztracených paketů vedl k nárůstu zpoždění. Po určité době by již opakovaně odeslaná data nemusela být pro službu relevantní. Pokud dojde ke ztrátě UDP datagramu, ztracený datagram se už nebude znovu přenášet. Funkci opakovaného přenosu nebo přenosu paketu ve správném pořadí nezajišťuje ani RTP.

Protokol RTP[5], definovaný v dokumentu RFC 3550 [6], se využívá pro přenos hlasu, respektive multimediálních dat. Jeho obvyklé využití je pro přenos audia a videa. RTP je nezávislý na typu sítě a přenosovém protokolu. Nejčastěji se využívá v kombinaci s protokolem UDP, protože stejně jako UDP, tak ani protokol RTP nezajišťuje doručení všech paketů. Dokáže pouze detekovat chybějící pakety na základě informací z hlaviček RTP paketů. Každý RTP paket se skládá z RTP záhlaví a dat. Vzhledem k velkému množství kódovacích algoritmů, kterými se zabezpečují přenášená data, musí být v hlavičce přenášena informace o použitém algoritmu [4]. Dalšími položkami v hlavičce RTP jsou:

- verze (Version – V) – značí současnou verzi RTP protokolu,
- výplň (Padding – P) – v případě, že velikost dat není dostačující k zaplnění minimální velikosti paketu, tak se tento paket opatří výplní; položka výplň značí, kolik oktetů je doplněno na konec paketu,
- rozšíření (Extension – E) – slouží k rozšíření záhlaví a vložení dodatečných informací,
- CSRC počet (CSRC Count – CC) – značí počet dalších zdrojů v multicastové relaci,
- značka (Marker – M) – význam položky je definován použitým médiem,
- typ média (Payload Type – P) – položka slouží k identifikaci kódovacího algoritmu,

- sekvenční číslo (Sequence Number) – jedná se o náhodně vygenerované číslo, které se s odeslanými pakety inkrementuje; tento mechanismus umožňuje detekovat ztracené pakety, nebo pakety přijaté ve špatném pořadí,
- časové razítko (Timestamp) – slouží k určení prvního oktetu v paketu,
- zdroj synchronizace (Synchronization Source – SSRC) – slouží k identifikaci zdroje synchronizace; ten nastavuje časové razítko a sekvenční číslo,
- přispívající zdroj (Contributing Source – CSRC) – idenfitikuje zdroje přispívající svým obsahem do relace.

Ke správné funkci přenosu dat pomocí RTP je potřeba zajistit řízení, které je realizované protokolem Real Time Control Protocol (RTCP), který je rovněž definován v dokumentu RFC 3550 [6]. Kromě řízení slouží tento protokol také k popisu zdroje a identifikaci případných problémů při odesílání RTP paketů [4]. K tomu se využívá pět typů RTCP zpráv.

- Zprávy nesoucí informace o zdroji dat (Sender report [SR]) – ve zprávách jsou přenášeny statistické informace o přenosu paketu směrem od zdroje k příjemci dat.
- Zprávy nesoucí informace o příjemci dat (Receiver Report [RR]) – ve zprávách jsou přenášeny statistické informace o přenosu paketu směrem od příjemce dat k zdroji.
- Popis zdroje (Source description [SDES]) – zprávy které obsahují identifikátor Canonical end–point Identifier (CNAME), což je kanonické jméno zdroje. Další položkou ze zprávy SDES jsou doplňující informace, jako například název aplikace a aktuální stav zdroje, dále například kontaktní údaje a jiné.
- Zpráva Bye – slouží k oznámení konce relace.
- Zpráva App – zprávy, jejichž funkci udává použitá aplikace.

RTCP sám nepřenáší žádná uživatelská data. Pomocí zpráv Sender report a Reception report je poskytována zpětná vazba ohledně probíhajícího přenosu. O výsledcích je zpravován zdroj a v případě potřeby bývá často možné přizpůsobit

vysílání například změnou kódování. Řídící data a data od uživatelů jsou oddělena a přenášena samostatně.

Alternativou RTP protokolu je jeho zabezpečená verze s názvem Secure RTP (SRTP), která je definována v dokumentu RFC 3711 [7], a k němu náležící řídící protokol Secure Real Time Control Protocol (SRTCP), jenž je rovněž nadefinován v dokumentu RFC 3711 [7]. Rozdíl mezi nimi je v tom, že při použití SRTCP se musí šifrovat i zprávy SIP, aby byla zajištěna bezpečnost spojení. Oba tyto protokoly využívají šifrování Advanced Encryption Standard (AES) jakožto výchozí variantu, ale je možné si vybrat z několika módů šifrování. Protokol AES je definován v dokumentu RFC 3826 [8]. Rozdíl mezi jednotlivými módy AES je v tom, jaké operace a jejich pořadí se využívá při šifrování. Patří sem módy ECB (Electronic codebook), CBC (Cipher block chaining), CFB (Cipher feedback), OFB (Output feedback) a CTR (Counter).

1.3 Signalizační protokoly

Signalizační protokoly slouží k vytváření spojení, jeho udržování a ukončení. Mezi často používané signalizační protokoly patří jednoznačně SIP, který je nadefinován v dokumentu RFC 3261 [9], a to hlavně díky své jednoduchosti. Také proto je využíván daleko častěji než protokol H.323 [10], který byl uveden dříve než SIP. Protokol je nadefinován v dokumentu ITU–T H.323 [11]. Rozdíl mezi nimi je také v tom, že zprávy odesílané protokolem H.323 mají binární obsah, zatímco SIP je textově orientovaný protokol, čímž jsou jeho zprávy pro uživatele jednodušší ke čtení. Existují ale i jiné signalizační protokoly. Patří sem již zmiňovaný H.323, dále IAX2, Media Gateway Control Protocol (MGCP), Skinny Client Control Protocol (SCCP) a další.

Protokol H.323 je založen na architektuře obsahující 4 prvky [4]. Těmi jsou terminály, gateway, gatekeeper a multicontrol unit (MCu) a každý má svou specifickou funkci v této architektuře. Terminály jsou jedinou povinnou komponentou ze všech čtyř zmiňovaných prvků. Slouží k navazování komunikace. Gateway je v překladu brána a plní funkci zprostředkování komunikace mimo síť H.323 a komunikaci s jinými typy sítí. Gatekeeper zastává funkci řízení zóny, kterou spravuje. Má na starost funkce jako je autorizace a autentizace jednotlivých prvků

v síti, překlad čísel na IP adresy koncových bodů nebo povolování a zakazování jednotlivých hovorů. Poslední ze zmiňovaných komponent je MCU. Ta má za úkol řídit komunikaci tří a více terminálů najednou a směšovat video a audio účastníků.

Protokol SIP slouží stejně jako ostatní signalizační protokoly k navázání, udržení modifikaci a ukončení spojení. Tím může být nejen přenos hlasu, ale i přenos videa a textových zpráv. SIP je textově založený protokol, který definuje vlastní strukturu zpráv. K přenosu zpráv může využívat UDP nebo TCP a porty 5060 a 5061. Podobně jako protokol H.323, tak i SIP zavádí nové prvky [10]. Mezi základní prvky patří uživatelský agent (User Agent – UA) a SIP proxy server. Ostatní prvky jsou rozšířením a patří sem: gateway, season border controller, redirect server a registrar server. Díky implementaci mapování adres a využití služeb redirect serveru podporuje SIP určitou mobilitu koncových stanic.

1.3.1 Uživatelský agent

Uživatelským agentem může být koncový prvek, který přijímá a generuje SIP zprávy. Nejčastěji se jedná o telefony, implementované hardwarově nebo softwarově, ale UA můžou být i jiná zařízení. UA pracuje jako klient–server. Jeden UA obsahuje klientskou část (User Agent Client – UAC), která slouží ke generování požadavků. Obsahuje také serverovou část (User Agent Server – UAS), k naslouchání těchto žádostí. K identifikaci jednotlivých koncových stanic se využívá speciální identifikátor uniform resource identifier (URI) [12]. Tvar adresy může vypadat například následovně sip:stanice@domena.cz. Adresa vždy začíná částí sip:, dále obsahuje část pro identifikaci stanice a část pro identifikaci sítě. Adresa může obsahovat i další části. Úplný tvar vypadá:
sip:[uživatel[:heslo]@]hostname[:port][;parametry][?hlavičky], kde hranaté závorky značí nepovinný údaj.

1.3.2 Gateway

Tento prvek má obdobnou funkci, jako má i stejnojmenný prvek architektury protokolu H.323. Slouží k propojení a umožnění komunikace mimo síť. Nejčastěji se jedná o bránu do sítě TDM, ale také může sloužit k propojení sítí s různými VoIP protokoly.

1.3.3 Season border controller

Také se označuje jako SIP firewall a slouží k oddělení privátní a veřejné sítě a k ochraně proti útokům (např. případy, kdy se útočník vydává za jiného uživatele). Také se využívá ke kontrole správnosti signalizace za účelem zvýšení její spolehlivosti.

1.3.4 Proxy server

Funkcí proxy serveru je přesměrování příchozích hovorů. Realizována je tak, že při příchodu žádosti o vytvoření hovoru se nejprve rozhodne, zda je povoleno tento hovor spojit a následně odešle žádost dál buď přímo k UA, nebo k dalšímu proxy serveru. Nevytváří nové relace, ale jednotlivé zprávy pouze přeposílá. Výhodou proxy serveru je i to, že se tímto způsobem dá vytvořit filtrační prvek a to díky tomu, že nově započaté hovory musí procházet právě tímto prvkem.

1.3.5 Back–to–Back User Agent – B2BUA

Tento prvek se skládá ze dvou částí – z uživatelské a klientské. Využitím obou částí je dosaženo end–to–end komunikace, a tím je možné zlepšit celkové zabezpečení. Oproti proxy serveru udržuje v paměti stav probíhajícího dialogu. Všechny zprávy musí procházet tímto prvkem.

1.3.6 Redirect server

Toto je server, který spolupracuje s proxy serverem. Redirect server generuje zprávy typu 3xx redirect. Tyto zprávy jsou využívány při požadavku o přesměrování, který přišel od klienta. Tomu předá požadovanou URI, naváže se spojení s požadovanou stanicí a ukončí se spojení.

1.3.7 Registrar server

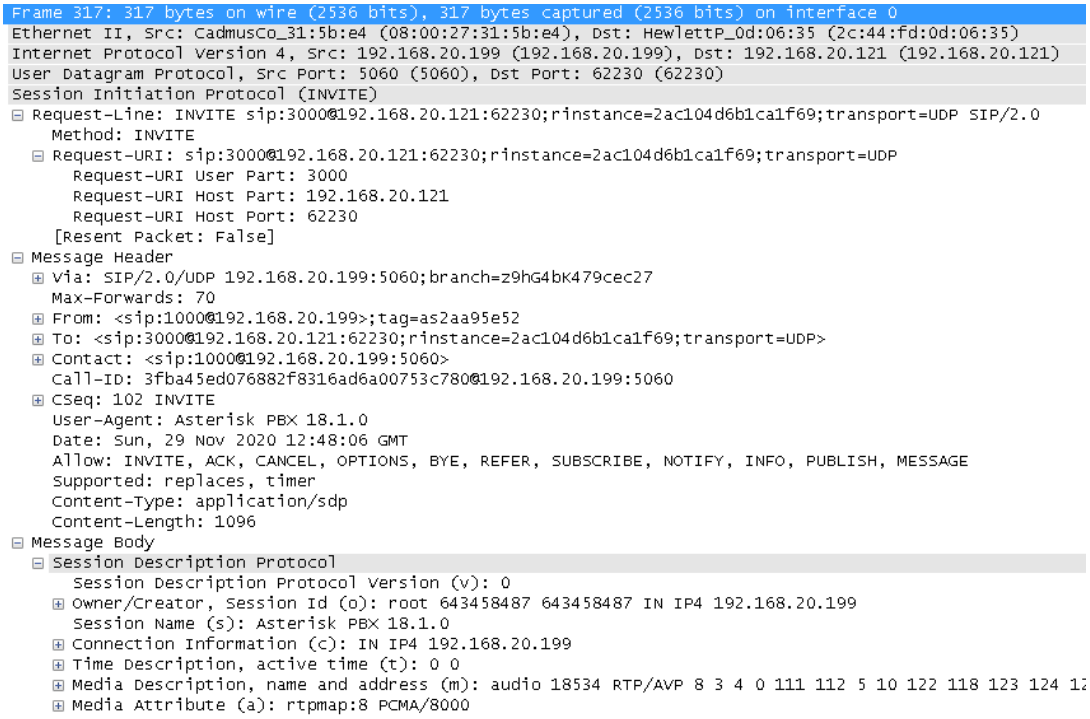
Jednou z vlastností SIP je mobilita účastníků. Ta je realizována právě pomocí registrar serveru. Všechny UA musí mít přidělenou URI předtím, než budou moci komunikovat. Ta jim je přidělena poté, co odešlou žádost REGISTER, která musí mít správné údaje. Pokud je žádosti vyhověno, server si zaznamená údaje z UA. Takto se

lze se stejným SIP účtem zaregistrovat na jiném UA a server si zaznamená novou IP adresu.

1.3.8 Typy zpráv

Kromě několika nových prvků architektury zavedl SIP i nové typy zpráv [12]. Dělí se na metody a odpovědi. Na následujícím obrázku je vidět zpráva zachycená pomocí programu Wireshark. SIP má podobnou strukturu zpráv jako například protokol HTTP. Obsahuje hlavičku a tělo zprávy. Hlavička zprávy obsahuje položky:

- Method – požadovaná metoda zprávy,
- request–URI – obsahuje URI dalšího prvku na cestě k cíli,
- Via – používá se k zaznamenání cesty mezi jednotlivými prvky na cestě ke koncovému bodu,
- From – udává, z jakého koncového bodu byla vytvořena zpráva,
- To – jakému koncovému bodu je zpráva určena,
- Contact – obsahuje jednu URI, na které lze kontaktovat odesílatele zprávy,
- call–ID – je globálně unikátní identifikátor hovoru,
- CSeq – zaznamenává počet odeslaných metod dle jejich typu,
- User–Agent – obsahuje informace UAC, ze kterého byla zpráva odeslána,
- Date – slouží k určení data a času,
- Allow – obsahuje seznam metod, které volající podporuje,
- Supported – zahrnuje seznam všech rozšíření, které volající podporuje,
- Content–Type – určuje typ přenášených médií,
- Content–length – určuje délku těla zprávy.



Obrázek 1.1 Zachycená zpráva SIP INVITE v programu Wireshark

Obrázek 1.1 zobrazuje zprávu SIP INVITE, zachycenou pomocí programu Wireshark. Jedná se o zprávu obsahující metodu INVITE, která přišla z adresy sip:3000@192.168.20.121. V těle zprávy se nachází položky protokolu Session Description Protocol (SDP) [4], který je nadefinován v dokumentu RFC 4566 [13]. SDP nepřenáší uživatelská data. Tento protokol je určený k popisu vlastností multimediální relace. K popisu relace je použito několik parametrů. Ty lze rozdělit na parametry pro popis relace, pro popis času a pro popis média. Mezi povinné položky popisu relace patří verze protokolu, identifikátor relace a volajícího a jméno relace. K nepovinným položkám popisu relace patří například informace o relaci, informace o šířce pásma, šifrovacím klíči, e–mail nebo telefonní číslo a další. Jako parametr k popisu času se používá povinná položka pro popis doby, po kterou je relace aktivní, a dále nepovinná položka udávající počet opakování relace. Popis médií obsahuje jednu povinnou položku pro název média a transportní adresy a dále obsahuje nepovinné položky pro označení média, informace o spojení a šířce pásma, šifrovací klíč a atributy relace.

Metody jsou odesílány z klientské části UA. V RFC 3261 je popsán protokol SIP a jsou zde definovány základní využívané typy metod. Sem patří REGISTER,

INVITE, ACK, BYE, CANCEL a OPTIONS. Ostatní typy metod jsou popsány až v přidaných RFC, a proto nemusí být podporovány některými UA.

- REGISTER – tuto metodu odesílají UA při žádosti o registraci u registrar serveru. Takto získá UA identifikátor URI.
- INVITE – touto metodou je zahajována komunikace mezi UA. Je s ní také možné měnit parametry probíhající komunikace.
- ACK – finální odpověď. Potvrzení na provedenou žádost.
- BYE – metoda odesílaná na žádost o ukončení relace.
- CANCEL – slouží k ukončení nepřijatých spojení, u kterých ještě neproběhla inicializace.
- UPDATE – využívá se k aktualizaci parametrů spojení.
- REFER – tento typ metody se využívá při potřebě přesměrování,
- SUBSCRIBE – metoda Subscribe se využívá, když je potřeba sledovat stav UA. Sledovaným stavem může být například, zda je UA aktivní.
- NOTIFY – metoda sloužící k informování účastníků. Používá se často s metodou SUBSCRIBE. Metodou SUBSCRIBE se sleduje událost a pomocí NOTIFY se odesílá výsledek.
- INFO – metoda sloužící k odeslání informací. Může být odeslána i v průběhu relace. Využívá se v implementaci s Interactive Voice Response (IVR).
- MESSAGE – textová zpráva.
- OPTIONS – slouží k zjištění dostupných prostředků UA. Používá se před navázáním spojení, aby se ověřila dostupnost UA.

Na tyto metody se odpovídá jednou nebo více odpověďmi [12]. Odpovědi jsou popsány třímístným číselným kódem, kdy první číslo udává typ odpovědi a následující číslice blíže specifikují tuto odpověď.

1xx: kladná odpověď. Žádost byla přijata a probíhá její zpracování.

2xx: kladná odpověď. Došlo k úspěšnému provedení žádosti.

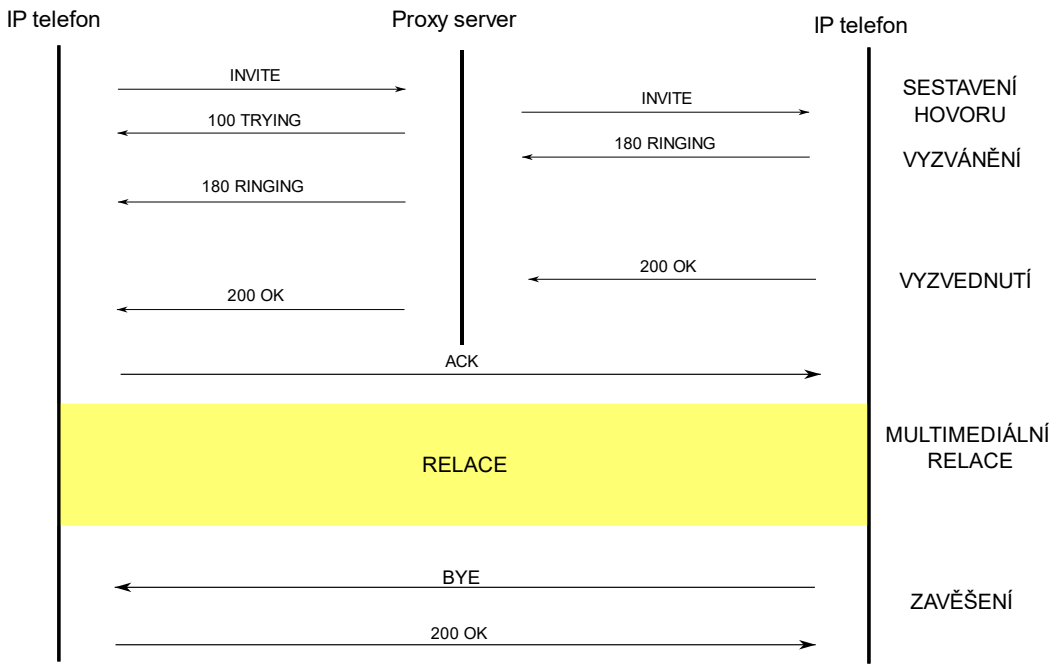
3xx: tato odpověď je žádostí o přesměrování, aby bylo možné dokončit požadavek.

4xx: záporná odpověď. Došlo k chybě způsobené klientem (chybná syntaxe metody).

5xx: záporná odpověď. Došlo k chybě způsobené serverem.

6xx: záporná odpověď. Obecná chyba, metoda nemohla být vyhodnocena.

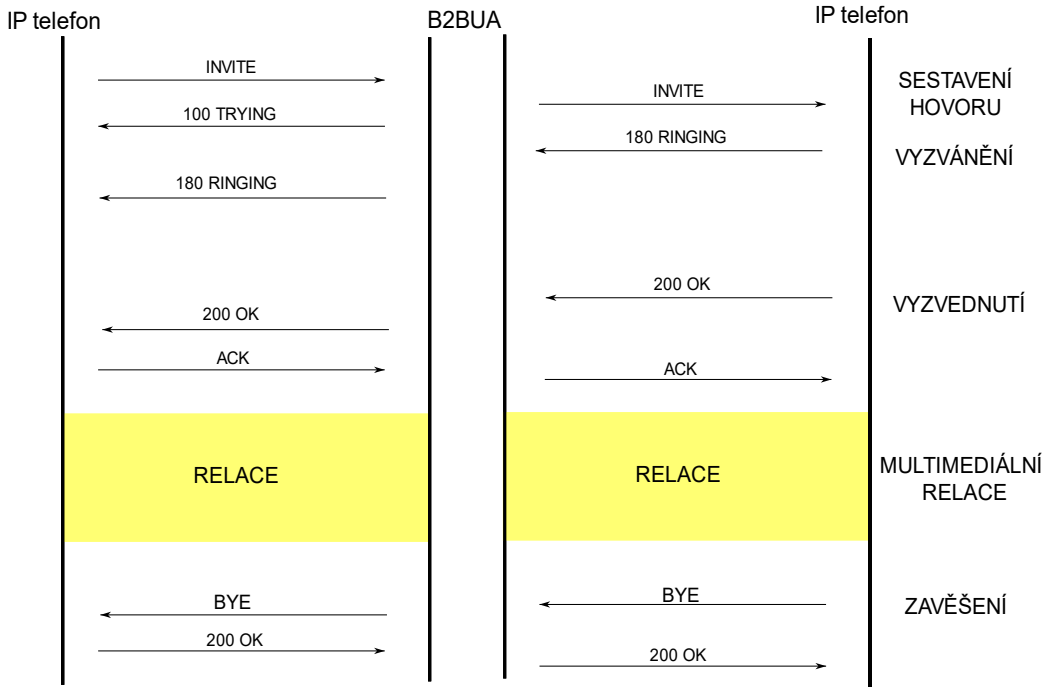
Mezi běžně se vyskytující zprávy patří 200 OK (odpověď o úspěšném provedení žádosti), 180 Ringing (koncový uživatel obdržel metodu INVITE, probíhá vyzvánění), 400 Bad Request (chybná syntaxe, metoda nemůže být vyhodnocena), 504 Server Time–out (došlo k vypršení času pro odpověď) nebo 600 Busy Everywhere (není možno přijmout hovor) všechny linky obsazené.



Obrázek 1.2 Komunikace s využitím proxy serveru

Obrázek 1.2 znázorňuje komunikaci mezi dvěma IP telefony s využitím proxy serveru. Nejprve je odeslána metoda INVITE jako žádost o navázání spojení. Je poslána na proxy server a ten ji následně odešle koncovému UA a zároveň pošle zpátky odpověď 100 TRYING. Mezitím ke druhému IP telefonu dorazila žádost INVITE. Odpovídá zprávou 180 RINGING, značící vizuální či zvukové upozornění. Jakmile uživatel přijme probíhající žádost (tedy zvedne telefon), jeho UA odešle odpověď 200 OK, na kterou je odpovídáno potvrzením ACK. Nyní může probíhat

multimediální relace (hlasová, s případným využitím zpráv nebo zakomponováním přenosu videa). Ta probíhá přímo mezi koncovými body. Na závěr se jeden z účastníků rozhodne ukončit hovor. Odešle metodu BYE, na kterou je odpovězeno 200 OK a hovor je tímto ukončen.



Obrázek 1.3 Komunikace s využitím B2BUA

Obrázek 1.3 zobrazuje komunikaci mezi dvěma UA. Nyní je jako prostředník využit B2BUA. Oproti proxy se liší v tom, že veškerá komunikace musí procházet přes tento element, kdežto proxy se využívá pouze k nalezení účastníků a navázání spojení. S využitím B2BUA v podstatě vznikají dvě relace pro jeden hovor. Jedna je mezi volajícím a B2BUA a druhá relace je mezi B2BUA a volaným účastníkem. Každá z nich má různý identifikátor Call–ID. Proxy pouze přeposílá/směruje jeden hovor.

1.4 PJSIP

Se vznikem a vývojem nových aplikací byly kladeny velké nároky na SIP stack, který byl vytvořen již v roce 2002 a poprvé popsán v RFC 2543, který byl následně rozšířen v RFC 3261. I přesto, že byl stack upravován a byly přidávány další funkce v podobě rozšiřujících RFC, jako jsou RFC 3262 (spolehlivost odpovědí), RFC 3263, RFC 3265, RFC 3311, RFC 6665 a dalších, tak stack nedokázal kompletně uspokojit

potřeby vývojářů. To mělo za následek vznik nového sip stacku PJSIP [14], který podporoval současné potřeby, ale byl navržen tak, aby mohl být využitelný i v budoucnu. V roce 2005 byla vydána první verze nového stacku. Do pobočkové ústředny Asterisk byl stack PJSIP plně implementován až ve verzi LTS 13 v roce 2014. PJSIP je založen na předešlých standardech a poskytuje zpětnou kompatibilitu se SIP stackem. Nativní stack Asterisku SIP byl monolitický jednokanálový ovladač, zatímco nová implementace přišla s modulární architekturou. To umožňuje jednodušeji přidávat nové funkce a stávající funkce upravovat.

PJSIP je open source SIP stack napsaný v jazyce C, zaměřující se převážně na vysoký výkon (tisíce hovorů v závislosti na použitém hardwaru), modularitu a podporu na velkém množství platforem. PJSIP stack byl navržen tak, aby se co nejvíce podobal fungování protokolu SIP. Realizuje funkce jako přenos multimediálního obsahu v reálném čase, signalizace a překlad adres. Díky modularitě, větším možnostem konfigurace a API je PJSIP lepší implementací SIP stacku. PJSIP zavedl možnost zaregistrovat větší množství uživatelů k jednomu účtu.

2. ASTERISK A JEHO TESTOVÁNÍ

Projekt Asterisk je jedna z nejznámějších open source realizací pobočkových ústředen na světě. K velké oblibě napomohla právě jeho dostupnost, ale i pestrost poskytovaných funkcí, jako je spojování hovorů, možnost vytváření konferenčních hovorů, odesílání textových zpráv, možnost využití Asterisku jako VoIP gateway anebo možnost vytvoření interaktivní hlasové odezvy (Interactive Voice Response System – IVR). Je zde také možnost upravovat si chování těchto ústředen a rozšiřovat je o další funkce pomocí modifikace původního kódu nebo doplněním o vlastní moduly a moduly třetích stran. Ty je potřeba nejprve přidat a načíst, aby mohly zpřístupnit další funkce.

2.1 Projekt Asterisk

Projekt vznikl jako řešení komunikace v malé firmě. Původně menší projekt se rychle uchytil, rozrostl a díky rozrůstající se komunitě uživatelů se z původně experimentálního kódu stal projekt, který nalezl uplatnění i v komerčním použití a je nasazován do reálných řešení PBX. Asterisk poskytuje základ (engine) [15], díky kterému je možné realizovat funkce PBX. O ty se pak starají signalizační a přenosové protokoly, které Asterisk podporuje. Mezi podporované signalizační protokoly patří protokoly SIP, H.323, IAX2, Jingle a další.

První verzi Asterisku představil Mark Spencer v roce 1999. Naprogramoval ji v jazyce C a lze ji spustit na velkém množství platforem. Mezi nejvíce využívané patří různé distribuce Linux, ale také FreeBSD, NetBSD, OpenBSD, MacOSX a v první verzi Asterisku i Windows. V roce 2001 byla založena firma Digium, která se ve spolupráci s komunitou věnuje vývoji dalších verzí Asterisku. V roce 2018 byla firma Digium koupena společností Sangoma, která od svého počátku financuje různé open source projekty. To umožňuje další růst projektu Asterisk, přidávání nových funkcí a zvyšování bezpečnosti aplikací [16].

Od počátku projektu vznikaly novější verze, které vylepšovaly původní projekt Asterisk, přidávaly se nové možnosti a funkcionality a opravovaly vzniklé bugy. Moduly, kterými jsou nové funkce realizovány, lze programovat opět v jazyce C v PHP, nebo s využitím skriptovacích jazyků, jako je Asterisk Gateway Interface

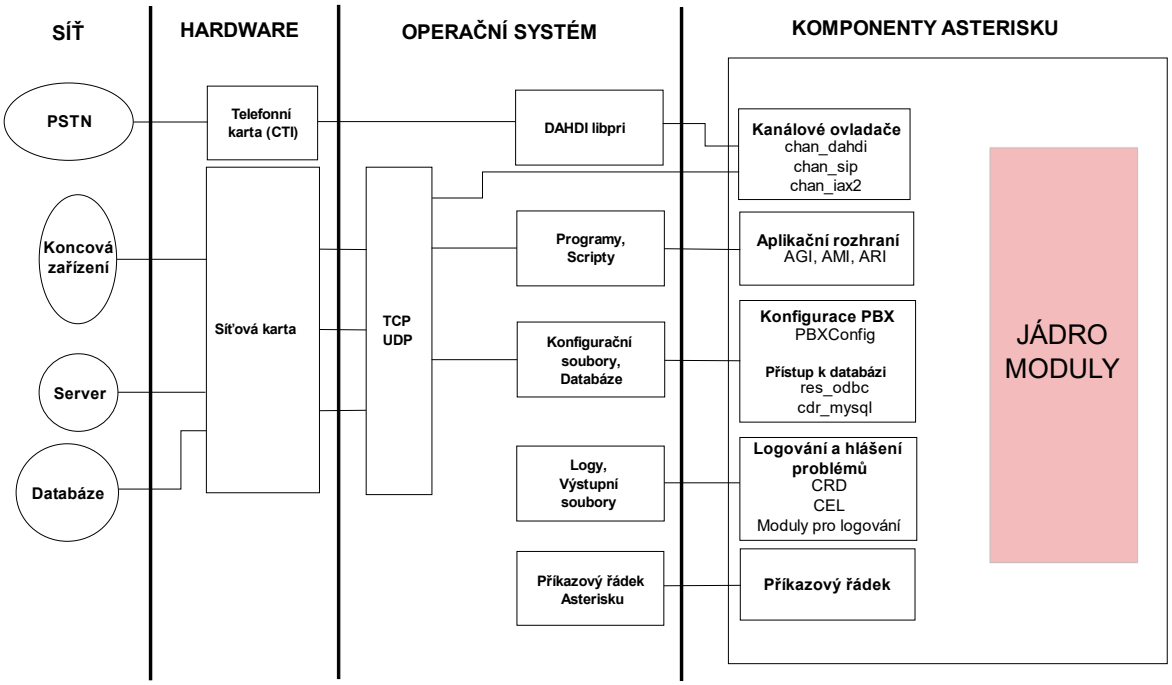
(AGI). Tabulka 2.1 obsahuje přehled jednotlivých verzí. Tyto verze jsou dvojího typu: standardní a verze s dlouhodobou podporou (Long Term Support – LTS). Verze se liší v délce podpory. Dále se liší v architektuře a struktuře souborů, ale to jen v případě LTS verzí. Verze standard vycházejí častěji, mají podporu alespoň jeden rok, mezi sebou zachovávají stejnou architekturu a strukturu souborů a přicházejí častěji, ale s menšími úpravami. Proto lze jednoduše upgradovat mezi jednotlivými verzemi standard. Zato verze LTS vycházejí s většími rozestupy mezi sebou, nabízí až čtyřletou podporu a ve struktuře souborů jsou větší či menší odlišnosti oproti přechozí LTS verzi. V průběhu vydávání nových verzí Asterisku se jednotlivé verze začaly dělit na tři hlavní větve. Těmi jsou: Asterisk, Certificated Asterisk a AsteriskNow (v současné době FreePBX).

Tabulka 2.1 Verze Asterisku

Číslo verze	Typ	Datum	Bezpečnostní opravy	Ukončení podpory
1.2.x	–	21.11.2005	07.08.2007	21.11.2010
1.4.x	LTS	23.12.2006	21.04.2011	21.04.2012
1.6.0.x	Standard	01.10.2008	01.05.2010	01.10.2010
1.6.1.x	Standard	27.04.2009	01.05.2010	27.04.2011
1.6.2.x	Standard	18.12.2009	21.04.2011	21.04.2012
1.8.x	LTS	21.10.2010	21.10.2014	21.10.2015
10.x	Standard	15.12.2011	15.12.2012	15.12.2013
11.x	LTS	25.10.2012	25.10.2016	25.10.2017
12.x	Standard	20.12.2013	20.12.2014	20.12.2015
13.x	LTS	24.10.2014	24.10.2020	24.10.2021
14.x	Standard	26.09.2016	26.09.2017	26.09.2018
15.x	Standard	03.10.2017	03.10.2018	03.10.2019
16.x	LTS	09.10.2018	09.10.2022	09.10.2023
17.x	Standard	28.10.2019	28.10.2020	28.10.2021
18.x	LTS	20.10.2020	20.10.2024	20.10.2025
19.x	Standard	02.11.2021	02.11.2022	02.11.2023

Tabulka 2.1 obsahuje jednotlivé vydané verze [17], datum jejich vydání, datum získávání oprav a datum ukončení podpory. Jelikož je verze 13.x verzí LTS, tak je datum ukončení její podpory pozdější než u dvou standardních verzí, vydaných po verzi LTS.

Kromě výše popsaných verzí vydává Asterisk také verze, které jsou vhodné pro komerční použití. Tyto verze se nazývají Asterisk Certificated. Jsou opět založeny na licenci GPLv2, ale oproti předešlým verzím dostávají méně často aktualizace a uživatel nemusí provádět tak časté změny.



Obrázek 2.1 Architektura PBX Asterisk

Obrázek 2.1 představuje zjednodušený pohled na architekturu Asterisku [18]. Jsou zde vidět komponenty Asterisku. U aplikačního rozhraní jsou uvedena jednotlivá rozhraní: Asterisk Manager Interface (AMI), Asterisk Gateway Interface (AGI) a Asterisk REST Interface. Všechny tři části jsou využívány ke kontrole Asterisku. AMI se využívá k vytváření kanálu nebo jeho modifikování během hovoru, AGI slouží k vytváření hovorů a práci s dialplanem a ARI umožňuje tvorbu externích aplikací. K logování se využívají mechanismy Channel Event Logging (CEL), pro zaznamenávání událostí v souvislosti s komunikačním kanálem. Dále Call Detail Record (CDR), určený k logování detailů o probíhajícím hovoru a poté další moduly určené k logování.

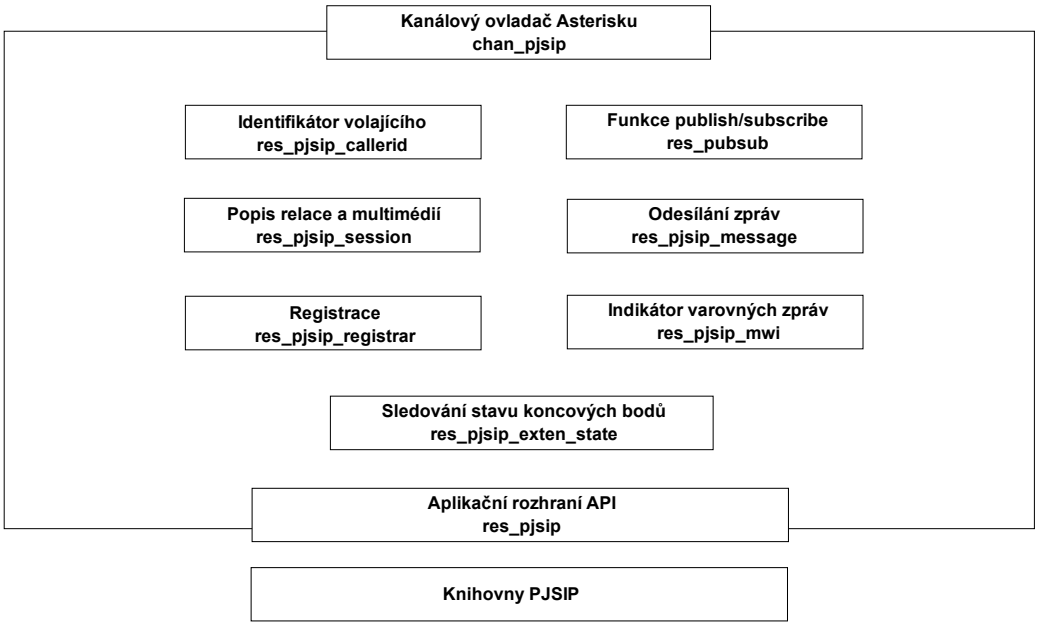
Jednotlivé komponenty Asterisku se vážou na další komponenty operačního systému a dále na hardware. Využívaným hardwarem jsou síťové karty pro připojení k jednotlivým koncovým zařízením, serverům nebo databázím a dále karty pro připojení analogových zařízení (Computer telephony Integration – CTI). Z výše popsaného obrázku je vidět, že lze komponenty Asterisku rozdělit na jádro a moduly,

s kterými jádro komunikuje. Jádro je základní komponentou každého Asterisku. Mezi jeho funkce patří mimo jiné načítání modulů a konfiguračních souborů. Ostatní funkce jsou realizovány pomocí modulů [19], kterých může být velké množství, a to také díky komunitě uživatelů. Základní moduly jsou distribuovány už se zvolenou verzí Asterisku a ostatní lze dodatečně stáhnout. Díky obrovské základně uživatelů, využívajících Asterisk, a také díky tomu, že projekt je již od začátku open source, tak mohou sami uživatelé vytvářet další moduly, dále tyto moduly modifikovat, šířit a rozšiřovat tak stávající systém o potřebné funkce. K základním modulům patří například Channel Driver (kanálový ovladač nebo také stack). Kanálové ovladače se starají o fungování signalizačních protokolů. Příkladem je nativní implementace chan_sip, která se stará o funkci SIP. Dalším příkladem kanálového ovladače může být chan_IAX2, využívaný pro funkci protokolu IAX2. Kanálové ovladače jsou tedy prostředky k propojení Asterisku s dalšími zařízeními díky využití k tomu určených protokolů. Dalším velmi důležitým modulem je Dialplan. Ten má na starost nejdůležitější funkci při inicializaci hovoru. Dialplan specifikuje, jak se má v ústředně směrovat hovor.

Toto bylo pouze pár základních modulů, celý výčet těchto modulů je velmi rozsáhlý. Spadají sem všechny moduly pro práci s kanálovými ovladači, moduly pro práci s dialplanem, dále moduly správy a využívání kodeků a zdrojů, moduly pro práci se soubory (převod streamu na soubory, ukládání v patřičném formátu) a moduly pro přepojování účastníků (bridging modules). Do výčtu ještě patří moduly pro záznam realizovaných hovorů a vzniklých událostí a jejich zápis na disk (Call Detail Record – CDR, Call Event Log – CEL Drivers).

2.2 Chan_PJSIP

Jak již bylo zmíněno v kapitole 1.4, tak kromě nativního stacku SIP (chan_sip) existuje ještě další implementace – PJSIP. Jelikož se jedná o jinou implementaci SIP stacku, může dojít ke konfliktu při současném používaní chan_sip a chan_pjsip. Proto je vhodné používat pouze jeden stack současně, případně nastavit, aby každý naslouchal na jiném portu. PJSIP má architekturu založenou na vysoké modularitě.



Obrázek 2.2 Stack PJSIP

Obrázek 2.2 znázorňuje pomocí bloků architekturu PJSIP stacku [14]. Zde jednotlivé bloky znázorňují funkční celky a obsahují popis funkce a název modulu. Jednotlivé funkce jsou realizované izolovanými moduly, které je možné jednoduše upravovat, přidávat nebo odebírat.

- chan_pjsip je kanálový ovladač, který zprostředkovaná funkci stacku PJSIP Asterisku.
- res_pjsip_callerid poskytuje funkce k identifikaci koncové stanice. Informacemi jsou identifikátory koncových účastníků a informace o přesměrování hovoru.
- res_pjsip_session slouží k poskytnutí informací protokolu SDP a zajišťuje tak jeho funkci. Vytváří a uzavírá relaci.
- res_pjsip_registrar se využívá k registraci koncových bodů k uživatelským účtům.
- res_pubsub realizuje funkce Publish a Subscription a slouží k odebírání informací o zařízení.
- res_pjsip_message se využívá pro práci s textovými zprávami.
- res_pjsip_mwi je modul, který sleduje stav kanálu a generuje zprávy MWI (Message warning indication).

- res_pjsip_exten_state slouží k poskytování informací o koncovém zařízení. Souvisí s modulem res_pubsub.

V implementaci s Asteriskem má stack PJSIP rozdílnou konfiguraci oproti nativnímu SIP stacku. Zatímco u původní implementace se nastavovalo obecné chování a následně se generovali klienti, u PJSIP lze detailněji nastavit chování jednotlivých klientů.

Ukázka konfigurace účtu u nativního SIP stacku:

```
[1000]
type=friend
username=1000
callerid=1000
secret=1234
context=TEST
host=dynamic
allow=all
```

V následujícím odstavci je vidět konfigurace uživatele s použitím PJSIP stacku. Konfigurace je mnohem rozsáhlejší. Jsou využívány sekce pro konfiguraci jednotlivých částí. Jednotlivé sekce jsou:

- ENDPOINT – sekce pro nastavení koncových bodů, obsahuje klíčová nastavení a propojuje ostatní sekce,
- TRANSPORT – sekce pro nastavení chování na transportní vrstvě,
- AUTH – sekce pro nastavení autentizace,
- AOR – sekce sdělující Asterisku, kde může být koncový bod kontaktován; bez této sekce ho nelze kontaktovat,
- REGISTRATION – sekce pro registraci koncových bodů mimo doménu Asterisku,
- DOMAIN ALIAS – sekce umožňuje zadat alias domény,
- ACL – sekce umožňující nastavit access control list pro koncové body uvnitř domény Asterisku,
- IDENTIFY – sekce pro kontrolu koncového bodu příchozího paketu,
- CONTACT – sekce pro udržování informací o registraci koncových bodů uvnitř domény Asterisku.

V příkladu konfigurace je použita sekce pro nastavení autentizace, pro nastavení přístupu více uživatelů k jednomu účtu a pro nastavení samotného účtu a propojení jednotlivých sekcí.

```
[auth_test]
type = auth
auth_type = userpass
username = 1000
password = 1234
[aors_test]
type = aor
max_contacts = 1
[1000]
type = endpoint
auth = auth_test
aors = aors_test
disallow = all
allow = ulaw
context = TEST
dtmfmode = rfc4733
```

2.3 Příprava pracoviště a instalace PBX

Před samotnou instalací některé z pobočkových ústředen bylo potřeba nejprve připravit pracoviště. Všechny testy se provádí na počítači, kde je spuštěn virtuální operační systém. Technické parametry počítače jsou:

Operační systém – Windows 7 Enterprice

Instrukční sada – 64bitová verze

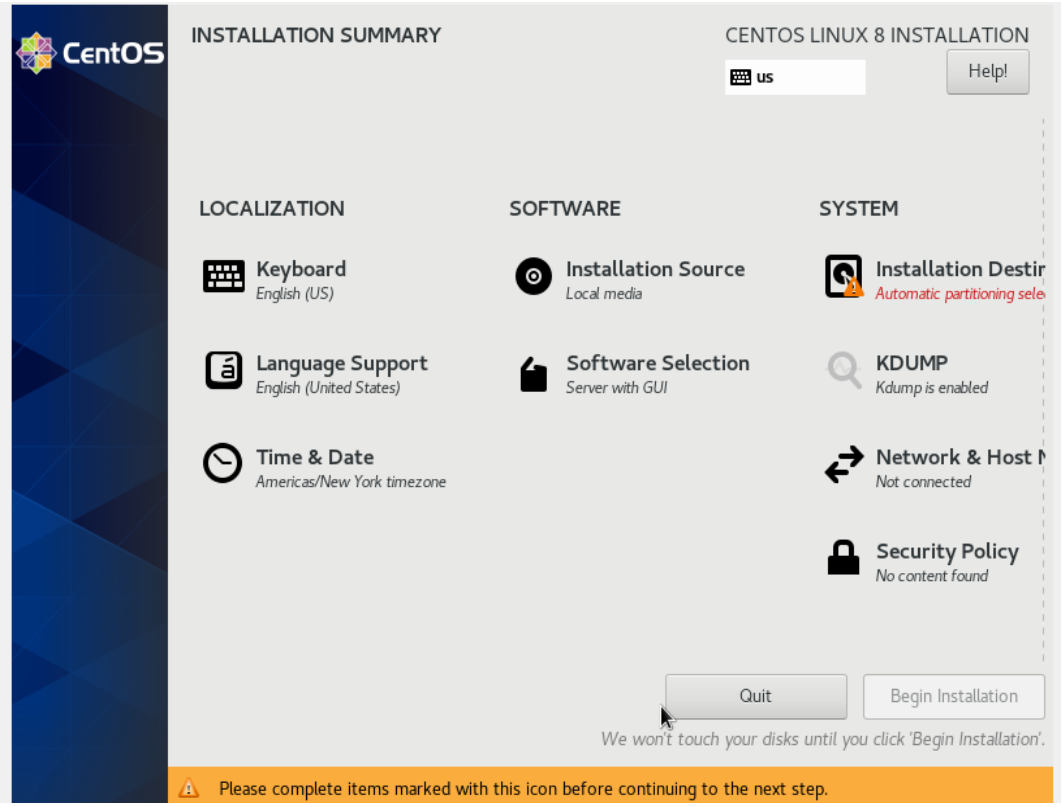
Procesor – Intel® Core™ i5 3570 s frekvencí 3,4 GHz

Operační paměť – 4,00 GB

Výhodami virtualizace jsou jednodušší instalace, možnost spuštění více virtuálních počítačů, operačních systémů a aplikací najednou na jednom počítači nebo serveru, což nabízí velkou flexibilitu, snížení provozních nákladů a nákladů pro implementaci. Dále možnost exportovat kopie virtuálních strojů a následně je zpětně importovat a pracovat s nimi samostatně. Díky této výhodě, tedy možnosti duplikování virtuálních operačních systémů, lze jednoduše dosáhnout vytvoření velkého množství kopií virtuálních systémů. Před samotným duplikováním se na jeden virtuální počítač nainstalují základní potřebné soubory a aktualizace

a následně se naklonuje. Toto poslouží jako základ pro všechny testované verze PBX Asterisk. Tím, že má každá z kopií všechny potřebné aktualizace a soubory už nainstalované, vznikl základ pro další práci. Na jednotlivé připravené kopie virtuálního počítače jsou následně nainstalovány různé verze Asterisku. Tak je zabráněno tomu, aby se pro každou testovanou verzi musely měnit všechny parametry a umístění všech souborů od jednotlivých verzí.

Pro první testování byl zvolen operační systém CentOS (Community ENTerprise Operating System), verze 8 s předinstalovaným grafickým rozhraním (v tomto případě se jedná o grafické rozhraní GNOME). CentOS je volně dostupná distribuce Linuxu, založená na Red Hat Enterprise Linux (RHEL). Z RHEL využívá pouze testované funkční aplikace. Díky tomu je tento operační systém velmi stabilní a bezpečný. Je možné ho implementovat pro desktopové využití, ale vhodnější uplatnění je jeho implementace jako serveru hlavně díky stabilitě a spolehlivosti. Instalace CentOS probíhá pomocí průvodce, který umožňuje uživateli přizpůsobení parametrů instalovaného OS ještě před nainstalováním celého systému.



Obrázek 2.3 Instalace CentOS

Obrázek 2.3 zobrazuje okno s průvodcem instalací operačního systému CentOS. Zde je možné upravit parametry systému, jako jsou čas a jazyk, síťový adaptér a rozdělení diskových oddílů. Dále je možné nastavit základní prostředí operačního systému. Z poskytovaných možností je na výběr server, server s grafickým prostředím, možnost minimální instalace, pracovní stanice anebo instalace s možností úpravy parametrů instalace. Nainstalovaný virtuální stroj má následující konfiguraci:

Operační systém: CentOS 8.2.2004

Verze jádra: 4.18.0–240

Instrukční sada: 64bit

Počet přidělených jader: 4

Operační paměť: 2 GB

Video paměť: 24 MB

Později po provedení první série testů bylo nutné navýšit operační paměť RAM. Ta byla zvýšena na hodnotu 4 GB.

Obdobně byl na virtuální počítač nainstalován i druhý operační systém zvolený k testování. Jedná se o Ubuntu pro servery bez grafického rozhraní. I tato distribuce je opensource, podobně jako CentOS. Instalace Ubuntu probíhala podobným způsobem jako instalace CentOS. Pomocí grafického rozhraní bylo možné upravovat parametry instalace. Virtualizovanému operačnímu systému byly nastaveny parametry:

Operační systém: Ubuntu 20.07.1

Verze jádra: 5.4.0–97

Instrukční sada: 64bit

Počet přidělených jader: 4

Operační paměť: 2 GB

Video paměť: 24 MB

Obdobně jako u CentOS, tak i zde došlo po první sérii testování k navýšení operační paměti na hodnotu 4 GB. K ukázce instalace byla zvolena aktuální LTS verze Asterisku. V následujících testech jsou využívány i další verze Asterisku a proxy server Kamailio. Před samotnou instalací ústředny byla ještě potřeba aktualizovat systém a doinstalovat všechny potřebné balíčky [20].

Příkaz pro aktualizaci u OS CentOS:

```
# yum -y update
```

Instalace balíčků u OS CentOS:

```
# yum install -y epel-release dmidecode gcc-c++ ncurses-devel libxml2-devel make wget openssl-devel newt-devel kernel-devel sqlite-devel libuuid-devel gtk2-devel jansson-devel binutils-devel libedit libedit-devel
#  yum -y groupinstall "Development Tools"
```

K doinstalování všech potřebných balíčků je možné použít také script, který se nachází v adresáři contrib pod názvem install_prereq. Jako další se instalovala knihovna Jansson, která se využívá pro práci s JSON daty.

```
# cd /usr/src/
```

```
# git clone https://github.com/akheron/jansson.git
# cd jansson
# autoreconf  -i
# ./configure --prefix=/usr/
# make && make install
```

Poté bylo ještě potřeba nainstalovat PJSIP stack. Ten je možný nainstalovat dvěma způsoby. První způsob je vidět na následujících řádcích, kdy se PJSIP stáhne jako balíček a nainstaluje. Druhou možností je přidání parametru *--with-pjproject-bundled* u */configure* při instalaci Asterisku.

```
# cd /usr/src/
# wget https://github.com/pjsip/pjproject/archive/2.10.tar.gz
# tar -xvf ${VER}.tar.gz
# cd pjproject-${VER}
#./configure CFLAGS="-DNDEBUG -DPJ_HAS_IPV6=1" --prefix=/usr --libdir=/usr/lib64 --enable-shared --disable-video --disable-sound --disable-opencore-amr
```

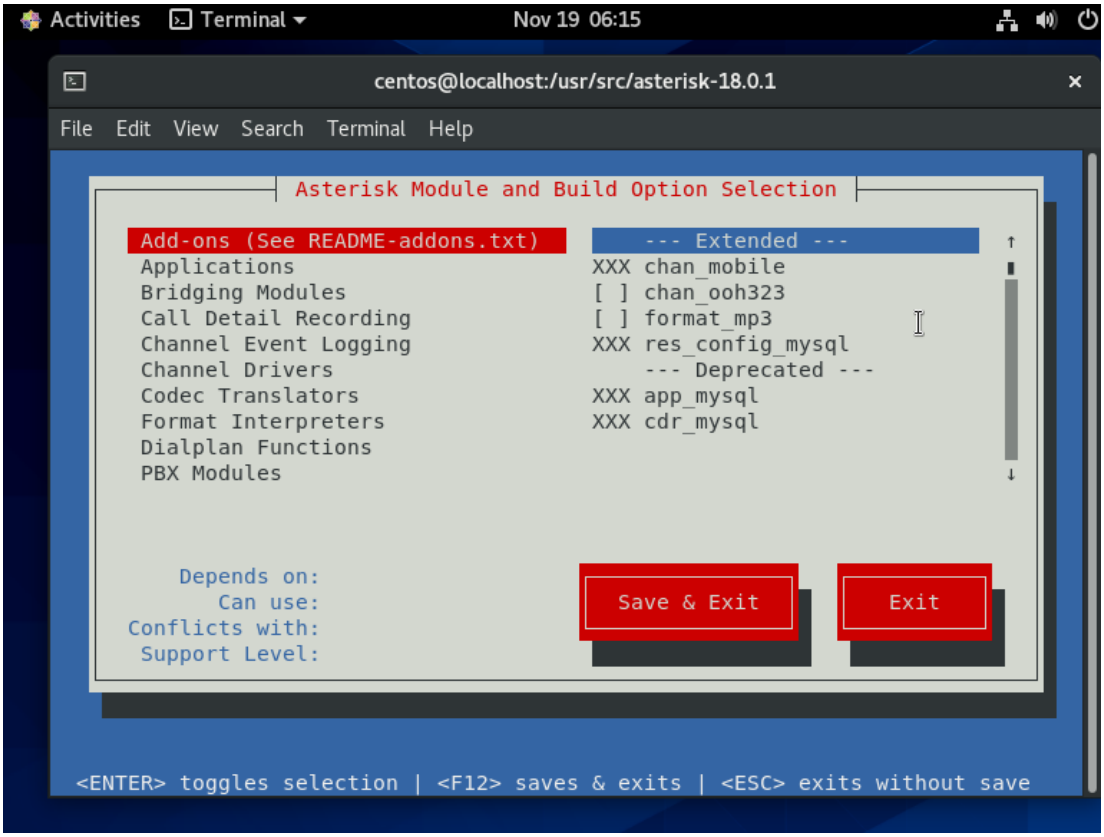
```
# make dep
# make
# make install
# ldconfig
```

V tento moment je virtuální stroj připravený k instalaci některé verze Asterisku, ale ještě před tím je provedeno již zmiňované vyexportování stávajícího stavu virtuálního stroje. Tím vznikne jeho kopie, kterou je možné naimportovat a dále s ní pracovat. Nyní se na připravený virtuální stroj nainstaluje Asterisk.

```
# wget http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-18-current.tar.gz
# tar xvfz asterisk-18-current.tar.gz
```

```
# cd asterisk-*
# ./configure --libdir=/usr/lib64
```

Příkaz *wget* slouží k získání balíčku dle zadané cesty k tomuto balíčku. Příkazem *tar xvfz* je balíček extrahován a příkazem ./configure je spuštěna konfigurace. Po konfiguraci se pomocí příkazu *#make* menuselect otevře grafické rozhraní, kde je možné přidávat nebo odebírat komponenty k instalaci.



Obrázek 2.4 Výběr instalovaných komponent

Obrázek 2.4 zachycuje okno průvodce instalace pobočkové ústředny Asterisk a možnosti výběru jednotlivých instalovaných komponent ústředny. Na výběr k přidání jsou různé aplikace, moduly, kodeky, funkce atd. Poté byly spuštěny příkazy:

```
# make
# make install
# make samples
# make config
# ldconfig
```

Po instalaci byl vytvořen nový uživatel s přístupovými právy. To muselo být upraveno i v konfiguračních souborech Asterisku. Na závěr byla služba PBX Asterisk nastavena na spouštění při startu virtuálního stroje příkazem:

```
# systemctl enable asterisk
```

V tomto stavu je již ústředna nainstalována a připravena k používání. Rozbor konfiguračních souborů je uveden v následující kapitole.

2.3.1 Konfigurace uživatelských účtů u stacku SIP a PJSIP

K předcházení konfliktů mezi stacky SIP a PJSIP bylo nutné nejprve definovat, na jakých portech bude každý stack naslouchat. Z toho důvodu byla provedena změna v souborech sip.chan a pjsip.chan, kde pro sip bylo nastaveno naslouchání na portu 5060 a pro pjsip naslouchání na portu 5066. To lze realizovat příkazem *bindport=číslo_portu* u stacku SIP a příkazem *bind=číslo_portu* u stacku PJSIP. Pro vytvoření velkého množství registrovaných účtů byl napsán bash script, který využívá šablony a vypadá následovně:

```
cat >> sip.conf << EOF1
[general]
bindport=5060
allow=all

[ucet](!)
type=friend
context=TEST
host=dynamic
allow=all
secret=1234
insecure=invite
EOF1

for i in {1000..3000}
do
cat >> sip.conf << EOF2
[$i](ucet)
username=$i
EOF2
done
```

Script vytvoří účty v rozsahu od 1000 do 3000 a každý bude mít definované své jméno, číslo, heslo, kodeky a kontext, do kterého spadá. Před spuštěním scriptu byl obsah souboru sip.conf vymazán. Důvodem je, že v defaultní podobě obsahoval příklady příkazů a komentáře. Po smazání obsahu byl spuštěn script. Výstup tohoto scriptu se zapsal do souboru sip.conf. Obsah souboru je:

```
[general]
bindport=5060
allow=all

[ucet](!)
type=friend
context=TEST
host=dynamic
allow=all
secret=1234
insecure=invite

[1000](ucet)
username = 1000

[1001](ucet)
username = 1001
.
.
.
[3000](ucet)
username = 3000
```

Následně byl v konfiguračním souboru extensions.conf přidán výše zmiňovaný kontext pro směrování hovorů do testeru na účet 3000. Server následně vybere ze seznamu volaných a realizuje hovor.

```
[TEST]
exten => _[1-9]XXX,1,DIAL(SIP/3000)
exten => _[1-9]XXX,2,Hangup
```

Obdobným způsobem, tedy s využitím šablon, byly vytvořeny účty i u stacku PJSIP. Kvůli odlišnosti konfigurace stacku SIP a PJSIP bylo potřeba upravit původní script do podoby, která by vytvářela sekce. Skript byl z toho důvodu rozdělen na několik šablon, kde každá z šablon byla využita pro jinou sekci. Do souboru *pjsip.conf,* jehož obsah byl stejně jako v předchozím případě nejprve vymazán, se nejprve zapíšou šablony jednotlivých sekcí a následně budou vytvořeny účty, které z těchto šablon budou čerpat. Nejprve je nastaveno, na jakém portu bude stack naslouchat a jaký transportní protokol použije. V sekci *endpoint* jsou nastaveny parametry o použitých kodecích a kontextech. Sekce *auth* nastavuje použitou autorizační metodu a sekce *aor* upravuje množství koncových bodů, které se můžou k účtu připojit.

```
cat >> pjsip.conf << EOF1
[transport]
type=transport
protocol=udp
bind=0.0.0.0:5066

[endpoint](!)
type=endpoint
context=pjsipTEST
disallow=all
allow=alaw

[authTest](!)
type=auth
auth_type=userpass

[aorTest](!)
type=aor
max_contacts=9999

EOF1

for i in {1000..3000}
do
cat >> pjsip.conf << EOF2
[$i](endpoint)
outbound_auth=$i
aors=$i
[$i](authTest)
password=1234
username=$i
[$i](aorTest)

EOF2
done
```

Po spuštění skriptu se na začátek souboru zapíše první část skriptu. Zápis se provede jen na začátku souboru pjsip.conf a zapíší se všechny šablony. Poté jsou vytvářeny a do souboru zapisovány jednotlivé účty. Ty vypadají následovně:

```
[1000](endpoint)
outbound_auth=1000
aors=1000
[1000](authTest)
password=1234
username=1000
[1000](aorTest)
```

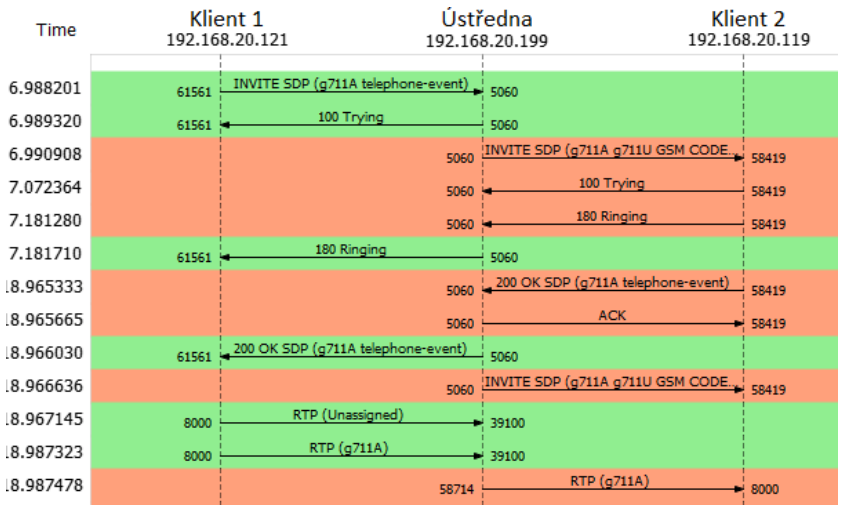
[1001](endpoint)
outbound_auth=1001
aors=1001
[1001](authTest)
password=1234
username=1001
[1001](aorTest)

.
.
.

[3000](endpoint)
outbound_auth=3000
aors=3000
[3000](authTest)
password=1234
username=3000
[3000](aorTest)

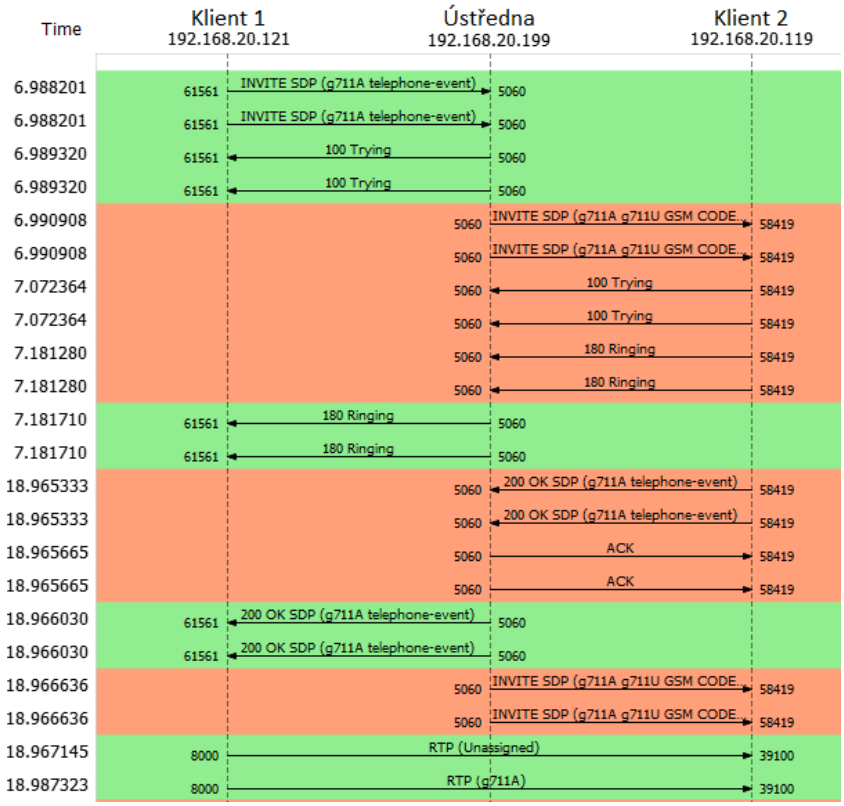
2.3.1.1 Chování v režimu B2BUA a proxy

Asterisk se v základním nastavení chová jako B2BUA. V tomto režimu prochází veškerá komunikace, včetně RTP paketů, skrze ústřednu. Jiným módem komunikace je proxy režim. Proxy režim lze nastavit v konfiguračním souboru. Výsledkem změny konfigurace je, že Asterisk změní své zacházení s hovory a výsledná komunikace bude vypadat, jako by se jednalo o proxy server.



Obrázek 2.5 Ukázka chování v režimu B2BUA

Obrázek 2.5 představuje ukázku defaultního zacházení s hovorem v ústředně Asterisk. Pro tento režim chování nejsou potřebná žádná další nastavení. Asterisk se v defaultním nastavení chová jako B2BUA. Na obrázku je vidět, že veškerá komunikace mezi klientem 1 a klientem 2 probíhá přes ústřednu. V podstatě se jedná o dvě samostatně probíhající relace, stejně jako to bylo nastíněno v kapitole 1.3.8. První relace probíhá mezi klientem 1 a ústřednou a druhá mezi ústřednou a klientem 2. Odlišným režimem komunikace, který byl popsán výše, je proxy režim. Základní režim komunikace Asterisku je režim B2BUA. Pro změnu módu komunikace do proxy režimu musí být upraven konfigurační soubor Asterisku.



Obrázek 2.6 Ukázka chování v proxy režimu

Obrázek 2.6 znázorňuje ukázku chování v proxy režimu. Nejprve dojde k sestavení hovoru. Po sestavení a navázání hovoru jsou přenášena samotná hlasová data mezi ústřednami. Pro použití tohoto režimu musí být upraveny konfigurační soubory obou stacků. U nativního stacku SIP je to provedeno přidáním řádku *directrtpsetup=yes* ke konfiguračnímu souboru *sip.conf*. V případě stacku PJSIP

se tento režim nastaví přidáním příkazu *direct_media=yes* do sekce *endpoint* v konfiguračním souboru *pjsip.conf*.

3. KAMAILIO A JEHO TESTOVÁNÍ

Práce na tomto projektu začaly roku 2001 a psaní programu se chopil Andrei Pelinescu–Onciul. Tehdy se projekt jmenoval SIP Express Router (SER) [21]. V průběhu čtyř let došlo k zrychlení kódu a přidání velkého množství funkcí a do projektu se zapojovalo více lidí. V roce 2005 došlo k oddělení několika hlavních vývojářů, kteří vytvořili projekt OpenSER. Ten se později přejmenoval na Kamailio.

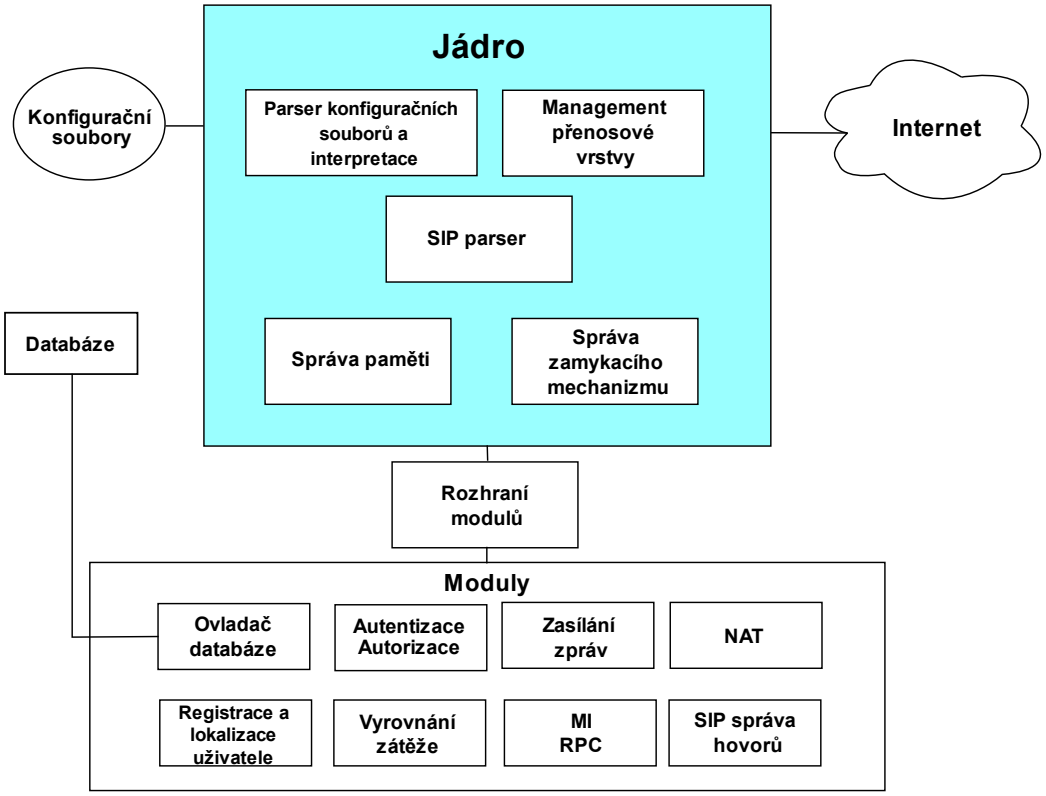
3.1 SIP server Kamailio

Kamailio je open source SIP Server. Zaměřuje se především na signalizaci sip a může zastávat funkce proxy serveru, aplikačního serveru, redirect a registrar serveru. Je naprogramován v jazyce C a stejně jako Asterisk, tak i Kamailio je postaveno na jádře, poskytující základní funkce a na modulech, přidávající další rozšiřující funkce. Jádro zajišťuje nízkoúrovňové funkce jako je správa paměti (slouží k přidělování privátní a sdílené paměti), SIP parser, řízení transportní vrstvy (podpora pro TCP, UDP a dále DNS), zamykací systém (locking systém), parser pro konfigurační soubory a interface pro správu a administraci. Ostatní funkce jsou realizovány moduly. Těch je opět obrovské množství. Programují se pomocí konfiguračních souborů.

V konfiguračních souborech se definují objekty a jejich parametry, pomocí kterých jsou řízeny vnitřní funkce. Moduly se načítají pomocí příkazu:

```
loadmodule "/cesta/modul.co"
```

Ke své funkci využívají také knihovny, které byly přidány ve verzi 3.0. Knihovny obsahují soubor funkcí. Jejich využívání má výhody v tom, že zabraňuje duplicitnímu kódu, opětovné využití poskytuje flexibilitu a funkce zahrnuté v knihovnách nemusí být obsaženy v jádře, tím pádem je samotné jádro menší. Knihovny stačí pouze nahrát do správného adresáře s knihovnami: *lib/*



Obrázek 3.1 Architektura Kamailia

Obrázek 3.1 znázorňuje zjednodušený pohled na architekturu SIP serveru Kamailio [22]. Jak bylo zmíněno výše, architektura se skládá z jádra a modulů. Jádro poskytuje základní funkce, ostatní je realizováno moduly. Základními prvky jsou parsery, management konfigurační vrstvy správa paměti a zamykacího mechanizmu. Konfigurace probíhá v konfiguračních souborech. Pomocí rozhraní modulů jsou zavedeny jednotlivé moduly. Na obrázku je pouze pár příkladů modulu, ale může jich být zavedeno více. Zde se může jednat o moduly přímo od vývojářů Kamailia, nebo moduly třetích stran.

3.2 Instalace

K instalaci SIP serveru Kamailio [23] se využila vyexportovaná kopie virtuálního stroje. Na kopii virtuálního stroje jsou již nainstalované balíčky, které využívá jak PBX Asterisk, tak i Kamailio, takže tím odpadá nutnost některé balíčky znovu instalovat. Ostatní balíčky, které jsou potřebné a nejsou nainstalovány, byly získány

z repozitáře projektu Kamailio. Ještě před samotnou instalací Kamailia bylo potřeba nainstalovat databázi, která je nutná ke správné funkci Kamailia. K tomu účelu byla využita MariaDB, což je databáze od tvůrců MySQL.

```
# yum -y install mariadb-server
# systemctl enable --now mariadb
# mysql_secure_installation
```

Po instalaci databáze byly z repozitáře staženy soubory potřebné k instalaci Kamailia.

```
# yum install kamailio kamailio-presence kamailio-ldap kamailio-mysql kamailio-debuginfo kamailio-xmpp kamailio-unixodbc kamailio-utils kamailio-tls kamailio-outbound kamailio-gzcompress
```

Poté byly v databázi vytvořeni dva základní uživatelé. Uživatel kamailio s právy pro čtení i zápis do databáze a uživatel kamailioro s možností čtení z databáze. K tomu byl využit příkaz:

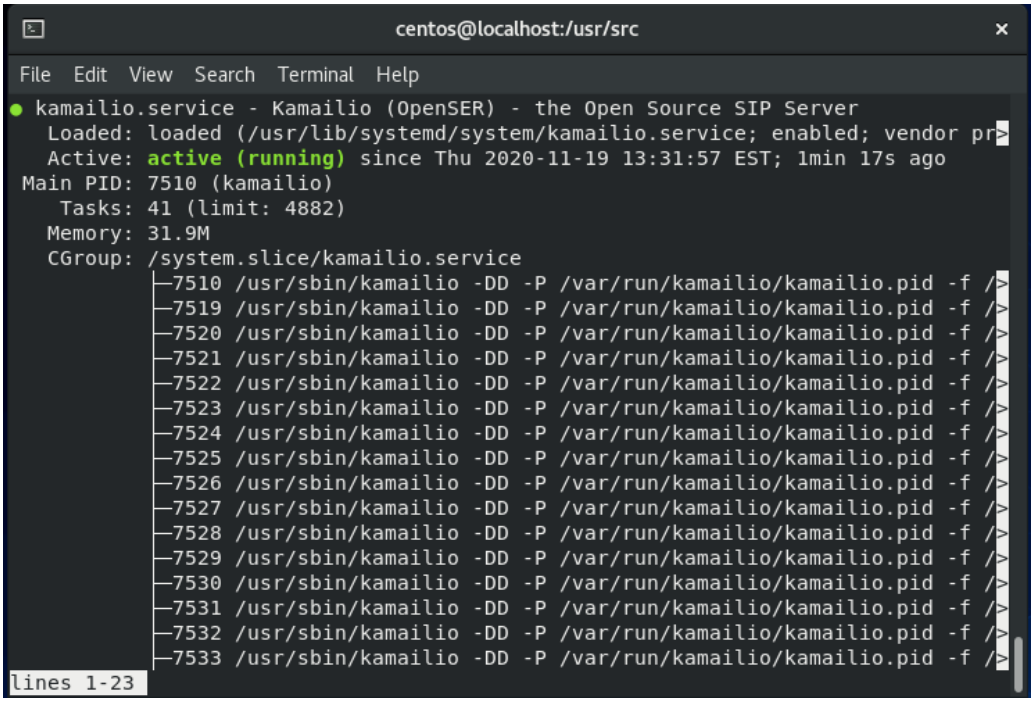
```
# kamdbctl create
```

Poté bylo potřeba upravit konfigurační soubory. Konkrétně do nich přidat řádky, které povolují využívání databáze, autentizaci uživatelů, zachování registračních záznamů uživatelů, průchod skrze NAT. To bylo realizováno přidáním následujících příkazů do konfiguračního souboru:

```
#!define WITH_MYSQL
#!define WITH_AUTH
#!define WITH_USRLOCDB
#!define WITH_NAT
#!define WITH_PRESENCE
#!define WITH_ACCDB
```

Následně se služba nastaví na spuštění při startu virtuálního stroje:

```
# systemctl enable kamailio
```



Obrázek 3.2 Nainstalované Kamailio

Obrázek 3.2 zobrazuje výpis stavu služby Kamailio. Je vidět, že služba je aktivní, dále je zde její PID, kolik spotřebovává paměti a další podrobnosti. V tomto stavu je proxy server Kamailio schopný plně fungovat a již s defaultním nastavením konfiguračních souborů je server schopný realizovat hovory. Aby bylo možné realizovat hovory, tak je nutné přidat jednotlivé účty koncových zařízení do databáze Kamailia. Přidávat účty do databáze lze několika způsoby. Jedním z nich je využití kamctl. Kamctl je zkratka pro nástroj Kamailio control tool. Tento nástroj slouží pro správu Kamailia, domén, nastavení a uživatelů. Jednotlivé příkazy tohoto nástroje jsou zadávány do příkazového řádku v následující syntaxi – *kamctl příkaz [parametry]*. Příkaz pro přidání uživatele následně vypadá: *kamctl add <username> <password>*, kde za username bude dosazeno požadované URI koncového zařízení a za password zvolené heslo. Po zadání příkazu bude záznam o novém účtu zapsán do databáze Kamailia a k tomuto účtu se nyní může registrovat koncové zařízení.

Druhou možností, jak přidávat a upravovat uživatelské účty, která byla využita v diplomové práci, je přímé vkládání a úprava záznamů v tabulkách databáze. Podobně jako i jiné databázové systémy, tak i MariaDB umožňuje pomocí dotazů přistupovat k jednotlivým záznamům v databázi, upravovat jejich atributy nebo s nimi dále jinak operovat. Přímá práce s databází byla využita i z toho důvodu,

že bylo potřeba vložit do databáze větší množství uživatelských účtů a práce s databází poskytuje nástroje pro jejich vložení. Snížila se tak celková časová náročnost oproti tomu, kdyby byli jednotliví uživatelé přidáváni pomocí příkazu *kamctl add*.

Účet v databázi je definován pomocí několika položek. Jimi jsou ID, uživatelské jméno, doména, heslo a hash. Pro testovací účely byly účty vytvářeny bez unikátního hashe. Pomocí bash skriptu byl vytvořen soubor obsahující všechny použité účty. Struktura skriptu s názvem *zapis.sh* je následující:

```
for i in {1000..3000}
do
cat >> zapis << EOF
$(($i – 999)), $i, 192.168.20.199, 1234
EOF
done
```

Výstupem tohoto skriptu je textový soubor, který obsahuje tři tisíce řádků a každý řádek reprezentuje jeden účet. Účty mají své ID začínající od čísla 1, uživatelské jméno z rozsahu 1000 až 3000, doménu a heslo. Obsah souboru zapis.sh je:

```
1, 1000, 1000, 192.168.20.199, 1234
2, 1001, 1001, 192.168.20.199, 1234
3, 1002, 1002, 192.168.20.199, 1234
.
.
2001, 3000, 1001, 192.168.20.199, 1234
```

Takto připravený soubor byl nahrán do databáze Kamailia do tabulky *subscribers.* Pro přístup do databázového systému byl použit příkaz

```
mysql -u root -p -h localhost
```

kde parametr *-u* specifikoval jméno, parametr *-p* specifikoval heslo (to bylo možné zadat hned za parametr *-pHESLO*, nebo ponechat pouze *-p* a uživatel bude vyzván

k zadání hesla v samostatném okně) a parametr *-h* specifikoval přihlašovaného uživatele. Po zadání hesla byla vybrána zvolená databáze pomocí příkazu:

```
use kamailio
```

Tabulky zvolené databáze je možné procházet a nahlížet do jejich záznamů pomocí:

```
show tables
SELECT * FROM subscriber
```

Posledním krokem bylo nahrání obsahu z požadovaného souboru do tabulky subscribers. K tomu bylo potřeba specifikovat, o jaký soubor se jedná, zadat jeho umístění, vybrat cílovou tabulku k vložení dat a separovat text z řádku na jednotlivé sloupce.

```
LOAD DATA LOCAL INFILE '/home/centos/test' INTO TABLE subscriber FIELDS TERMINATED BY ', ';
```

Tímto byly nastaveny jednotlivé účty. Pro dosažení funkčnosti bylo potřeba upravit i nastavení směrování hovorů. Základní konfigurace by mohla být dostačující například pro využití u méně rozsáhlých sítí, čítající jednotky až desítky koncových bodů. V případě, že je server nasazen do rozsáhlejší sítě nebo je požadováno specifické zacházení s některými hovory, jako například směrování hovorů k vybraným koncovým bodům, tak stávající konfigurace nemusí splňovat požadavky a je potřeba implementovat složitější schémata směrování hovorů. Tyto úpravy jsou prováděny v konfiguračním souboru Kamailia *kamailio.cfg*. Tento soubor obsahuje globální konfigurační parametry, příkazy pro načítání modulů a script pro spouštění příkazů. Soubor je strukturovaný do sekcí, kde první sekce obsahuje nastavení globálních parametrů Kamailia, v další sekci se načítají jednotlivé moduly a upravují se jejich specifická nastavení a v poslední sekci je obsaženo nastavení přepojování hovorů. Jednotlivé moduly mohou být závislé na jiných modulech a je potřeba zavést všechny požadované moduly pro zajištění správného fungování. V opačném případě nebude možné spustit Kamailio.

Úpravy prováděné v konfiguračním souboru byly cíleny na úpravu globálního nastavení Kamailia a úpravu směrování hovorů. V globálním nastavení byla upravena položka children, jejíž hodnota by měla odpovídat počtu jader procesoru.

```
children=4
```

Následně byl nahrán modul *uac.so*, který potřeboval ke svému fungování modul *rr.so*. Modul *uac.so* poskytuje základní funkce pro koncové body a modul *rr.so* se stará o směrování hovorů. Modul *rr.so* byl již v základní konfiguraci zaveden a bylo nutné upravit pouze parametr *append_fromtag*, čímž bylo zajištěno, že se k některým položkám v SIP hlavičce přistupuje jako k tagu. Tag je následně reference pro tuto hodnotu.

```
modparam("rr", "append_fromtag", 1)
loadmodule "uac.so"
```

Nyní, když byl modul uac.so úspěšně zaveden, bylo potřeba spravit směrovací plán Kamailia, aby došlo k přesměrování hovoru na správný účet testeru. Podobně jako u Asterisku, tak i zde bylo nutné přesměrovat hovor na specifický účet. Konfigurace se prováděla v sekci směrovací logiky – v sekci *ROUTE[LOCATION]*, kam byl přidán řádek s kódem, který upravuje URI ze všech příchozích hovorů na jedno specifické URI. Tím bylo docíleno, že tester dokáže realizovat hovory skrze proxy server Kamailio.

```
ROUTE[LOCATION] {
rewriteuri("sip:3000@192.168.20.199");
.
.
}
```

4. POROVNÁNÍ FUNKCIONALIT PROJEKTŮ ASTERISK A KAMAILIO

Rozdíl obou systémů spočívá v tom, jak jsou oba navrženy. Kamailio je systém, který je navržen jako SIP proxy server. Charakteristika serveru tak umožňuje manipulovat pouze s hlavičkami SIP protokolu. SIP server tak dokáže pouze přijímat data a dle hlaviček, do kterých nahlédne, nebo je modifuje, se následně rozhodne, jak se bude dále zpracovávat. Od toho se také odvíjí škála funkcionalit, které SIP proxy server poskytuje. U Asterisku je situace odlišná. V základu se chová jako B2BUA a poskytuje velké množství funkcí pro zpracování hovoru. Je zde také možnost nastavení pobočkové ústředny jako proxy server.

V následujících tabulkách je vypsán přehled funkcí a možností jednotlivých systémů, které jsou uvedeny na oficiálních stránkách Asterisku a Kamailia [24][25][26]. V první části jsou funkcionality, které byly získány ze stránek Asterisku. K nim bylo hledáno v dokumentaci Kamailia, v dokumentaci jednotlivých modulů a na diskusním fóru Kamailia, zda Kamailio disponuje stejnou funkcionalitou, nebo alespoň alternativou této funkcionality. V druhé tabulce jsou uvedeny funkcionality získané z oficiálních stránek Kamailia a obdobně jako v předchozím případě jsou funkcionality srovnávány s funkcionalitami Asterisku. Zde již byly vynechány funkcionality, které jsou uvedeny v první tabulce.

Tabulka 4.1 Přehled funkcionalit uvedených na stránkách Asterisku

Funkcionality uváděné na oficiální stránce Asterisku			
Funkcionalita	**Asterisk**	**Kamailio**	**Popis**
ADSI	Ano	Ne	Implementace protokolu ADSI
Alarm Receiver	Ano	Ne	Nastavení upozornění pomocí telefonu
Append Message	Ano	Ano	Doručení zprávy na e–mail
Authentication	Ano	Ano	Autentizace klientů
Automated Attendant	Ano	Ne	Automatické předání
Blacklists	Ano	Ano	Seznam hovorů, které budou zamítnuty
Blind Transfer	Ano	Ne	Přepojení hovoru na jinou linku
Call Detail Records	Ano	Ano	Detaily o zprostředkovaném hovoru
Call Forward on Busy	Ano	Ano	Přesměrování v případě, kdy je volaná linka obsazená
Call Forward on No Answer	Ano	Ano	Přesměrování v případě, když není získána odpověď na hovor

Call Forward Variable	Ano	Ano	Přesměrování pomocí kódu
Call Monitoring	Ano	Ne	Monitorování a evidence proběhlých, příchozích a odchozích hovorů
Call Parking	Ano	Ne	Přidržení hovoru s možností vyzvednutí na stejné nebo jiné lince
Call Queuing	Ano	Ano	Automatická distribuce příchozích hovorů
Call Recording	Ano	Ano	Nahrávání hovorů a jejich následné uložení
Call Retrieval	Ano	Ne	Volání hovorů pozastavených funkcí Call parking
Call Routing	Ano	Ano	Směrování hovorů
Call Snooping	Ano	Ne	Naslouchání hovoru bez přenosu hlasu naslouchajícího
Call Transfer	Ano	Ne	Předání hovoru na jinou linku
Call Waiting	Ano	Ano	Funkce, která při dvou příchozích hovorech umožňuje mezi těmito hovory přepínat
Caller ID	Ano	Ano	Zobrazení čísla a jména volajícího
Caller ID Blocking	Ano	Ne	Blokování na základě identifikátoru volajícího
Caller ID on Call Waiting	Ano	Ne	Zobrazení čísla a jména dalšího volajícího
Conference Bridging	Ano	Ne	Vytvoření konference mezi účastníky
Database Store / Retrieve	Ano	Ano	Využití databáze pro ukládání a nahrávání dat
Database Integration	Ano	Ano	Poskytování informací o hovoru před jeho zvednutím nebo v jeho průběhu
Dial by Name	Ano	Ne	Vytáčení pomocí záznamu kontaktu
Direct Inward System Access	Ano	Ne	Přístup k funkcím z vnější straněy sítě
Distinctive Ring	Ano	Ne	Rozdílné vyzvánění dle volajícího
Distributed Universal Number Discovery (DUNDi™)	Ano	Ne	Funkce pro sdílení informací z dialplanu mezi telefony
Do Not Disturb	Ano	Ne	Funkce pro nastavení nedostupnosti linky
E911	Ano	Ano	Funkce pro nouzová volání
ENUM	Ano	Ano	Funkce pro mapování čísla a internetové adresy
Fax Transmit and Receive	Ano	Ne	Funkce pro přijímání a odesílání faxů
Flexible Extension Logic	Ano	Ne	Flexibilní směrování hovorů
Interactive Directory Listing	Ano	Ne	Vyhledávání kontaktu podle jména ve firemním adresáři
Interactive Voice Response (IVR)	Ano	Ne	Interaktivní hlasový automat
Local and Remote Call Agents	Ano	Ano	Funkce pro přihlášení uživatele i na jiném telefonu a jeho využívání jako vlastní telefon

Macros	Ano	Ano	Možnost využití maker
Music On Hold	Ano	Ano	Hudba při pozdrženém hovoru
Music On Transfer:	Ano	Ne	Hudba při transferu hovoru
Privacy	Ano	Ne	Možnost využít privacy managementu, uživatel je požádán o zadání kódu pro realizaci hovoru
Open Settlement Protocol (OSP)	Ano	Ano	Protokol pro autorizaci a účtování služeb
Overhead Paging	Ano	Ano	Funkce pro odesílání page zpráv skupině; interkom
Protocol Conversion	Ano	Ne	Funkce pro konverzi komunikačních protokolů u různých druhů sítí
Remote Call Pickup	Ano	Ne	Vzdálené vyzvednutí hovoru
Remote Office Support	Ano	Ne	Funkce pro vzdálené připojení telefonu mimo území PBX
Roaming Extensions	Ano	Ne	Možnost připojení se k jiným telefonům, identifikace kódem
Route by Caller ID	Ano	Ano	Směrování hovoru dle identifikátoru volajícího
SMS Messaging	Ano	Ano	SMS zprávy
Spell / Say	Ano	Ne	Přečtení specifického čísla nazpět volajícímu
Supervised Transfer	Ano	Ne	Transfer, při kterém je dohlíženo na to, zda proběhl úspěšně
Talk Detection	Ano	Ne	Detekce hlasu ke spouštění různých událostí
Text–to–Speech	Ano	Ne	Funkce předčítání textu
Three–way Calling	Ano	Ne	Přidání dalšího účastníka do již probíhajícího hovoru
Time and Date	Ano	Ano	Funkce pro nastavení času a data
Transcoding	Ano	Ano	Umožňuje překódovat hovory, které využívají různé kodeky pro přenos zvuku
Trunking	Ano	Ano	Možnost připojení pobočkové ústředny do jiného systému
VoIP Gateways	Ano	Ano	Možnost využití jako brány pro různé systémy
Voicemail	Ano	Ano	Funkce k nahrávání vzkazů a odeslání do hlasové schránky – funkce indikace přijaté hlasové zprávy, možnost odeslání zprávy na určitý e–mail nebo skupině
Web Voicemail Interface	Ano	Ne	Triviální rozhraní pro správu
Zapateller	Ano	Ne	Blackhole pro nevyžádané marketingové hovory

Tabulka 4.2 Přehled funkcionalit uvedených na stránkách Kamailia

Funkcionality uváděné na oficiální stránce Kamailia			
Funkcionalita	**Asterisk**	**Kamailio**	**Popis**
Accounting	Ano	Ano	Účtování na základě události
Benchmark	Ne	Ano	Nástroje pro výkonové testování
CLI	Ano	Ano	Podpora uživatelského rozhraní v podobě příkazového řádku
CPL – Call Processing Language	Ne	Ano	Podpora kombinovaného programovacího jazyka CPL
Call blocking	Ne	Ano	Funkce pro omezení provozu v případě přetížení
Call limitation	Ano	Ano	Možnost omezit hovory na určitý počet nebo na určitou délku dle parametrů hovorů
DNSsec support	Ano	Ano	Podpora DNSsec
Dialplan	Ano	Ano	Nástroj směrování hovorů. Dle parametrů příchozích hovorů a nastavených pravidel
Diameter support	Ne	Ano	Podpora AAA protokolu diameter
Erlang Programming Language	Ne	Ano	Podpora programovacího jazyka Erlang
Flexible debug and error message logging system	Ano	Ano	Možnost debugování a ladění chybových zpráv
Flexible least cost routing	Ne	Ano	Směrování na základě nejlepší ceny spoje
Forking/multiple registration	Ano	Ano	Možnost registrace jednoho uživatele u více zařízení
Gateway to sms or xmpp and other IM services	Ano	Ano	Možnost využití jako brány pro služby okamžitého zasílání zpráv IM (instant messaging)
IPv4 and IPv6	Ano	Ano	Podpora protokolů IPv4 a IPv6
JSON implementation	Ano	Ano	Možnost získání dat SS7 a převod na objektový zápis – JSON
Java programming language	Ano	Ano	Podpora programovacího jazyka Java
JavaScript programming language	Ne	Ano	Podpora programovacího jazyka JavaScript
LDAP integration	Ano	Ano	Podpora protokolu LDAP
Load balancing	Ne	Ano	Vyvažování zátěže – směrování hovorů dle vybraného algoritmu
Location server	Ano	Ano	Funkce lokačního serveru
Lua programming language	Ano	Ano	Podpora programovacího jazyka Lua
Managed Code (C#) programming language	Ne	Ano	Podpora programovacího jazyka C#
Memcached connector	Ne	Ano	Podpora Memcashed pro efektivní práci s pamětí
Message body compression/decompression	Ano	Ano	Komprese a dekomprese těla zprávy využitím kodeků

Modular architecture	Ano	Ano	Možnost rozšíření funkcí pomocí modulů
NAT traversal	Ano	Ano	Podpora překonání NAT
Offline messaging	Ano	Ano	Umožňuje přijímat SMS i klientům, kteří jsou offline a ukládat je do databáze
Over 1000 parameters, variables and functions exported to config file	Ne	Ano	Velké množství funkcí, proměnných a parametrů pro konfiguraci chování
Perl programming language	Ano	Ano	Podpora programovacího jazyka Perl
Plug&play modules	Ano	Ano	Jednoduchost zavádění nových modulů
Presence	Ano	Ano	Funkce pro dohled nad dostupností uživatelů a reakce na různé události
Proxy server	Ano	Ano	Funkce proxy server
Python programming language	Ano	Ano	Podpora programovacího jazyka Python
QOS	Ano	Ano	Umožňuje sledovat dialog a poskytovat různé úrovně QoS
RPC control interface – via XMLRPC, JSONRPC, UDP or TCP	Ne	Ano	Remote Procedure Call – rozhraní pro připojení externích aplikací
Radius support	Ano	Ano	Podpora AAA protokolu Radius
Redirect server	Ano	Ano	Funkce serveru přesměrování
Registrar server	Ano	Ano	Funkce registrar serveru
Remote control via XMLRPC	Ne	Ano	Možnost vzdáleného přístupu a volání procedur pomocí XMLRPC
Repository	Ano	Ano	Repozitáře se zdrojovými kódy a přídavným obsahem jako jsou repozitáře
Resource monitoring	Ne	Ano	Nástroje pro sledování zatížení
Routing failover	Ne	Ano	Využití záložní cesty při výpadku
Ruby Programming Language	Ano	Ano	Rozhraní pro programovací jazyk Ruby
Runtime update framework	Ne	Ano	Možnost upravovat parametry v provozu bez nutnosti restartu
SCTP multi–homing and multi–streaming	Ano	Ano	Podpora multihoamingu a multistreamingu
SIP Application server	Ne	Ano	Funkce aplikačního serveru
SNMP – interface to Simple Network Management Protocol	Ano	Ano	Rozhraní pro správu pomocí protokolu SNMP
Small footprint	Ano	Ano	Nízká paměťová náročnost
Straightforward interconnection with PSTN gateways	Ano	Ano	Možnost využití jako brány pro veřejnou telefonní síť
Suport of databases with have unixodbc drivers	Ne	Ano	Podpora databází
Support for SRV and NAPTR DNS lookups	Ano	Ano	Funkce pro zjišťování upřesňujících informací z SRV a NAPTZ záznamů

Support of API	Ano	Ano	Využití aplikačního programovacího rozhraní
Support of UDP, TCP, TLS and SCTP	Ano	Ano	Podpora komunikačních protokolů UDP, TCP, TLS a SCTP
Support of scripts	Ano	Ano	Podpora skriptů a jejich možné využití pro konfigurační soubory
TLS support	Ano	Ano	Podpora TLS pro signalizaci
Web Management Interface: Siremis	Ne	Ano	Nástroj pro správu Siremis
Whitelist	Ano	Ano	Seznam hovorů, které mají být povoleny
XCAP client capabilities	Ne	Ano	Možnost modifikace konfiguračních dat uložených v XML souborech pomocí XML Configuration Access Protocol

Obě tabulky obsahují funkcionality jednotlivých ústředen. Tabulka 4.1 obsahuje přehled funkcionalit pobočkové ústředny Asterisk a jejich porovnání se SIP serverem Kamailio. Tabulka 4.2 obsahuje přehled funkcionalit SIP serveru Kamailio a jejich porovnání s pobočkovou ústřednou Asterisk. Z výše popsaných funkcionalit vyplývá, že SIP proxy server Kamailio poskytuje menší množství funkcionalit pro práci přímo s RTP streamem, jak bylo nastíněno na začátku kapitoly. Je to způsobeno právě možnostmi proxy serveru, které škálu funkcionalit omezují. Na druhou stranu pobočková ústředna Asterisk dokáže zpracovávat i obsah RTP paketů a poskytovat tak širší paletu funkcí.

Pobočková ústředna Asterisk nabízí větší množství funkcí než SIP server Kamailio. To ovšem neznamená, že Asterisk je správnou volbou pro všechny typy aplikací, konfigurací a sítí. Na rozdíl od Asterisku je Kamailio navrženo tak, aby mělo menší paměťové nároky, a proto bylo rychlejší a stabilnější než Asterisk. Kamailio jako pobočkovou ústřednu je proto vhodné implementovat tehdy, kdy není potřeba manipulace s médii.

Jako poslední byla porovnávána jednoduchost ovládání a konfigurace. Dle mých zkušeností je u Asterisku mnohem jednodušší k pochopení, jak se zachází s hovory, jak probíhá konfigurace nebo jak se odladí některé problémy. Zároveň existuje rozsáhlá dokumentace a fóra, kde je možné dohledat další informace a podpůrné materiály. Poslední zmíněné platí i v případě Kamailia. Zde je ovšem, na základě zkušenosti, mnohem náročnější na pochopení, jak funguje logika Kamailia [27] a jak je pohlíženo na jednotlivé hovory. Po pochopení logiky Kamailia pak samotná konfigurace nabízí rozsáhlé možnosti úprav.

5. SPIRENT TESTCENTER C1

Veškeré testování výkonnosti ústředen Asterisk a Kamailio bylo prováděno na testeru Spirent TestCenter C1. Jedná se o přenosný generátor zátěže. Je určen pro generování zátěže na vrstvách L2 – L7 modelu OSI/ISO. Využívá se pro testování různých prvků sítě, kde se může jednat o směrovače nebo přepínače, servery, databáze, firewally a další. Z toho důvodu ho lze využít k testování sítě ať už při jejím budování, modifikaci, ale také při ověřování kvality a spolehlivosti jednotlivých prvků. Zařízení, využívané pro testování v diplomové práci, má 4 ethernetové porty, každý pro 1Gbps. Na vrstvách L2 – L3 je tedy schopný generovat provoz k testování kvality služeb QoS, který dosahuje maximální linkové rychlosti 1 Gbps. Na vrstvách L4 – L7 dokáže vygenerovat zátěž o velikosti až 5 Gbps a slouží k testování aplikačních protokolů a aplikací. Mezi podporované protokoly patří IPV4 a IPV6, TCP, UDP, SSLv3, HTTP, HTTPS, FTP, DNS, DNSTCP, DNSSEC, TELNET, SMTP, POP3, ICMP, Network Access Control (NAC), Radius. Další podporované protokoly a technické parametry testeru lze dohledat v katalogovém listu [28].

Obrázek 5.1 Spirent TestCenter C1

Obrázek 5.1 obsahuje tester Spirent TestCenter C1 [29] použitý pro realizaci testů. Pro potřeby diplomové práce bylo potřeba generovat VoIP provoz. O tu se stará část testeru, pracující na vrstvách L4 – L7. S testerem jsou distribuována i další softwarová řešení v podobě programů pro práci s testerem. Mezi ně patří například Avalanche DHCP testing, Avalanche attack designer, Avalanche Extreme Scale & Performance nebo Avalanche SIPNG.

5.1 Aplikace SIPNG

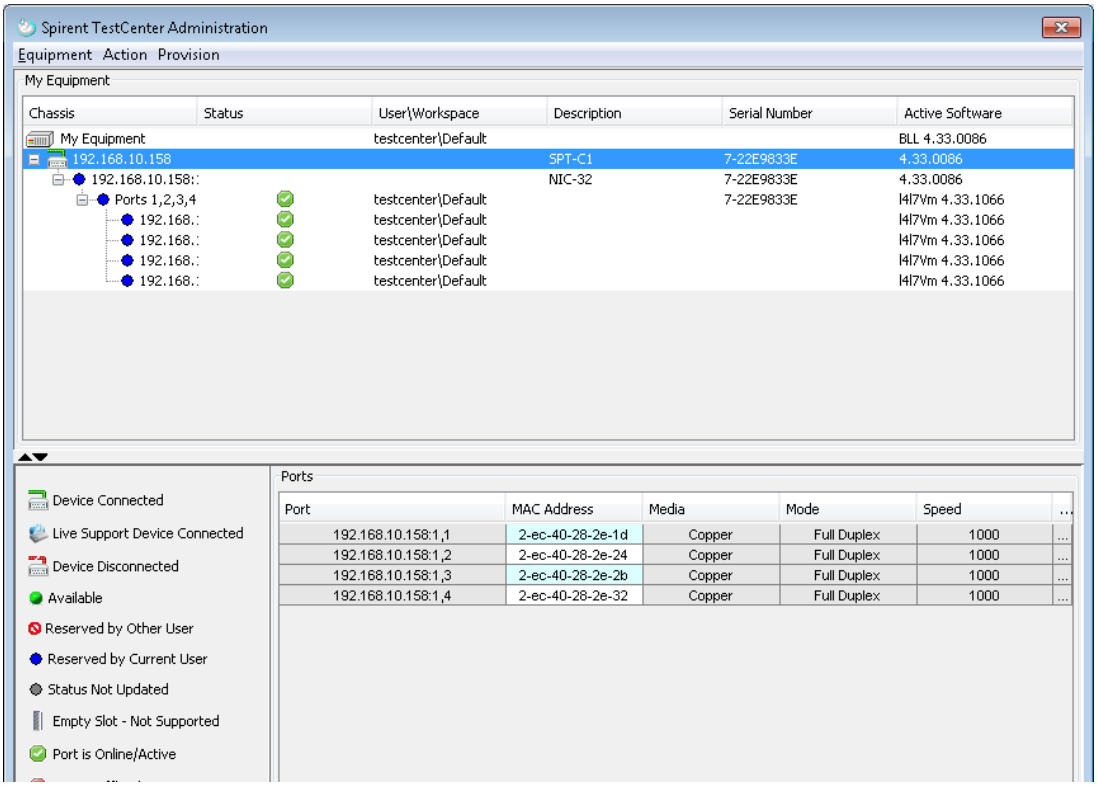
Tester od společnosti Spirent dokáže pracovat s velkým množstvím protokolů, jak bylo popsáno výše. Pro simulaci datového provozu VoIP využívá aplikaci

SIPNG [30]. Aplikace je vhodná pro testování škálovatelnosti sítě a pro sledování jejího chování za zvýšeného datového provozu. Dokáže emulovat chování koncových zařízení i serveru. To umožňuje simulovat komunikaci mezi klienty i mezi klienty a serverem. Pro komunikaci mezi emulovanými koncovými body používají signalizační metody a odpovědi protokolu SIP. Mezi podporované metody patří: REGISTER, INVITE, ACK a BYE a z odpovědí jsou podporovány všechny třídy (1xx, 2xx…6xx). Mezi klienty lze přenášet i RTP data. Tady se jedná o předem nahrané zprávy, nikoliv o hlasový záznam pořizovaný v průběhu testu. Aplikace podporuje značné množství kodeků. Mezi ně patří: G.711u, G.711a, G.723, G.726, G.728, G.729A a G.729AB. Kromě běžné struktury hlavičky SIP protokolu umožňuje aplikace SIPNG vytvářet i modifikované záhlaví SIP.

5.2 Spirent TestCenter Layer 4–7 Application

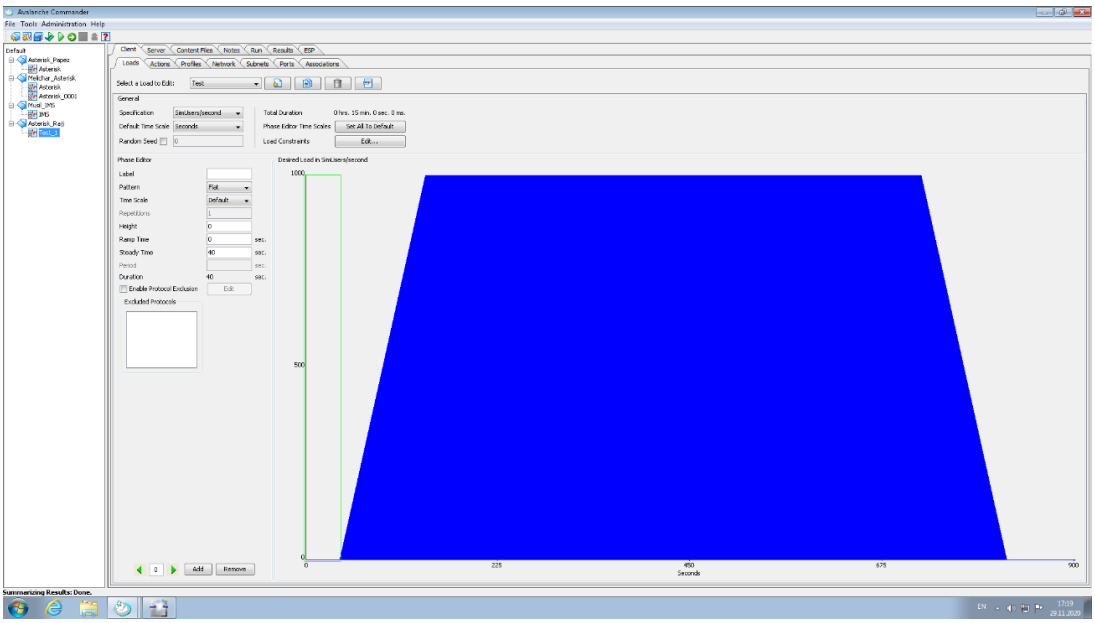
Pro ovládání, kontrolu a získávání výsledků z testeru se používá program Spirent TestCenter Layer 4–7 Application. Jak již název vypovídá, jedná se o aplikaci pro práci s protokoly na vrstvách L4 – L7, které tester podporuje. Po spuštění aplikace se otevře okno programu Avalanche Commander. Tento program poskytuje uživatelské rozhraní pro práci s testerem Spirent TestCenter C1. V tomto programu lze vytvářet projekty a v nich sdružovat jednotlivé testovací scénáře. Ty lze nadefinovat pomocí zjednodušeného typu testu (EZ Test), rychlého testu (Quick) a pokročilého testu (Advanced). Všechny tři typy se od sebe odlišují množstvím možností nastavení, kde EZ Test nabízí nejmenší množství možností, ale jeho konfigurace je jednodušší a nejrychlejší, zatímco Advanced test nabízí největší množství možností pro konfiguraci testu.

Před testováním je nutné nejprve zajistit komunikaci mezi programem Avalanche Commander a testerem. To se provádí v kartě Administration v položce Spirent Testcenter Chassis.



Obrázek 5.2 Okno registrace portů

Obrázek 5.2 zachycuje okno pro registraci portů. V tomto případě je k testeru připojena jedna karta – NIC–32, která je osazena čtyřmi porty. Tlačítkem refresh lze zjistit dostupnost portů a následně tlačítkem reserve si je lze zarezervovat.

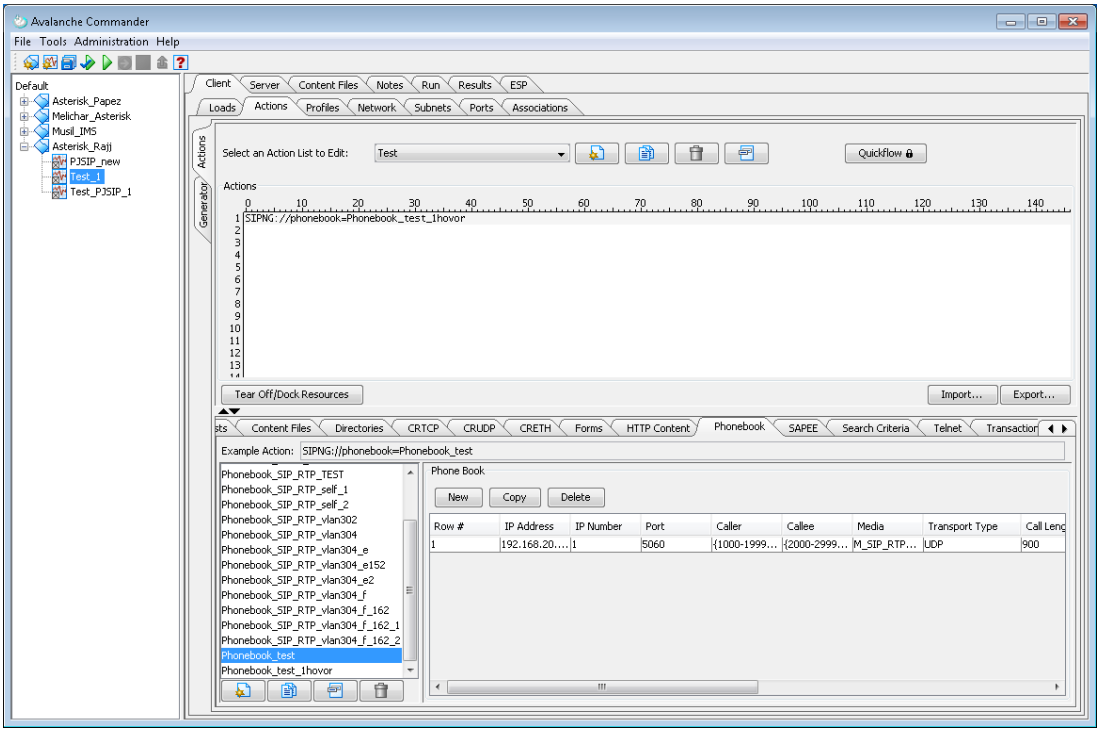


Obrázek 5.3 Prostředí programu Avalanche Commander

Obrázek 5.3 obsahuje uživatelské rozhraní programu Avalanche Commander. V jednotlivých kartách programu lze nastavovat, jak se bude chovat testovací scénář. Toto nastavení se dělí na nastavení klienta (karta Client) a nastavení serveru (karta Server).

Do nastavení klientské části spadá nastavení zátěže (Load), akce (Action), profilu (Profiles), sítě (Network), podsítě (Subnet), portů (Ports) a karta asociace (Association). Na obrázku nahoře je vidět karta pro nastavení zátěže testu. Na této kartě lze nadefinovat, jak intenzivní bude generovaný datový provoz. Mezi nejdůležitější položky patří specifikace testu (položka Specification). Ta nabízí na výběr několik možností. První možností je položka Simusers. Toto je nejvhodnější nastavení pro aplikaci SIPNG. Simusers odpovídá počtu virtuálních uživatelů, kteří budou vygenerováni. Množství odpovídá velikosti nastavené zátěže. Každý virtuální uživatel provede jednu z předem definovaných akcí ze seznamu akcí. Ze specifikací lze ještě vybrat položku Simusers/sec a Simusers/hod. Při použití těchto položek se tester snaží udržovat požadovaný počet přírůstků uživatelů za sekundu/hodinu. Mezi zbylé možnosti položky Specification patří položky Transactions, Connections, Bandwidth a BodyBytes. S položkou Transactions tester generuje takový provoz, aby udržel požadovaný počet transakcí. Tyto transakce jsou pouze HTTP a HTTPS, proto se tato možnost pro aplikaci SIPNG nepoužívá. Další možností je položka Connections, která zaručuje generaci TCP spojení. BodyBytes generuje HTTP požadavky a Bandwidth určuje množství dat pro odesílání přes všechna dostupná rozhraní.

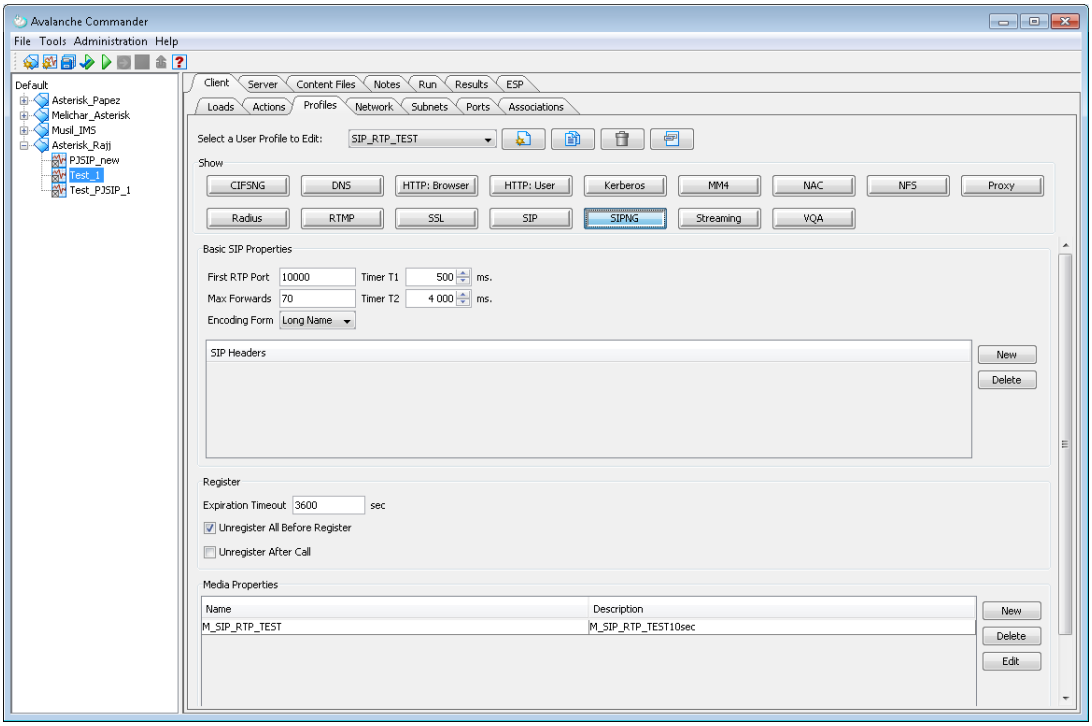
Profil zátěže lze nastavovat v sekci Phase editor. Grafická reprezentace nadefinované zátěže je vidět ve vedlejším okně s názvem Designated load. Nastavení probíhá po oddílech. Nejprve je přidán oddíl, který lze libovolně pojmenovat. Ten má zprvu nulovou hodnotu. Lze zvolit jeden z přednastavených tvarů, jako je schodovitý tvar, tvar sinusoidy nebo nahodilý tvar, popřípadě ponechat rovný tvar a nadefinovat si průběh samostatně. Upravováním položek výšky, doby nárůstu a doby stálosti (height, ramp time, steady time) lze formovat výsledný tvar zátěže. Součet doby trvání všech oddílů odpovídá celkové době testu a je vidět u položky Total Duration.



Obrázek 5.4 Karta nastavení akcí

Obrázek 5.4 obsahuje kartu nastavení akce v programu Avalanche Commander. Dle použité aplikace se může obsah akčního listu lišit, ale vždy platí, že každý vytvořený objekt provede každou akci z akčního listu pouze jednou. V případě SIPNG jsou to simulovaní uživatelé, kteří provádějí akce. Obsah akčního listu se může lišit dle použitého protokolu a může mít rozsah jednoho až několika řádků. Některé protokoly jsou dále rozšířeny o další parametry, na které se odkazují. Příkladem může být Phonebook, který se používá právě pro SIPNG a obsahuje další rozšiřující parametry.

Rozšiřujícími parametry v Phonebooku jsou: IP adresa a port, volající, volaný, média, transportní protokol, minimální a maximální délka hovoru, IP adresa a port registrar serveru, na které se budou uživatelé registrovat a heslo k registraci. Nastavením volajícího a volaného (Caller a Callee) lze nastavit, z jakých URI se bude volat a na jaké URI bude voláno. Nastavit lze jedna specifická URI, ale i rozsah URI. Parametr média nastavuje, jaká média se budou přenášet. Tato volba se blíže specifikuje v kartě Profiles. Transportní protokol lze použít UDP nebo TCP. Nastavení minimální a maximální délky hovoru ovlivňuje, jak dlouho budou probíhat jednotlivé hovory. Avalanche vybere pro každý hovor náhodnou hodnotu ze zadaného rozsahu.



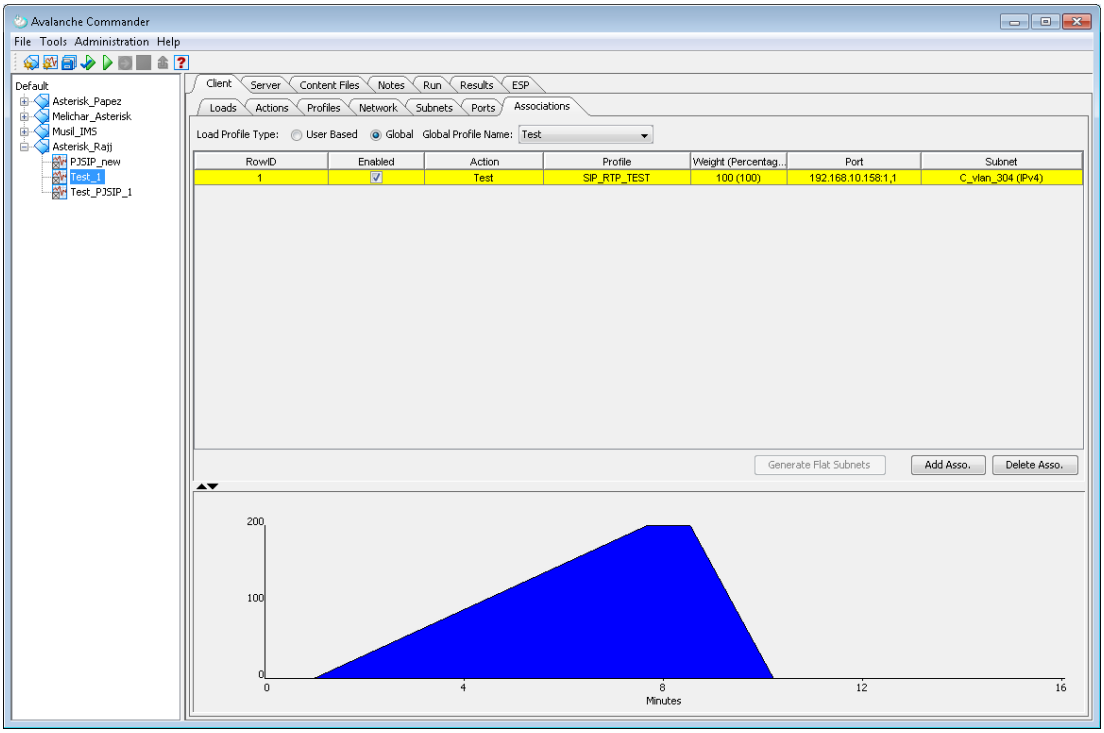
Obrázek 5.5 Karta nastavení profilu protokolu

Obrázek 5.5 zachycuje nastavení a jejich nastavení v kartě Profiles. Zde lze zvolit použitý protokol a upravit některé parametry. Pro SIPNG tady může být upraveno základní nastavení SIP protokolu, jako je první RTP port, časovače T1 a T2 nebo přidávat vlastní hlavičku k metodě INVITE. Další nastavení ovlivňuje registrace virtuálních uživatelů. Zde může být nastavena doba vypršení platnosti registrace a zda má být provedeno odregistrování uživatelů před registrací a po provedení hovoru.

V sekci Media Properties se definuje, jaká data budou přenášena v RTP paketech. Po přidání nové položky tlačítkem New a následně editací tlačítkem Edit se otevře okno pro přidání přenášených dat. Ta lze nastavit jako náhodná data, data ze souboru nebo žádná data. Při nastavení žádných dat nebudou v průběhu testu přenášeny žádné RTP pakety. Nastavení náhodných dat vytváří RTP pakety s náhodným obsahem, a tento obsah je možné ještě blíže specifikovat nastavením velikosti těchto dat, typem dat a nastavením intervalu mezi jednotlivými RTP pakety. Poslední možností, která byla použita v diplomové práci, je nastavení obsahu ze souboru. Soubor lze nahrát ve formátu .wav a pro jeho přenos musí být nastaven některý z kodeků.

Dalšími kartami, kterými lze upravovat další chování testu, ale v diplomové práci byly jejich parametry ponechány v defaultních hodnotách, jsou karty Network, Subnet a Ports. Karta Network dovoluje upravovat TCP parametry, porty, směrování

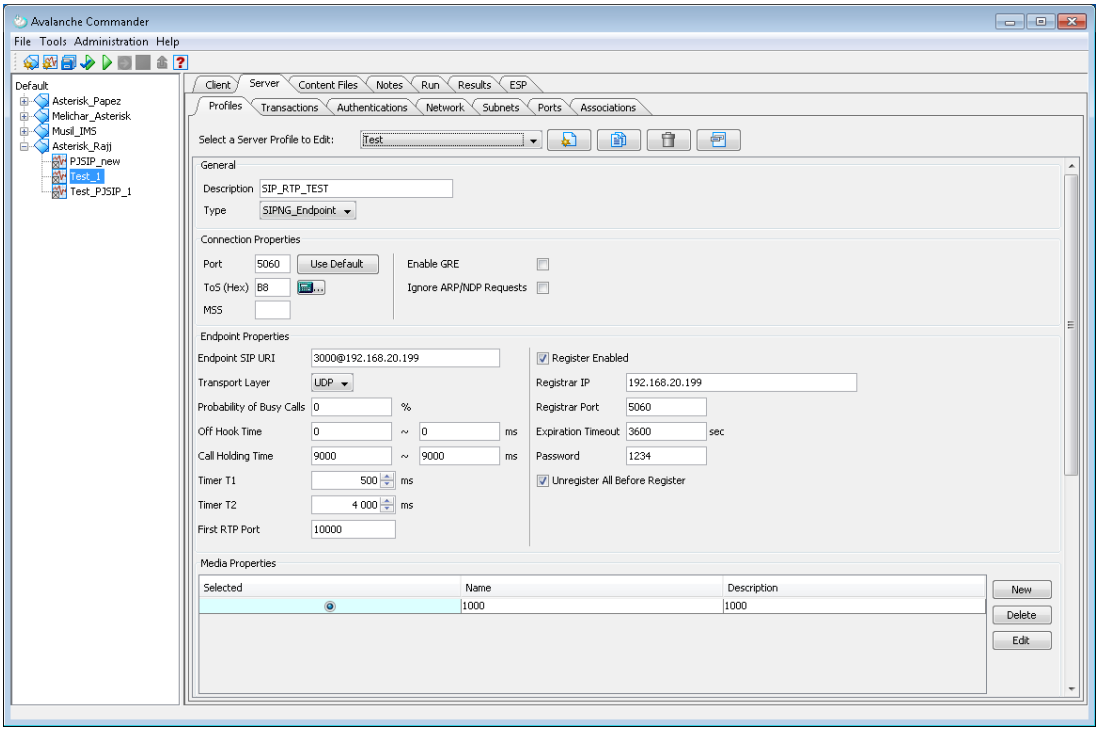
a další. Karta Subnet dovoluje kromě nastavení podsítí také nastavit statické směrování, fragmentaci IP paketů, DHCP, VLAN a realistické chování linky. V kartě Ports došlo pouze k nastavení portů, přes které budou odesílaná data.



Obrázek 5.6 Karta nastavení asociace

Poslední kartou v nastavení klientské části je karta asociace. Obrázek 5.6 obsahuje náhled karty asociace. V této kartě jsou spojena všechna předchozí nastavení z ostatních karet do jednoho celku a tvoří tak výsledný testovací scénář. Z jednotlivých karet jsou navolena různá nastavení. Příkladem může být vytvořený průběh testu z obrázku nahoře, uložený pod názvem Test. V kartě Association je pak vybrán odpovídající průběh dle jeho názvu. Takto lze jednoduše upravovat testovací scénář nebo jen jeho části bez nutnosti vytváření celého scénáře znovu.

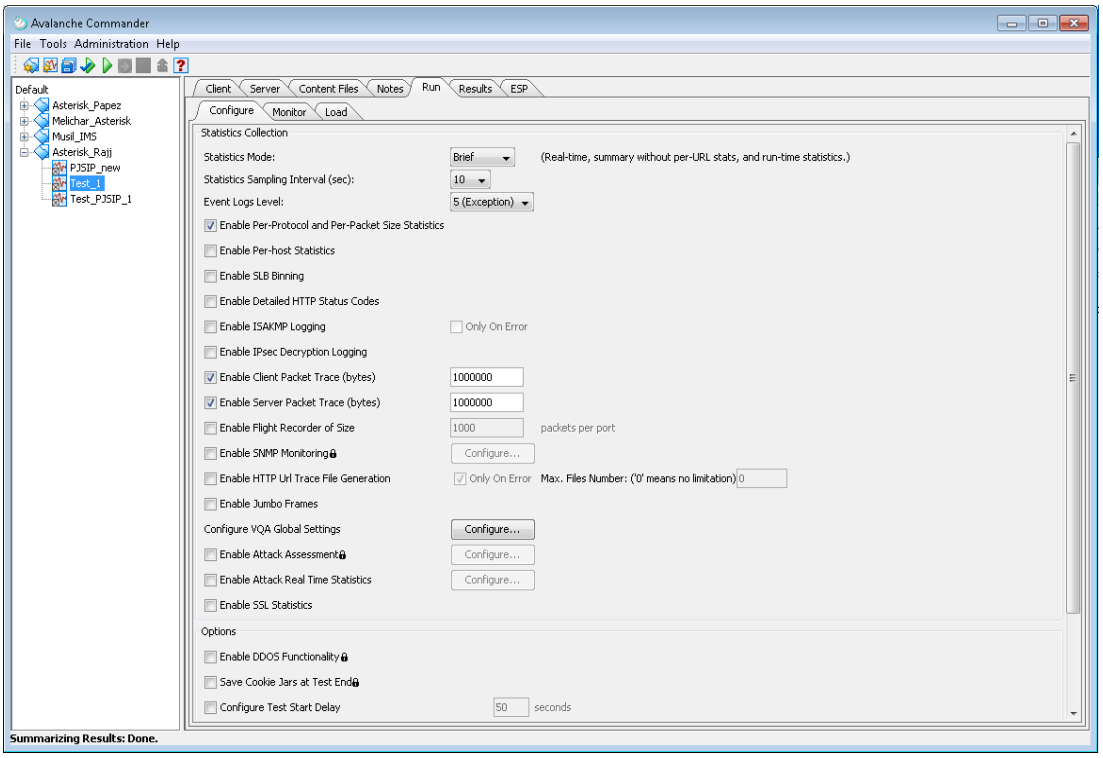
Na straně serveru, která se v aplikaci SIPNG nastavuje, aby simulovala volanou protistranu a odpovídala na SIPNG žádosti generované virtualizovanými uživateli, se změny parametrů provádí v kartách profily (Profiles), transakce (Transactions), autentizace (Authentication), sít (Network), podsíť (Subnet), porty (Ports) a asociace (Association).



Obrázek 5.7 Karta nastavení profilu na straně serveru

Karta profilů je první z karet k nastavení v serverové části. Obrázek 5.7 zachycuje jednotlivá nastavení v kartě profilu. Nejdůležitějším parametrem je typ (Type). Od výběru tohoto parametru se následně odvíjí obsah dalších nastavení. Parametr nastavuje, jak se bude chovat protistrana. Pro SIPNG jsou dostupné dvě možnosti: SIPNG_Endpoint a SIPNG_Proxy a použita byla možnost simulace koncového bodu (SIPNG_Endpoint). Z dalších parametrů jsou zde obsaženy všechny protokoly, jejichž chování dokáže Avalanche simulovat, a k nim příslušná nastavení. Příkladem je DHCP, DNS, FTP, HTPP, POP3, SMTP a další. Mezi nastavení výše zmiňovaného typu SIPNG_Endpoint patří nastavení portu a nastavení URI koncového bodu. Dále se nastavuje pravděpodobnost, že bude volaná strana obsazená, poté doba (v určitém časovém rozsahu) než bude hovor vyzvednut a doba (v časovém rozsahu) ukončení hovoru z volané strany. Tyto tři parametry mají za úkol simulovat reálné chování VoIP datového provozu. Dále se nastavují časovače T1, T2 a podobně jako na klientské straně, tak i zde se nastavují parametry k registraci v registrar serveru. Obdobně jako na straně klienta, tak i zde se nastavují vlastnosti médií. Kromě tří možností nastavení, které zůstaly shodné s klientskou částí, přibyla ještě možnost zrcadlení médií. Takto bude serverová část vkládat média z přijatých RTP paketů do svých RTP paketů a odesílat je zpět.

Karty nastavení transakcí, autentizace, sítě a podsítě zůstaly v rámci diplomové práce v defaultních hodnotách. V kartě Transactions se nastavují parametry pro HTTP a HTTPS server. Karta Authentication se používá v případě, že je potřeba provést autentizaci se serverem. Karta Network a karta Subnet obsahuje podobné nastavení jako na klientské straně. V kartě Network lze nastavovat TCP parametry sítě a v kartě Subnet lze opět měnit nastavení podsítí, statické směrování, fragmentace IP paketů a nastavení VLAN. Na kartě Ports se opět nastavují porty, ze kterých se budou odesílat data. Poslední kartou je karta Association a ta zastává stejnou funkci jako na serverové straně. Seskupuje jednotlivá nastavení z ostatních karet a vytváří celkový testovací scénář.



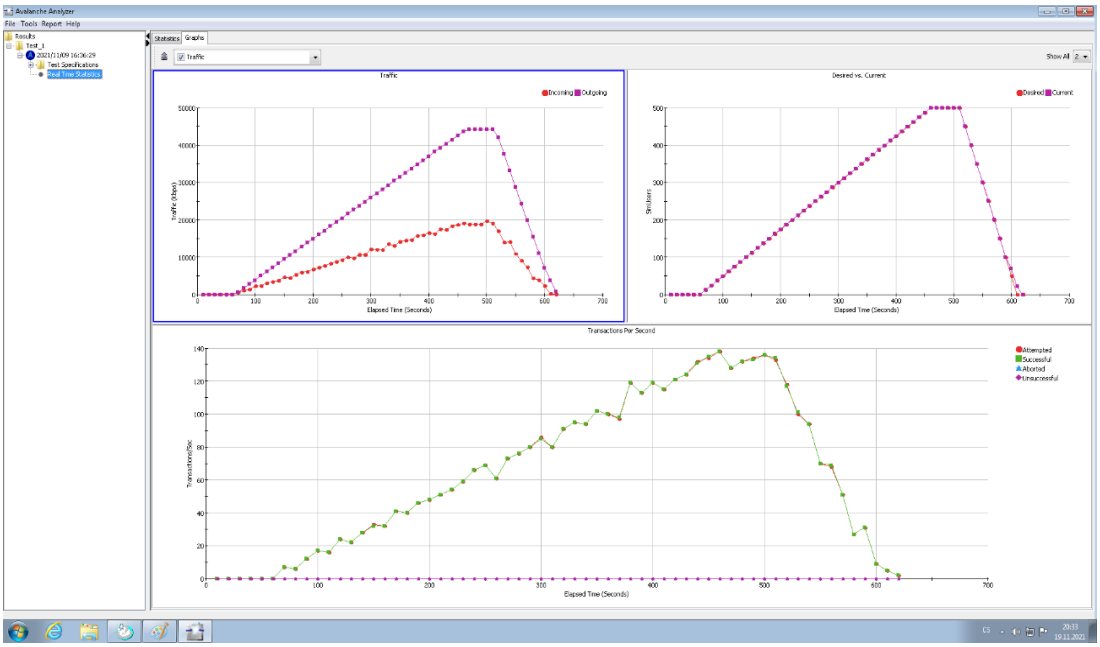
Obrázek 5.8 Konfigurace testeru

Obrázek 5.8 obsahuje nastavení karty Run. Samotný testovací scénář lze ještě dále ovlivnit nastavením testeru. To se provádí v kartě Run a Configure. Karta slouží k nastavení statistik, které budou během testu sesbírány. Toto nastavení dokáže přímo ovlivnit i výkon testeru. Proto je vhodné nastavit menší množství zaznamenávaných statistik a kratší vzorkovací interval, aby nedošlo právě k negativnímu ovlivnění výsledků.

V kapitole byl uveden zkrácený postup a popis některých položek a parametrů, jaký byl použit ke konfiguraci testovacích scénářů pro testování ústředen. Podrobnější návod se všemi možnostmi konfigurace je možné získat přímo ze stránek výrobce [31].

5.3 Spirent TestCenter Layer 4–7 Results Analyzer

Po dokončení výsledků vygeneruje tester dle konfigurace několik výsledků. Jedním z nich je soubor realtime.csv. V tomto souboru je zaznamenáno, jakých hodnot bylo dosaženo v daném čase dle vzorkovacího intervalu testu. Spirent TestCenter Layer 4–7 Results Analyzer, zkráceně také Avalanche Analyzer, je program, který slouží k zobrazení výsledků a práci s nimi.

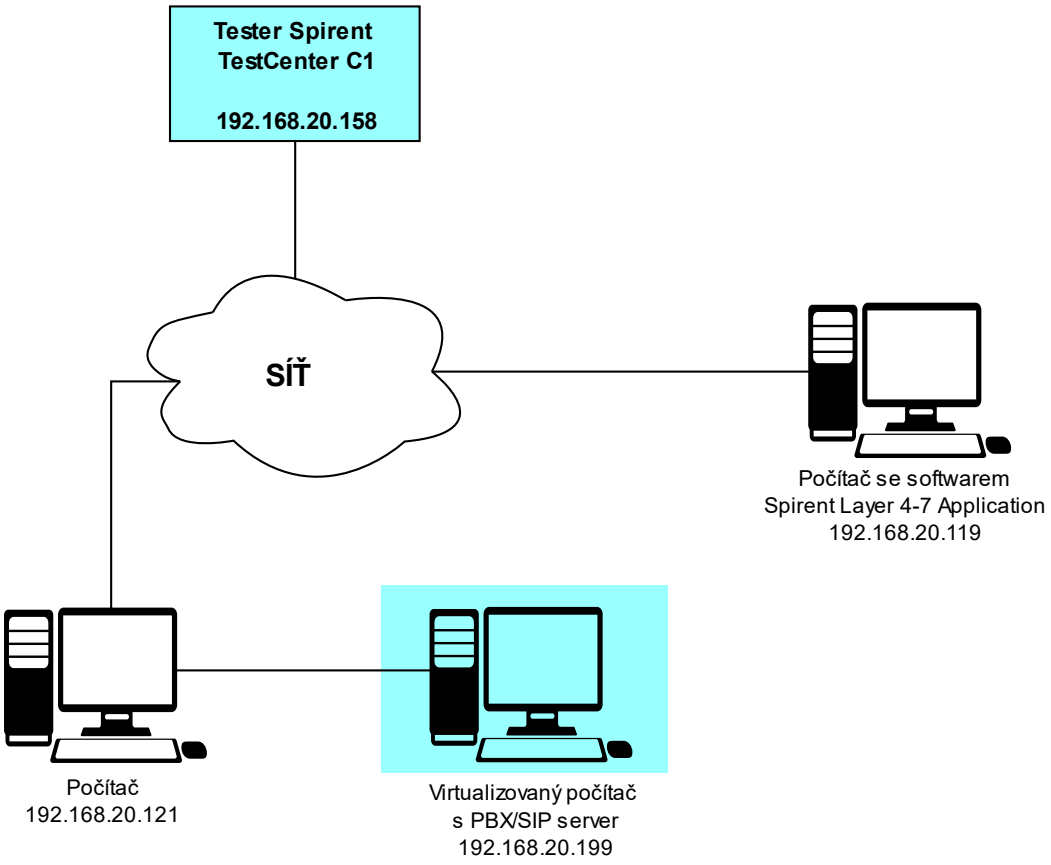


Obrázek 5.9 Výsledky získané ze souboru realtime.csv

Obrázek 5.9 obsahuje rozhraní programu Avalanche Analyzer s již načteným souborem s výsledky realtime.csv. Umožňuje zobrazit specifikace proběhlého testu, jako jsou jméno testu, použitý protokol, velikost zátěže ve všech krocích, ale hlavně dokáže zobrazit libovolné časové průběhy testu dle preferencí uživatele. Zvolené grafy umožňuje exportovat do několika různých formátů.

5.4 Experimentální pracoviště VUT

Pro testování bylo využito experimentální pracoviště. Cílem testování je zjistit rozdíl ve výkonnosti nativního stacku SIP, stacku PJSIP v konfiguraci B2BUA a proxy u pobočkových ústředen Asterisk a porovnat hodnoty s proxy serverem Kamailio.



Obrázek 5.10 Schéma experimentálního pracoviště

Testování bylo prováděno na experimentálním pracovišti VUT. Na virtuálním počítači je spuštěna pobočková ústředna. Na počítači s testovacím softwarem jsou vytvořeny testovací scénáře pro jednotlivé testy. Tester generuje datový provoz. Nejprve se odešlou žádosti REGISTER pro registraci účtů v testeru. Následně jsou v závislosti na testovaném scénáři realizovány telefonní hovory. Obrázek 5.10 představuje zjednodušené schéma experimentálního pracoviště VUT.

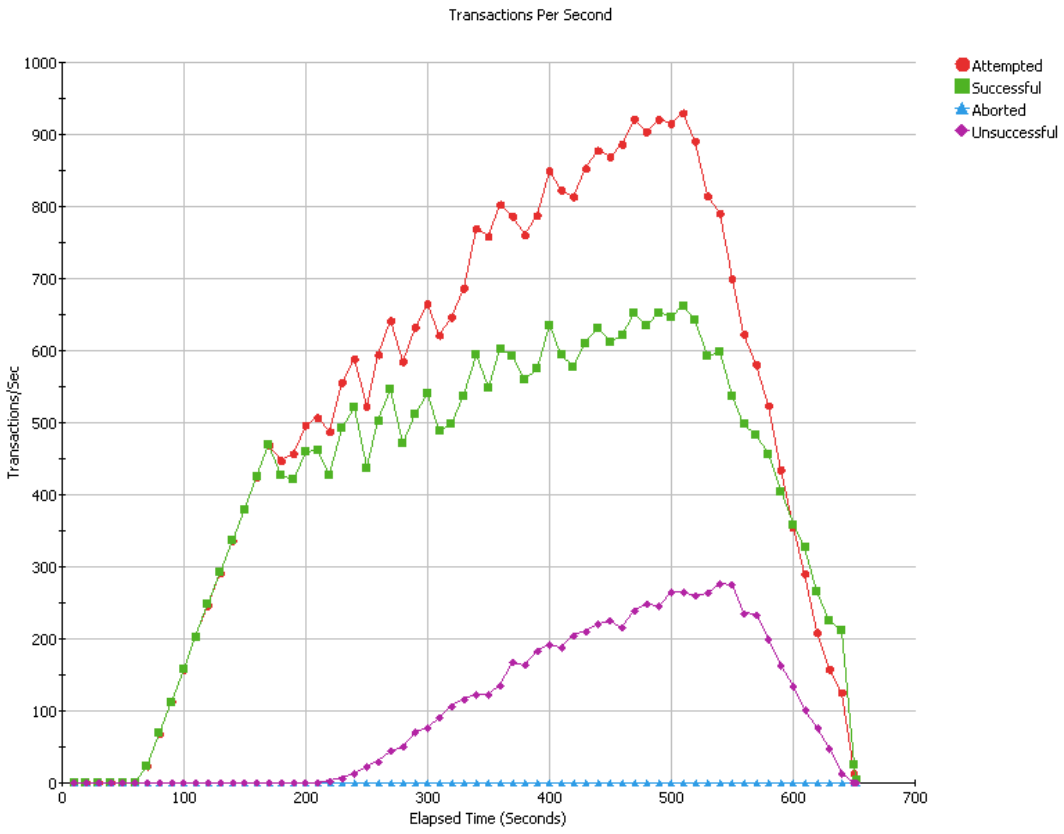
5.5 Metodika testování

Testování bylo prováděno za účelem zjišťování, kolik hovorů a transakcí dokážou jednotlivé ústředny na dané konfiguraci realizovat, dokud nedojde k zahazování

dalších transakcí a hovorů, které již ústředna nebude moct obsloužit z důvodu dosažení limitů dané ústředny nebo vyčerpání zdrojů. Testováno bylo chování Asterisku v režimu B2BUA, což je jeho defaultní režim zacházení s hovory a dále chování v režimu proxy, což je jediný způsob zacházení s hovory v Kamailiu. V případě Asterisku lze tohoto chování dosáhnout přidáním parametru do konfiguračního souboru. U Asterisku byly tyto testy prováděny pro stack SIP i PJSIP. Pro výše popisované testování byly připraveny dva testovací scénáře. Ty byly nadefinovány tak, aby po prvních pár sekundách od spuštění začaly realizovat hovory. Navyšování hovorů trvá po stanovenou dobu, poté již nejsou realizovány nové hovory a jejich hodnota je několik sekund udržována na konstantní úrovni. V poslední etapě dochází už jen k ukončování hovorů. Těmto hovorům byl nastaven kodek, zvuková stopa, která se má přehrávat, ale hlavně délka hovorů. Rozdíl mezi testovacím scénářem na maximální dosažený počet transakcí a hovorů byl v tom, že se oba scénáře liší právě délkou hovorů a způsobem vytváření nových hovorů. První testovací scénář se zaměřuje na množství transakcí, které zvládne ústředna uskutečnit. Délka hovorů je zde nastavena na nejkratší možný čas, aby byly hovory pouze zaznamenány, ale celkové měření nebylo ovlivněno přetrvávajícími hovory. Neukončené hovory, které by ústředna neustále udržovala v paměti, by mohly mít negativní vliv na celkové měření. Druhý scénář se zaměřuje na zjištění maximálního množství hovorů, které je možné realizovat, než začne docházet k zahazování paketů, a tedy k neúspěšným transakcím. Zde byla délka hovorů nastavena tak, aby jednotlivé hovory probíhaly až do konce testu. Toto nastavení bylo podpořeno nastavením serveru, aby hovory neukončil dřív, než je jejich nastavená délka. Tím bylo dosaženo, že hovory nejsou přerušovány ze strany serveru a ke stávajícím hovorům v průběhu testu kumulativně přibývají další hovory. Testování bylo doplněno skripty pro sledování zdrojů virtuálního počítače. Sledováno bylo zatížení procesoru a množství spotřebovávané a dostupné paměti RAM. Veškeré testy byly prováděny na virtuálních počítačích s operačními systémy CentOS a Ubuntu. Na každém operačním systému byly nainstalovány tři poslední verze LTS verze Asterisku a poslední verze Kamailia. U Asterisku byly navíc provedeny testy pro stack SIP i PJSIP a u každého stacku se ještě testovalo chování v režimu B2BUA a proxy. Po dokončení všech testů na stávající konfiguraci byla zvýšena přidělená paměť pro jednotlivé ústředny

z původních 2 GB na 4 GB. Paměť RAM se zdvojnásobila a na nově získaných výsledcích byl zkoumán vliv tohoto zvýšení na počet hovorů a transakcí.

Po dokončení každého testu byly zaznamenány výsledky. Tester dokáže zaznamenat velké množství výsledků. Pro odečítání počtu transakcí a hovorů byly nejdůležitější výsledky ze souboru realtime.csv, které byly vygenerovány testerem. Pro zpracování a analýzu dat byl použit výše popsaný program Spirent Analyzer.



Obrázek 5.11 Výsledky testu na transakce ze souboru realtime.csv

Obrázek 5.11 prezentuje výsledky z testovacího scénáře na počet transakcí převedené do grafu. Červená křivka znázorňuje požadovaný průběh transakcí. Tento průběh odpovídá průběhu, nadefinovaném v profilu zátěže testovacího scénáře. Zelená křivka odpovídá počtu úspěšných transakcí a fialová naopak značí počet neúspěšných transakcí. V ideálním případě bude zelená křivka překrývat červenou, což znamená, že byly provedeny všechny transakce. Pro získání maximálního počtu transakcí musela být zátěž nastavena tak, aby se v průběhu testu začaly vyskytovat

i neúspěšné transakce. Na obrázku nahoře je vidět, jak přestal růst počet úspěšných transakcí za sekundu a začínají narůstat neúspěšné transakce.

Obdobný postup byl použit pro získání hodnoty o maximálním počtu hovorů před tím, než začalo docházet k neúspěšným transakcím. Z časového údaje výskytu prvních neúspěšných transakcí byl následně odvozen počet hovorů, kterého bylo dosaženo, než k těmto transakcím začalo docházet. Neúspěšné transakce nejsou doručeny k druhému účastníkovi hovoru a v případě, že se jedná o nějakou kritickou zprávu či metodu, tak musí být odeslána znovu. To by vedlo k nárůstu zpoždění a z toho důvodu bylo zvoleno odečítání hovorů při výskytu neúspěšných transakcí.

Tabulka 5.1 Přehled nastavení testovacích scénářů

Přehled nastavení testovacích scénářů		
	Test na maximální počet transakcí za sekundu	Test na maximální počet hovorů
	Nastavení klientské části	
Load		
Specification	Simusers/sec	Simusers
Default Time Scale	Seconds	
Steady Time (nulová hodnota)	60 sec.	
Ramp Time (nárůst)	400 sec.	
Height (liší se dle ústředny)	V řádu stovek	V řádu tisíců
Steady Time (na maximu)	50 sec.	
Ramp Time (sestup)	100 sec.	
Steady Time (nulová hodnota)	350 sec.	
Actions		
IP adress	192.168.20.199	
Port	SIP – 5060, PJSIP – 5066	
Caller	{1000–1999}@192.168.20.199	
Callee	{2000–2999}@192.168.20.199	
Media	M_SIP_RTP_TEST	
Transport type	UDP	
Call length min	1	900000 ms
Call length max	1	900000 ms
Registrar IP adress	192.168.20.199	
Registrar Port;	SIP – 5060, PJSIP – 5066	
Password	1234	
Profiles		
Protocol	SIPNG	
First RTP port	10000	

Unregister All Before Register	YES	
Media Properties		
Content	voice_10_sec.wav	
Codecs	G.711a	
Subnet		
Enable DHCP	YES	
Ports		
Port	192.168.10.158:1,1	
	Nastavení serverové části	
Profiles		
Type	SIPNG_Endpoint	
Port	SIP – 5060, PJSIP – 5066	
Endpoint SIP URI	3000@192.168.20.199	
Transport Layer	UDP	
Probability of Busy Calls	0	
Off Hook Time	0	
Call Holding Time	1ms – 1ms	900000 ms – 900000 ms
Registrar IP	192.168.20.199	
Registrar Port	SIP – 5060, PJSIP – 5066	
Password	1234	
Unregister All Before Register	YES	
Media Properties		
Content	voice_10_sec.wav	
Codecs	G.711a	
Ports		
Port	192.168.10.158:1,1	
	Nastavení sběru statistik	
Statistic Mode	Brief	
Statistic Sampling Interval (sec)	10	
Enable Logs Level	5 (Exception)	
Enable Per–Protocol and Per–Packet Size Statistics	YES	
Enable Client Packet Trace (Client)	YES	
Enable Client Packet Trace (Server)	YES	

Tabulka 5.1 obsahuje přehled nastavení, která byla použita pro testovací scénáře.

6. VÝSLEDKY TESTŮ

Realizovány byly testy na zjištění maximálního počtu transakcí a maximálního počtu hovorů, než dojde k neúspěšným transakcím. Testy byly prováděny u tří LTS verzí Asterisku: Asterisk 18 – LTS, Asterisk 16 – LTS a Asterisk 13 – LTS a u SIP serveru Kamailio – Kamailio v5.3. U Asterisku se testovaly oba stacky SIP a PJSIP a také oba režimy chování B2BUA a proxy. Všechny testy byly následně zopakovány na nové konfiguraci virtuálního počítače s dvojnásobkem paměti RAM. Takto vzniklo celkem sto naměřených hodnot, které byly seskupeny dle prováděného testu (transakce nebo hovory) a v případě Asterisku ještě podle použitého stacku a režimu chování.

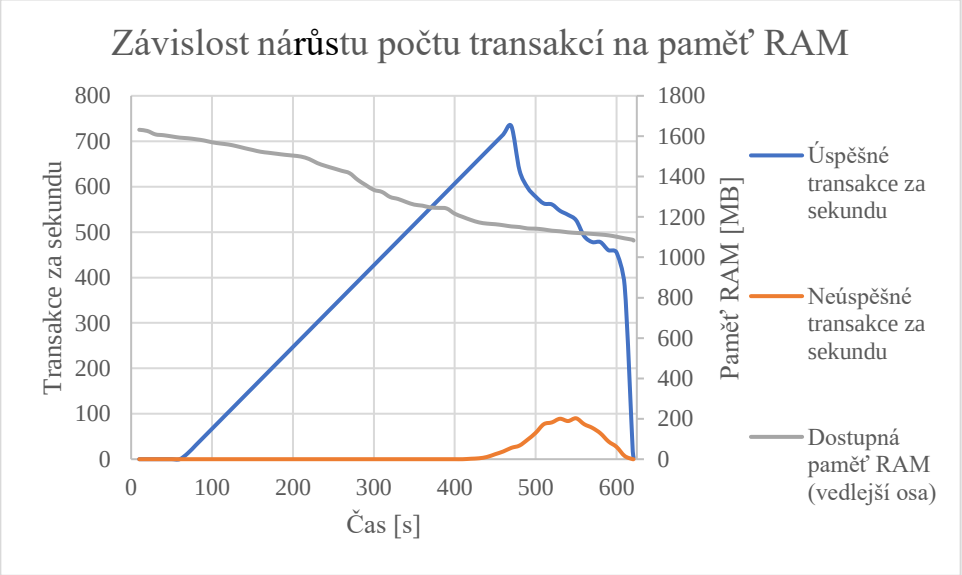
V průběhu testu byly zaznamenávány hodnoty o zatížení CPU a dostupné paměti RAM. Testování bylo cíleno tak, aby jednotlivé ústředny dosáhly svých limitů. Toho bylo ve většině případů dosaženo a u jednotlivých virtuálních počítačů byl zaznamenán nedostatek paměti RAM, nadměrné zatížení CPU, případně obojí.

Obrázek 6.1 Upozornění na nedostatek paměti

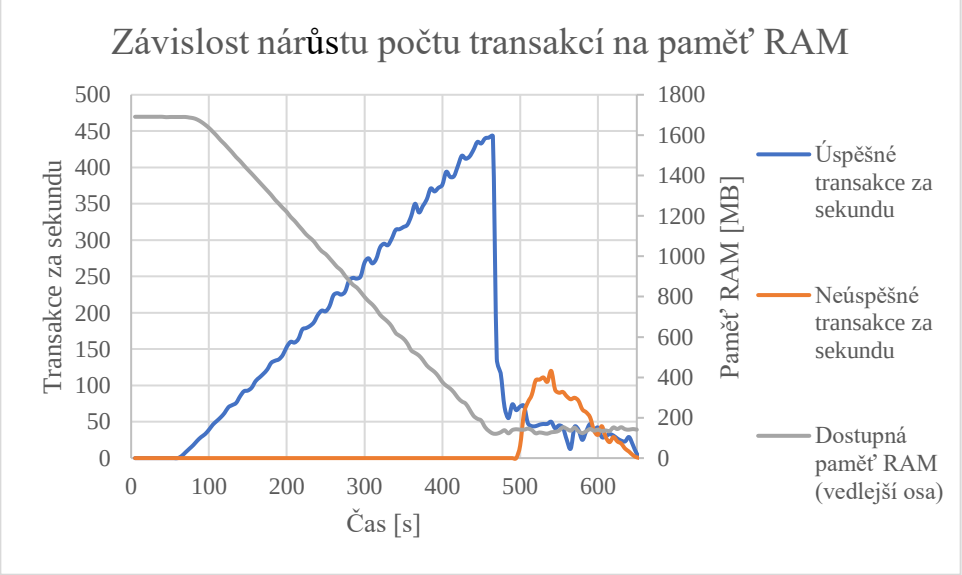
Obrázek 6.1 ukazuje, že došlo k ukončení procesu asterisk z důvodu nedostatku paměti RAM. Toto se stávalo ve velké většině případů právě tehdy, když byla vyčerpána veškerá dostupná paměť virtuálního stroje. Samovolný pád ústředny byl jedním z jevů, které se vyskytovaly v průběhu testování. Druhým jevem, ke kterému docházelo v průběhu testování a po ukončení testu, byla nemožnost práce s ústřednou. Důsledkem byl opět nedostatek paměti, ale i zatížení CPU v průběhu testu. Ústředna nereagovala na zadávané příkazy, nebo na ně reagovala s velkým zpožděním. Tento stav přetrvával i po ukončení testu a ústřednu bylo proto potřeba restartovat pro obnovení běžné funkce.

Vzhledem k odlišnému charakteru jednotlivých testů bylo vypozorováno, že každý z testů měl jiný vliv na zdroje ústředny. U testu na maximální počet hovorů docházelo nejčastěji k nadměrnému vytížení CPU. Tento nárůst byl způsoben tím, že hovory měly nastavenou délku hovoru takovou, aby zůstaly aktivní v průběhu celého hovoru. Kumulativně se k těmto hovorům přidávaly další a další hovory,

až do té doby, než se začaly objevovat neúspěšné transakce a určila se hranice počtu hovorů, které budou obslouženy bez problému. Z toho důvodu musela ústředna udržovat tyto hovory aktivní a mělo to vliv právě na vytížení CPU.



Obrázek 6.2 Závislost počtu transakcí na RAM u ústředny Asterisk (nativní stack SIP)



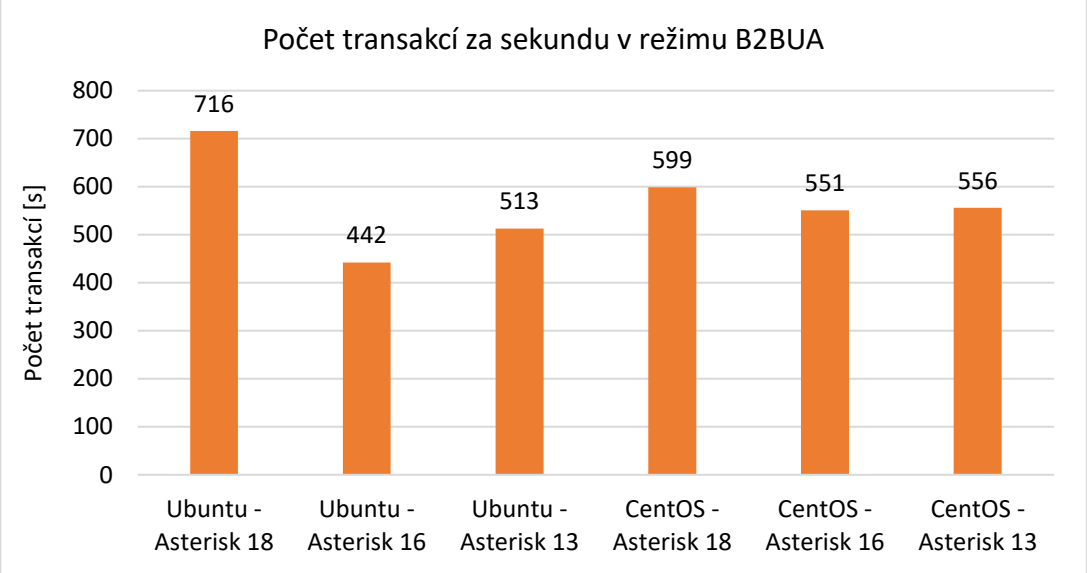
Obrázek 6.3 Závislost počtu transakcí na RAM u ústředny Asterisk (stack PJSIP)

U testu na maximální počet transakcí docházelo k vyčerpání paměti RAM. Obrázek 6.2 obsahuje hodnoty úspěšných a neúspěšných transakcí za sekundu. Dále obsahuje křivku představující dostupnou paměť RAM. Při snížení množství dostupné paměti začaly být některé transakce neúspěšné. Obrázek 6.3 obsahuje stejně jako

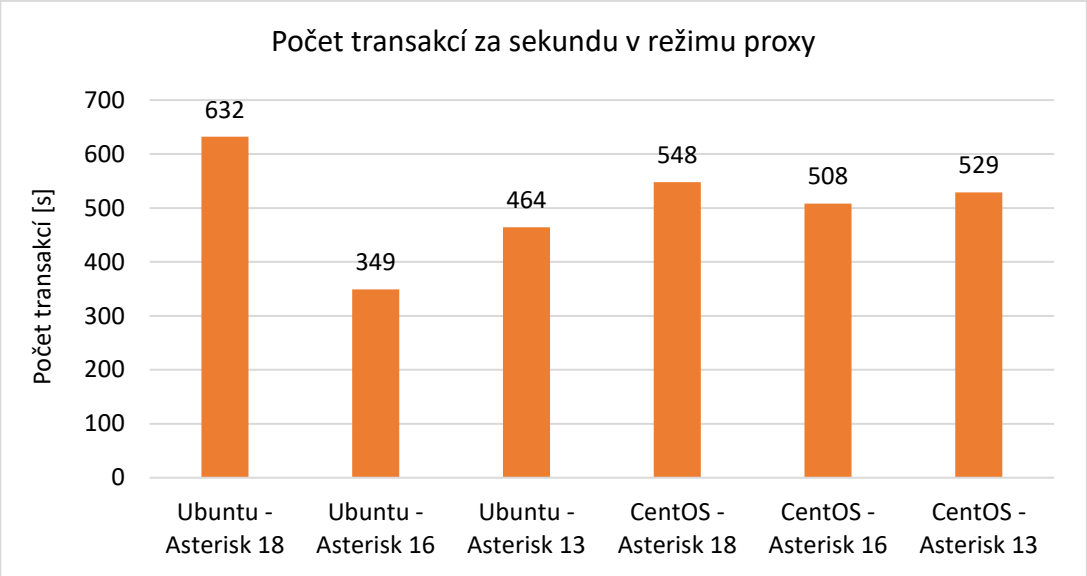
v přechozím případě závislost transakcí na paměti RAM u stacku PSJIP. Zde je vidět, že stack PJSIP je více paměťově náročný než stack SIP, ale oproti stacku SIP využil PJSIP efektivněji dostupnou paměť a neúspěšné transakce vznikaly až při minimu dostupné paměti. Jelikož byla délka hovoru nastavena na co nejkratší možný čas, byly hovory ukončeny ihned po přenesení prvních transakcí. Těchto hovorů bylo větší množství než v případě testu na počet hovorů. Z toho důvodu nebylo příliš zatěžováno CPU, ale velké množství hovorů rychle vyčerpalo paměť RAM.

6.1 Měření 1: Počet transakcí u stacku SIP – 2 GB

První z provedených testů byl test na počet transakcí u stacku SIP. V případě Kamailia se jedná se jediný implementovaný stack, a proto jsou jeho výsledky zahrnuty v této kapitole. Rovněž jsou zde zahrnuty oba režimy chování B2BUA, i proxy režim.



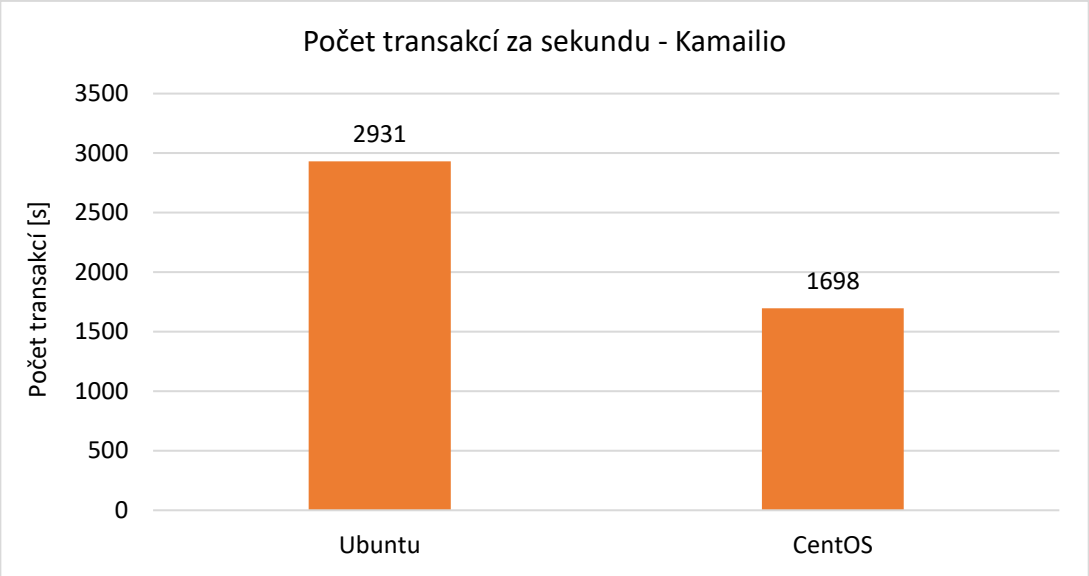
Obrázek 6.4 Počet transakcí za sekundu (SIP/B2BUA/2 GB)



Obrázek 6.5 Počet transakcí za sekundu (SIP/PROXY/2 GB)

Obrázek 6.4 zobrazuje výsledky získané z testu na počet transakcí v režimu B2BUA. Obrázek 6.5 rovněž zobrazuje výsledky na počet transakcí, ale v tomto případě v proxy režimu. V režimu B2BUA dosáhla nejlepších výsledků ústředna s OS Ubuntu a s Asteriskem 18, tedy poslední aktuální LTS verzí Asterisku. Maximální počet transakcí, kterých ústředna dosáhla, je 716 transakcí za sekundu, což byl nejlepší výsledek z této sady testů. Ústředna Asterisk 18 dosáhla nejvyššího množství transakcí i s OS CentOS. Zde bylo dosaženo 599 transakcí za sekundu.

V proxy režimu bylo řádově dosaženo obdobných hodnot jako v režimu B2BUA. Mezi jednotlivými ústřednami jsou menší rozdíly než v případě B2BUA. Opět dosáhla nejlepších výsledků ústředna Asterisk 18 na operačním systému Ubuntu s 632 transakcemi za sekundu. Na virtuálním stroji s operačním systémem CentOS dosáhla nejlepších výsledků ústředna Asterisk 18 s 548 transakcemi za sekundu.

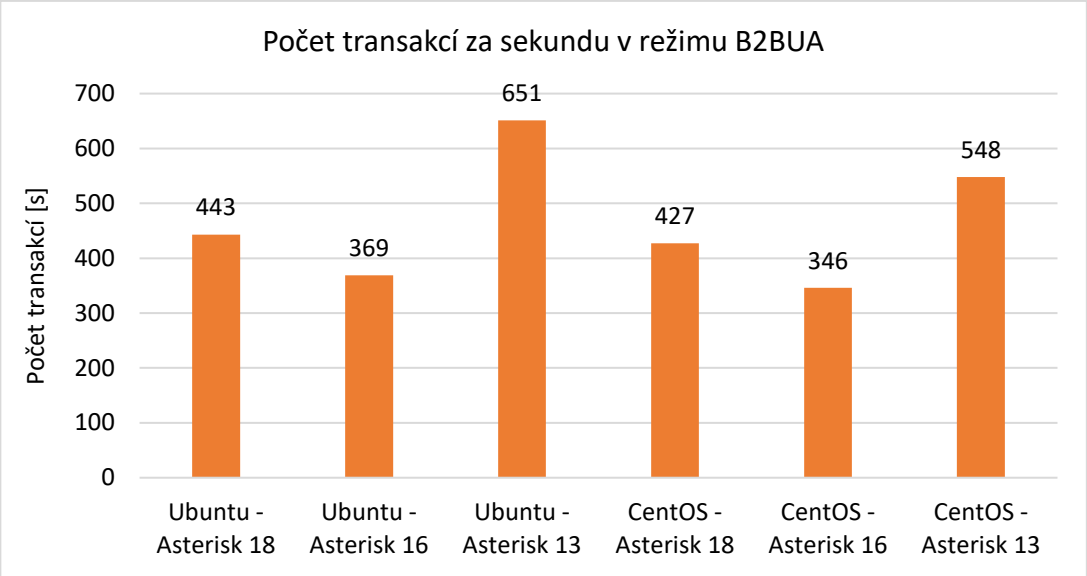


Obrázek 6.6 Počet transakcí za sekundu (Kamailio/2 GB)

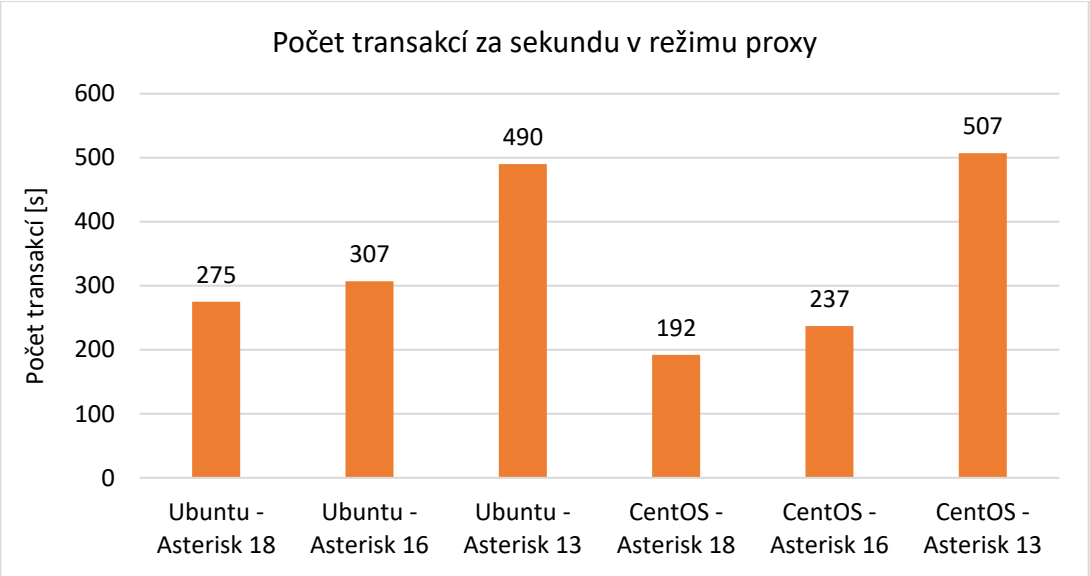
Posledním testem, který spadá do této kapitoly, je test na počet transakcí provedených na proxy serveru Kamailio. Obrázek 6.6 zobrazuje výsledky tohoto měření. Zde je vidět, že počet transakcí za sekundu přesahuje hodnoty získané u ústředen Asterisk. V případě CentOS bylo zaznamenáno 2204 transakcí za sekundu a v případě Ubuntu až 2931. I přesto může být celkový počet úspěšných transakcí za sekundu ještě větší, než ukazuje graf, protože nebyly vyčerpány zdroje virtuálního stroje, tudíž nebylo dosaženo plného potenciálu SIP proxy serveru. Použitý tester nedokázal více zvýšit počet těchto transakcí a výsledná hodnota může být vyšší.

6.2 Měření 2: Počet transakcí u stacku PJSIP – 2 GB

Jako další byl testován stack PJSIP na ústřednách Asterisk. Použit byl shodný testovací scénář jako v případě testu na počet transakcí u nativního stacku SIP. Rozdíl byl jen v použitém portu, na kterém stack naslouchal.



Obrázek 6.7 Počet transakcí za sekundu (PJSIP/B2BUA/2 GB)



Obrázek 6.8 Počet transakcí za sekundu (PJSIP/PROXY/2 GB)

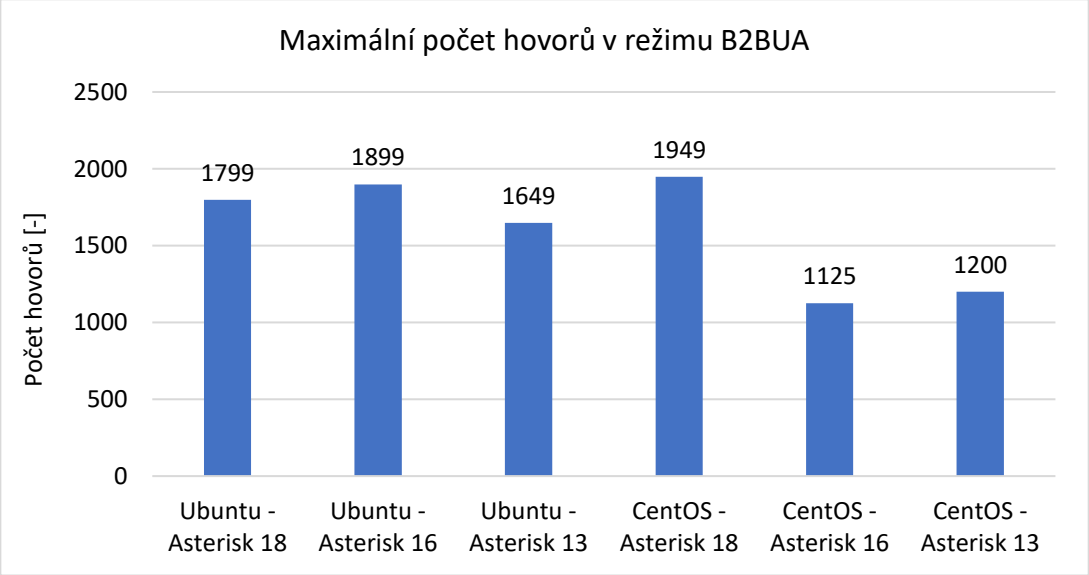
Obrázek 6.7 zobrazuje výsledky testu na maximální počet transakcí u stacku PJSIP v režimu B2BUA. Obrázek 6.8 zobrazuje výsledky počtu transakcí stacku PJSIP v proxy režimu. V režimu B2BUA i v proxy režimu dosáhly nejlepších výsledků starší verze ústředen. V režimu B2BUA dosáhla nejlepších výsledků ústředna Asterisk 13 na virtuálním počítači Ubuntu s počtem 651 úspěšných transakcí za sekundu. Mezi ústřednami nainstalovanými na virtuálním stroji CentOS dosáhla nejlepších výsledků opět ústředna Asterisk 13 s 548 úspěšně přenesenými transakcemi.

U proxy režimu je opět vidět mírně nižší počet úspěšných transakcí, ale výsledné hodnoty se pořád do určité míry podobají hodnotám z režimu B2BUA. I v tomto případě vykazovala lepší výsledky právě starší verze Asterisku. V případě Ubuntu dosáhla ústředna Asterisk 13 490 úspěšných transakcí za sekundu a v případě CentOS 507 úspěšných transakcí za sekundu.

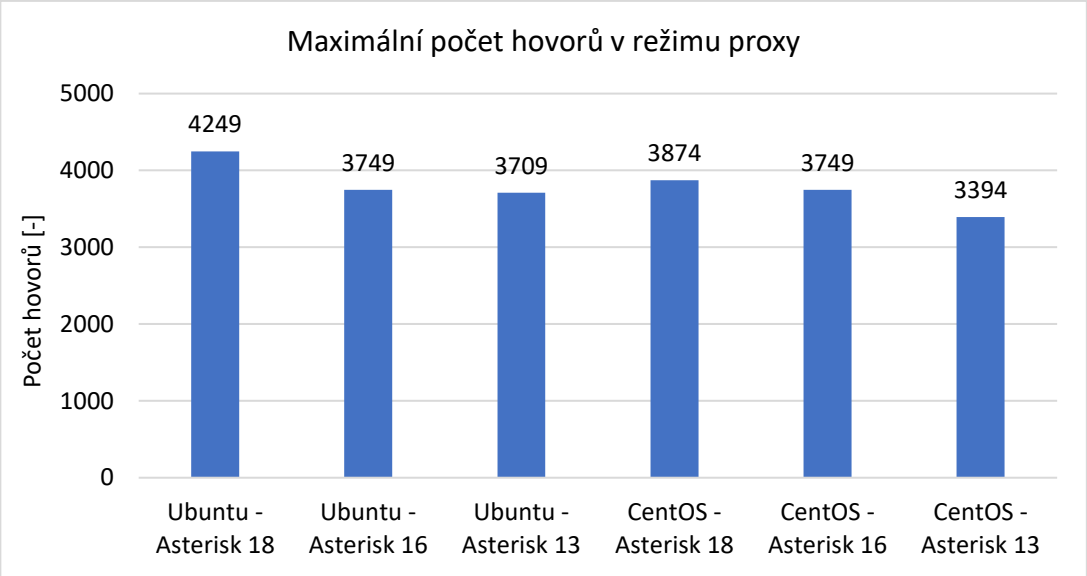
Při porovnání obou stacků SIP i PJSIP dosáhl mnohem lepších výsledků právě SIP. Zde je ovšem důležité podotknout, že tyto výsledky jsou vztaženy na stávající konfiguraci, tedy na paměť 2 GB, což je velmi nízká hodnota a ve valné většině případů bude mít server, na kterém bude nainstalovaná ústředna, mnohem větší paměť. Zároveň se již od 13. verze Asterisku začalo přecházet na novější stack PJSIP a v novějších verzích ústředen Asterisk už je upřednostňován právě stack PJSIP kvůli svým výhodám oproti SIP.

6.3 Měření 3: Počet hovorů u stacku SIP – 2 GB

Jako další byl testován počet hovorů, které ústředna zvládne realizovat, než začne docházet ke ztrátě transakcí. I v tomto případě byly do grafu sloučeny naměřené hodnoty jednotlivých ústředen a rozděleny dle použitého režimu přenosu B2BUA nebo proxy režimu.



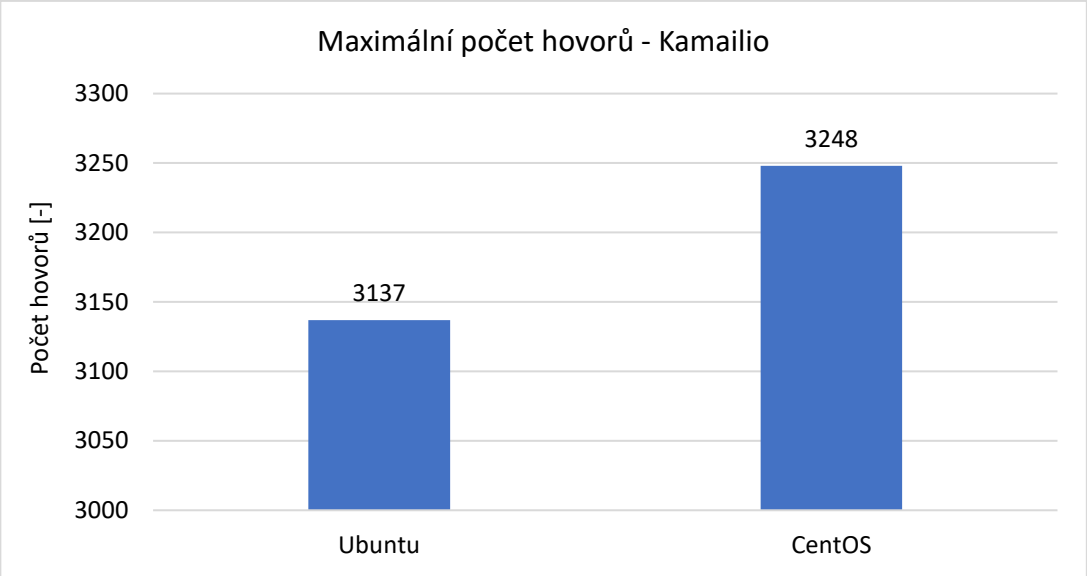
Obrázek 6.9 Počet hovorů (SIP/B2BUA/2 GB)



Obrázek 6.10 Počet hovorů (SIP/PROXY/2 GB)

Obrázek 6.9 reprezentuje výsledky počtu hovorů, než došlo k výskytu neúspěšných transakcí v režimu B2BUA. Obrázek 6.10 reprezentuje shodné výsledky, ale v proxy režimu. V režimu B2BUA dosahovaly ústředny maximálního počtu hovorů v řádu tisíců. Za Ubuntu měla nejlepší výsledky ústředna Asterisk 18 s 1899 realizovanými hovory, než došlo k výskytu neúspěšných transakcí a za CentOS dosáhla nejlepších výsledků ústředna Asterisk 18 s počtem 1949 dosažených hovorů.

V porovnání s režimem B2BUA došlo v režimu proxy k zvýšení počtu hovorů. To je způsobeno tím, že značná část komunikace probíhá mezi koncovými zařízeními. V tomto režimu dosáhla u virtuálního stroje Ubuntu nejlepších výsledků ústředna Asterisk 18 s 4249 realizovanými hovory a u virtuálního stroje CentOS opět ústředna Asterisk 18 s 3874 hovory.

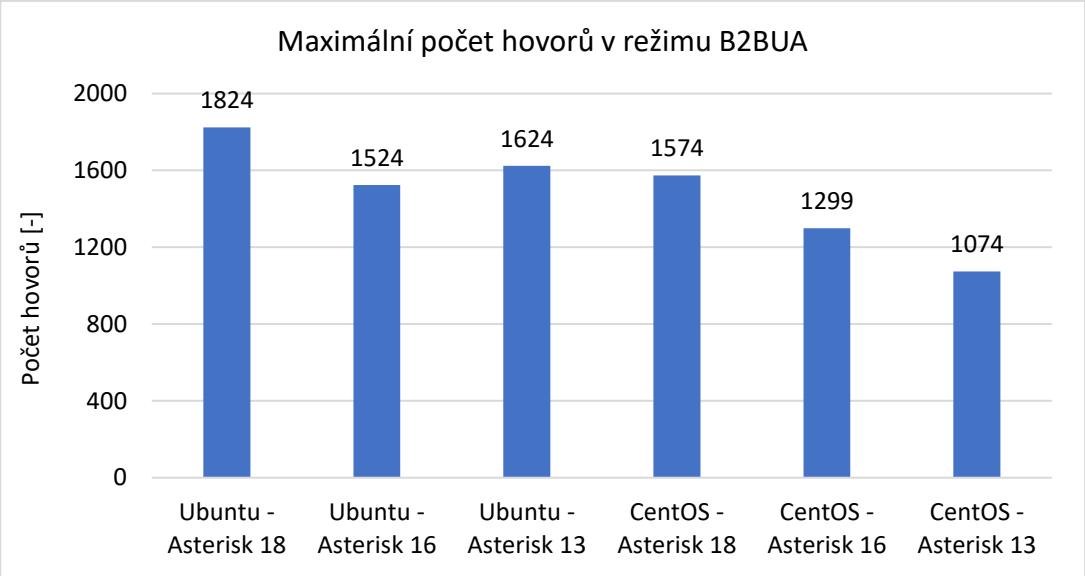


Obrázek 6.11 Maximální počet hovorů (Kamailio/2 GB)

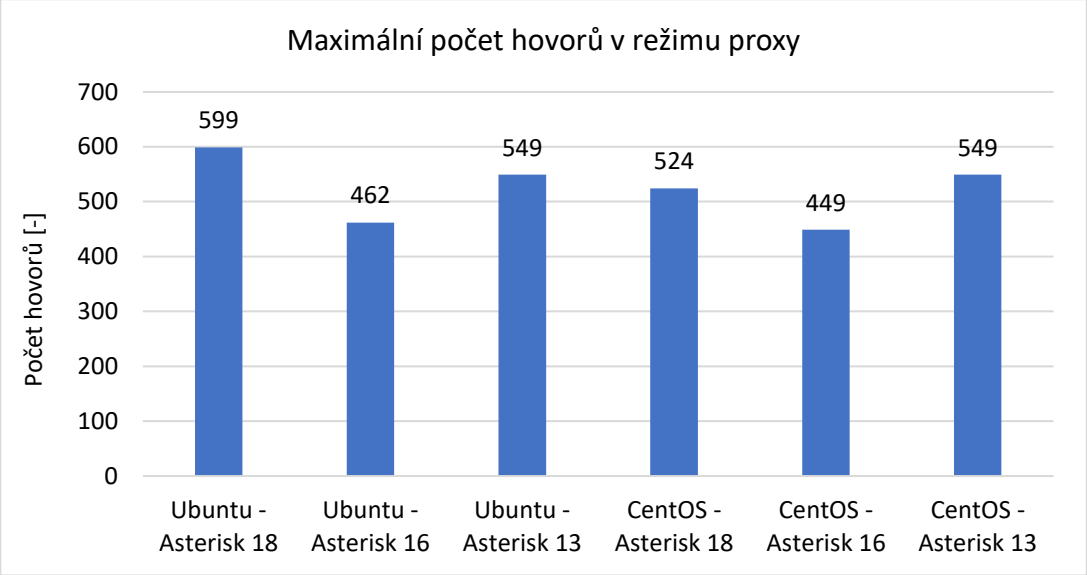
Obrázek 6.11 zobrazuje výsledky maximálního počtu hovorů u SIP proxy serveru Kamailio. Na virtuálním stroji s OS Ubuntu bylo dosaženo 3137 hovorů a virtuální stroj s CentOS dosáhl 3248 hovorů. Zde je nutno dodat, že při těchto hodnotách se nevyskytovaly neúspěšné transakce a Kamailio mělo dostatek zdrojů. Při dalších pokusech o navýšení počtu hovorů se předpokládané množství hovorů zvyšovalo, ale reálně proběhl jen takový počet hovorů, který byl zaznamenán na obrázku nahoře.

6.4 Měření 4: Počet hovorů u stacku PJSIP – 2 GB

Posledním testem se stávající konfigurací byl test na počet hovorů, které ústředna zvládne realizovat, než začne docházet ke ztrátě transakcí. V ústředně bylo opět nastaveno naslouchání na jiném portu, než tomu bylo u stacku SIP.



Obrázek 6.12 Počet hovorů (PJSIP/B2BUA/2 GB)



Obrázek 6.13 Počet hovorů (PJSIP/PROXY/2 GB)

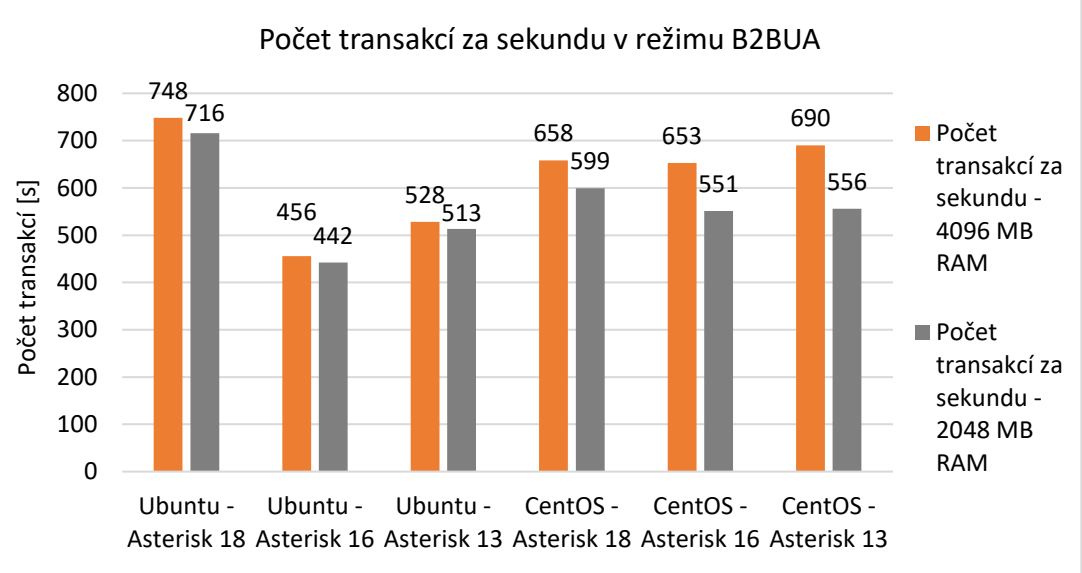
Obrázek 6.12 znázorňuje výsledky testů na počet transakcí u stacku PJSIP. Obrázek 6.13 znázorňuje výsledky testu na počet hovorů u stacku PJSIP v proxy režimu. V režimu B2BUA bylo u virtuálního stroje s Ubuntu dosaženo na ústředně Asterisk 18 maxima 1824 hovorů a virtuální stroj s CentOS dosáhl maxima rovněž s ústřednou Asterisk 18 s 1574 hovory. V režimu proxy je vidět značné zhoršení v počtu dosažených hovorů a to ať ve srovnání s režimem B2BUA, tak i s předešlými výsledky u stacku SIP. Nejvyšší dosažené hodnoty u stroje s Ubuntu dosáhla ústředna

Asterisk 18 s pouhými 599 hovory. U virtuálního stroje s CentOS bylo nejlepšího výsledku dosaženo u ústředny Asterisk 13 s výsledkem 549 hovorů.

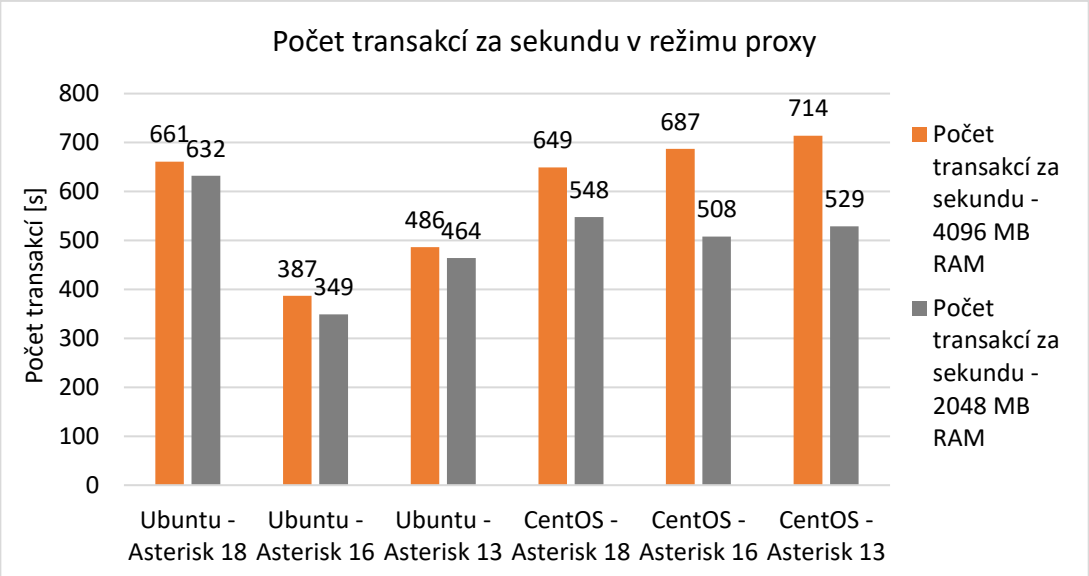
S přihlédnutím k získaným výsledkům na stávající konfiguraci dosáhl stack SIP lepších výsledků než stack PJSIP. Z jednotlivých ústředen si nejlépe vedl Asterisk 18.

6.5 Měření 5: Počet transakcí u stacku SIP – 4 GB

Další testování bylo provedeno na konfiguraci s vyšší přidělenou pamětí. Paměť byla zvýšena z 2 GB na 4 GB. Došlo tedy k dvojnásobnému zvýšení paměti oproti původní konfiguraci. Následně byl testován vliv tohoto navýšení na jednotlivé ústředny. Z doposud získaných hodnot vyplývá, že stack SIP by měl zvládnout přenést větší množství transakcí a realizovat větší množství hovorů. S novou konfigurací vyplyne, zda byl tento předpoklad potvrzen či vyvrácen.



Obrázek 6.14 Počet transakcí za sekundu (SIP/B2BUA/4 GB)

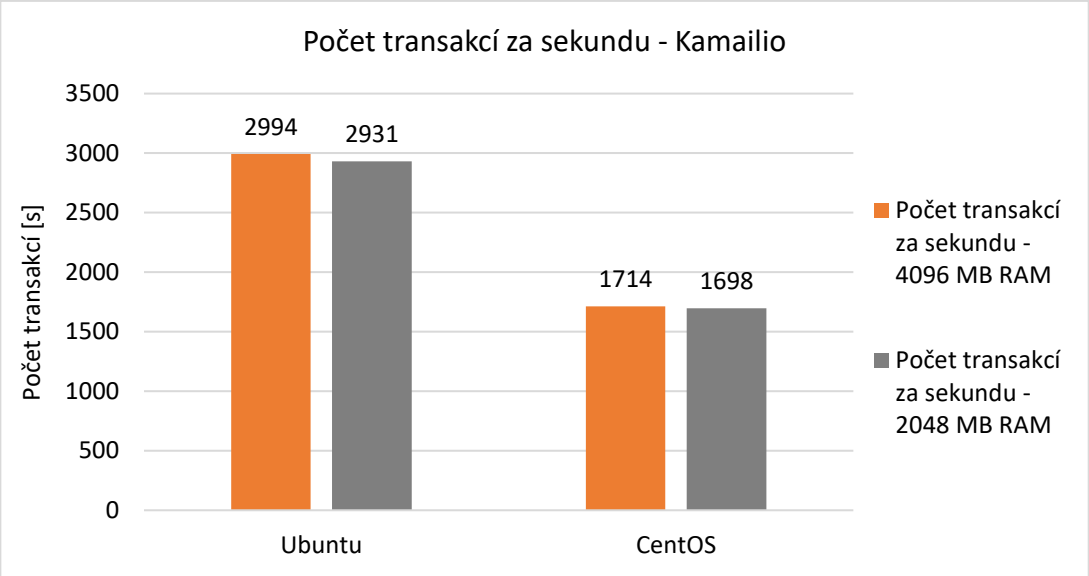


Obrázek 6.15 Počet transakcí za sekundu (SIP/PROXY/4 GB)

Navýšení paměti RAM mělo vliv na počet transakcí. Obrázek 6.14 představuje vliv tohoto zvýšení paměti na počet transakcí. Obrázek 6.15 představuje vliv tohoto zvýšení na transakce v proxy režimu. U režimu B2BUA dosáhla nejlepších výsledků ústředna Asterisk 18 nainstalovaná na Ubuntu. Tato ústředna dosáhla 748 transakcí za sekundu. Ústředna Asterisk 13, nainstalovaná na OS CentOS, dokázala ve stejném režimu realizovat 690 transakcí za sekundu.

V proxy režimu se počet transakcí velmi blížil počtu transakcí v defaultním režimu B2BUA. Ve všech případech nastala změna, která nepřekročila 100 transakcí za sekundu. Virtuální stroj s Ubuntu získal nejlepší výsledek s ústřednou Asterisk 18 a dokázal realizovat 661 transakcí za sekundu. U virtuálního stroje s CentOS měl nejlepší výsledek Asterisk 13 s 714 transakcemi za sekundu.

Dvojnásobné zvětšení paměti nemělo na celkový počet transakcí až tak zásadní vliv. Při srovnání výsledků z kapitoly 7.1 je patrné, že stack SIP pravděpodobně dosáhl na použité konfiguraci svého maxima, protože došlo pouze k nepatrnému nárůstu transakcí, a to pouze v řádu desítek transakcí za ústřednu. Zvětšení paměti pomohlo celkové stabilitě ústředny i virtuálního stroje, ale na počet transakcí to nemělo zásadní vliv.

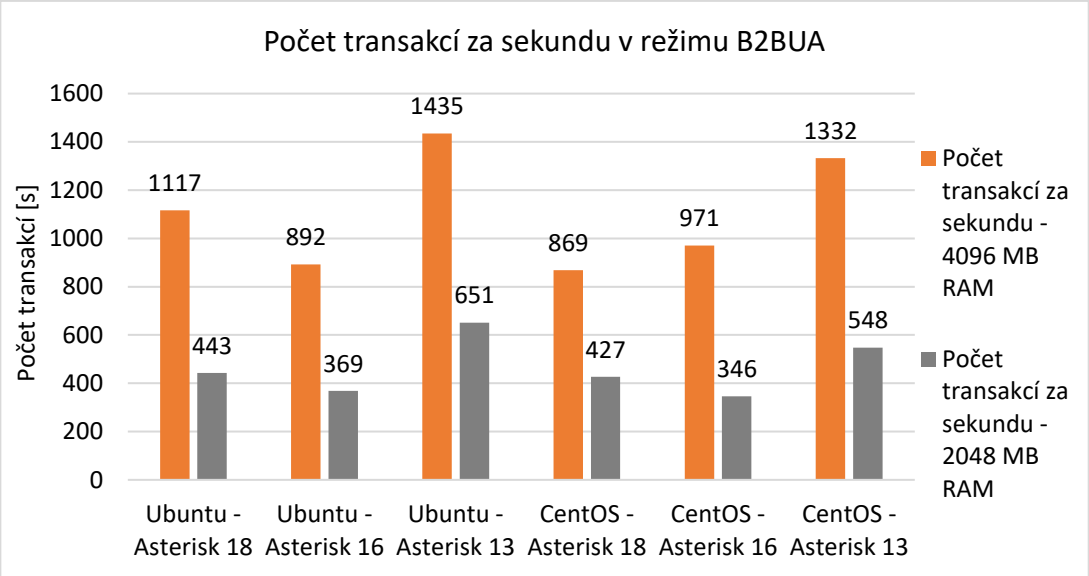


Obrázek 6.16 Počet transakcí za sekundu – (Kamailio/4 GB)

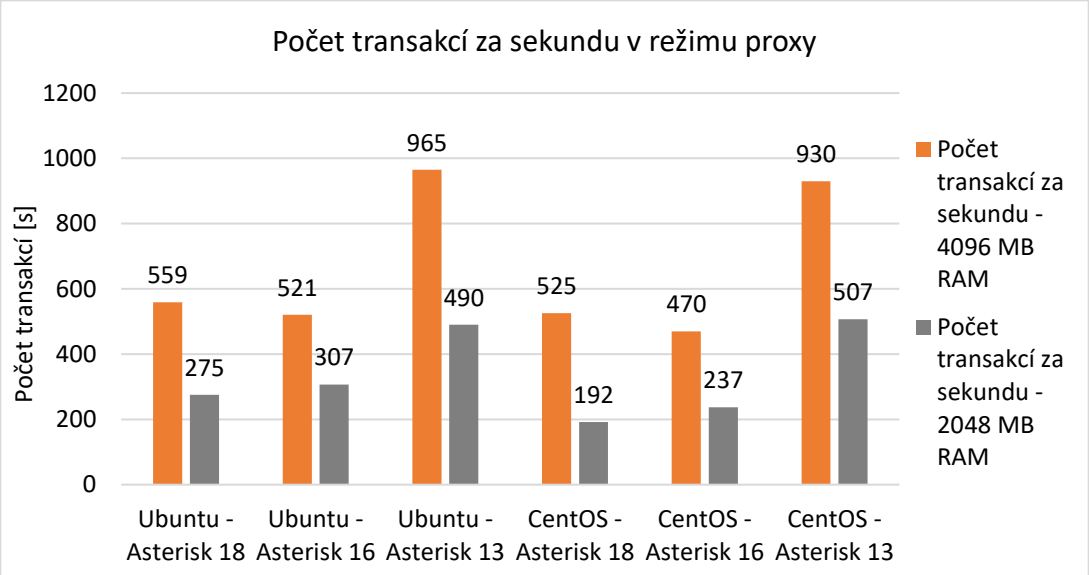
Stejně jako v předchozím případě, tak i zde byl společně se stackem SIP otestován SIP proxy server Kamailio. Obrázek 6.16 prezentuje výsledky měření proxy serveru Kamailio. Nárůst paměti zde rovněž neměl skoro žádný vliv na počet transakcí. To mohlo být způsobeno výše popisovaným faktem, že i přes snahy zvýšení zátěže (navýšení počtu transakcí) se celkový počet dosažených transakcí dále nezvyšoval a s největší pravděpodobností bylo dosaženo limitu testeru, nikoliv však proxy serveru. Kamailio nainstalované na virtuálním stroji s Ubuntu dosáhlo 2994 transakcí za sekundu a na virtuálním stroji CentOS uskutečnilo 1714 transakcí za sekundu.

6.6 Měření 6: Počet transakcí u stacku PJSIP – 4 GB

Jako další byl se zvýšenou přidělenou pamětí otestován stack PJSIP u ústředen Asterisk a byl zkoumán vliv tohoto zvýšení na maximální počet transakcí. Výsledné hodnoty byly porovnány s hodnotami získanými na předešlé konfiguraci.



Obrázek 6.17 Počet transakcí za sekundu (PJSIP/B2BUA/4 GB)



Obrázek 6.18 Počet transakcí za sekundu (PJSIP/PROXY/4 GB)

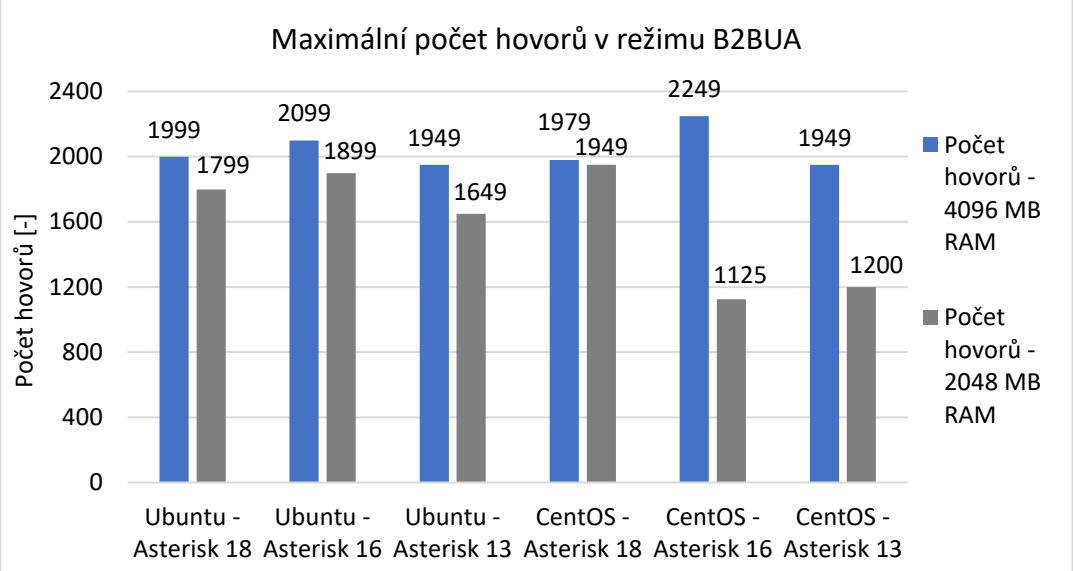
Obrázek 6.17 zobrazuje výsledky testů na počet transakcí s novou konfigurací virtuálních strojů u stacku PJSIP. Obrázek 6.18 zobrazuje výsledky na nové konfiguraci v proxy režimu. Nejlepších výsledků u tohoto stacku dosahovala ústředna Asterisk 13. V režimu B2BUA dokázala tato ústředna na virtuálním stroji s Ubuntu realizovat 1435 transakcí za sekundu. Stejná ústředna, nainstalovaná na virtuálním stroji CentOS v režimu B2BUA, dokázala realizovat 1332 transakcí za sekundu.

V konfiguraci proxy režimu dosáhla nejlepších výsledků ústředna Asterisk 13, stejně jako tomu bylo u B2BUA. Ústředna v kombinaci s Ubuntu dosáhla 965 transakcí za sekundu, což byl nejlepší výsledek ze série měření zanesených do grafu nahoře. V kombinaci s CentOS ústředna přenesla 930 transakcí za sekundu.

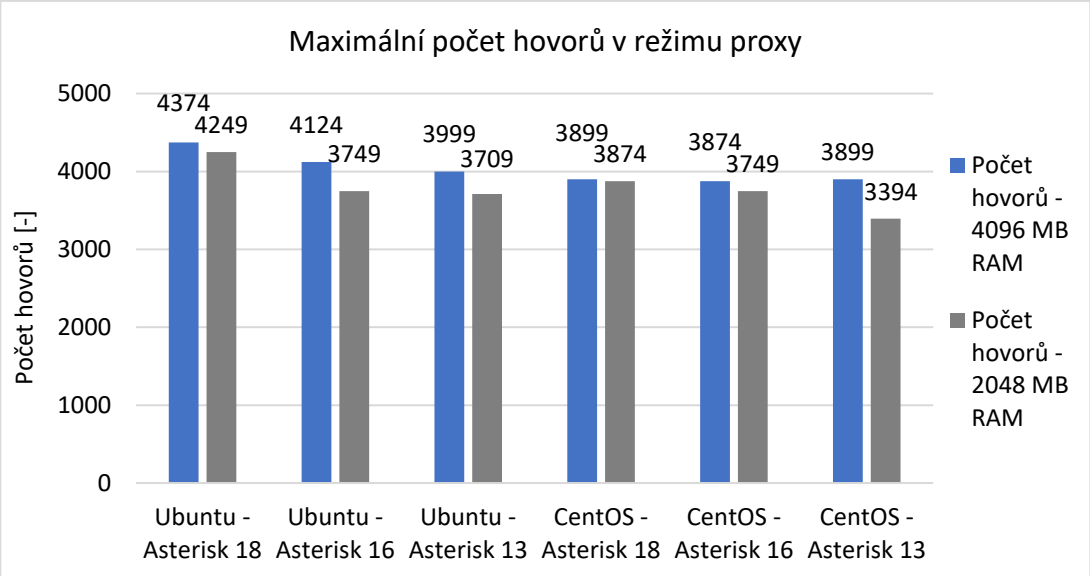
Při srovnání výsledků s předešlou konfigurací došlo téměř ve všech případech k několikanásobnému zvýšení počtu transakcí. Již s pamětí 4 GB přesáhl dosažený počet transakcí výsledky, které byly získány s využitím stacku SIP. Zatímco u stacku SIP počet hovorů spíše stagnoval a došlo k nepatrnému nárůstu hovorů, tak u stacku PJSIP mělo zvýšení paměti razantní dopad na počet transakcí. Plyne z toho, že předpoklad ze začátku kapitoly je mylný. Stack SIP sice dosáhl většího množství transakcí s 2 GB přidělené paměti, ale při jejím zvýšení ho stack PJSIP překonal. Obdobná situace nastává při dalším zvyšování paměti.

6.7 Měření 7: Počet hovorů u stacku SIP – 4 GB

Předposledním provedeným testem byl test na maximální počet transakcí. I v tomto případě byly výsledky zaneseny do grafu a porovnány s předešlou konfigurací.



Obrázek 6.19 Počet hovorů (SIP/B2BUA/4 GB)

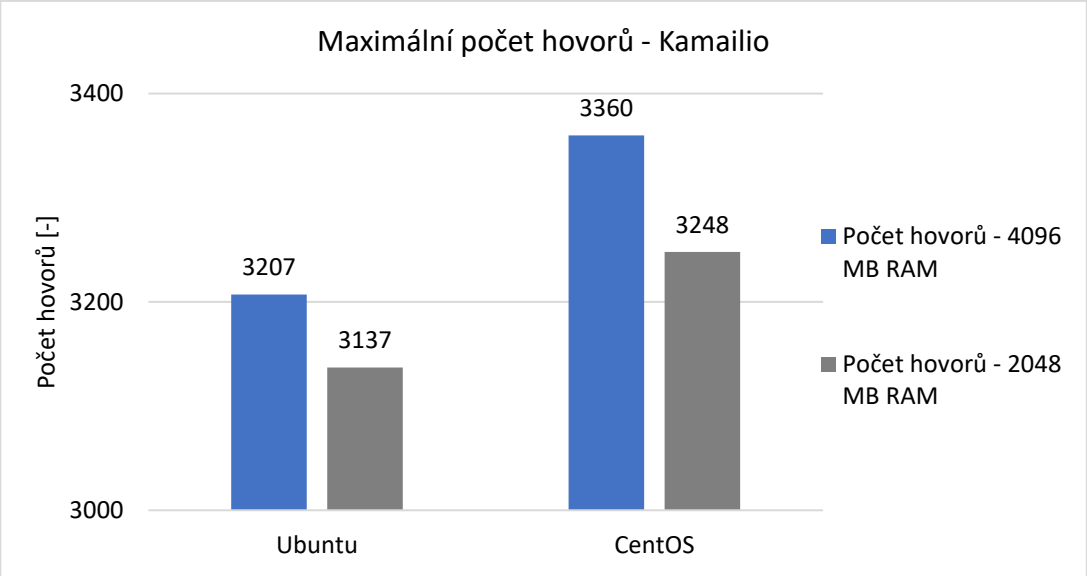


Obrázek 6.20 Počet hovorů (SIP/PROXY/4 GB)

Obrázek 6.19 obsahuje výsledky testů na počet transakcí s novou konfigurací virtuálních strojů pro režim B2BUA. Obrázek 6.20 obsahuje výsledky shodného testu, spuštěného na ústřednách pracujících v proxy režimu. V režimu B2BUA dosáhla s oběma operačními systémy nejlepšího výsledku ústředna Asterisk 16. V kombinaci s Ubuntu dosáhla 2099 hovorů před tím, než se začaly vyskytovat neúspěšné transakce a v kombinaci s CentOS dosahovalo maximum až k 2249 hovorům.

V režimu proxy měla nejlepší výsledky ústředna Asterisk 18. V konfiguraci s virtuálním strojem s OS Ubuntu dosáhla ústředna 4374 hovorů a v konfiguraci s CentOS dosáhla 3899.

Zvýšení paměti mělo vliv na celkový počet hovorů. Ve většině případů došlo k nárůstu v řádu stovek hovorů. Stejně jako v případě testu na transakce, tak i zde je vidět nárůst, který ovšem není tak razantní.

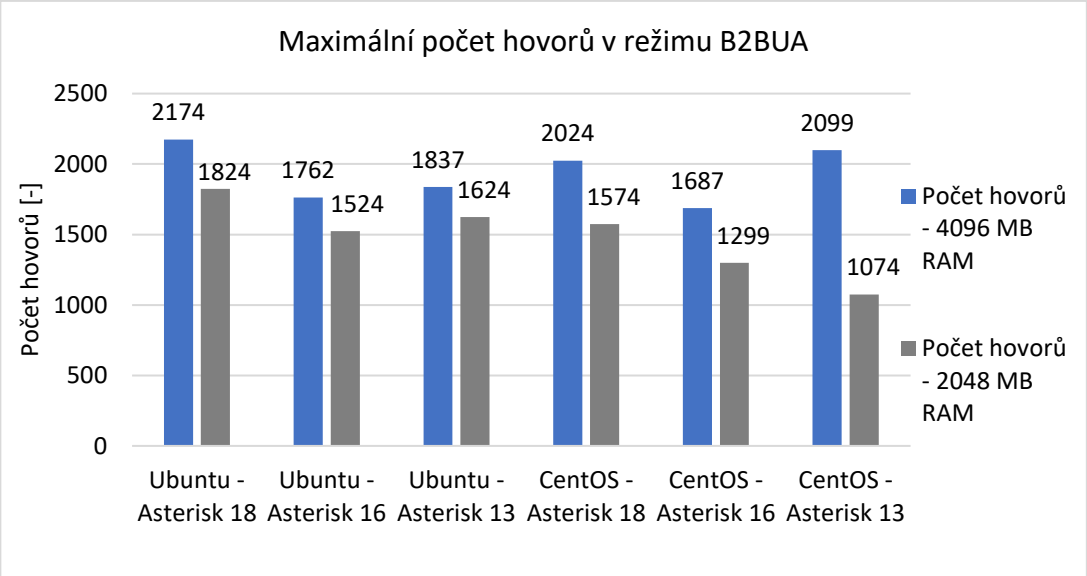


Obrázek 6.21 Maximální počet hovorů (Kamailio/4 GB)

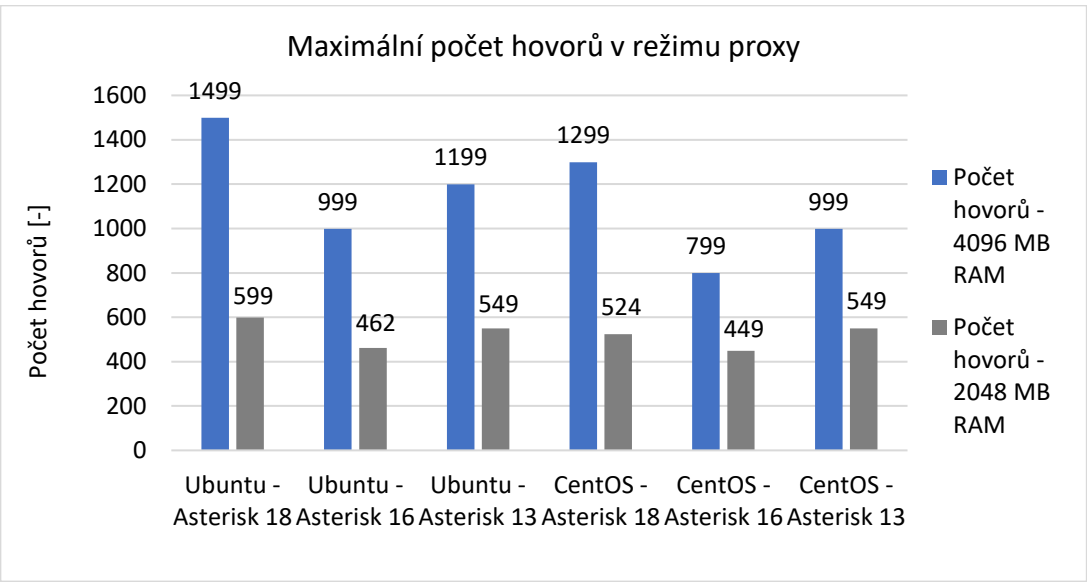
I u Kamailia byla testována závislost zvýšení paměti na počet hovorů. Toto zvýšení mělo nepatrný efekt, ale stejně jako v předešlém případě v konfiguraci s 2 GB, tak i zde při zvyšování zátěže nebylo možné dosáhnout dalšího zvýšení počtu hovorů. Obrázek 6.21 představuje získané hodnoty měření. Kamailio v konfiguraci s Ubuntu realizovalo 3207 hovorů a v konfiguraci s CentOS realizovalo 3360 hovorů.

6.8 Měření 8: Počet hovorů u stacku PJSIP – 4 GB

Poslední test se zvýšenou pamětí byl test na maximální počet hovorů u stacku PJSIP. Hodnoty byly vyneseny do grafu s porovnány s výsledky z kapitoly 6.4. I v tomto případě byl použit režim chování B2BUA a proxy režim.



Obrázek 6.22 Počet hovorů (PJSIP/B2BUA/4 GB)



Obrázek 6.23 Počet hovorů (PJSIP/PROXY/4 GB)

Obrázek 6.22 obsahuje výsledky testů na počet transakcí u stacku PJSIP, v konfiguraci se zvýšenou pamětí v režimech B2BUA. Obrázek 6.23 obsahuje výsledky ze stejného testovacího scénáře, ale pro ústředny v proxy režimu. V režimu B2BUA dosáhla nejlepšího výsledku ústředna Asterisk 18, nainstalovaná na virtuálním počítači s OS Ubuntu, s počtem 2174 souběžně realizovaných hovorů. Ve stejném režimu, ale na virtuálním stroji s OS Centos, měla nejlepší výsledky ústředna Asterisk 13 s 2099 souběžnými hovory.

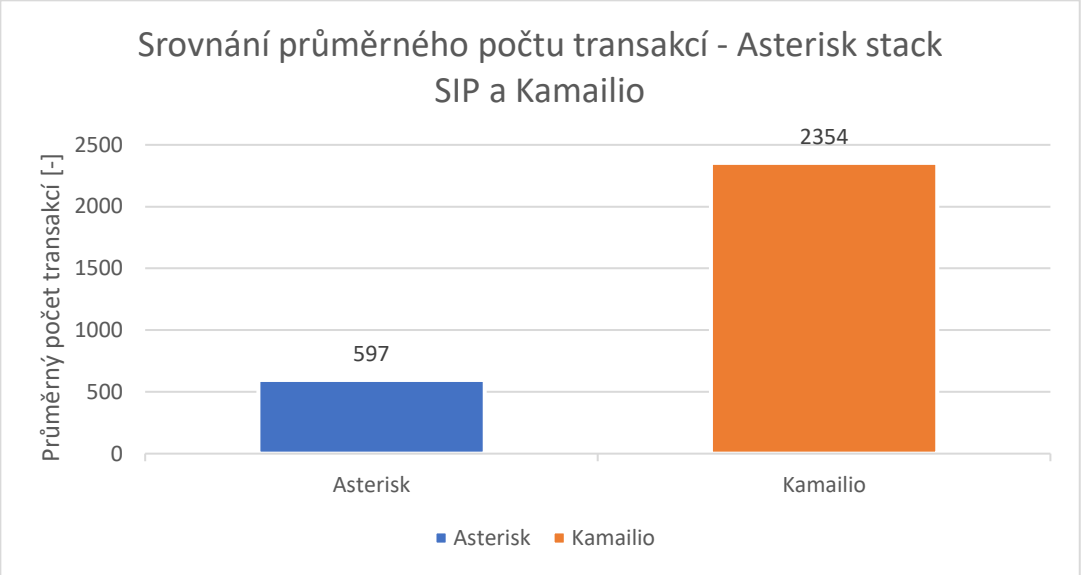
Pro proxy režim měla v konfiguraci virtuálního stroje s OS Ubuntu nejlepší výsledky ústředna Asterisk 18 s maximálním počtem 1499 hovorů a s OS CentOS zase ústředna Asterisk 18 s 1299 souběžnými hovory.

Při srovnání se stejnými výsledky pro stack SIP je vidět, že i zde mělo zvýšení paměti vliv na počet hovorů. V režimu B2BUA došlo k nárůstu v řádu stovek hovorů, čímž bylo dosaženo většího nárůstu než v případě nativního stacku SIP. I v proxy režimu došlo k mnohem většímu nárůstu, než tomu bylo u stacku SIP na shodné konfiguraci. Obecně mělo zvýšení paměti větší efekt na ústředny, které byly nainstalovány na virtuálním stroji s OS CentOS.

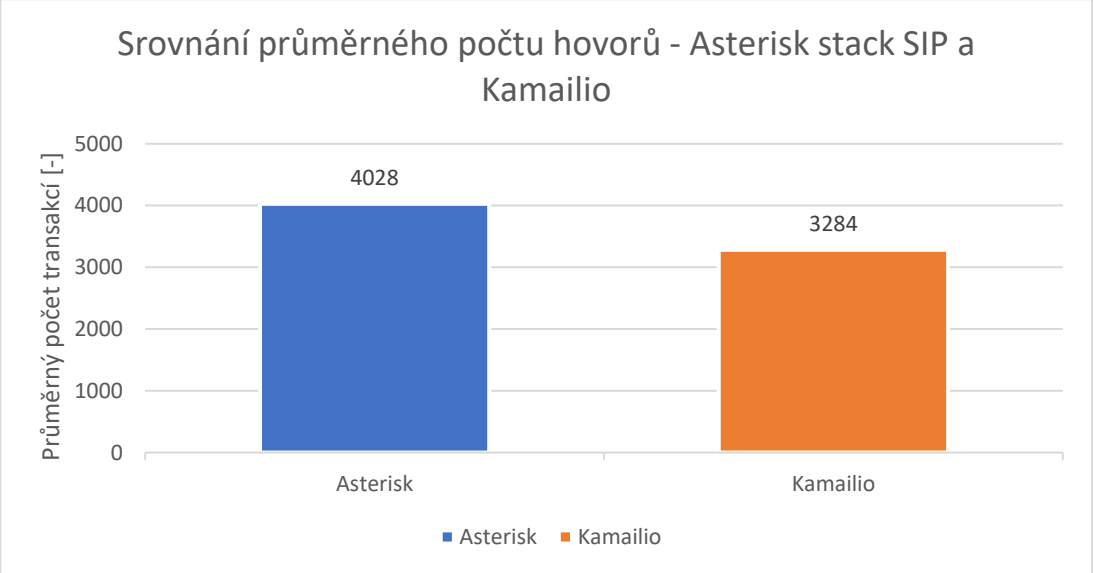
Při srovnání stacků SIP a PJSIP je vidět, že v případě hovorů došlo u obou stacků k nárůstu počtu hovorů. U stacku PJSIP byl tento nárůst větší než u stacku SIP, kde počet hovorů rostl střídměji. Na konfiguraci se 4 GB se stack PJSIP vyrovnal stacku SIP v režimu B2BUA. V proxy režimu nedošlo k překonání nativního stacku SIP. Přidělení většího množství zdrojů mělo razantní vliv na počet transakcí, kde stack PJSIP značně překonal nativní stack.

7. SROVNÁNÍ VÝSLEDKŮ

V rámci získaných výsledků byly srovnány jednotlivé ústředny, vliv operačního systému, rozdíl mezi výkonností nativního stacku SIP a stacku PJSIP u ústředen Asterisk a režimy komunikace u ústředen Asterisk. Hodnoty byly pro přehlednost seskupeny do tabulek a grafů.

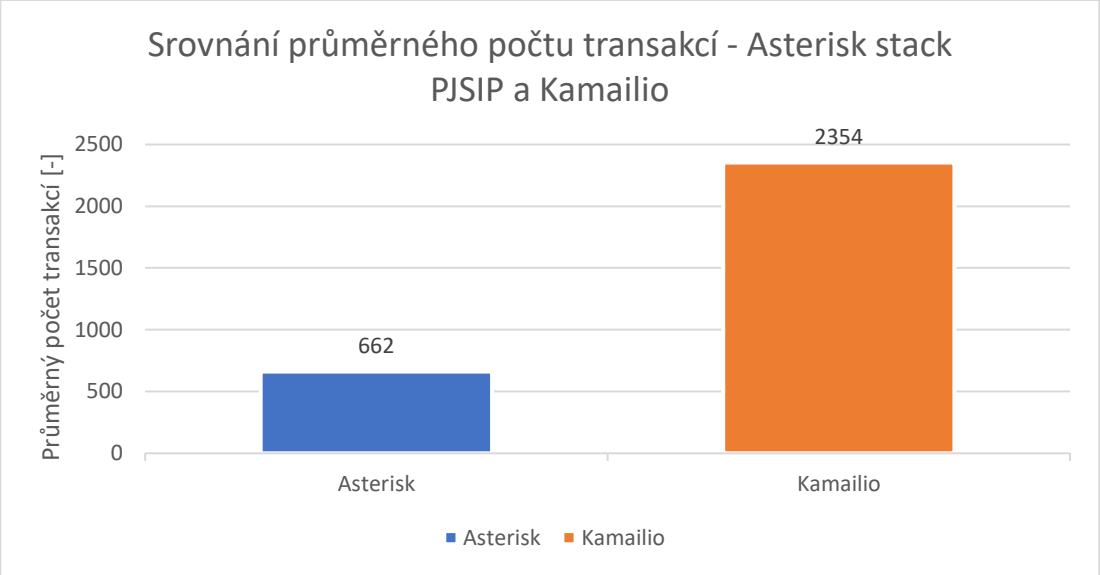


Obrázek 7.1 Srovnání průměrného počtu transakcí – Asterisk stack SIP a Kamailio

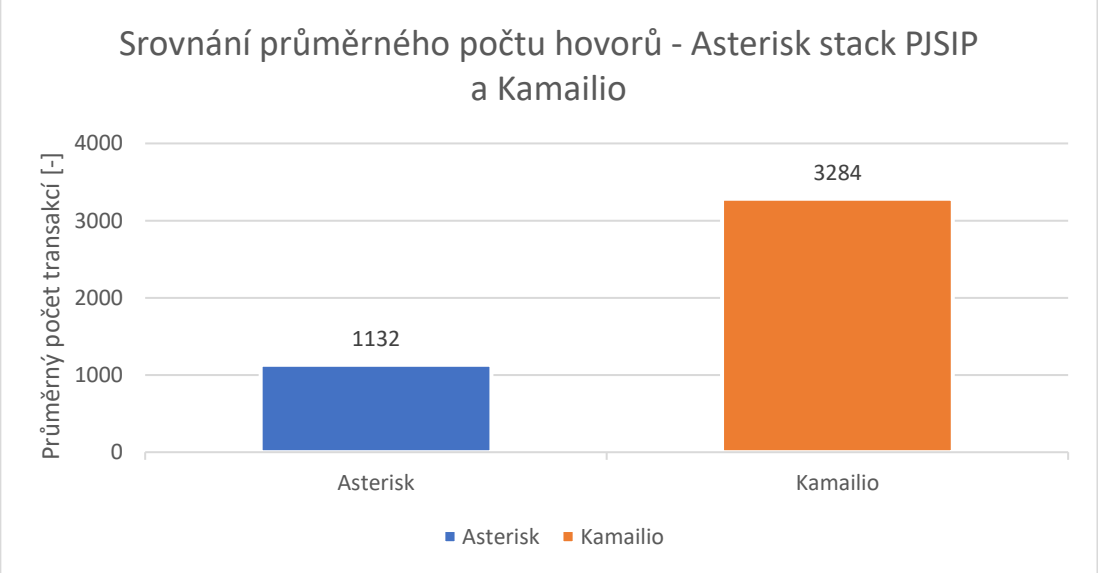


Obrázek 7.2 Srovnání průměrného počtu hovorů – Asterisk stack SIP a Kamailio

Obrázek 7.1 obsahuje průměrný počet transakcí ze všech výsledků ústředen Asterisk pracujících v režimu proxy, využívající stack SIP a porovnává je s průměrnou hodnotou transakcí dosažených u ústředny Kamailio. Obrázek 7.2 obsahuje průměrné hodnoty transakcí u ústředny Asterisk pracující v režimu proxy s využitím stacku PJSIP a u ústředny Kamailio.



Obrázek 7.3 Srovnání průměrného počtu transakcí – Asterisk stack PJSIP a Kamailio

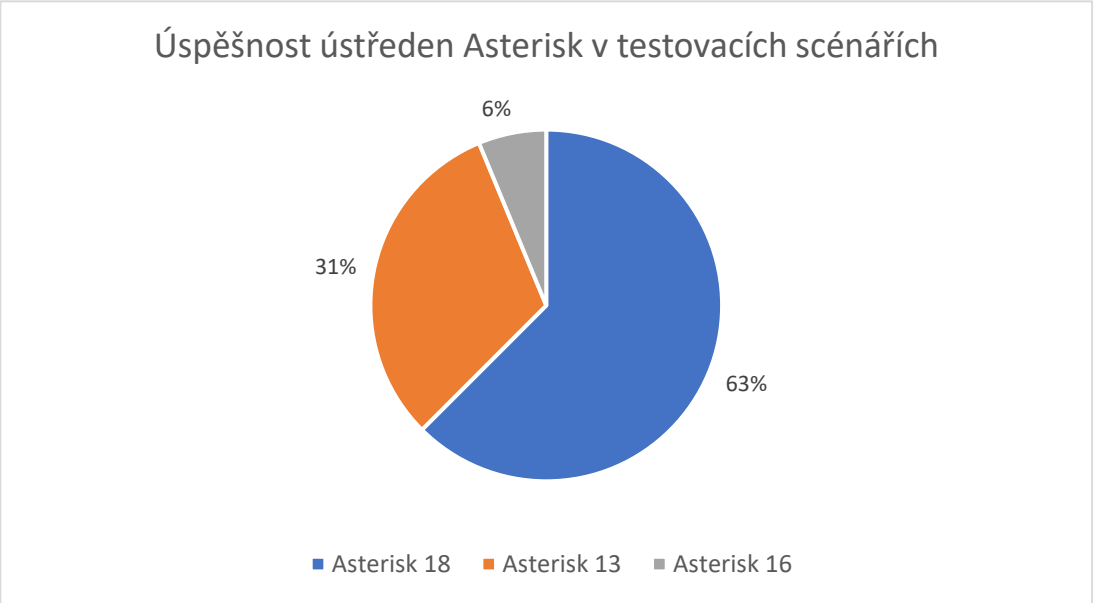


Obrázek 7.4 Srovnání průměrného počtu hovorů – Asterisk stack PJSIP a Kamailio

Obrázek 7.3 obsahuje průměrné hodnoty transakcí obou ústředen Asterisk i Kamailio, kde u Asterisku je použit proxy režim a stack PJSIP. Obrázek 7.4 obsahuje

průměrné hodnoty hovorů obou ústředen Asterisk i Kamailio, kde u Asterisku je použit proxy režim a stack PJSIP. Ve výše uvedených grafech byl porovnán počet hovorů a transakcí dosažených ústřednami Asterisk a Kamailio. Pro porovnání byl u ústředny Asterisk zvolen režim proxy, protože se více podobá realizaci hovorů u proxy serveru Kamailio. V rámci srovnání byly použity průměrné hodnoty pro danou ústřednu v rámci testovací série (např. všechny hodnoty ústředen Asterisk v režimu proxy na konfiguraci s 4 GB) a pro přehlednost byly srovnávány jen výsledky získané z měření na konfiguraci s 4 GB. Z grafů je patrné, že Kamailio dosáhlo většího počtu transakcí než ústředny Asterisk a to s nativní implementací SIP i implementací PJSIP. Ústředna Asterisk se stackem SIP dosáhla lepších výsledků, než Kamailio.

V rámci testování na počet hovorů je možné si povšimnout, že Kamailio nepřekonalo výsledný počet hovorů, kterých dosáhly ústředny Asterisk. Nestalo se tak na žádné z obou použitých konfigurací. Vzhledem k provedeným testům na počet transakcí, kde dosahovalo Kamailio mnohem lepších výsledků než ústředny Asterisk, se může zdát, že množství hovorů realizovaných Kamailiem je zatíženo chybou. Touto chybou je pravděpodobně velké vytížení testeru. Při porovnání testovacích scénářů na počet transakcí a hovorů je patrné, že jejich hlavním rozdílem je délka hovorů. Zatímco u testu na počet transakcí jsou hovory ukončovány téměř okamžitě a u žádného z nich nedochází k přenosu multimediální relace, tak v případě testu na počet hovorů je situace opačná. Jednotlivé hovory zůstávají aktivní po celou dobu probíhajícího testu a přenáší mezi sebou multimediální relaci. Právě ta může mít zásadní vliv na výkon testeru. Způsobeno je to tím, že veškerá koncová zařízení simuluje tester (simusers). V případě Kamailia je pak veškerý provoz přenášen právě mezi těmito koncovými body, čímž je značně zatěžován server. Nikoliv však ústředna nebo virtuální počítač, na kterém byla spuštěna. U Asterisku se tento jev neprojevoval patrně proto, že se nejedná čistě o proxy server, ale komunikaci podobnou proxy režimu dokáže do jisté míry simulovat. Ulevit serveru od nadměrné zátěže je možné úpravou charakteru přenášeného multimediálního obsahu, ale ještě lepších výsledků lze dosáhnout jeho úplným odstraněním.



Obrázek 7.5 Úspěšnost ústředen Asterisk

Obrázek 7.5 obsahuje procentuální úspěšnost ústředen Asterisk v testovacích scénářích. Vyhodnoceny byly jednotlivé série testů a v rámci nich byla zvolena jedna ústředna, která měla v dané sérii nejlepší výsledky bez ohledu na použitý OS. Bylo provedeno celkově 16 sérií testů. V každé sérii byla zvolena nejúspěšnější ústředna. Nejúspěšnější ústředna s výsledkem devíti nejlepších výsledků v rámci série testů je ústředna Asterisk 18. To činí 63 % nejlepších výsledků z celkového počtu testů. Ústředna Asterisk 13 získala 6 nejlepších výsledků v sérii testování, což činí 31 % výsledků a ústředna Asterisk 16 měla pouze jeden nejlepší výsledek. Při porovnání projektů Kamailio a Asterisk mělo Kamailio lepší výsledky v rámci testů na počet transakcí. V rámci testu na počet hovorů mělo lepší výsledky než ústředny pracující se stackem PJSIP, ale horší výsledky než ústředny pracující s nativním stackem SIP v režimu proxy. Zde se ovšem vyskytla událost, popisovaná výše, při které se nezvyšoval počet hovorů ani při opakovaných pokusech o navýšení zátěže v testovacím scénáři a ústředně přitom zůstalo dostatečné množství nevyužitých zdrojů. Teoreticky může být výsledné množství hovorů větší.

Tabulka 7.1 Srovnání použitých OS – nižší paměť RAM

Asterisk – nižší paměť RAM				
Průměrný počet transakcí za sekundu v režimu B2BUA a proxy režimu (stack SIP)				
Transakce – B2BUA	1126		Transakce – proxy	1010
Ubuntu [%]	49,5		Ubuntu [%]	47,7
CentOS [%]	50,5		CentOS [%]	52,3
Průměrný počet transakcí za sekundu v režimu B2BUA a proxy režimu (stack PJSIP)				
Transakce – B2BUA	928		Transakce – proxy	669
Ubuntu [%]	52,6		Ubuntu [%]	53,4
CentOS [%]	47,4		CentOS [%]	46,6
Průměrný počet hovorů v režimu B2BUA (stack SIP)				
Hovory – B2BUA	3207		Hovory – proxy	7575
Ubuntu [%]	55,6		Ubuntu [%]	51,5
CentOS [%]	44,4		CentOS [%]	48,5
Průměrný počet hovorů v režimu B2BUA (stack PJSIP)				
Hovory – B2BUA	2973		Hovory – proxy	1044
Ubuntu [%]	55,7		Ubuntu [%]	51,4
CentOS [%]	44,3		CentOS [%]	48,6

Tabulka 7.2 Srovnání použitých OS – vyšší paměť RAM

Asterisk – vyšší paměť RAM				
Průměrný počet transakcí za sekundu v režimu B2BUA a proxy režimu (stack SIP)				
Transakce – B2BUA	1244		Transakce – proxy	1195
Ubuntu [%]	46,4		Ubuntu [%]	42,8
CentOS [%]	53,6		CentOS [%]	57,2
Počet transakcí za sekundu v režimu B2BUA a proxy režimu (stack PJSIP)				
Transakce – B2BUA	2205		Transakce – proxy	1323
Ubuntu [%]	52,1		Ubuntu [%]	51,5
CentOS [%]	47,9		CentOS [%]	48,5
Maximální počet hovorů v režimu B2BUA (stack SIP)				
Hovory – B2BUA	4075		Hovory – proxy	8056
Ubuntu [%]	49,5		Ubuntu [%]	51,7
CentOS [%]	50,5		CentOS [%]	48,3
Maximální počet hovorů v proxy režimu (stack PJSIP)				
Hovory – B2BUA	3861		Hovory – proxy	2265
Ubuntu [%]	49,8		Ubuntu [%]	54,4
CentOS [%]	50,2		CentOS [%]	45,6

Tabulka 7.3 Srovnání použitých OS – Kamailio

Kamailio – nižší paměť RAM			Kamailio – vyšší paměť RAM	
Počet transakcí za sekundu	4629		Počet transakcí za sekundu	4708
Ubuntu [%]	63,3		Ubuntu [%]	63,6
CentOS [%]	36,7		CentOS [%]	36,4
Počet hovorů	6385		Počet hovorů	6567
Ubuntu [%]	49,1		Ubuntu [%]	48,8
CentOS [%]	50,9		CentOS [%]	51,2

Tabulka 7.1 obsahuje výsledky měření u ústředen Asterisk. Jsou zde zahrnuty pouze výsledky měření v konfiguraci s nižší pamětí RAM. Tabulka 7.2 obsahuje výsledky měření u ústředen Asterisk v konfiguraci s vyšší pamětí RAM. Tabulka 7.3 obsahuje výsledky měření u ústředny Kamailio pro obě konfigurace přidělené paměti. V prvních dvou tabulkách jsou nejprve zprůměrovány výsledky ze všech ústředen na dané operační systémy a výsledky za oba operační systémy jsou sečteny. Z této hodnoty a z průměru výsledků z ústředen na daném OS je vypočítáno procentuální zastoupení výsledků vzhledem k celkovému počtu.

Po vyhodnocení úspěšnosti ústředen byl vyhodnocen vliv operačního systému na ústředny Asterisk i Kamailio. V jednotlivých sériích testů byly zprůměrovány výsledky u daného operačního systému a výsledné hodnoty byly porovnány. Z hodnot vyplynulo, že z dvaceti srovnávaných výsledků byly v šedesáti procentech případů úspěšnější ústředny nainstalované na operačním systému Ubuntu. Do vyhodnocení byly zahrnuty i výsledky z ústředny Kamailio, nainstalované na obou operačních systémech. Procentuální rozložení dokázalo, že rozdíl mezi jednotlivými OS je maximálně několik procent a nelze tak jednoznačně určit operační systém, který by byl lepší nebo horší pro instalaci. Z celkového počtu dvaceti testovacích sérií měl operační systém Ubuntu lepší výsledky ve dvanácti případech a operační systém CentOS měl lepší výsledky ve zbylých osmi případech.

Tabulka 7.4 Průměrný počet transakcí a hovorů – nižší paměť

Průměrný počet transakcí za sekundu v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack SIP					Průměrný počet transakcí za sekundu v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack PJSIP				
Ubuntu	557		Ubuntu	482	Ubuntu	488		Ubuntu	357
CentOS	569		CentOS	528	CentOS	440		CentOS	312
Průměrný počet hovorů v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack SIP					Průměrný počet hovorů v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack PJSIP				
Ubuntu	1782		Ubuntu	3902	Ubuntu	1657		Ubuntu	537
CentOS	1425		CentOS	3672	CentOS	1316		CentOS	507

Tabulka 7.5 Celkové a procentuální vyjádření výsledků pro nativní stack SIP (vlevo) a implementaci stacku PJSIP (vpravo)– nižší paměť

Transakcí celkem – SIP	2136	Transakcí celkem – PJSIP	1597
Transakcí procentuálně	57,2	Transakcí procentuálně	42,8
Hovorů celkem – SIP	10782	Hovorů celkem – PJSIP	4017
Hovorů procentuálně	72,9	Hovorů procentuálně	27,1

Tabulka 7.6 Průměrný počet transakcí a hovorů – vyšší paměť

Průměrný počet transakcí za sekundu v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack SIP					Průměrný počet transakcí za sekundu v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack PJSIP				
Ubuntu	577		Ubuntu	511	Ubuntu	1148		Ubuntu	682
CentOS	667		CentOS	683	CentOS	1057		CentOS	642
Průměrný počet hovorů v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack SIP					Průměrný počet hovorů v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack PJSIP				
Ubuntu	2016		Ubuntu	4166	Ubuntu	1924		Ubuntu	1232
CentOS	2059		CentOS	3891	CentOS	1937		CentOS	1032

Tabulka 7.7 Celkové a procentuální vyjádření výsledků pro nativní stack SIP a implementaci stacku PJSIP – vyšší paměť

Transakcí celkem – SIP	2439	Nárůst transakcí po zvětšení paměti [%]	12,44		Transakcí celkem – PJSIP	3529	Nárůst transakcí po zvětšení paměti [%]	54,73
Transakcí procentuálně	40,9				Transakcí procentuálně	59,1		
Hovorů celkem – SIP	12131	Nárůst hovorů po zvětšení paměti [%]	11,12		Hovorů celkem – PJSIP	6126	Nárůst hovorů po zvětšení paměti [%]	34,42
Hovorů procentuálně	66,4				Hovorů procentuálně	33,6		

K porovnání stacků byly využity 4 tabulky. Tabulka 7.4 obsahuje průměrné hodnoty z výsledků měření ústředen na operačních systémech Ubuntu a CentOS. Tabulka 7.5 obsahuje součet transakcí a hovorů v rámci využitého stacku a jejich procentuální zastoupení vzhledem k celkovému počtu. Pro vyhodnocení bylo zvlášť nahlíženo na výsledky v konfiguraci s nižší a vyšší přidělenou pamětí RAM. Tabulka 7.6 obsahuje průměrné hodnoty výsledků ústředen, nainstalovaných na obou OS v konfiguraci s vyšší pamětí RAM. Tabulka 7.7 obsahuje součet transakcí a hovorů za jednotlivé stacky a jejich procentuální zastoupení v rámci celkového počtu.

Z tabulek je vidět, že na konfiguraci s nižší pamětí RAM realizoval nativní stack SIP větší procento hovorů i transakcí. V konfiguraci s dvojnásobným množstvím přidělené paměti RAM došlo k nárůstu počtu transakcí i hovorů u obou stacků. I v tomto případě má stack SIP lepší výsledky v proxy režimu, kde realizoval 66,4 % z celkového počtu hovorů. V případě transakcí má lepší výsledky implementace stacku PJSIP. Vzhledem k výsledkům získaným na konfiguraci s nižší pamětí RAM došlo u stacku SIP po zdvojnásobení paměti k nárůstu transakcí o 12,44 % a navýšení počtu hovorů o 11,12 %, zatímco pro stack PJSIP došlo v režimu B2BUA k nárůstu transakcí až o 54,73 % a v proxy režimu došlo k nárůstu o 34,42 %.

Tabulka 7.8 Celkové a procentuální vyjádření výsledků pro režimy B2BUA a proxy – nižší paměť

Transakce v režimu B2BUA	2054		Transakce v proxy režimu	1679
Procentuální zastoupení transakcí	55,0		Procentuální zastoupení transakcí	45,0
Hovory v režimu B2BUA	6180		Hovory v proxy režimu	8619
Procentuální zastoupení hovorů	41,8		Procentuální zastoupení hovorů	58,2

Tabulka 7.9 Celkové a procentuální vyjádření výsledků pro režimy B2BUA a proxy – vyšší paměť

Transakce v režimu B2BUA	3450	Nárůst transakcí [%]	40,47	Transakce v proxy režimu	2518	Nárůst transakcí [%]	33,31
Procentuální zastoupení transakcí	57,8			Procentuální zastoupení transakcí	42,2		
Hovory v režimu B2BUA	7936	Nárůst hovorů [%]	22,12	Hovory v proxy režimu	10321	Nárůst hovorů [%]	16,49
Procentuální zastoupení hovorů	43,5			Procentuální zastoupení hovorů	56,5		

Tabulka 7.8 obsahuje celkový počet transakcí a hovorů v režimech B2BUA a proxy. Tabulka 7.9 obsahuje obdobné výsledky, ale v konfiguraci s vyšší pamětí RAM. Hodnoty v tabulkách jsou získány z již dříve uvedených tabulek 7.4 a 7.6. V tomto případě se ale sčítaly průměrné hodnoty transakcí v rámci režimu komunikace a tím byl získán celkový počet transakcí v daném režimu. Po sečtení transakcí v obou režimech bylo následně vypočítáno procentuální zastoupení v jednotlivých režimech komunikace. Shodný postup byl aplikován i pro počet hovorů.

Z tabulky je patrné, že v konfiguraci s nižší pamětí RAM bylo v režimu B2BUA dosaženo většího množství transakcí a v proxy režimu bylo naopak dosaženo většího množství hovorů. Tento trend byl zachován i u výsledků v konfiguraci s vyšší pamětí. Po navýšení paměti RAM pak v rámci režimu B2BUA došlo k navýšení transakcí o 40,47 % a zvýšení počtu hovorů o 22,12 %. U proxy režimu navýšení počtu transakcí čítalo zlepšení o 33,31 % a zvýšení počtu hovorů bylo o 16,49 %.

ZÁVĚR

Diplomová práce se zabývala testováním open source pobočkových ústředen Asterisk a Kamailio. Na začátku práce byly uvedeny kapitoly, zabývající se problematikou pobočkových ústředen. Zde byl zahrnut popis pobočkových ústředen a jejich dělení dle jednotlivých generací, se zaměřením právě na poslední generaci, zabývající se IP telefonií. S návazností na tento popis byl v následující kapitole přidán popis přenosových a signalizačních protokolů, které pobočkové ústředny páté generace využívají. V této kapitole byly dále popsány různé druhy používaných metod a zpráv, dále druhy serverů používaných v IP telefonii a na závěr byl přiložen popis novějšího stacku PJSIP.

Následující kapitoly se věnovaly jednotlivým ústřednám Asterisk a Kamailio. V kapitole zabývající se ústřednou Asterisk byla v úvodu popsána samotná pobočková ústředna a krátce byl zmíněn vznik tohoto projektu. Další popis se zabýval modulární architekturou této ústředny a implementací novějšího stacku PJSIP. Do této kapitoly byl umístěn i popis přípravy pracoviště, použitého pro následující testy. Zde byl uveden postup instalace operačního systému na virtuální stroj a následná instalace ústředny, konfigurace účtů pro oba stacky SIP i PJSIP. V závěru kapitoly byl popsán rozdíl ve způsobu chování ústředny Asterisk v režimu B2BUA a proxy režimu.

Obdobná kapitola se zabývala SIP proxy serverem Kamailio. I zde byl v úvodu kapitoly popsán vznik proxy serveru Kamailio, obecná charakteristika a modulární architektura tohoto serveru. Dále byla popsána instalace serveru na již připravený virtuální stroj, vytvoření účtů v databázi a úprava směrovací logiky pro jednotlivé hovory.

V další kapitole byl uveden přehled funkcionalit jednotlivých ústředen a bylo porovnáváno, zda jsou dané funkcionality implementovány do obou ústředen. Celý seznam byl shrnut do přehledné tabulky. Jednotlivé funkcionality ústředen byly vždy získávány z oficiálních webových stránek pro danou ústřednu. K funkcionalitám Asterisku bylo následně hledáno, zda jsou implementovány v Kamailio a obráceně. Z porovnání vyplynulo, že Asterisk je jednodušší, uživatelsky přívětivější a poskytuje

více možností pro práci s médii. Kamailio je naopak složitější a neumožňuje některé funkce stejně jako Asterisk, ale je mnohem více škálovatelné, stabilní a rychlejší.

Šestá kapitola se zabývala popisem pracoviště VUT, přípravou testeru Spirent TestCenter C1 a konfigurací testů. Do této kapitoly byla umístěna také podkapitola zabývající se metodikou prováděného testování a způsobu zpracování výsledků.

V poslední kapitole jsou zahrnuty veškeré výsledky z provedeného testování. Byly prováděny dva druhy testů. První z nich se zaměřoval na maximální počet transakcí, kterých dokáže ústředna dosáhnout. K tomu byl přizpůsoben i samotný testovací scénář. Druhý test se zaměřoval na maximální množství souběžných hovorů, které ústředna zvládne realizovat, než začne docházet k zahazování. Všechny ústředny byly nainstalovány na samostatných virtuálních strojích. Toto umožnilo jednoduchou správu ústředen, kde pro každou byl vytvořen vlastní klon virtuálního stroje. Samotné využití virtuálních strojů následně poskytovalo volnost ve volbě konfigurace a vzdáleného přístupu k počítači, ale dovolovalo přidělení pouze omezeného množství zdroje, aby nedošlo ke snížení stability hostitelského PC. Jako operační systém pro instalaci ústředen byl zvolen CentOS. Pro porovnání, zda bude mít tato volba zásadní vliv na výkonnost systému, byly ústředny nainstalovány i na virtuální počítač s operačním systémem Ubuntu. Testování bylo prováděno se dvěma konfiguracemi paměti RAM.

První testy na konfiguraci se 2 GB paměti udaly hodnoty počtu transakcí i hovorů. Na této konfiguraci si ve většině testů vedla nejlépe ústředna Asterisk 18 v konfiguraci s OS Ubuntu. Při porovnání výsledků s Kamailiem je vidět, že v případě transakcí Kamailio značně překonává ústřednu Asterisk, jak bylo nastíněno v kapitole zabývající se porovnáním funkcionalit obou ústředen. V případě hovorů se vyskytl problém s přetížením testeru, který bránila dalšímu navyšování počtu hovorů. Při srovnání stacku SIP a PJSIP bylo z výsledků vidět, že stack SIP si vedl mnohem lépe v případě transakcí i v případě realizovaných hovorů. Tyto výsledky udaly předpoklad, zda dokáže stack PJSIP překonat nativní stack SIP například při zvýšení dostupných zdrojů jednotlivých ústředen. Z toho důvodu došlo ke zvětšení paměti RAM u virtuálních strojů, na kterých byly ústředny spuštěny.

Po zvýšení paměti RAM došlo i k navýšení počtu transakcí a hovorů. U stacku PJSIP byl nárůst mnohem větší než u stacku SIP. Stack PJSIP tak na nové konfiguraci

překonal stack SIP v počtu transakcí za sekundu. V případě režimu B2BUA se po zvýšení paměti RAM vyrovnal stack PJSIP nativnímu stacku SIP. V proxy režimu se tohoto nepodařilo docílit. Na Kamailio nemělo zvýšení paměti žádný větší vliv. Došlo sice k navýšení počtu transakcí, ale virtuální počítač nepostihl nedostatek zdrojů ani při prvním testu, proto také jejich zvýšení nemá příliš velký vliv na počet transakcí. Zde došlo pravděpodobně k vytížení testeru, ale ústředna by zvládla ještě větší počet transakcí. Zvýšení paměti nemělo vliv ani na počet hovorů. I zde při zvyšování zátěže nedošlo k navýšení hovorů.

V rámci srovnání ústředen Asterisk měla nejlepší výsledky ústředna Asterisk 18. Ze všech testovacích sérií dosáhla tato ústředna nejlepšího výsledku 63 % z celkového počtu osmnácti provedených testovaných sérií. Asterisk 16 měl nejlepší výsledek pouze v jednom případě, což činí 6 % z celkového počtu a ústředna Asterisk 13 měla nejlepší výsledek v 31 % procentech z celkového počtu testovaných sérií.

Pro porovnání výsledků za jednotlivé operační systémy, na kterých byly ústředny nainstalovány, dosáhly lepších výsledků ústředny, nainstalované na operačním systému Ubuntu. Ubuntu mělo lepší výsledky v 60 % výsledků a CentOS ve zbývajících 40 %. Z výsledků ovšem vychází, že rozdíl mezi jednotlivými ústřednami není až tak velký a lepší výsledky oproti OS CentOS se nevyskytly ve sto procentech případů, aby se dalo jednoznačně určit, že Ubuntu by mělo být vhodnější pro instalaci pobočkových ústředen Asterisk nebo Kamailio.

Porovnány byly rovněž stacky. Porovnávaly se výsledky nativního stacku SIP a implementace stacku PJSIP. U konfigurace s nižší pamětí RAM dosáhl lepších výsledků nativní stack SIP. Zde bylo dosaženo lepších výsledků jak v případě transakcí, tak i v případě hovorů. Při srovnání bylo s použitím nativního stacku realizováno 56,9 % z celkového počtu transakcí a 72,7 % z celkového počtu hovorů, zatímco s použitím implementace stacku PJSIP se realizovalo 43,1 % z celkového počtu transakcí a 27,3 % z celkového počtu hovorů. Při zvýšení paměti z 2 GB na 4 GB došlo rovněž ke zvýšení transakcí a hovorů. Toto zvýšení mělo rozdílný dopad na oba stacky. U stacku SIP se nyní realizovalo 40,9 % z celkového počtu transakcí a 66,2 % z celkového počtu hovorů, zatímco u implementace stacku PJSIP se realizovalo 59,1 % z celkového počtu transakcí a 33,8 % z celkového počtu hovorů. Vzhledem k výsledkům na konfiguraci s nižší pamětí RAM došlo u nativního stacku

k navýšení transakcí o 13,5 % a navýšení počtu hovorů o 10,96 %. U implementace stacku PJSIP mělo zvýšení paměti razantnější dopad. Počet transakcí se zvýšil až o 54,7 % a počet hovorů o 34,4 %.

V závěru poslední kapitoly byly porovnávány režimy komunikace pobočkových ústředen Kamailio. V konfiguraci s nižší pamětí RAM se realizovalo v režimu B2BUA 55,4 % z celkového počtu transakcí a 42,1 % z celkového počtu hovorů. V proxy režimu se realizovalo 44,6 % z celkového počtu transakcí a 57,9 % z celkového počtu hovorů. Při nižší hodnotě paměti RAM dosáhl většího množství transakcí režim B2BUa a většího množství hovorů proxy režim. V konfiguraci s vyšší hodnotou paměti RAM bylo v režimu B2BUA dosaženo 57,8 % z celkového počtu transakcí a 43,8 % z celkového počtu hovorů. V proxy režimu bylo dosaženo 42,2 % z celkového počtu transakcí a 56,2 % z celkového počtu hovorů. Zde zůstalo rozložení výsledků shodné jako v předešlém případě. V režimu B2BUA bylo realizováno více transakcí a v proxy režimu více hovorů. Po zvýšení paměti došlo k nárůstu transakcí i hovorů, které ústředny zvládly realizovat. V tomto případě nebyl mezi režimy komunikace tak zásadní rozdíl. Počet transakcí vzrostl v režimu B2BUA o 40,5 % a v proxy režimu o 34,4 %. Počet hovorů vzrostl v režimu B2BUA o 22,1 % a v proxy režimu o 16,4 %.

Na základě realizovaných testů jsem dospěl k těmto závěrům: Ústředny nainstalované na OS Ubuntu dosahovaly lepších výsledků než ty, které byly nainstalované na OS CentOS. Výsledek byl očekáván vzhledem k použitým jádrům obou OS. Kamailio realizovalo významně větší množství maximálních transakcí než Asterisk, zatímco v počtech hovorů mělo srovnatelné výsledky. Z ústředen Asterisk dosáhla nejlepších výsledků nejnovější LTS verze – Asterisk 18. Implementace stacku PJSIP je paměťově náročnější, ale zároveň i výkonnější než nativní stack SIP.

LITERATURA

[1] BAZALA, David. Telekomunikace & VoIP telefonie I. 1. vyd. Praha: BEN - technická literatura, 2006, 222 s. ISBN 80-7300-201-9. [cit. 2022-02-25].

[2] Principy – klasická telefonie. PhoNet – telefonní ústředny 5. generace [online]. Praha 10: ProTel engineering, c1996-2020 [cit. 2020-10-15]. Dostupné z: http://www.phonet.cz/old_tel.html

[3] SS7. Introduction to CCITT Signalling System No. 7 [online]. 1994 [cit. 2022-01-17]. Dostupné z: https://www.itu.int/rec/T-REC-Q.700-199303-I/en

[4] COLLINS, Daniel. Carrier grade voice over IP. Druhá edice. New York: McGraw-Hill, c2003, 522 s. ISBN 0071406344. [cit. 2022-02-25].

[5] Real – Time Transport Protocol a aplikační rozhraní RTP Java Media Framework. ElectroRevue [online]. Brno: Elektrorevue, 2003 [cit. 2020-10-15]. Dostupné z: http://www.elektrorevue.cz/clanky/03018/index.html

[6] RFC 3550. RTP: A Transport Protocol for Real-Time Applications [online]. 2003 [cit. 2022-01-17]. Dostupné z: https://www.rfc-editor.org/rfc/rfc3550.html

[7] RFC 3711. The Secure Real-time Transport Protocol (SRTP) [online]. 2004 [cit. 2022-01-17]. Dostupné z: https://www.rfc-editor.org/rfc/rfc3711.html

[8] RFC 3826. The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model [online]. 2004 [cit. 2022-01-17]. Dostupné z: https://www.rfc-editor.org/rfc/rfc3826.html

[9] RFC 3261. SIP: Session Initiation Protocol [online]. 2002 [cit. 2022-01-17]. Dostupné z: https://www.rfc-editor.org/rfc/rfc3261.html

[10] Úvod do VoIP. MB DATA [online]. PRAHA: MB DATA, 2006 [cit. 2020-10-15]. Dostupné z: https://www.mbdata.cz/uvoddovoip.html

[11] H.323. Packet-based multimedia communications systems [online]. 2010 [cit. 2022-01-17]. Dostupné z: https://www.itu.int/rec/T-REC-H.323-200912-I/en

[12] Signalizační protokol pro přenos hlasu přes datové sítě – SIP. ElectroRevue [online]. Brno: Elektrorevue, 2003 [cit. 2020-10-15]. Dostupné z: http://www.elektrorevue.cz/clanky/03003/index.html

[13] RFC 4566. SDP: Session Description Protocol [online]. 2006 [cit. 2022-01-17]. Dostupné z: https://www.rfc-editor.org/rfc/rfc4566.html

[14] Asterisk 12 Part IV: The SIP Stack of the Future. Sangoma [online]. Sangoma: Matt Jordan [cit. 2020-12-04]. Dostupné z: https://www.sangoma.com/articles/asterisk-12-part-iv-sip-stack-future/

[15] Asterisk as a Swiss Army Knife of Telephony. Asterisk Project Wiki [online]. Malcolm Davenport, Sean Bright, 2020 [cit. 2020-10-27]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Asterisk+as+a+Swiss+Army+Knife+of+Telephony

[16] A Brief History of the Asterisk Project. Asterisk Project Wiki [online]. Malcolm Davenport, Rusty Newton, 2020, 2019 [cit. 2020-10-27]. Dostupné z:

https://wiki.asterisk.org/wiki/display/AST/A+Brief+History+of+the+Asterisk+Project

[17] Asterisk Versions. Asterisk Project Wiki [online]. Russell Bryant, Sean Bright, 2020 [cit. 2020-10-27]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Asterisk+Versions

[18] Asterisk Architecture, The Big Picture. Asterisk Project Wiki [online]. Malcolm Davenport, Rusty Newton, 2020 [cit. 2020-10-27]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Asterisk+Architecture%2C+The+Big+Picture

[19] Types of Asterisk Modules. Asterisk Project Wiki [online]. Malcolm Davenport, Rusty Newton, 2020 [cit. 2020-10-27]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Types+of+Asterisk+Modules

[20] How To Install Asterisk 16 PBX on CentOS 7. ComputingforGeeks [online]. Josphat Mutai, 2020 [cit. 2020-11-20]. Dostupné z: https://computingforgeeks.com/how-to-install-asterisk-16-centos-7-linux/

[21] Kamailio SIP Server v3.2.0 Development Guide. Asipto [online]. Asipto: Daniel-Constantin Mierla, Elena-Ramona Modroiu, c2011 [cit. 2020-11-10]. Dostupné z: https://www.asipto.com/pub/kamailio-devel-guide/#c01osss

[22] Kamailio Architecture, Core and Modules. Telecom R & D [online]. Bengalúru: Altanai, 2014 [cit. 2020-11-20]. Dostupné z: https://telecom.altanai.com/2014/11/18/kamailio-modules/

[23] Install Latest Kamailio SIP Server on CentOS 8 / CentOS 7. ComputingforGeeks [online]. Josphat Mutai, 2020 [cit. 2020-11-20]. Dostupné z: https://computingforgeeks.com/how-to-install-latest-kamailio-sip-server-on-centos-linux/

[24] Features. Asterisk [online]. Sangoma Technologies, c2021 [cit. 2021-12-02]. Dostupné z: https://www.asterisk.org/get-started/features/

[25] Features. Kamailio [online]. The Kamailio SIP Server Project, 2014 [cit. 2021-12-02]. Dostupné z: https://www.kamailio.org/w/features/

[26] NOHEJLOVÁ, Alice. Inovace stavu IP telefonie na Ústavu výzkumu globální změny Akademie Věd ČR, v. v. i. [online]. Brno, 2020 [cit. 2022-01-09]. Dostupné z: https://is.muni.cz/th/d28sf/. Diplomová práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Aleš ROČEK.

[27] Kamailio vs Asterisk. Nick vs Networking [online]. Melbourne, Austrálie: Nick J., 2019 [cit. 2021-12-02]. Dostupné z: https://nickvsnetworking.com/kamailio-vs-asterisk/

[28] Spirent Hardware Referenc. *Spirentcom.com* [online]. Calabasas, USA: Spirent Communications, 2021 [cit. 2021-12-02]. Dostupné z: https://support-kb.spirent.com/resources/sites/SPIRENT/content/live/DOCUMENTATION/10000/DOC10031/en_US/Spirent%20Hardware%20Reference.pdf

[29] Malý, přenosný zátěžový generátor/analyzátor Spirent TestCenter C1. TR instruments spol. s.r.o. [online]. Brno: TR instruments spol. [cit. 2020-12-04]. Dostupné z: http://www.trinstruments.cz/spirent-testcenter-c1

[30] SPIRENT AVALANCHE. Spirentcom.com [online]. Calabasas, USA: Spirent Communications, c2013 [cit. 2021-12-02]. Dostupné z: https://www.toyo.co.jp/files/user/img/products/ict/pdf/spirent_NA/11_Avalanche_SIPNG_Datasheet.pdf

[31] Testing SIPNG. Spirentcom.com [online]. Calabasas, USA: Spirent Communications, c2021 [cit. 2021-12-02]. Dostupné z: http://kms.spirentcom.com/CSC/Avalanche_WebHelp/index.htm#testing_sipng.htm

SEZNAM PŘÍLOH

Příloha 1: Konfigurace a scripty

Konfigurace pobočkové ústředny Asterisk

Konfigurace Sip.conf

```
[general]
bindport=5060
allow=all

[ucet](!)
type=friend
context=TEST
host=dynamic
allow=all
secret=1234
insecure=invite
directrtpsetup=yes


[1000](ucet)
username = 1000

[1001](ucet)
username = 1001
.
.
.
[3000](ucet)
username = 3000
```

Script pro vytvoření SIP účtů

```
cat >> sip.conf << EOF1
[general]
bindport=5060
allow=all

[ucet](!)
type=friend
context=TEST
host=dynamic
allow=all
secret=1234
insecure=invite
directrtpsetup=yes
EOF1
```

```
for i in {1000..3000}
do
cat >> sip.conf << EOF2
[$i](ucet)
username=$i
EOF2
done
```

Konfigurace pjsip.conf

```
[transport]
type=transport
protocol=udp
bind=0.0.0.0:5066

[endpoint](!)
type=endpoint
context=pjsipTEST
disallow=all
allow=alaw
direct_media=yes

[authTest](!)
type=auth
auth_type=userpass

[aorTest](!)
type=aor
max_contacts=9999


[1000](endpoint)
outbound_auth=1000
aors=1000
[1000](authTest)
password=1234
username=1000
[1000](aorTest)

[1001](endpoint)
outbound_auth=1001
aors=1001
[1001](authTest)
password=1234
username=1001
[1001](aorTest)
```

.
.
.

```
[3000](endpoint)
outbound_auth=3000
aors=3000
[3000](authTest)
password=1234
username=3000
[3000](aorTest)
```

Konfigurace scriptu pro vytvoření účtu PJSIP

```
cat >> pjsip.conf << EOF1
[transport]
type=transport
protocol=udp
bind=0.0.0.0:5066

[endpoint](!)
type=endpoint
context=pjsipTEST
disallow=all
allow=alaw
direct_media=yes

[authTest](!)
type=auth
auth_type=userpass

[aorTest](!)
type=aor
max_contacts=9999

EOF1

for i in {1000..3000}
do
cat >> pjsip.conf << EOF2
[$i](endpoint)
outbound_auth=$i
aors=$i
[$i](authTest)
password=1234
```

username=$i
[$i](aorTest)

EOF2
done

Konfigurace extensions.conf

[general]
static=yes
writeprotection=yes

[TEST]
exten => _[1-9]XXX,1,DIAL(SIP/3000)
exten => _[1-9]XXX,2,Hangup

[pjsipTest]
exten => _[1-9]XXX,1,DIAL(PJSIP/3000)
exten => _[1-9]XXX,2,Hangup

Konfigurace proxy serveru Kamailio

Script pro vygenerování účtů

for i in {1000..3000}

do

cat >> zapis << EOF

$(($i – 999)), $i, 192.168.20.199, 1234

EOF

done

Obsah vytvořeného souboru

1, 1000, 192.168.20.199, 1234

2, 1001, 192.168.20.199, 1234

.

.

.

2001, 3000, 192.168.20.199, 1234

Upravované sekce v konfiguračním souboru Kamailio.cfg

#!KAMAILIO

```
#
#!define WITH_MYSQL
#!define WITH_AUTH
#!define WITH_USERLOCDB
#!define WITH_ACCDB
#

memdbg=5
memlog=5

log_facility=LOG_LOCAL0
log_prefix="{$mt $hdr(CSeq) $ci} "

/* number of SIP routing processes for each UDP socket
 * - value inherited by tcp_children and sctp_children when not set explicitely */
children=4

/* uncomment the next line to disable TCP (default on) */
# disable_tcp=yes

/* number of SIP routing processes for all TCP/TLS sockets */
# tcp_children=8

/* uncomment the next line to disable the auto discovery of local aliases
 * based on reverse DNS on IPs (default on) */
# auto_aliases=no

/* add local domain aliases - it can be set many times */
# alias="sip.mydomain.com"

/* listen sockets - if none set, Kamailio binds to all local IP addresses
 * - basic prototype (full prototype can be found in Wiki - Core Cookbook):
```

```
 *      listen=[proto]:[localip]:[lport] advertise [publicip]:[pport]
 * - it can be set many times to add more sockets to listen to */
# listen=udp:10.0.0.10:5060

/* life time of TCP connection when there is no traffic
 * - a bit higher than registration expires to cope with UA behind NAT */
tcp_connection_lifetime=3605

/* upper limit for TCP connections (it includes the TLS connections) */
tcp_max_connections=2048

#!ifdef WITH_JSONRPC
tcp_accept_no_cl=yes
#!endif

######## Modules Section ########

/* set paths to location of modules */
# mpath="/usr/lib/x86_64-linux-gnu/kamailio/modules/"

#!ifdef WITH_MYSQL
loadmodule "db_mysql.so"
#!endif

#!ifdef WITH_JSONRPC
loadmodule "xhttp.so"
#!endif
loadmodule "jsonrpcs.so"
loadmodule "kex.so"
loadmodule "corex.so"
loadmodule "tm.so"
loadmodule "tmx.so"
```

```
loadmodule "sl.so"
loadmodule "rr.so"
loadmodule "pv.so"
loadmodule "maxfwd.so"
loadmodule "usrloc.so"
loadmodule "registrar.so"
loadmodule "textops.so"
loadmodule "textopsx.so"
loadmodule "siputils.so"
loadmodule "xlog.so"
loadmodule "sanity.so"
loadmodule "ctl.so"
loadmodule "cfg_rpc.so"
loadmodule "acc.so"
loadmodule "counters.so"

modparam ("rr", "append_fromtag", 1);
#loadmodule "uac.so"



# ----- rr params -----
# set next param to 1 to add value to ;lr param (helps with some UAs)
modparam("rr", "enable_full_lr", 0)
# do not append from tag to the RR (no need for this script)
#modparam("rr", "append_fromtag", 0)

######## Routing Logic ########

# User location service
route[LOCATION] {
rewriteuri("sip:3000@192.168.20.199");
```

```
#!ifdef WITH_SPEEDDIAL
        # search for short dialing - 2-digit extension
        if($rU=~"^[0-9][0-9]$") {
                if(sd_lookup("speed_dial")) {
                        route(SIPOUT);
                }
        }
#!endif
# PSTN GW routing
route[PSTN] {
#!ifdef WITH_PSTN
        # check if PSTN GW IP is defined
        if (strempty($sel(cfg_get.pstn.gw_ip))) {
                xlog("SCRIPT: PSTN routing enabled but pstn.gw_ip not defined\n");
                return;
        }

        # route to PSTN dialed numbers starting with '+' or '00'
        #     (international format)
        # - update the condition to match your dialing rules for PSTN routing
        if(!($rU=~"^(\+|00)[1-9][0-9]{3,20}$")) return;

        # only local users allowed to call
        if(from_uri!=myself) {
                sl_send_reply("403", "Not Allowed");
                exit;
        }
        # normalize target number for pstn gateway
        # - convert leading 00 to +
        if (starts_with("$rU", "00")) {
                strip(2);
                prefix("+");
```

```
        }

        if (strempty($sel(cfg_get.pstn.gw_port))) {
                $ru = "sip:" + "3000" + "@" + $sel(cfg_get.pstn.gw_ip);
        } else {
                $ru = "sip:" + "3000" + "@" + $sel(cfg_get.pstn.gw_ip) + ":"
                                        + $sel(cfg_get.pstn.gw_port);
        }

        route(RELAY);
        exit;
#!endif

        return;
}
```

Script pro záznam zdrojů virtuálního počítače

```
#pamet RAM
DATE=$(date "+%H:%M:%S")
RAMFREE=$(free --mega | grep Mem | awk '{print ($2-$3)}')
RAMFREE_TOTAL="$DATE $RAMFREE"
echo $RAMFREE_TOTAL >> RAMFREE.txt

RAM=$(free --mega | grep Mem | awk '{print ($2-$4)}')
RAM_TOTAL="$DATE $RAM"
echo $RAM_TOTAL >> RAM.txt


#vyuziti CPU
CPU=$(mpstat | grep all | awk '{print(100-$13)}')
CPU_TOTAL="$DATE $CPU%"
echo $CPU_TOTAL >> CPU.txt
```

Příloha 2: Použité tabulky

Verze Asterisku

Číslo verze	Typ	Datum	Bezpečnostní opravy	Ukončení podpory
1.2.x	–	21.11.2005	07.08.2007	21.11.2010
1.4.x	LTS	23.12.2006	21.04.2011	21.04.2012
1.6.0.x	Standard	01.10.2008	01.05.2010	01.10.2010
1.6.1.x	Standard	27.04.2009	01.05.2010	27.04.2011
1.6.2.x	Standard	18.12.2009	21.04.2011	21.04.2012
1.8.x	LTS	21.10.2010	21.10.2014	21.10.2015
10.x	Standard	15.12.2011	15.12.2012	15.12.2013
11.x	LTS	25.10.2012	25.10.2016	25.10.2017
12.x	Standard	20.12.2013	20.12.2014	20.12.2015
13.x	LTS	24.10.2014	24.10.2020	24.10.2021
14.x	Standard	26.09.2016	26.09.2017	26.09.2018
15.x	Standard	03.10.2017	03.10.2018	03.10.2019
16.x	LTS	09.10.2018	09.10.2022	09.10.2023
17.x	Standard	28.10.2019	28.10.2020	28.10.2021
18.x	LTS	20.10.2020	20.10.2024	20.10.2025
19.x	Standard	02.11.2021	02.11.2022	02.11.2023

Přehled funkcionalit uvedených na stránkách Asterisku

Funkcionality uváděné na oficiální stránce Asterisku			
Funkcionalita	**Asterisk**	**Kamailio**	**Popis**
ADSI	Ano	Ne	Implementace protokolu ADSI
Alarm Receiver	Ano	Ne	Nastavení upozornění pomocí telefonu
Append Message	Ano	Ano	Doručení zprávy na e–mail
Authentication	Ano	Ano	Autentizace klientů
Automated Attendant	Ano	Ne	Automatické přepojování
Blacklists	Ano	Ano	Seznam hovorů, které budou zamítnuty
Blind Transfer	Ano	Ne	Předání hovoru na jinou linku
Call Detail Records	Ano	Ano	Detaily o zprostředkovaném hovoru
Call Forward on Busy	Ano	Ano	Přesměrování v případě, kdy je volaná linka obsazená
Call Forward on No Answer	Ano	Ano	Přesměrování v případě, když není získána odpověď na hovor
Call Forward Variable	Ano	Ano	Přesměrování pomocí kódu
Call Monitoring	Ano	Ne	Monitorování a evidence proběhlých, příchozích a odchozích hovorů
Call Parking	Ano	Ne	Pozastavení a odložení hovoru s možností vyzvednutí na stejné nebo jiné lince

Call Queuing	Ano	Ano	Možnost řazení příchozích hovorů do front a jejich vyzvedávání účastníky
Call Recording	Ano	Ano	Nahrávání hovorů a jejich následné uložení
Call Retrieval	Ano	Ne	Volání hovorů pozastavených funkcí Call parking
Call Routing	Ano	Ano	Směrování hovorů
Call Snooping	Ano	Ne	Naslouchání hovoru bez přenosu hlasu naslouchajícího
Call Transfer	Ano	Ne	Přenos hovoru na jinou linku
Call Waiting	Ano	Ano	Funkce, která při dvou příchozích hovorech umožňuje mezi těmito hovory přepínat
Caller ID	Ano	Ano	Zobrazení čísla a jména volajícího
Caller ID Blocking	Ano	Ne	Blokování na základě identifikátoru volajícího
Caller ID on Call Waiting	Ano	Ne	Zobrazení čísla a jména dalšího volajícího
Conference Bridging	Ano	Ne	Vytvoření konference mezi účastníky
Database Store / Retrieve	Ano	Ano	Využití databáze pro ukládání a nahrávání dat
Database Integration	Ano	Ano	Poskytování informací o hovoru před jeho zvednutím nebo v jeho průběhu
Dial by Name	Ano	Ne	Vytáčení pomocí záznamu kontaktu
Direct Inward System Access	Ano	Ne	Přístup k funkcím z vnější strany sítě
Distinctive Ring	Ano	Ne	Rozdílné vyzvánění dle volajícího
Distributed Universal Number Discovery (DUNDi™)	Ano	Ne	Funkce pro sdílení informací z dialplanu mezi telefony
Do Not Disturb	Ano	Ne	Funkce pro nastavení nedostupnosti linky
E911	Ano	Ano	Funkce pro nouzová volání
ENUM	Ano	Ano	Funkce pro mapování čísla a internetové adresy
Fax Transmit and Receive	Ano	Ne	Funkce pro přijímání a odesílání faxů
Flexible Extension Logic	Ano	Ne	Flexibilní směrování hovorů
Interactive Directory Listing	Ano	Ne	Vyhledávání kontaktu podle jména ve firemním adresáři
Interactive Voice Response (IVR)	Ano	Ne	Interaktivní hlasový automat
Local and Remote Call Agents	Ano	Ano	Funkce pro přihlášení uživatele i na jiném telefonu a využívání ho jako vlastní telefon
Macros	Ano	Ano	Možnost využití maker
Music On Hold	Ano	Ano	Hudba při pozdrženém hovoru
Music On Transfer:	Ano	Ne	Hudba při transferu hovoru
Privacy	Ano	Ne	Možnost využít privacy managementu, uživatel je požádán o zadání kódu pro realizaci hovoru
Open Settlement Protocol (OSP)	Ano	Ano	Protokol pro autorizaci a účtování služeb

Overhead Paging	Ano	Ano	Funkce pro odesílání page zpráv skupině – interkom
Protocol Conversion	Ano	Ne	Funkce pro konverzi komunikačních protokolů u různých druhů sítí
Remote Call Pickup	Ano	Ne	Vzdálené vyzvednutí hovoru
Remote Office Support	Ano	Ne	Funkce pro vzdálené připojení telefonu mimo území PBX
Roaming Extensions	Ano	Ne	Možnost připojení se k jiným telefonům, identifikace kódem
Route by Caller ID	Ano	Ano	Směrování hovoru dle identifikátoru volajícího
SMS Messaging	Ano	Ano	SMS zprávy
Spell / Say	Ano	Ne	Přečtení specifického čísla nazpět volajícímu
Supervised Transfer	Ano	Ne	Transfer, při kterém je dohlíženo na to, zda proběhl úspěšně
Talk Detection	Ano	Ne	Detekce hlasu k spouštění různých událostí
Text–to–Speech	Ano	Ne	Funkce předčítání textu
Three–way Calling	Ano	Ne	Přidání dalšího účastníka do již probíhajícího hovoru
Time and Date	Ano	Ano	Funkce pro nastavení času a data
Transcoding	Ano	Ano	Umožňuje překódovat hovory, které využívají různé kodeky pro přenos zvuku
Trunking	Ano	Ano	Možnost připojení pobočkové ústředny do jiného systému
VoIP Gateways	Ano	Ano	Možnost využití jako brány pro různé systémy
Voicemail	Ano	Ano	Funkce k nahrávání vzkazů a odeslání do hlasové schránky – funkce indikace přijaté hlasové zprávy, možnost odeslání zprávy na určitý e–mail nebo skupině
Web Voicemail Interface	Ano	Ne	Triviální rozhraní pro správu
Zapateller	Ano	Ne	Blackhole pro nevyžádané marketingové hovory

Přehled funkcionalit uvedených na stránkách Kamailia

Funkcionality uváděné na oficiální stránce Kamailia			
Funkcionalita	**Asterisk**	**Kamailio**	**Popis**
Accounting	Ano	Ano	Účtování na základě události
Benchmark	Ne	Ano	Nástroje pro výkonové testování
CLI	Ano	Ano	Podpora uživatelského rozhraní v podobě příkazového řádku
CPL – Call Processing Language	Ne	Ano	Podpora kombinovaného programovacího jazyka CPL
Call blocking	Ne	Ano	Funkce pro omezení provozu v případě přetížení
Call limitation	Ano	Ano	Možnost omezit hovory na určitý počet nebo na určitou délku dle parametrů hovorů
DNSsec support	Ano	Ano	Podpora DNSsec

Dialplan	Ano	Ano	Nástroj pro práci s hovory. Dle parametrů příchozích hovorů a nastavených pravidel.
Diameter support	Ne	Ano	Podpora AAA protokolu diameter
Erlang Programming lang.	Ne	Ano	Podpora programovacího jazyka Erlang
Flexible debug and error message logging system	Ano	Ano	Možnost debugování a zaznamenávání chybových hlášek
Flexible least cost routing	Ne	Ano	Směrování na základě nejlepší ceny spoje
Forking/multiple registration	Ano	Ano	Možnost registrace jednoho uživatele u více zařízení
Gateway to sms or xmpp and other IM services	Ano	Ano	Možnost využití jako brány pro služby okamžitého zasílání zpráv IM (instant messaging)
IPv4 and IPv6	Ano	Ano	Podpora protokolů IPv4 a IPv6
JSON implementation	Ano	Ano	Možnost získání dat SS7 a převod na objektový zápis – JSON
Java programming language	Ano	Ano	Podpora programovacího jazyka Java
JavaScript programming language	Ne	Ano	Podpora programovacího jazyka JavaScript
LDAP integration	Ano	Ano	Podpora protokolu LDAP
Load balancing	Ne	Ano	Vyvažování zátěže
Location server	Ano	Ano	Funkce lokačního serveru
Lua programming language	Ano	Ano	Podpora programovacího jazyka Lua
Managed Code (C#) programming language	Ne	Ano	Podpora programovacího jazyka C#
Memcached connector	Ne	Ano	Podpora Memcached pro efektivní práci s pamětí
Message body compression/decompression	Ano	Ano	Komprese a dekomprese těla zprávy využitím kodeků
Modular architecture	Ano	Ano	Možnost rozšíření funkcí pomocí modulů
NAT traversal	Ano	Ano	Možnost průchodu skrze NAT
Offline messaging	Ano	Ano	Umožňuje přijímat SMS i klientům, kteří jsou offline a ukládat je do databáze
Over 1000 parameters, variables and functions exported to config file	Ne	Ano	Velké množství funkcí, proměnných a parametrů pro konfiguraci chování
Perl programming language	Ano	Ano	Podpora programovacího jazyka Perl
Plug&play modules	Ano	Ano	Jednoduchost zavádění nových modulů
Presence	Ano	Ano	Funkce pro dohled nad přítomností uživatelů a reakce na různé události
Proxy server	Ano	Ano	Funkce proxy server
Python programming language	Ano	Ano	Podpora programovacího jazyka Python
QOS	Ano	Ano	Umožňuje sledovat dialog a poskytovat různé úrovně QoS

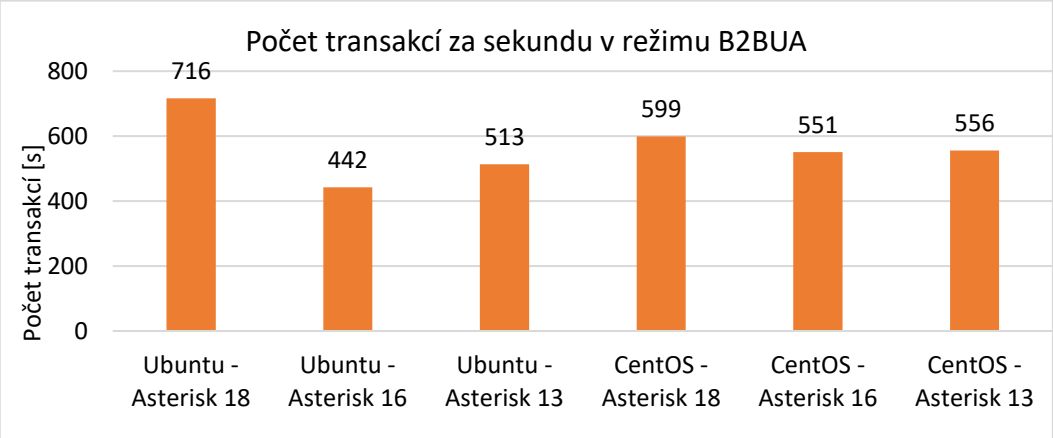
RPC control interface – via XMLRPC, JSONRPC, UDP or TCP	Ne	Ano	Remote Procedure Call – rozhraní pro připojení externích aplikací
Radius support	Ano	Ano	Podpora AAA protokolu Radius
Redirect server	Ano	Ano	Funkce serveru přesměrování
Registrar server	Ano	Ano	Funkce registrar serveru
Remote control via XMLRPC	Ne	Ano	Možnost vzdáleného přístupu a volání procedur pomocí XMLRPC
Repository	Ano	Ano	Repozitáře se zdrojovými kódy a přídavným obsahem jako jsou repozitáře
Resource monitoring	Ne	Ano	Nástroje pro sledování zatížení
Routing failover	Ne	Ano	Využití záložní cesty při výpadku
Ruby Programming Language	Ano	Ano	Rozhraní pro programovací jazyk Ruby
Runtime update framework	Ne	Ano	Možnost upravovat parametry v provozu bez nutnosti restartu
SCTP multi–homing and multi–streaming	Ano	Ano	Podpora multihoamingu a multistreamingu
SIP Application server	Ne	Ano	Funkce aplikačního serveru
SNMP – interface to Simple Network Management Protocol	Ano	Ano	Rozhraní pro správu pomocí protokolu SNMP
Small footprint	Ano	Ano	Nízká paměťová náročnost
Straightforward interconnection with PSTN gateways	Ano	Ano	Možnost využití jako brány pro veřejnou telefonní síť
Suport of databases with have unixodbc drivers	Ne	Ano	Podpora databází
Support for SRV and NAPTR DNS lookups	Ano	Ano	Funkce pro zjišťování upřesňujících informací z SRV a NAPTZ záznamů
Support of API	Ano	Ano	Využití aplikačního programovacího rozhraní
Support of UDP, TCP, TLS and SCTP	Ano	Ano	Podpora komunikačních protokolů UDP, TCP, TLS a SCTP
Support of scripts	Ano	Ano	Podpora skriptů a jejich možné využití pro konfigurační soubory
TLS support	Ano	Ano	Podpora TLS pro signalizaci
Web Management Interface: Siremis	Ne	Ano	Nástroj pro správu Siremis
Whitelist	Ano	Ano	Seznam hovorů, které mají být povoleny
XCAP client capabilities	Ne	Ano	Možnost modifikace konfiguračních dat uložených v XML souborech pomocí XML Configuration Access Protocol

Přehled nastavení testovacích scénářů

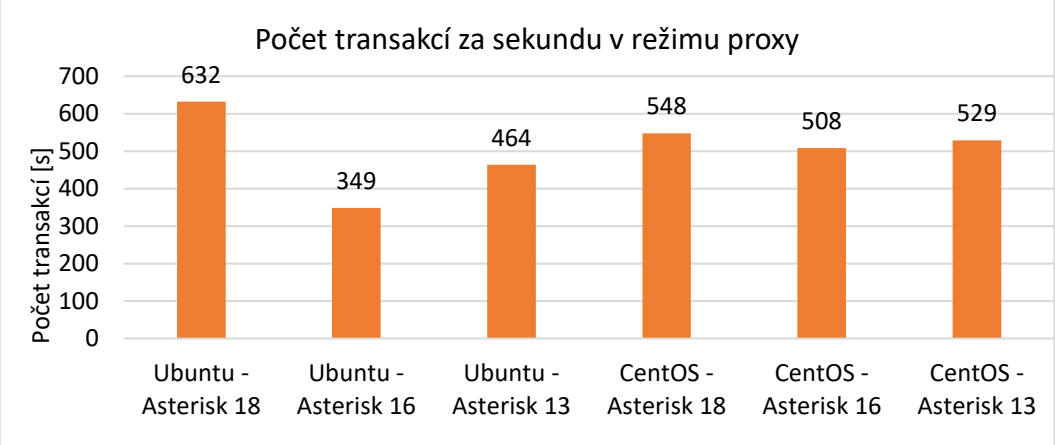
Přehled nastavení testovacích scénářů		
	Test na maximální počet transakcí za sekundu	Test na maximální počet hovorů
	Nastavení klientské části	
Load		
Specification	Simusers/sec	Simusers
Default Time Scale	Seconds	
Steady Time (nulová hodnota)	60 sec.	
Ramp Time (nárůst)	400 sec.	
Height (liší se dle ústředny)	V řádu stovek	V řádu tisíců
Steady Time (na maximu)	50 sec.	
Ramp Time (sestup)	100 sec.	
Steady Time (nulová hodnota)	350 sec.	
Actions		
IP adress	192.168.20.199	
Port	SIP – 5060, PJSIP – 5066	
Caller	{1000–1999}@192.168.20.199	
Callee	{2000–2999}@192.168.20.199	
Media	M_SIP_RTP_TEST	
Transport type	UDP	
Call length min	1	900000 ms
Call length max	1	900000 ms
Registrar IP adress	192.168.20.199	
Registrar Port	SIP – 5060, PJSIP – 5066	
Password	1234	
Profiles		
Protocol	SIPNG	
First RTP port	10000	
Unregister All Before Register	YES	
Media Properties		
Content	voice_10_sec.wav	
Codecs	G.711a	
Subnet		
Enable DHCP	YES	
Ports		
Port	192.168.10.158:1,1	
	Nastavení serverové části	
Profiles		
Type	SIPNG_Endpoint	
Port	SIP – 5060, PJSIP – 5066	
Endpoint SIP URI	3000@192.168.20.199	

Transport Layer	UDP	
Probability of Busy Calls	0	
Off Hook Time	0	
Call Holding Time	1ms – 1ms	900000 ms – 900000 ms
Registrar IP	192.168.20.199	
Registrar Port	SIP – 5060, PJSIP – 5066	
Password	1234	
Unregister All Before Register	YES	
Media Properties		
Content	voice_10_sec.wav	
Codecs	G.711a	
Ports		
Port	192.168.10.158:1,1	
	Nastavení sběru statistik	
Statistic Mode	Brief	
Statistic Sampling Interval (sec)	10	
Enable Logs Level	5 (Exception)	
Enable Per–Protocol and Per–Packet Size Statistics	YES	
Enable Client Packet Trace (Client)	YES	
Enable Client Packet Trace (Server)	YES	

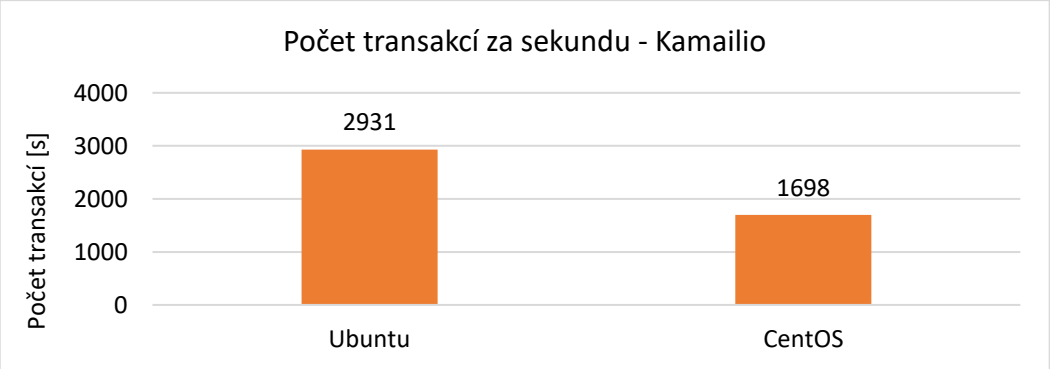
Příloha 3: Výsledky měření



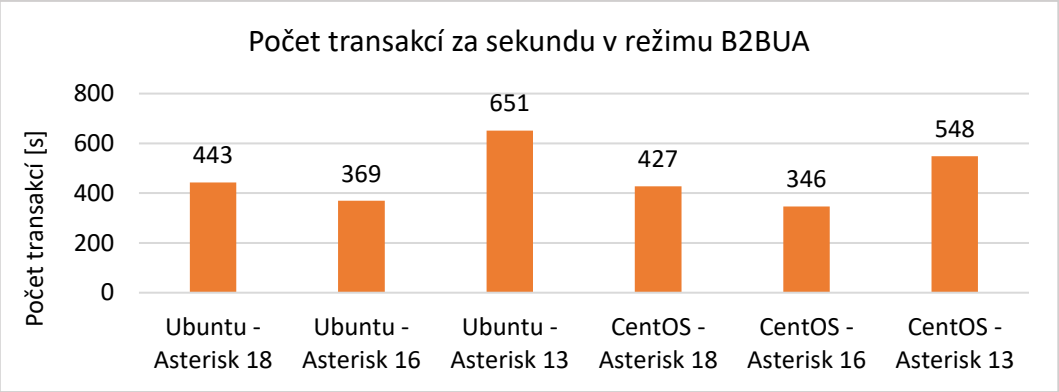
Počet transakcí za sekundu (SIP/B2BUA/2 GB)



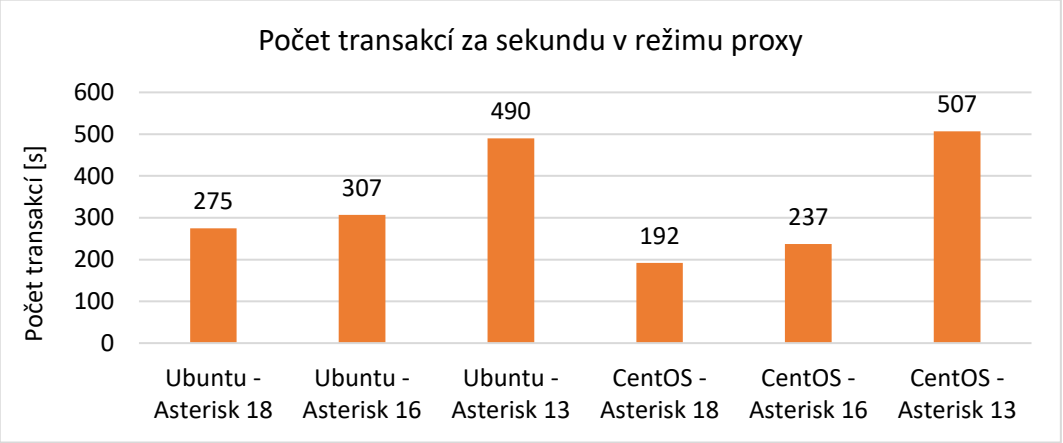
Počet transakcí za sekundu (SIP/PROXY/2 GB)



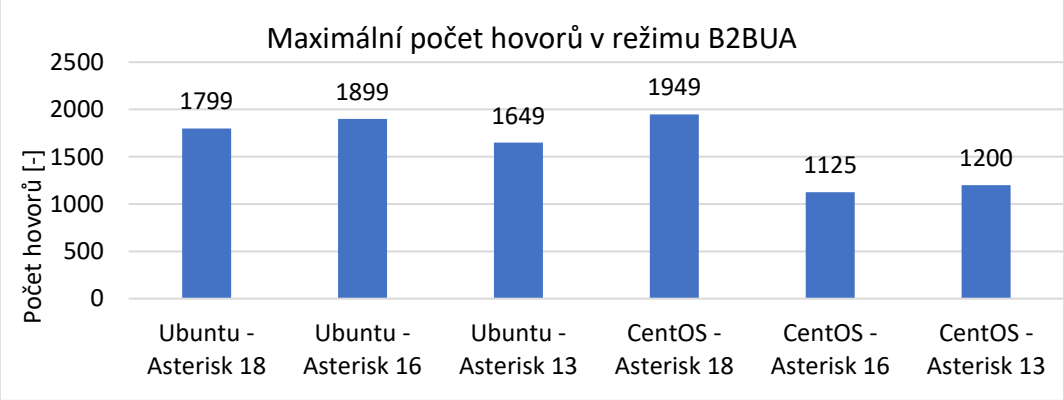
Počet transakcí za sekundu (Kamailio/2 GB)



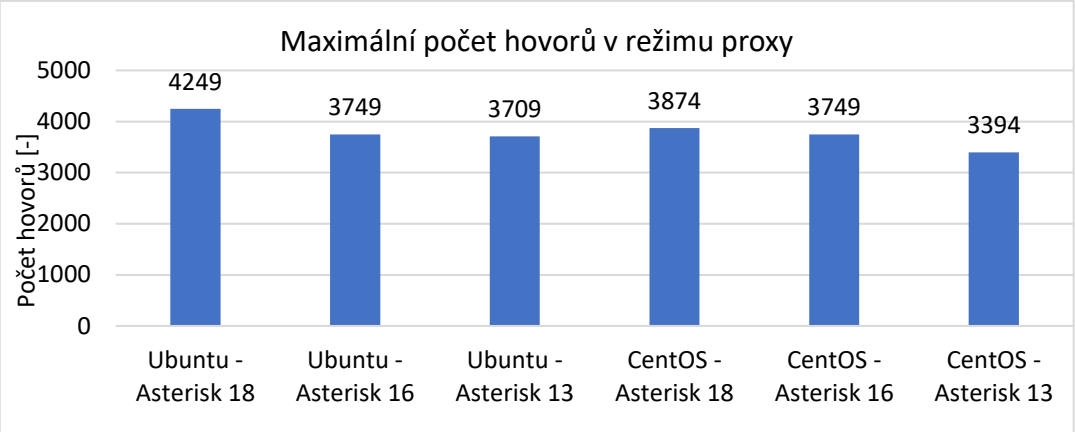
Počet transakcí za sekundu (PJSIP/B2BUA/2 GB)



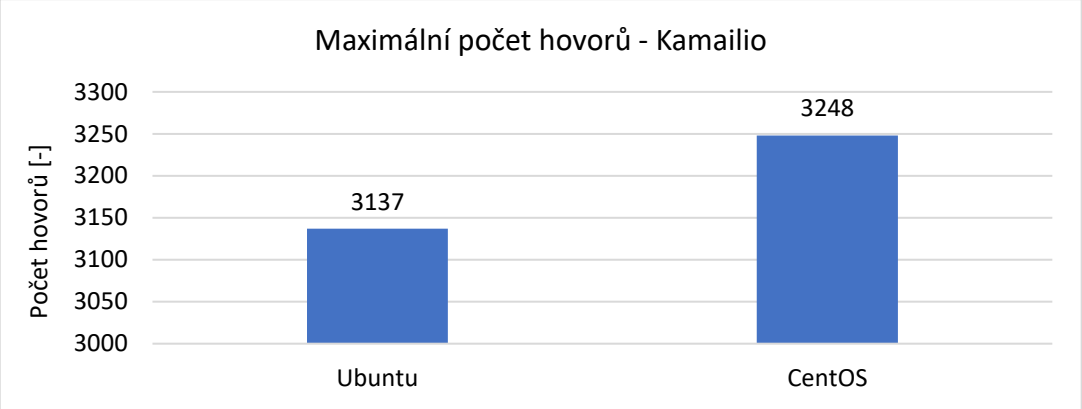
Počet transakcí za sekundu (PJSIP/PROXY/2 GB)



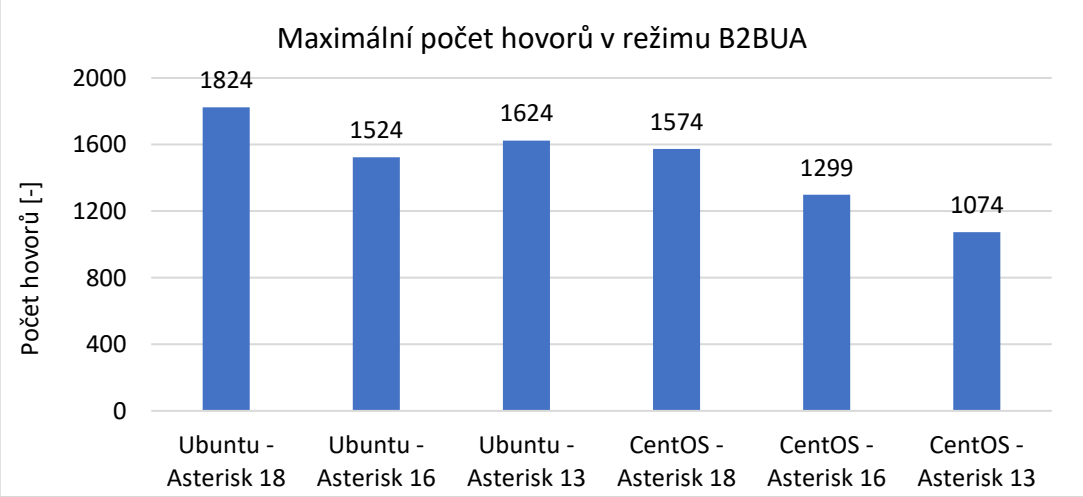
Počet hovorů (SIP/B2BUA/2 GB)



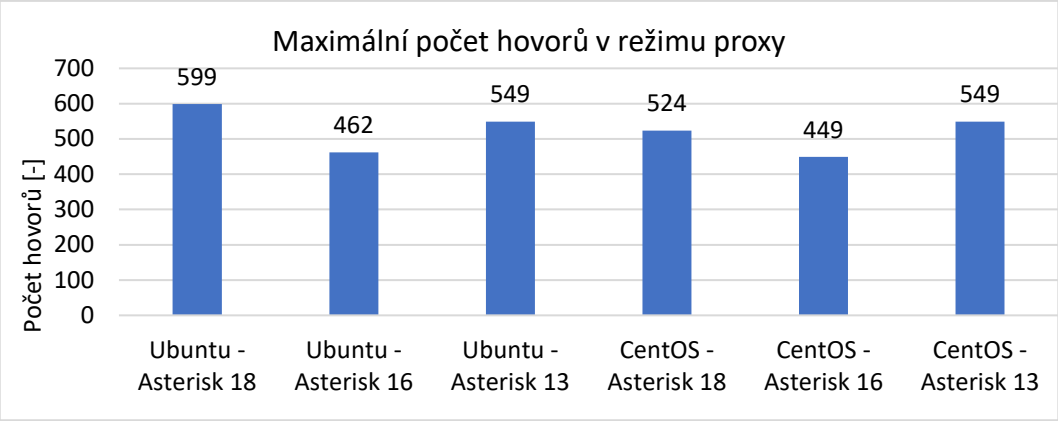
Počet hovorů (SIP/PROXY/2 GB)



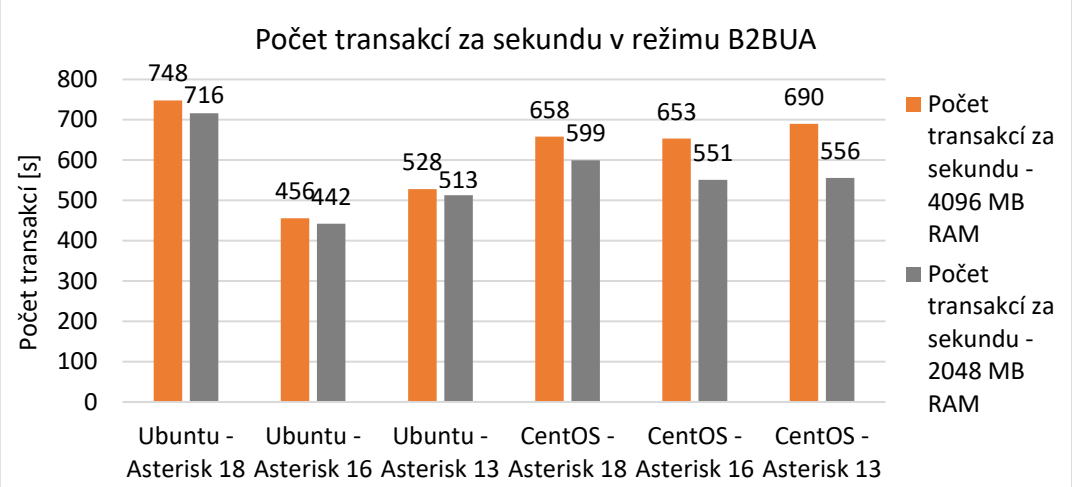
Maximální počet hovorů (Kamailio/2 GB)



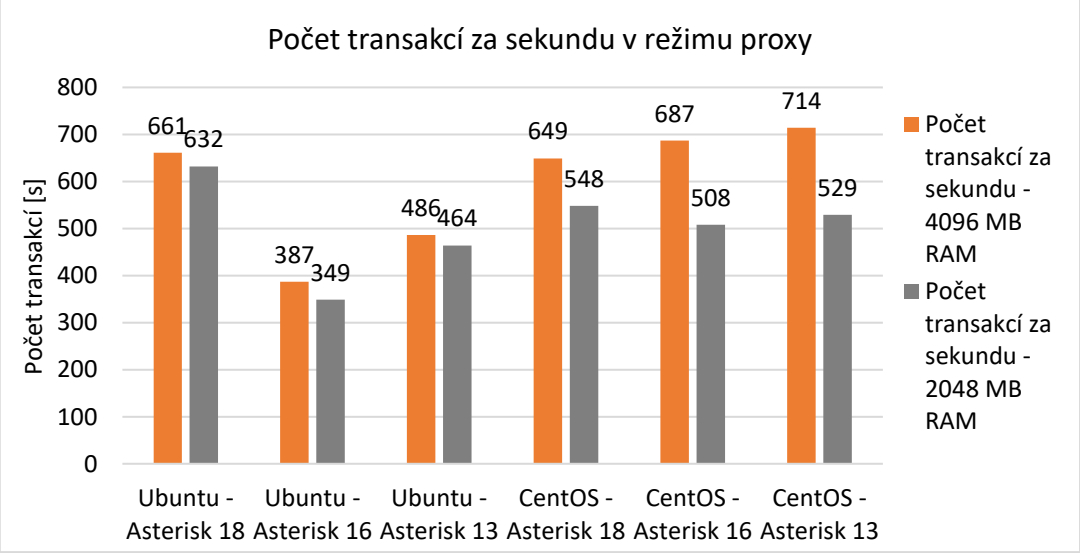
Počet hovorů (PJSIP/B2BUA/2 GB)



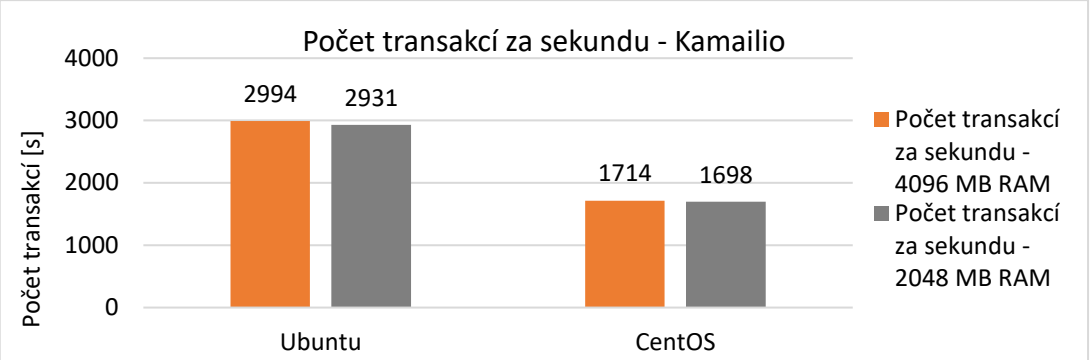
Počet hovorů (PJSIP/PROXY/2 GB)



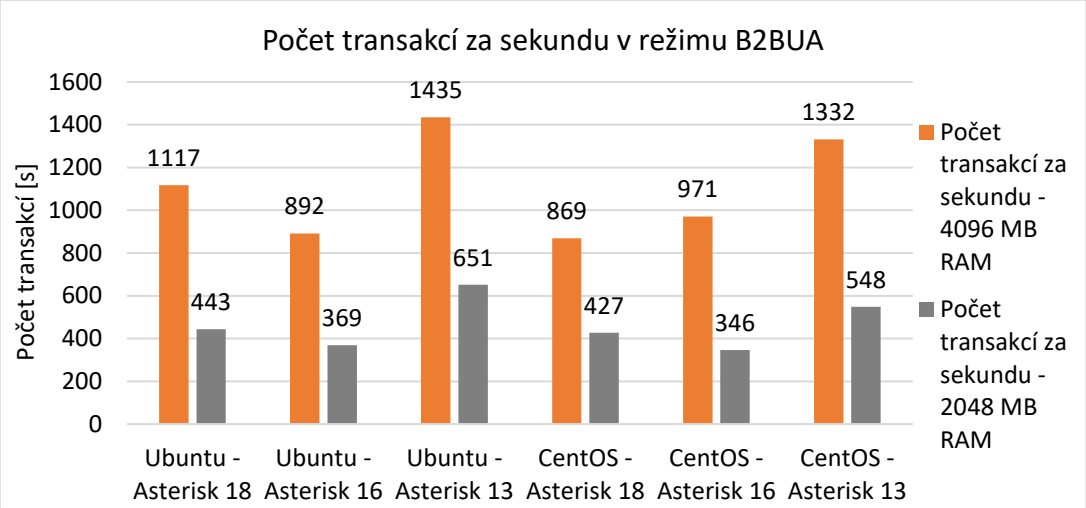
Počet transakcí za sekundu (SIP/B2BUA/4 GB)



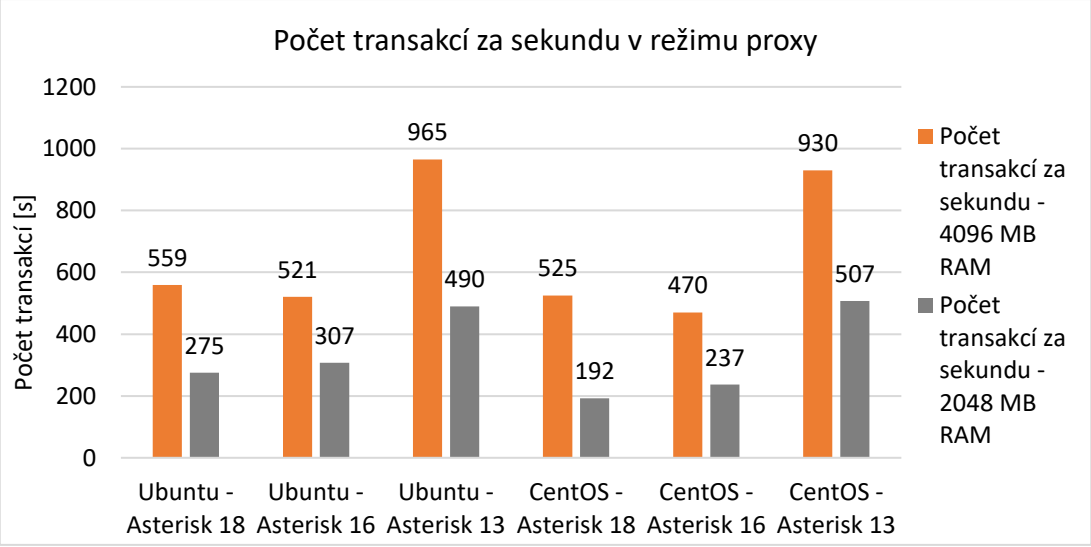
Počet transakcí za sekundu (SIP/PROXY/4 GB)



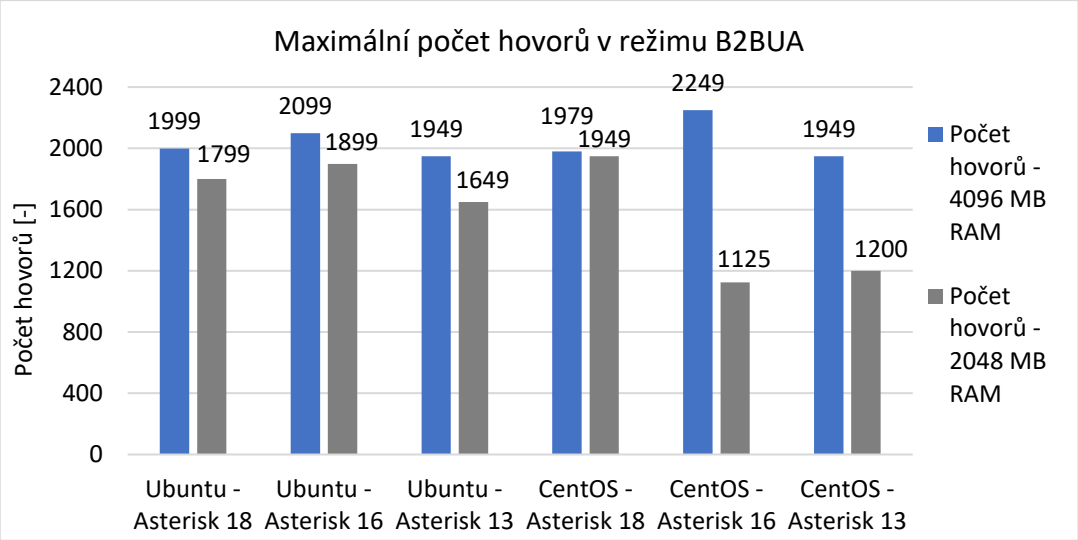
Počet transakcí za sekundu (Kamailio/4 GB)



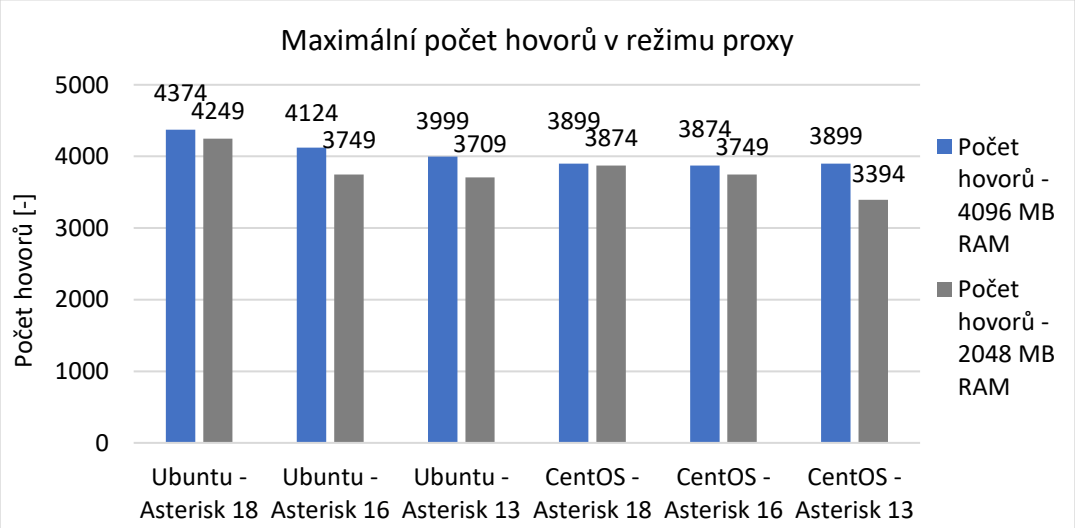
Počet transakcí za sekundu (PJSIP/B2BUA/4 GB)



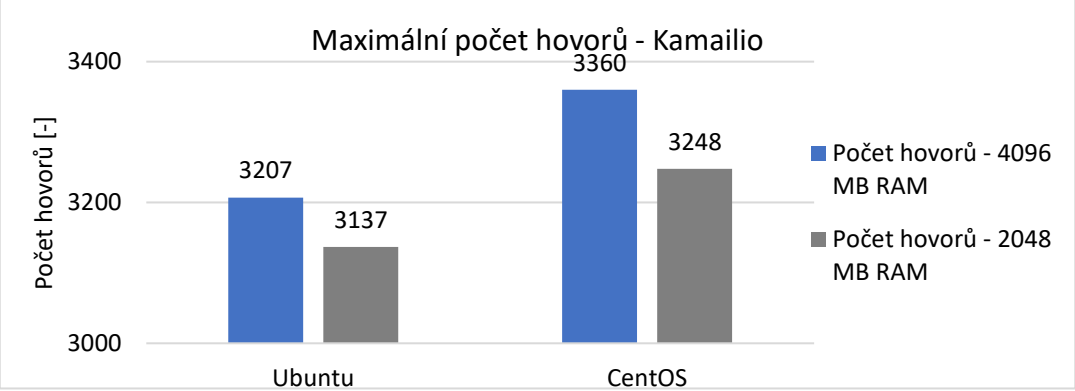
Počet transakcí za sekundu (PJSIP/PROXY/4 GB)



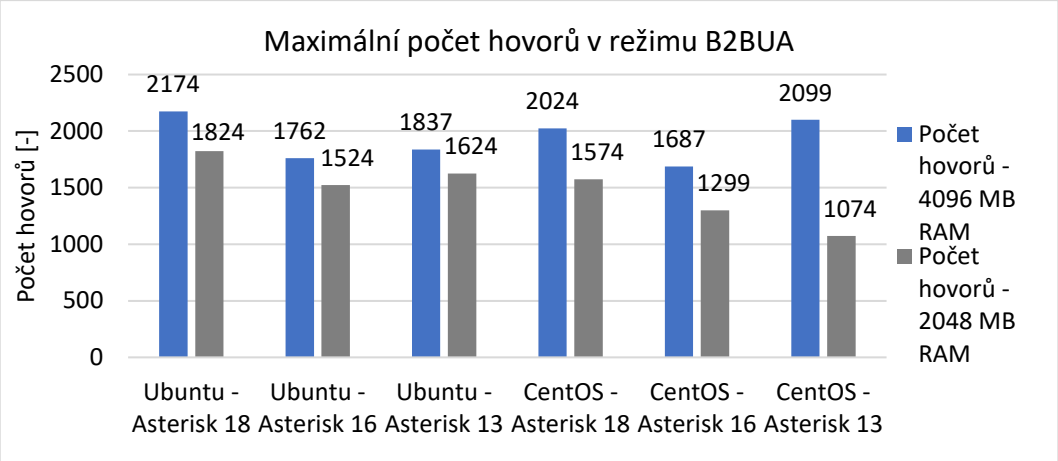
Počet hovorů (SIP/B2BUA/4 GB)



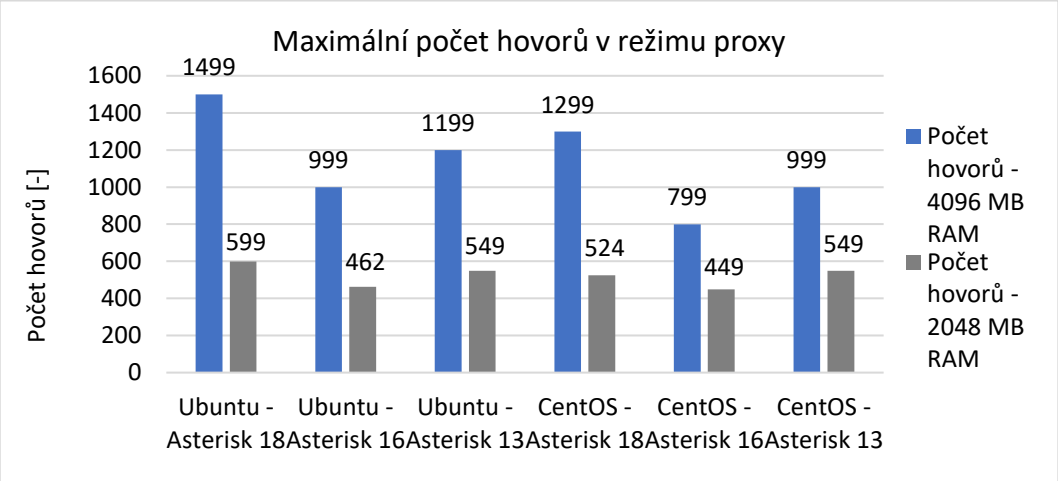
Počet hovorů (SIP/PROXY/4 GB)



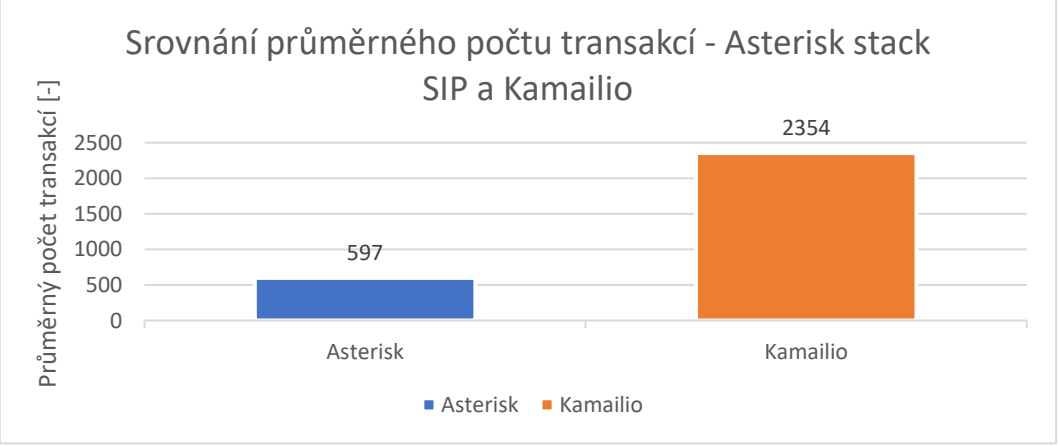
Maximální počet hovorů (Kamailio/4 GB)



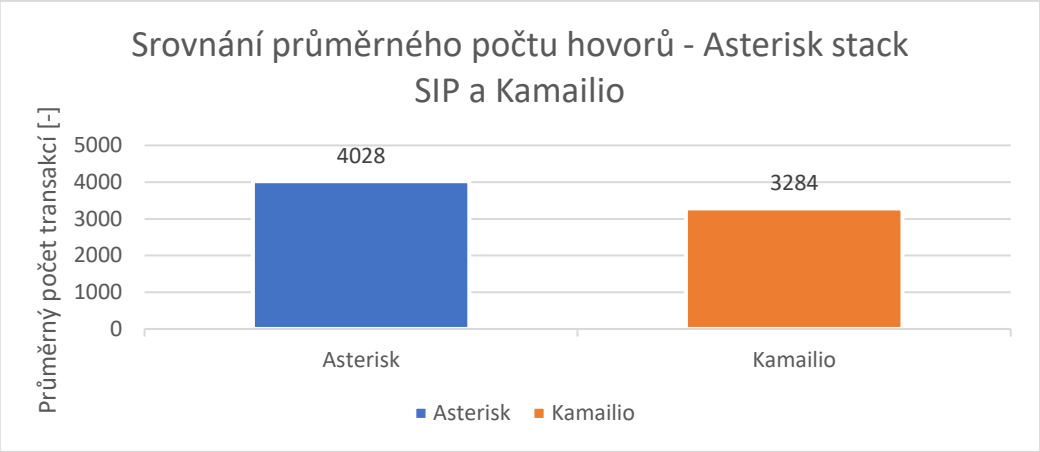
Počet hovorů (PJSIP/B2BUA/4 GB)



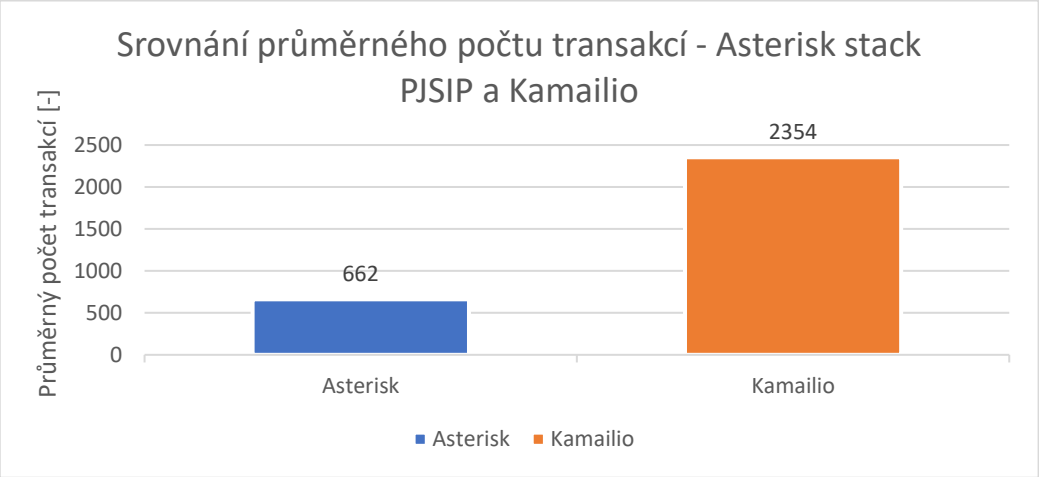
Počet hovorů (PJSIP/PROXY/4 GB)



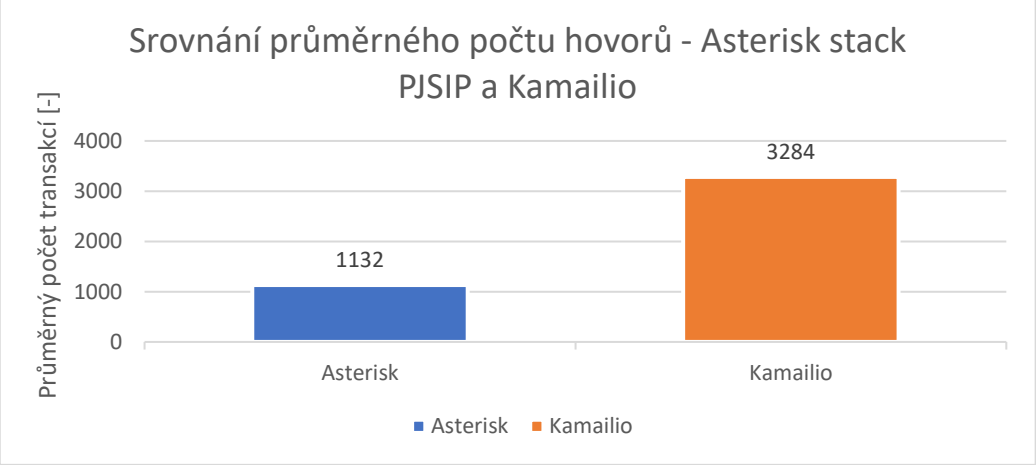
Srovnání průměrného počtu transakcí – Asterisk stack SIP a Kamailio



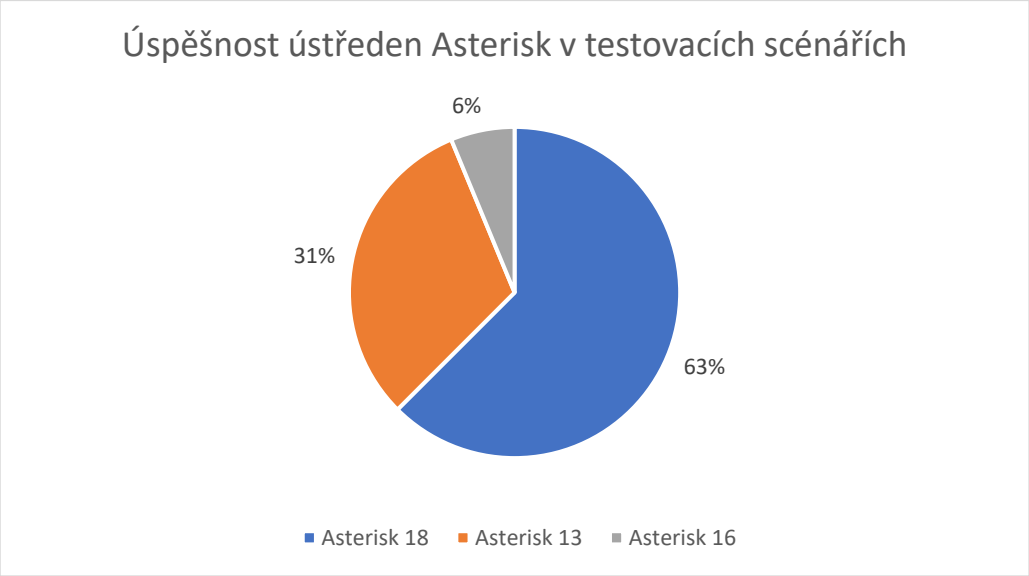
Srovnání průměrného počtu hovorů – Asterisk stack SIP a Kamailio



Srovnání průměrného počtu transakcí – Asterisk stack PJSIP a Kamailio



Srovnání průměrného počtu hovorů – Asterisk stack PJSIP a Kamailio



Úspěšnost ústředen Asterisk

Srovnání použitých OS – nižší paměť RAM

Asterisk – nižší paměť RAM				
Průměrný počet transakcí za sekundu v režimu B2BUA a proxy režimu (stack SIP)				
Transakce – B2BUA	1126		Transakce – proxy	1010
Ubuntu [%]	49,5		Ubuntu [%]	47,7
CentOS [%]	50,5		CentOS [%]	52,3
Průměrný počet transakcí za sekundu v režimu B2BUA a proxy režimu (stack PJSIP)				
Transakce – B2BUA	928		Transakce – proxy	669
Ubuntu [%]	52,6		Ubuntu [%]	53,4
CentOS [%]	47,4		CentOS [%]	46,6
Průměrný počet hovorů v režimu B2BUA (stack SIP)				
Hovory – B2BUA	3207		Hovory – proxy	7575
Ubuntu [%]	55,6		Ubuntu [%]	51,5
CentOS [%]	44,4		CentOS [%]	48,5
Průměrný počet hovorů v režimu B2BUA (stack PJSIP)				
Hovory – B2BUA	2973		Hovory – proxy	1044
Ubuntu [%]	55,7		Ubuntu [%]	51,4
CentOS [%]	44,3		CentOS [%]	48,6

Srovnání použitých OS – vyšší paměť RAM

Asterisk – vyšší paměť RAM				
Průměrný počet transakcí za sekundu v režimu B2BUA a proxy režimu (stack SIP)				
Transakce – B2BUA	1244		Transakce – proxy	1195
Ubuntu [%]	46,4		Ubuntu [%]	42,8
CentOS [%]	53,6		CentOS [%]	57,2
Počet transakcí za sekundu v režimu B2BUA a proxy režimu (stack PJSIP)				
Transakce – B2BUA	2205		Transakce – proxy	1323
Ubuntu [%]	52,1		Ubuntu [%]	51,5
CentOS [%]	47,9		CentOS [%]	48,5
Maximální počet hovorů v režimu B2BUA (stack SIP)				
Hovory – B2BUA	4075		Hovory – proxy	8056
Ubuntu [%]	49,5		Ubuntu [%]	51,7
CentOS [%]	50,5		CentOS [%]	48,3
Maximální počet hovorů v proxy režimu (stack PJSIP)				
Hovory – B2BUA	3861		Hovory – proxy	2265
Ubuntu [%]	49,8		Ubuntu [%]	54,4
CentOS [%]	50,2		CentOS [%]	45,6

Srovnání použitých OS – Kamailio

Kamailio – nižší paměť RAM			Kamailio – vyšší paměť RAM	
Počet transakcí za sekundu	4629		Počet transakcí za sekundu	4708
Ubuntu [%]	63,3		Ubuntu [%]	63,6
CentOS [%]	36,7		CentOS [%]	36,4
Počet hovorů	6385		Počet hovorů	6567
Ubuntu [%]	49,1		Ubuntu [%]	48,8
CentOS [%]	50,9		CentOS [%]	51,2

Průměrný počet transakcí a hovorů – nižší paměť

Průměrný počet transakcí za sekundu v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack SIP				
Ubuntu	557		Ubuntu	482
CentOS	569		CentOS	528
Průměrný počet hovorů v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack SIP				
Ubuntu	1782		Ubuntu	3902
CentOS	1425		CentOS	3672

Průměrný počet transakcí za sekundu v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack PJSIP				
Ubuntu	488		Ubuntu	357
CentOS	440		CentOS	312
Průměrný počet hovorů v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack PJSIP				
Ubuntu	1657		Ubuntu	537
CentOS	1316		CentOS	507

Celkové a procentuální vyjádření výsledků pro nativní stack SIP (vlevo) a implementaci stacku PJSIP (vpravo) – nižší paměť

Transakcí celkem – SIP	2136		Transakcí celkem – PJSIP	1597
Transakcí procentuálně	57,2		Transakcí procentuálně	42,8
Hovorů celkem – SIP	10782		Hovorů celkem – PJSIP	4017
Hovorů procentuálně	72,9		Hovorů procentuálně	27,1

Průměrný počet transakcí a hovorů – vyšší paměť

Průměrný počet transakcí za sekundu v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack SIP				
Ubuntu	577		Ubuntu	511
CentOS	667		CentOS	683
Průměrný počet hovorů v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack SIP				
Ubuntu	2016		Ubuntu	4166
CentOS	2059		CentOS	3891

Průměrný počet transakcí za sekundu v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack PJSIP				
Ubuntu	1148		Ubuntu	682
CentOS	1057		CentOS	642
Průměrný počet hovorů v režimu B2BUA (vlevo) a proxy režimu (vpravo) – stack PJSIP				
Ubuntu	1924		Ubuntu	1232
CentOS	1937		CentOS	1032

Celkové a procentuální vyjádření výsledků pro nativní stack SIP a implementaci stacku PJSIP – vyšší paměť

Transakcí celkem – SIP	2439	Nárůst transakcí po zvětšení paměti [%]	12,44		Transakcí celkem – PJSIP	3529	Nárůst transakcí po zvětšení paměti [%]	54,73
Transakcí procentuálně	40,9				Transakcí procentuálně	59,1		
Hovorů celkem – SIP	12131	Nárůst hovorů po zvětšení paměti [%]	11,12		Hovorů celkem – PJSIP	6126	Nárůst hovorů po zvětšení paměti [%]	34,42
Hovorů procentuálně	66,4				Hovorů procentuálně	33,6		

Celkové a procentuální vyjádření výsledků pro režimy B2BUA a proxy – nižší paměť

Transakce v režimu B2BUA	2054	Transakce v proxy režimu	1679
Procentuální zastoupení transakcí	55,0	Procentuální zastoupení transakcí	45,0
Hovory v režimu B2BUA	6180	Hovory v proxy režimu	8619
Procentuální zastoupení hovorů	41,8	Procentuální zastoupení hovorů	58,2

Celkové a procentuální vyjádření výsledků pro režimy B2BUA a proxy – vyšší paměť

Transakce v režimu B2BUA	3450	Nárůst transakcí [%]	40,47		Transakce v proxy režimu	2518	Nárůst transakcí [%]	33,31
Procentuální zastoupení transakcí	57,8				Procentuální zastoupení transakcí	42,2		
Hovory v režimu B2BUA	7936	Nárůst hovorů [%]	22,12		Hovory v proxy režimu	10321	Nárůst hovorů [%]	16,49
Procentuální zastoupení hovorů	43,5				Procentuální zastoupení hovorů	56,5		