VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

OPEN SOURCE PBX KAMAILO A OPENSIPS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

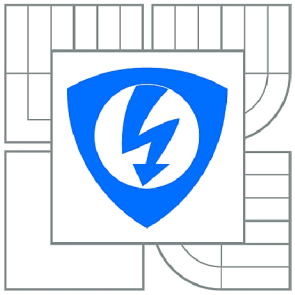
AUTOR PRÁCE
AUTHOR

Bc. VÁCLAV JANEČEK

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

OPEN SOURCE PBX KAMAILO A OPENSIPS

KAMAILIO AND OPENSIPS OPEN SOURCE PBX

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

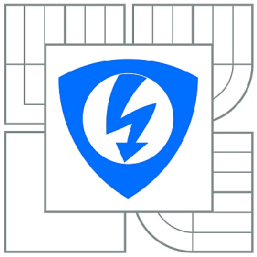
AUTOR PRÁCE — Bc. VÁCLAV JANEČEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVEL ŠILHAVÝ, Ph.D.

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Václav Janeček	***ID:***	156467
Ročník:	2	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Open source PBX Kamailo a OpenSIPs

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte možnosti Open Source pobočkových ústředen Kamailo a OpenSIPs. Realizujte instalace a konfigurace těchto PBX na linuxové distribuci Ubuntu. Text vlastní práce bude obsahovat podrobné návody instalace a konfigurace. Realizujte testy umožňující srovnání tohoto open source řešení s PBX Asterisk. Zaměřte se na výkonnostní srovnaní, paměťovou náročnost a vytížení dalších prostředků operačního systému. Na základě vlastních testů proveďte podrobné porovnání tohoto open source řešení s PBX Asterisk.

DOPORUČENÁ LITERATURA:

[1] Goncalves, Flavio E. Building Telephony Systems With OpenSER: A Step-by-step Guide to Building a High-performance Telephony System. Birmingham: Packt, 2008. ISBN: 978-1849510745.
[2] Meggelen, J.V, Smith, J., Madsen, L. Asterisk™ The Future of Telephony. Sevastopol: O'Reilly Media, Inc., 2005. ISBN 0-596-00962-3.
[3] Bosse, J.G.. Signaling in telecommunication networks. John Wiley & Sons, Ltd. En-gland 2002 , ISBN 0-471-66288-7.

Termín zadání:	10.2.2014	***Termín odevzdání:***	28.5.2014

Vedoucí práce: Ing. Pavel Šilhavý, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Diplomová práce open source PBX Kamailio a OpenSIPS se zabývá seznámením se jmenovanými SIP ústřednami a jejich výkonovým porovnáním. Cílem práce je i detailní návod instalace na operačním systému Ubuntu. Práce obsahuje historický vývoj telefonních ústředen se zaměřením na poslední generaci. Dále je uveden základní popis protokolu SIP a jsou popsány komponenty, ze kterých může být SIP ústředna složena. Další část textu je věnována vývoji ústředen Kamailio a OpenSIPS. Je popsána jejich architektura a popis konfiguračního souboru. Praktická část diplomové práce se věnuje výkonovému porovnání ústředen a to z pohledu paměťových a výpočetních nároků. U vybraných měření jsou ústředny porovnány s ústřednou Asterisk.

KLÍČOVÁ SLOVA

Kamailio, OpenSIPS, SIP Express Router, pobočková ústředna, SIP, VoIP

ABSTRACT

Open source PBX Kamailio and OpenSIPS diploma thesis covers familiarization with appointed SIP exchanges and with their power comparing. A detailed installation instructions on the operating system Ubuntu is the aim of this work too. The work includes the historical development of telephone exchanges with a focus on the latest generation. The following is SIP protocol basic description and components that can be composed SIP exchanges. Another part is devoted to the development of exchanges Kamailio and OpenSIPS. The thesis contain the archutecture and configuration file description. The practical part of the thesis deals with high-capacity switches, and comparing it in terms of memory and computational demands. Selected measurements are compared with the Asterisk PBX.

KEYWORDS

Kamailio, OpenSIPS, SIP Express Router, PBX, SIP, VoIP

JANEČEK, V. *Open Source PBX Kamailo a OpenSIPS*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2014. 58 s., 27 s. příloh. Diplomová práce. Vedoucí práce: Ing. Pavel Šilhavý, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma Open source PBX Kamailo a OpenSIPS 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/nebo majetkových a~jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, 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

Děkuji vedoucímu diplomové práce Ing. Pavlu Šilhavému, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé semestrální práce.

V Brně dne ....

(podpis autora)

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Purkynova 118, CZ-61200 Brno, Czechia

http://www.six.feec.vutbr.cz

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V poslední době, díky obrovskému rozmachu internetu a konvergenci datových sítí, se u hlasové komunikace stále častěji využívá technologie Voice over Internet Procotol (VoIP), která funguje na principu přepojování paketů. Mezi jeden z hlavních protokolů zajišťující VoIP komunikaci patří Session Initiation Procol (SIP).

Díky digitalizaci hlasového signálu, dostupnosti sítí založených na přepojování paketů, cenově dostupných počítačů se stále narůstajícím výpočetním výkonem, ale hlavně možnosti oddělení signalizačních informací od datového toku kromě proprietárních telefonních ústředen začali vznikat i open source software. Taková řešení lze rozdělit na 2 skupiny.

Pobočkové ústředny, které slouží k hlavně k připojení koncových uživatelů do telefonní domény a dále pro uživatele nabízí přídavné služby, jako funkci spojovatelky a podobně. K takovým ústřednám lze připojit účastníky používající různé komunikační protokoly a to ať už se jedná o digitální, jako jsou SIP nebo IAX, tak po zakoupení potřebného přídavného zařízení i účastníky využívající analogové telefonní přístroje. Mezi typické představitele pobočkových open source ústředen patří Asterisk, nebo YATE.

Do druhé skupiny se počítají ústředny, jež jsou určeny ke směrování v páteři telefonní sítě. Z důvodu co největší rychlosti směrování navíc nemají aplikovány takovou škálu služeb, jako nabízí ústředny pobočkové. Do takové skupiny ústředen se řadí Kamailio a OpenSIPS, jež jsou určeny pro směrování signalizace protokolu SIP.

Diplomová práce, jež si klade za úkol základní představení a porovnání ústředen Kamailio a OpenSIPS je rozdělena do 4 hlavních částí.

První část je věnována úvodu do problematiky, kde je stručně popsána historie vývoje telefonních ústředen. U protokolu SIP jsou popsány jeho hlavní funkce a nejčastěji používané zprávy. Nakonec jsou rozděleny a popsány komponenty SIP ústředen, které se nejčastěji využívají.

Druhá část práce se věnuje základnímu představení ústředen Kamailio a OpenSIPS, kde na začátku je popsána jejich historie. Dále jsou popsány samotné ústředny, se zaměřením na jejich architekturu. Nakonec je provedené základní porovnání a vysvětlení rozdílů ústředen Kamailio a OpenSIPS s pobočkou ústřednou Asterisk.

Ve třetí části lze najít podrobný popis instalací a základní sprovoznění ústředen, jež byla provedena v operačním systému Ubuntu 12.04 LTS. Kromě instalace samotných ústředen a potřebných komponent jako MySQL databáze, je popsáno i vytvoření a implementace certifikátu a klíčů potřebných pro navázání zabezpečené komunikace pomocí implementace openssl.

Poslední část diplomové práce je věnována praktickému porovnání ústředen, kde je postupně porovnána paměťová náročnost lokačního serveru se zaměřením na funkci nastavení sdílené paměti a zápisu do MySQL databáze a velikost využité paměti při sestavování hovorů. Dále je porovnávan výkon ústředen při generování SIP odpovědí a sestavování hovorů. Také je porovnána rychlost odezvy na požadavky a konec je

věnován ochraně proti útokům, kde je i výkonové porovnání zabezpečené komunikace pomocí TLS spojení.

1 ÚVOD DO PROBLEMATIKY

1.1 Historický vývoj telefonních ústředen

Od roku 1878 kdy ve státě Connecticut (USA) byla uvedena do provozu první telefonní ústředna [1] prošly ústředny několika radikálními změnami. Ústředny pak lze rozdělit do několika generací:

- **0. generace** – Jednalo se ústředny s manuálním přepínáním analogových hovorů (spojovatelka), kdy na jednom vedení probíhal jeden hovor.
- **1. generace** – Přepínání analogových hovorů probíhalo již pomocí elektromechanických voličů. Pomocí fantomového vedení snaha o vedení více hovorů po stejném vedení. Signalizace probíhala pomocí pulzů.
- **2. generace** – Díky technologii Frequency Division Multiplex (FDM) šlo na jednom vedení provozovat vice hovorů, kdy signalizace byla realizována nad hovorovým pásmem pomocí frekvenčních značek.
- **3. generace** – Jednalo se o poloekeltronické ústředny, které využívaly FDM systémy vyšších řádů a dokázali spolupracovat s ústřednami 2. generace.
- **4. generace** – Již plně digitalizované ústředny pracující s časovým přepínáním digitálních hovorů, technologií Time Division Multipex (TDM). Transportní síť již nerozlišovala, jestli přenášený signál je hovorový, nebo signalizační.
- **5. generace** – Ústředny, jež realizují spojení hovorů pomocí VoIP technologie pracují na principu přepojování paketů. Ústředny mohou zpracovávat pouze signalizaci a samotný hovor přes ně nemusí procházet.

Ústředny až do 5. generace, kdy fungovaly na principu přepojování okruhů, musely obsahovat spojovací pole, přes které se hovory spojovaly. Díky tomu existovala pouze proprietární řešení od velkých výrobců. Od nástupu poslední generace a VoIP technologie již fyzické spojovací pole není nutné. Díky tomuto faktu a i dalším výhodám spojených s přepojováním paketů, kromě proprietárních ústředen začal vznikat i počítačový software, jež dokázal vše realizovat i na běžně dostupných zařízeních. Takovýmto řešením se začalo říkat softwarová ústředna.

Softwarová ústředna (angl. Softwsitch) existuje i jako volně dostupné open-source řešení. K přepínání hovorů nepotřebuje spojovací pole, ale pro zlepšení výkonu lze dokoupit specifické hardwarové karty, stejně jako lze dokoupit karty pro připojení do sítě TDM a podobně.

V rámci VoIP technologie se hovorová data mezi účastníky přenáší pomocí nezabezpečeného protokolu Real-time Transport Protocol (RTP), nebo jeho zabezpečené variantě Secure RTP. Signalizačních protokolů určených pro řízení VoIP spojení existuje více variant. Kromě rodiny protokolů H.323, která je nyní na ústupu, protokolu Inter-Asterisk eXchange (IAX), jež je oblíbený díky rozšířenosti ústředny Asterisk a dalších zejména proprietárních protokolů, je dominantním využívaným signalizační protokolem Session Initiation Protocol (SIP).

Díky takové rozmanitosti existují ústředny, které jsou schopné pracovat s větším počtem signalizačních protokolů a jsou nasazovány většinou jako pobočkové ústředny. Zatím co ústředny pracující s čistě jedním signalizačním protokolem pracují jako páteřní směrovače.

1.2 Session Initiation Protocol (SIP)

Jedná se o signalizační protokol, který se stará o sestavování, modifikaci a ukončování multimediálních a real-time spojení. Pro samotný přenos dat využívá protokol RTP. Pro správu multimédií pak využívá Session Description Protocol (SDP). SIP spojení lze navázat jak pomocí User Datagram Protocol (UDP), tak pomocí Transmission Control Protocol (TCP). Pro komunikaci využívá port 5060. Existuje i šifrovaná varianta SIPS, která využívá služeb protokolů TCP a Transport Layer Security (TLS). Pro zabezpečenou komunikaci se využívá port 5061.

Pro sestavování a ukončování multimediálních spojení protokol využívá tyto funkce [2]:

- **Vyhledávání uživatelů** – Nalezení adresy požadovaného koncového bodu.
- **Zjištění dostupnosti** – Zjištění jestli s nalezeným koncovým bodem lze navázat spojení.
- **Dohodnutí parametrů** – Zjištění možností účastníků (přenosová rychlost, typ kodeku atd.).
- **Sestavení spojení** - Sestavení spojení mezi koncovými body a informace oběma stranám o průběhu spojení (vyzvánění, zahlcení atd).
- **Řízení spojení** – Řízení a ukončení aktivního spojení.

SIP je textově orientovaný protokol, jenž vychází z protokolu Hypertext Transfer Protocol (HTTP). Komunikace funguje na způsobu zasílání zpráv a odpovědi na ně. Mezi základní zprávy patří [2]:

- **ACK** – Potvrzení zprávy INVITE
- **BYE** – Ukončení aktivního spojení
- **CANCEL –** Zrušení nevyřízeného požadavku
- **INFO** – Hovorové signalizační informace
- **INVITE** – Navázání spojení
- **MESSAGE** – Přenos zprávy
- **NOTIFY** – Odesílání účastníkům zprávy o stavech zdrojů
- **OPTIONS** – Informace o možnostech účastníků
- **PRACK** – Potvrzení prozatimní odpovědi

- **PUBLISH** – Odesílání status informací na server
- **REGISTER** – Registrace nových uživatelů a aktualizace lokačních tabulek
- **REFER** – Možnost kontaktování třetích stran pomocí URI (Uniform Resource Identifier) adresy.
- **SUBSCRIBE** – Přihlášení k odběru aktualizací
- **UPDATE** – Aktualizace stavových informací o spojení

Na každou zprávu musí příjemce odpovědět. Odpověď se stejně jako v protokolu HTTP skládá z třímístného číslo, kde první číslo udává rodinu odpovědi a další 2 číslice dopřesňují odpověď. Rodiny odpovědí jsou:

- **1XX** - Informační nebo prozatimní odpověď (100 Trying, 180 Ringing)
- **2XX** – Úspěšné převzetí zprávy (200 OK, 202 Accepted)
- **3XX** – Informace o přesměrování (301 Moved Permanently, 305 Use Proxy)
- **4XX** – Chybové hlášení (400 Bad Request, 404 Not Found)
- **5XX** – Chyba serveru (500 Internal Server Error, 502 Bad Gateway)
- **6XX** – Globální chyby (603 Decline, 606 Not Accepted)

1.3 Komponenty SIP ústředen

Softwarové ústředny lze chápat jako servery, jež nabízí účastníkům vybrané služby. Různé řešení od různých vývojových týmů pak dokáží funkce serverů kombinovat. Dále v textu jsou vyjmenovány a popsány druhy serverů, se kterými se lze při komunikaci pomocí SIP setkat [3].

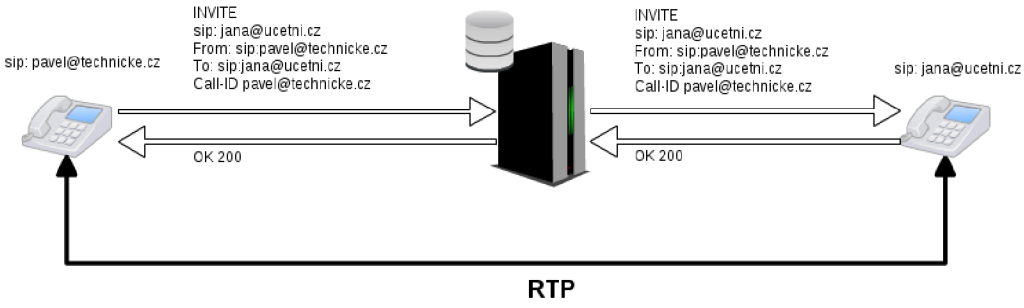
1.3.1 Registrar Server

Registrar server je určený k vyřizování SIP požadavků typu REGISTER. Registrace ve většině případů se skládá z uživatelského jména a hesla. Pokud registrace proběhne úspěšně, jsou informace o přihlášeném uživateli poslány do databáze lokačního serveru patřícího pod spravovanou doménu. Z toho důvodu se často registrar a lokační server spojují do jednoho. Lokační server může uložit k jednomu SIP URI záznamu více IP adres. Pokud chce klient navázat spojení, lokační server má za práci najít polohu volaného klienta.

1.3.2 Proxy Server

Jedná se o server nacházející se vždy mezi 2 koncovými uživateli, jež zpracovává SIP signalizaci po celou dobu navázaného spojení. Díky tomu lze implementovat účtování hovorů. Hlavním úkolem serveru je aby INVITE požadavek byl směrován k správnému

cíli. Proxy server nezpracovává mediální tok. Většinou je proxy server implementován společně s registrar a lokačním serverem. Průběh navázání spojení pomocí proxy serveru je zobrazen na obr. 1.1



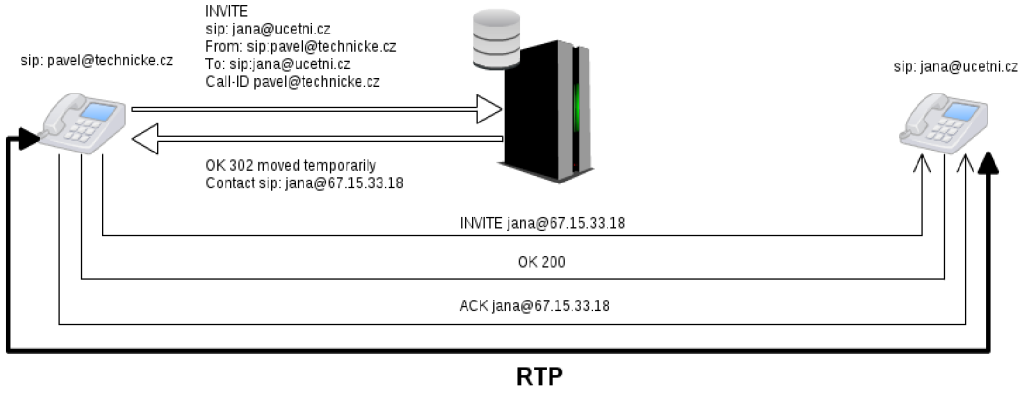
Obr. 1.1: Realizace spojení pomocí proxy serveru

1.3.3 Redirect Server

Jedná se o speciální variantu proxy serveru v módu přesměrování, která na INVITE požadavek vrací odpověď typu 3xx společně s adresou hledaného koncového zařízení a sestavení spojení již je plně v režii koncových zařízení. Výhodou takového řešení je, že server i s omezenými prostředky dokáže zpracovat velké množství požadavků. Nedokáže ale nabízet účtovací služby. Průběh navázání spojení pomoci redirect serveru je zobrazen na obr. 1.2.

1.3.4 Media Server

Jedná se o typ, jež v rámci spojení dokáže vystupovat jako koncový uživatel, jež dokáže přehrávat automaticky nahrané vzkazy a oznámení (IVR), přijímat tónovou volbu (DTMF) nebo hlasové volby kvůli poskytování služeb jako voicemail nebo tele-banking.



Obr. 1.2: Realizace spojení pomocí Redirect serveru

1.3.5 Media Proxy

Media proxy zpracovává kromě SIP signalizace i mediální datový tok. Tudíž je umístěn přímo mezi koncovými zařízeními. Z toho důvodu modifikuje SIP signalizaci tak aby

adresa pro vysílání mediálního toku nebyla adresou koncového uživatele, ale adresou serveru. Media proxy se využívá hlavně pro přemostění překladu adres (NAT), kdy server má vždy veřejnou IP adresu, díky které jsou obě koncová zařízení, jež mají adresy privátní, schopné se serverem navázat spojení a využít ho ve chvíli, kdy na sebe koncová zařízení nevidí.

1.3.6 Gateway Server

Jedná se o servery, které spojují VoIP síť s veřekjnou telefonní sítí (PSTN), nebo se sítí mobilního operátora. Jeho hlavním úkolem je tedy konverze signalizace i transkódování mediálního toku.

1.3.7 B2BUA/Media Server

Back to Back User agent (B2BUA) jehož nejznámějším představitelem je Asterisk nabízí stejné funkce jak proxy server. Na rozdíl od něj se ale dokáže chovat jako koncový uživatel. Dokáže hovory transkódovat pro potřeby odlišných sítí. Na rozdíl od gateway serveru se B2BUA považuje jako přemostění mezi 2 VoIP sítěmi.

1.3.8 Session Border Controller

Udává se jako server implementovaný na hranici privátní sítě a internetu, jehož úkolem je kontrola VoIP komunikace. Mezi jeho nejdůležitější funkce patří zabezpečení (zabránění útoků odmítnutí služby (DoS), skrývání topologie, nebo kódování signalizace i mediálního toku), řízení konektivity (NAT přemostění, IPv4-to-IPv6 konverze, virtuální privátní sítě (VPN)), zajištění kvality služeb (QoS) nebo poskytování mediálních služeb.

2 ÚSTŘEDNY KAMAILIO A OPENSIPS

2.1 Projekt SER/OpenSER

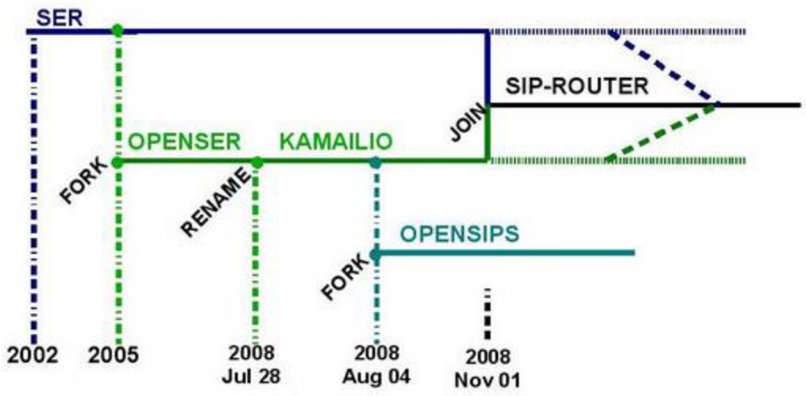
První myšlenky o vzniku SIP směrovače vznikaly již v roce 1995 v komunitě studentů kolem prof. Schulzrinneho, jež je považován za jednoho z konstruktérů SIP protokolu, který právě pracoval ve Forschungs-Institut für Offene Kommunikationssysteme (FOKUS) v Německu [4].

V roce 2001 začala práce na psaní kódu programu s názvem SIP Express Router (SER). Jednalo se o SIP registrar, proxy nebo redirect server, který byl v počátku šířen pod GNU General Public License. Mezi jeho vlastnosti patřilo účtování a autentizace pomocí RADIUS/syslog serverů nebo XML-RPC dálková kontrola. SER šlo nastavit jak pro využití v malé kanceláři, tak pro funkci velké korporátní ústředny. SER lze z části považovat i za český program a to díky práci Jana Janáka nebo Karla Kozlíka.

Jelikož už v roce 2004 SER patřil mezi hojně nasazované SIP servery mezi poskytovateli internetového připojení (např. freenet, sipgate) [4], bylo potřeba projekt přesunout z akademické půdy pod hlavičku nově vzniklé firmy iptelorg GmbH na kterou byly převedeny práva na komerční využití autorských práv a tvorba komerčních doplňku. Na nezávislé stránce iptel.org pak byl uveřejněný open-source SER, volně dostupné SIP služby a základní informace o SIP protokolu.

V roce 2005 (viz. obr.2.1) došlo k odtržení části tvůrců, kteří nesouhlasili se směřováním projektu a založili vlastní projekt s názvem openSER, který začal kolem sebe budovat silnou komunitu zajišťující funkcionalitu, flexibilitu a dokumentaci. Ale práce na SER se nezastavily a v Praze byla založena nová pobočka, zabývající se vývojem SIP infrastruktury pro mobilní zařízení. Ve stejném roce byla firma odkoupena vlastním zákazníkem firmou Tekelec. A v roce 2006 bylo uveřejněno nové vydání SER.

Další velké změny nastaly až v roce 2008, kdy z důvodu problémů s ochrannou známkou se musel program openSER přejmenovat na Kamailio. Kromě změny názvu ale mezi vývojáři opět začali sílit spory, které vedly k dalšímu rozdělení týmu a založení dalšího projektu, který nese název openSIPS.



Obr. 2.1: Časová historie projektů SER, Kamailio a OpenSIPS (převzato z [4])

Ze situace, která nastala, kdy souběžně existovaly 3 open-source projekty vykonávající stejnou službu (SIP Signalizaci), rostla nejistota mezi uživateli, který z projektů je pro jejich řešení nejlepší, ale hlavně jestli jimi zvolený produkt časem neukončí svoji činnost. Z toho důvodu začal v komunitě sílit tlak na možné opětovné spojení projektů pod jeden. A ještě v listopadu 2008 se podařilo domluvit integraci projektů SER a Kamailio pod hlavičku The SIP Router Project. Zatím co vývoj projektu openSIPS pokračoval nezávisle, SER a Kamailio postupně integrovali svá řešení, aby v lednu 2010 mohlo vyjít první společné řešení Kamailio (OpenSER) 3.0.0.

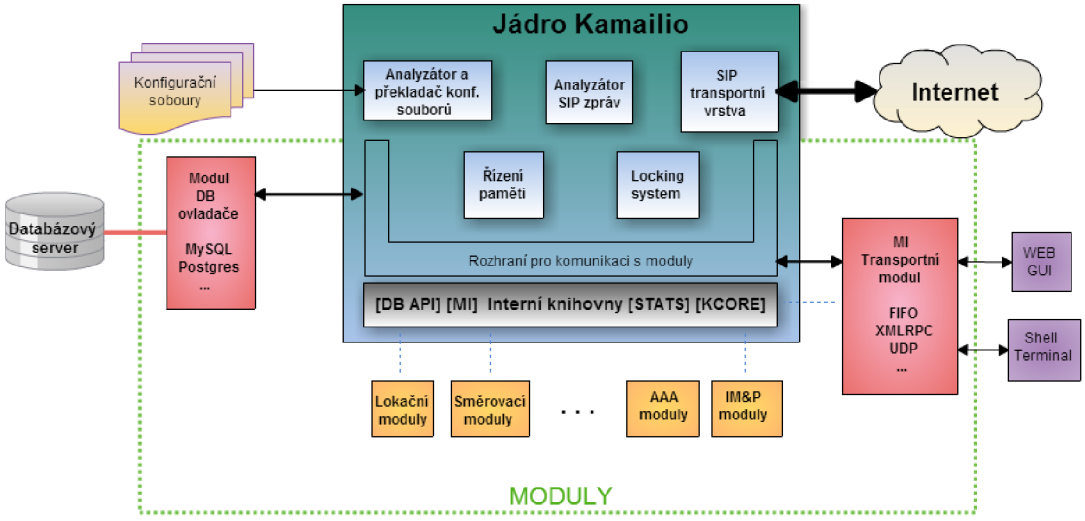
2.2 Kamailio

Projekt Kamailio jehož název vychází z havajského slova Kama'ilio znamená v překladu mluvit, konverzovat [5]. Vznikl v roce 2005 pod názvem openSER odštěpením od projektu SIP Express Router. V roce 2008 po přejmenování na současný název a odchodem části tvůrců přišlo opět sblížení s původním projektem SIP Express Router a začaly práce na integraci. Ta byla hotová při uveřejnění verze programu v3.0.0. V dnešní době je nejaktuálnější verze programu v4.1.x.

Jedná se o Open Source řešení napsané v programu C šířené pod veřejnou licencí GPL. Oproti ústředně Asterisk se nejedná o plnohodnotné řešení, ale nýbrž o platformu, která pomocí přídavných modulů (jež lze psát například pomocí jazyků Lua, Perl, Java nebo Python) získává požadovanou funkcionalitu [6].

Nejčastěji se Kamailio používá jako Registrar/Lokační server, proxy server nebo jako SBC.

Kamailio používá modulární architekturu jež je zobrazena na obr. 2.2. Architektura obsahuje základní 2 části a to jádro s interními knihovnami a moduly [7].



Obr. 2.2: Architektura Kamailio

2.2.1 Jádro

Obsahuje komponenty pro poskytování služeb nižších vrstev a interní knihovny, které sbírají informace z některých modulů, jež nemají obecný účel. Moduly tudíž nepatří

mezi součásti jádra. Jádro se skládá:

- Jádro,
- řízení paměti,
- analyzátor SIP zpráv,
- Jednoduché vývojové rozhraní tzv. Locking system,
- řízení doménových jmen (DNS) a transportních služeb (UDP, TCP, TLS a další)
- a další…

2.2.2 Interní knihovny

- Některé komponenty z jádra Kamailia v1.5.x,
- databázovou abstraktní vrstvu,
- rozhraní pro programování aplikací řízení rozhraní (MI API),
- statistické jádro.

2.2.3 Moduly

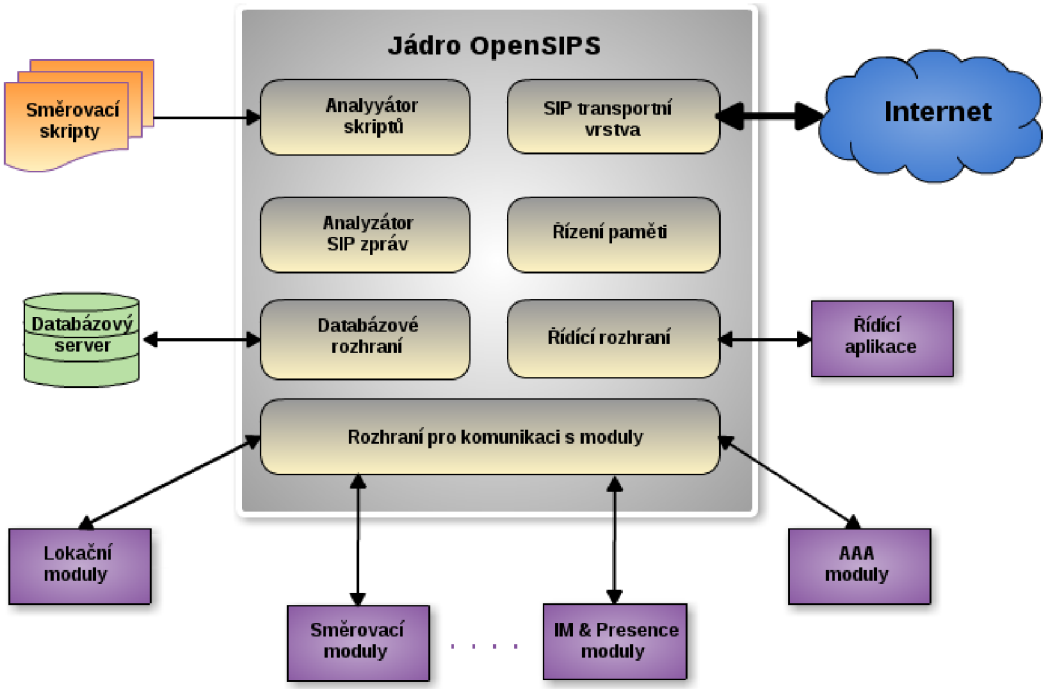
Jedná se o volitelné komponenty, jež vykonávají samotné funkce jako například:

- Autentikační, autorizační a účtovací služby (AAA),
- sledování SIP dialogů,
- Structured Quary Language (SQL) a non-SQL dababáze,
- NAT přemostění a skrývání topologie,
- vyvažování zátěže a hledání nejkratší cesty
- a další…

2.3 OpenSIPS

Projekt OpenSIPS vznikl v roce 2008 odštěpením od projektu Kamailio. I přes vyvíjený tlak komunitou o sloučení s projekty OpenSER a Kamailio je OpenSIPS dále vyvíjen nezávisle. V dnešní době je nejaktuálnější verze programu 1.10.0, ale podle vývojářů probíhá intenzivní práce na verzi 2.0, která je fázi testování [8].

OpenSIPS je Open Source řešení psané v jazyce C a šířené pod licencí GPL. Nejčastěji se používá jako Registrar/lokační server, proxy server nebo jako SBC. Stejně jako Kamailio je OpenSIPS navržen jako modulární platforma (viz obr. 2.3), kde jádro obsahuje pouze nezbytné prvky k fungování. Rozmanitost služeb zajišťují připojené moduly, jež lze napsat v jazycích jako Perl nebo Java [9].



Obr. 2.3: Architektura OpenSIPS

2.3.1 Jádro

Obsahuje komponenty pro poskytování služeb nižších vrstev.

- Transportní vrstva,
- řízení paměti,
- analyzátor SIP zpráv,
- synchronizační mechanizmus,
- řídící rozhraní,
- databázové rozhraní,
- konfigurační soubory a skripty.

2.3.2 Moduly

Jedná se o volitelné komponenty, jež vykonávají samotné funkce jako například:

- AAA služby,
- tElephone NUmber Mapping (ENUM) směrování,
- vyvažování zátěže,
- NAT přemostění,
- MySQL databáze,
- a další ...

3 POROVNÁNÍ KAMAILIO A OPENSIPS S PBX ASTERISK

Pro správnou představu jaké funkce projekty Kamailio a OpenSIPS nabízejí, budou v následujících podkapitolách porovnány s nejrozšířenější open source pobočkovou ústřednou Asterisk[10].

3.1 Architektura

V základu jsou Kamailio a OpenSIPS primárně Proxy servery spojené s Registrar serverem, jejichž hlavním úkolem je směrování SIP signalizace. Oproti ústředně Asterisk, která také dokáže pracovat jako proxy server však nekladou takové nároky na výpočetní výkon a tudíž dokáží obsloužit větší počet účastníků. Tenhle rozdíl je dán hlavně faktem, že jádro Kamailio a OpenSIPS obsahuje minimální počet prvků a to hlavně komponenty starající se o správné vyřízení SIP požadavků a rozhraní pro dodatečné přípojné moduly, jež rozšiřují funkcionalitu ústředen. Asterisk je navržen jako B2BUA, díky čemuž jádro ústředny musí obsahovat větší počet komponent. Jako například část starající se o konverzi různorodých hlasových kodeků, nebo konverzi různých signalizačních protokolů, jež v ústředna Kamailio a OpenSIPS nejsou.

3.2 Druhy telefonních spojení

Kamailio a OpenSIPS jsou navrženy jako čistě signalizační VoIP ústředny, jež dokáží obsluhovat pouze protokol SIP. Díky přístupu k SIP hlavičce a možnosti její editace dokáží překládat mezi SIP implementacemi od různých výrobců, jež nemusí být mezi sebou kompatibilní. Jelikož mediální tok mezi koncovými uživateli ústředny nezpracovávají, nedokáží transkódovat data pořízena jinou modulační technikou. Asterisk oproti tomu dokáže zpracovávat mediální tok a díky externím kartám dokáže nejen transkódovat data na požadovaný kodek, ale dokáže zpracovávat hovory přijaté z PSTN. Kromě singalizace SIP dokáže zpracovat i ostatní druhy (IAX, MGCP nebo TDM).

3.3 Zabezpečení

V problematice bezpečnosti žádné výrazné odlišnosti nejsou. Všechny ústředny podporují autentizaci pomocí RADIUS serveru, stejně jako protokol pro ukládání a přístup k datům (LDAP). Zabezpečení komunikace lze provézt pomocí kryptografického protokolu TLS.

3.4 Směrování

V Asterisku lze směrování definovat pomocí atribut uživatelské jméno, ruri, callid a dalších atribut. Lze také nastavit směrování při výpadku linky. U ústředen Kamailio a

OpenSIPS lze také definovat atributy uživatelského jména, ruri, callid a dalších atribut. Ke směrování se ale využívá speciální hashovací algoritmus, díky němuž výběr probíhá rychleji.

3.5 Media služby

Jak již bylo výše napsáno, Kamailio a OpenSIPS zpracovávají pouze signalizaci, tudíž nenabízejí služby jako IVR, hlasovou poštu nebo transkódování, které Asterisk zvládá.

3.6 Konfigurace

Díky rozdílné architektuře je konfigurace Kamailio a OpenSIPS oproti Asterisku rozdílná. Zatím co v Asterisku existuje konfigurační soubor pro každou službu zvlášť a lze do nich ukládat i provozní data (uživatele, vytáčecí plán atd.), ostatní dvě ústředny mají vždy pouze jediný konfigurační soubor, jež obsahuje například základní nastavení nebo seznam načtených modulů. Provozní data jsou pak uložené v databázi.

3.7 Shrnutí

Ačkoliv se v kapitole srovnávala ústředna Asterisk s ústřednami Kamailio a OpenSIPS, nelze tyhle projekty navzájem srovnávat. Porovnání bylo provedeno k pochopení odlišností, které jsou zapříčiněné různými rolemi v rámci komunikace a tudíž i rozdílnou architekturou. Naopak velice často se využívá spolupráce kdy Kamailio a OpenSIPS jsou využity jako páteřní směrovače nebo SBC, zatím co Asterisk je implementován jako pobočková ústředna s IVR, nebo jako PSTN brána.

4 INSTALACE A KONFIGURACE ÚSTŘEDNY KAMAILIO

Návod popisuje krok po kroku instalaci ústředny Kamailio v4.1.x na nově nainstalovém operační systému Ubuntu 12.04 [11].

4.1 Instalce programu

Spustíme terminál a přepneme se permanentně do root uživatele:

```
kamailio@kamailio:~$ sudo -s
[sudo] password for kamailio:
root@kamailio:~#
```

Instalace prerekvizit:

```
apt-get install mysql-server git gcc flex bison libmysqlclient-dev
make libssl-dev libcurl4-openssl-dev libxml2-dev libpcre3-dev
```

Vytvoříme složku pro instalační soubory a přepneme se do ní:

```
mkdir -p /usr/local/src/kamailio
cd /usr/local/src/kamailio
```

Stáhneme instalační soubory pomocí příkazu GIT a přepneme se do instalačního adresáře:

```
git clone --depth 1 git://git.sip-router.org/kamailio kamailio
cd kamailio
git checkout -b 4.1 origin/4.1
```

Vygenerujeme konfigurační soubory:

```
make cfg
```

Zeditujeme soubor `modules.lst`:

```
vi modules.lst
```

V souboru nastavíme prefix pro instalaci programu a povolíme moduly, které se standardně nekompilují:

```
include_modules = db_mysql tls
```

Po nastavení všech modulů a uložení souboru, provedeme kompilaci programu:

```
make all
```

Po správné kompilaci program nainstalujeme:

```
make install
```

Po úspěšné instalaci by měli ve složce `/usr/local/sbin` být vytvořené skripty:

```
kamailio - Kamailio SIP server
kamdbctl - vytvářející a řídící skript pro práci s databází
kamctl - řídící a kontrolní skript pro Kamailio SIP server
sercmd - CLI pro komunikaci s Kamailio SIP serverem
```

Aby mohly být skripty spouštěné mimo složku ve které jsou uložené je potřeba aby složka byla uložená v systémové proměnné PATH. Proto zkontrolujeme zda-li je cesta ke skriptům v proměnné uložena:

```
echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Pokud ve výpisu cesta `/usr/local/sbin` není vypsaná, je potřeba cestu zadat ručně do souboru `/root/.bash_profile`:

```
vim /root/.bash_profile
```

Kde na konec souboru přidáme odkaz na cestu ke skriptům:

```
PATH = $PATH:/usr/local/sbin
Export PATH
```

4.2 Vytvoření MySQL databáze

Před vytvořením MySQL databáze je potřeba ve vytvářecím skriptu databázi povolit:

```
vi /usr/local/etc/kamailio/kamctlrc
```

Kde je potřeba povolit popřípadě podle potřeby modifikovat řádky:

```
DBENGINE=MYSQL
DBHOST=localhost
DBNAME=kamailio
DBRWUSER="kamailio"
DBRWPW="kamailiorw"
DBROUSER="kamailioro"
DBROPW="kamailioro"
DBACCESSHOST=127.0.0.1
DBROOTUSER="root"
```

Samotná databáze se vytvoří pomocí skriptu kamdbctl, při které všechny volby potvrdíme:

```
/usr/local/sbin/kamdbctl create
```

4.3 Úprava konfigurace a inicializace programu

Konfiguraci nainstalovaného programu lze provést v souboru kamailio.cfg, ve kterém povolíme povolíme práci s MySQL databází:

```
vim /usr/local/etc/kamailio/kamailio.cfg
```

Na začátek souboru přidáme řádky:

```
#!define WITH_MYSQL
#!define WITH_AUTH
#!define WITH_USRLOCDB
```

Pokud jsme při vytváření databáze změnili uživatelské jméno nebo heslo, je potřeba změnit hodnotu v řádku v souboru za nastavené:

```
#!define DBURL "mysql://kamailio:kamailiorw@localhost/kamailio"
```

Po nastavení konfigurace překopírujeme inicializační soubor, kterému změníme práva aby mohl být program spuštěn i jinými uživateli, než rootem:

```
cp usr/local/src/kamailio/kamailio/pkg/kamailio/deb/debian/
kamailio.init /etc/init.d/kamailio
chmod 755 /etc/init.d/kamailio
```

Následně spouštěcí skript editujeme:

```
vi /etc/init.d/kamailio
```

A v souboru upravíme řádky odkazující na cestu k spouštěcímu skriptu a konfiguračnímu souboru:

```
DAEMON=/usr/local/sbin/kamailio
CFGFILE=/usr/local/etc/kamailio/kamailio.cfg
```

Dále překopírujeme soubor se základním nastavením `kamailio.default`:

```
cp /usr/local/src/kamailio/kamailio/pkg/kamailio/deb/debian/
kamailio.default /etc/default/kamailio
```

A v souboru povolíme spouštění programu změnou řádku:

```
RUN_KAMAILIO=yes
```

Vytvoříme složku pro možnost ukládání dočasných souborů:

```
mkdir -p /var/run/kamailio
```

V souboru pro základní nastavení se určuje uživatel a skupina aplikace. Proto je potřeba obojí vytvořit. Pokud se hodnota v soboru nezmění je název uživatele i název skupiny kamailio.

```
adduser --quiet --system --group --disabled-password \
        --shell /bin/false --gecos "Kamailio" \
        --home /var/run/kamailio kamailio
```

Jelikož jsme soubor se základním nastavením kopírovali pod uživatelem root je potřeba změnit vlastníka souboru na uživatele kamailio:

```
chown kamailio:kamailio /var/run/kamailio
```

Tím je základní nastavení hotovo a aplikace je připravené spuštění. Start, stop a restart aplikace lze provézt příkazy:

```
/etc/init.d/kamailio start
/etc/init.d/kamailio restart
```

```
/etc/init.d/kamailio stop
```

4.4 Vytvoření uživatelského účtu

Nový uživatelský účet lze vytvořit pomocí řídícího a kontrolního skriptu kamctl pomocí příkazu:

```
kamtcl add <jméno-uživatele> <heslo> <email>
```

Kde email je nepovinný údaj. Jméno uživatele musíme zadat ve tvaru jméno@SIP_DOMAIN.

Pokud budou uživatelé patřit do stejné domény a budeme chtít při tvorbě zadávat pouze jméno, bez adresy domény, musíme jí deklarovat jako systémovou proměnnou:

```
export SIP_DOMAIN=127.0.0.1
```

4.5 Nastavení TLS modulu

Před samotným nastavením komunikace pomocí TLS je potřeba vytvořit certifikát. Ten lze zakoupit u některé z certifikačních autorit. Pro potřeby diplomové práce bude popsáno vytvoření vlastní certifikační autority a následné vytvoření vlastního certifikátu [12]

Vytvoříme adresáře, do kterých se budou soubory ukládat:

```
mkdir /etc/certs
chmod 0700 /etc/certs
mkdir /etc/certs/CA
mkdir /etc/certs/CA/newcerts
```

Vytvoříme soubory `serial` a `index.txt`:

```
echo '01' > /etc/certs/CA/serial
touch /etc/certs/CA/index.txt
```

Přepneme se do adresáře certifikační autority a vytvoříme certifikát certifikační autority platný na 10 let:

```
cd /etc/certs/CA
openssl req -new -x509 -extensions v3_ca -keyout key.pem -out
cert.pem -days 3650
```

Vytvoříme adresář pro certifikační soubory ústředny a vytvoříme vlastní certifikát podepsaný vytvořenou certifikační autoritou:

```
mkdir /etc/certs/pbx
cd /etc/certs/pbx
openssl req -new -nodes -keyout key.pem -out req.pem
cd ..
openssl ca -days 730 -out pbx/cert.pem -keyfile CA/key.pem -cert
CA/cert.pem -infiles pbx/req.pem
```

V konfiguračním souboru /etc/ssl/openssl.cnf změníme cesty k certifikační autoritě:

```
vim /etc/ssl/openssl.cnf
dir = /etc/certs/CA
certs = $dir/
```

Po vytvoření certifikátu aktidivujeme funkci TLS v konfiguračním souboru ústředny:

```
vim /etc/kamailio/etc/kamailio/kamailio.cfg
```

Kde na začátek souboru přidáme řádek:

```
#!define WITH_TLS
```

Definujeme ip adresu a port na kterém zabezpečená komunikace bude probíhat:

```
listen=tls:127.0.0.1:5061
```

Nakonec nakofigurejeme cesty k certifikátům v souboru `/usr/local/kamailio/etc/kamailio/tls.cfg`:

```
vim /usr/local/kamailio/etc/kamailio/tls.cfg

vim /usr/local/kamailio/etc/kamailio/tls.cfg

[server:default]

method = TLSv1
verify_certificate = yes
require_certificate = no
private_key = /etc/certs/pbx/key.pem
certificate = /etc/certs/pbx/cert.pem
ca_list = /etc/certs/CA/cert.pem

[server:127.0.0.1:5061]
method = SSLv23
verify_certificate = no
require_certificate = no
private_key = /etc/certs/pbx/key.pem
certificate = /etc/certs/pbx/cert.pem
ca_list = /etc/certs/CA/cert.pem

[client:default]
verify_certificate = no
require_certificate = no
```

4.6 Adresářová struktura Kamailio

4.6.1 Konfigurační soubory

Složka `/usr/local/kamailio/etc/kamailio` obsahuje hlavní konfigurační soubor kamailio.cfg a také zdrojový soubor pro skript kamctlrc.

Výpis adresáře `/usr/local/kamailio/etc/kamailio/`:

```
-rw-r--r-- 1 root root  1745 pro  1 14:53 dictionary.kamailio
```

```
-rw-r--r-- 1 root root 22237 pro  1 14:53 kamailio-advanced.cfg
-rw-r--r-- 1 root root 14484 pro  1 14:53 kamailio-basic.cfg
-rw-r--r-- 1 root root 21550 pro  1 14:53 kamailio.cfg
-rw-r--r-- 1 root root  3754 pro  1 14:53 kamctlrc
```

4.6.2 Moduly

Všechny moduly, které dokáží spolupracovat s jádrem Kamailio lze najít v adresáři `/usr/local/kamailio/lib/kamailio/modules/`.

Výpis adresáře `/usr/local/kamailio/lib/kamailio/modules/`:

```
acc.so              dmq.so              pdb.so              speeddial.so
alias_db.so         domainpolicy.so     pdt.so              sqlops.so
async.so            domain.so           permissions.so      sst.so
auth_db.so          drouting.so         pike.so             statistics.so
auth_diameter.so    enum.so             pipelimit.so        stun.so
auth.so             exec.so             prefix_route.so     textops.so
avpops.so           group.so            print_lib.so        textopsx.so
avp.so              htable.so           print.so            timer.so
benchmark.so        imc.so              p_usrloc.so         tls.so
blst.so             ipops.so            pv.so               tmrec.so
call_control.so     kex.so              qos.so              tm.so
cfg_db.so           malloc_test.so      ratelimit.so        tmx.so
cfg_rpc.so          mangler.so          registrar.so        topoh.so
cfgutils.so         matrix.so           rr.so             uac_redirect.so
cnxcc.so            maxfwd.so           rtimer.so           uac.so
corex.so            mediaproxy.so       rtpproxy-ng.so      uid_auth_db.so
counters.so         mi_datagram.so      rtpproxy.so         uid_avp_db.so
ctl.so              mi_fifo.so          sanity.so           uid_domain.so
db_cluster.so       mi_rpc.so           sca.so              uid_gflags.so
db_flatstore.so     mohqueue.so         sdpops.so           uid_uri_db.so
db_mysql.so         mqueue.so           seas.so             uri_db.so
db_text.so          msilo.so            sipcapture.so     userblacklist.so
db2_ops.so          msrp.so             siptrace.so         usrloc.so
debugger.so         mtree.so            sipt.so             xhttp_rpc.so
dialog.so           nathelper.so        siputils.so         xhttp.so
dispatcher.so       nat_traversal.so    sl.so               xlog.so
diversion.so        path.so             sms.so              xprint.so
```

4.6.3 Binární soubory

Binární soubory skriptů jsou uloženy v adresáři `/usr/local/kamailio/sbin`. Pro libovolné spouštění skriptů lze adresář přidat do systémové proměnné PATH, nebo skripty zkopírovat do adresáře, který již v proměnné PATH uložený je.

Výpis adresáře `/usr/local/kamailio/sbin`:

```
-rwxr-xr-x 1 root root 6632114 pro  1 14:53 kamailio
-rwxr-xr-x 1 root root  121233 pro  1 14:53 kamcmd
-rwxr-xr-x 1 root root   56886 pro  1 14:53 kamctl
-rwxr-xr-x 1 root root   10501 pro  1 14:53 kamdbctl
```

4.6.4 Dodatečné informace a návody

Informace o tvůrcích, novinkách a pokyny pro instalaci programu a modulů lze nalést

v adresáři `/usr/local/kamailio/share/doc/kamailio/`.

Výpis adresáře `/usr/local/kamailio/share/doc/kamailio/`:

```
-rw-r--r-- 1 root root  2149 pro  1 14:53 AUTHORS
-rw-r--r-- 1 root root 29537 pro  1 14:53 INSTALL
drwxr-xr-x 2 root root  4096 pro  1 14:53 modules
-rw-r--r-- 1 root root 78984 pro  1 14:53 NEWS
-rw-r--r-- 1 root root  4312 pro  1 14:53 README
-rw-r--r-- 1 root root  8857 pro  1 14:53 README-MODULES
```

5 INSTALACE A KONFIGURACE ÚSTŘEDNY OPENSIPS

Návod popisuje krok po kroku instalaci ústředny OpenSIPS ver. 1.8 na nově nainstalovém operační systému ubuntu 12.04 [13].

5.1 Instalace programu

Spustíme terminál a přepneme se permanentně do root uživatele:

```
opensips@opensips:~$ sudo -s
[sudo] password for opensips:
root@opensips:~#
```

Instalace prerekvizit:

```
apt-get install subversion flex bison libncurses-dev
libmysqlclient-dev libsctp-dev m4 mysql-server
```

Přepneme se do pracovní složky:

```
cd /usr/local/src/
```

Stáhneme instalační soubory:

```
svn co https://svn.code.sf.net/p/opensips/svn/branches/1.8
opensips_1_8
```

Přejdeme do staženého adresáře a spustíme konfigurační soubor:

```
cd opensips_1_8/
make menuconfig
```

Vzhled konfiguračního souboru lze vidět na obr. 5.1

Obr. 5.1: Vzhled konfiguračního souboru OpenSIPS

Zkontrolujeme, případně nastavíme další instalované atributy:

```
Configure Compile Options → Configure Compile Flags
```

Povolíme ukládání informací do mySql databáze:

```
Configure Compile Options → Configure Exclude Modules → db_mysql
```

Zadáme instalační cestu:

Configure Compile Options → Configure Install Prefix → /usr/local/opensips.

Nainstalujeme program:

```
Compile And Install OpenSIPS
```

Přejdeme do nainstalované složky a editujeme soubor `opensipsctlrc`:

```
cd /usr/local/opensips/etc/opensips
vi opensipsctlrc
```

Ukládání dat do databáze MySQL nastavíme odkomenotováním řádků:

```
DBENGINE=MYSQL
DBHOST=localhost
DBNAME=opensips
DBRWUSER=opensips
DBRWPW="opensipsrw"
DBROOTUSER="root"
```

Přejdeme do složky /usr/local/opensips/sbin/ a vytvoříme databázi MySQL

```
cd /usr/local/opensips/sbin/
./opensipsdbctl create
MySQL password for root: (heslo)
INFO: test server charset
INFO: creating database opensips ...
INFO: Core OpenSIPS tables succesfully created.
Install presence related tables? (y/n): n
Install tables for imc cpl siptrace domainpolicy carrierroute
userblacklist? (y/n): n
```

Vytvoříme konfigurační soubor:

```
./osipsconfig
```

Zvolíme funkce, které chceme používat:

Generate OpenSIPS Script → Residential Script → Configure Residential Script

Vygenerujeme konfikurační soubor:

Generate OpenSIPS Script → Residential Script → Generate Residential Script

Přepneme se do složky `/usr/local/opensips/etc/opensips` a editujeme

vytvořený konfigurační soubor:

```
cd /usr/local/opensips/etc/opensips
vi opensips_residential_2013-11-18_0:2:12.cfg
```

Upravíme cestu k modulům programu:

```
mpath="/usr/local/opensips/lib/opensips/modules/"
```

V sekci `URI module` doplníme řádek:

```
modparam("uri","db_url",
"mysql://opensips:opensipsrw@localhost/opensips")
```

Přepneme se do složky `/usr/local/src/opensips_1_8/packaging/debian:`

```
cd /usr/local/src/opensips_1_8/packaging/debian
```

Zkopírujeme inicializační soubor a změníme jeho práva pro spuštění:

```
cp opensips.init /etc/init.d/opensips
chmod -755 /etc/init.d/opensips
```

Editujeme inicializační soubor:

```
vi /etc/init.d/opensips
```

Změníme cestu k deamonu:

```
DAEMON=/usr/local/opensips/sbin/opensips
```

Zadáme cestu ke konfiguračnímu souboru:

```
OPTIONS="-P $PIDFILE -m $S_MEMORY -M $P_MEMORY -u $USER -g $GROUP -
f    /usr/local/opensips/etc/opensips/opensips_residential_2013-11-
18_0:2:12.cfg "
```

Vymažeme možnost debugování:

```
if [ "$1" != "debug" ]; then
 check_fork
fi
```

Zkopírujeme soubor se základním nastavením:

```
cp opensips.default /etc/default/opensips
```

Přepneme se do složky `/etc/default/` a editujeme soubor opensips:

```
cd /etc/default/
vi opensips
```

Změníme položky:

```
RUN_OPENSIPS=yes
USER=root
GROUP=root
```

Přepneme se do složky /usr/local/opensips/etc/opensips/ a editujeme konfigurační soubor:

```
cd /usr/local/opensips/etc/opensips/
vi opensips_residential_2013-11-18_0\:2\:12.cfg
```

V souboru upravíme řádek:

```
log_facility=LOG_LOCAL1
```

V konfiguračním souboru programu syslog povolíme ukládání logů do souboru /var/log/opensips.log:

```
vi/etc/rsyslog.conf
```

f

Na konec souboru přidáme řádku:

```
local1.*-/var/log/opensips.log
```

Restartujeme syslog deamona:

```
/etc/init.d/rsyslog restart
```

Spustíme program opensips:

```
/etc/init.d/opensips start
```

Pokud je vše správně nastavené v terminálu se vypíše:

```
Starting opensips: opensipsListening on
            udp: 127.0.0.1 [127.0.0.1]:5060
            tcp: 127.0.0.1 [127.0.0.1]:5060
Aliases:

 already running.
```

5.2 Vytvoření domény a obsluha uživatelských účtů

Jméno uživatele musíme zadat ve tvaru jméno@SIP_DOMAIN. Pokud budou uživatelé patřit do stejné domény a budeme chtít při tvorbě zadávat pouze jméno, bez adresy domény, musíme jí deklarovat jako systémovou proměnnou:

```
export SIP_DOMAIN=127.0.0.1
```

Nový uživatelský účet lze vytvořit pomocí řídícího a kontrolního skriptu opensipsctl v adresáři /usr/local/opensips/sbin/ pomocí příkazu:

```
./opensipsctl add <jméno-uživatele> <heslo>
```

Odstranění uživatele lze pomocí stejného skriptu příkazem:

```
./opensipsctl rm <jméno-uživatele>
```

Skript zadané hodnoty ukládá do tabulky subscriber v databázi opensips. Případné přidání, odebrání, či editování vytvořených uživatelů lze pomocí MySQL příkazů k editaci entit v tabulce.

5.3 Nastavení TLS komunikace

Po instalaci je v adresáři /usr/local/opensips/etc/opensips vytvořena složka s vytvořenou certifikační autoritou a již vytvořenými certifikačními soubory. Pokud adresář neexistuje, lze certifikační autoritu i certifikační soubory vytvořit pomocí skriptu opensipsctl [14]:

```
opensipsctl tls rootCA
opensipsctl tls userCERT
```

Další možností je postup popsaný v kapitole 4.5.

Po vytvoření certifikačních souborů povolíme použití komunikace v konfiguračním souboru `/usr/local/opensips/etc/opensips/opensips.cfg`:

```
vim /usr/local/opensips/etc/opensips/opensips.cfg
```

Kde na začátek souboru přidáme následující řádky:

```
disable_tls = no
listen = tls:127.0.0.1:5061
tls_verify_server = 0
tls_verify_client = 0
tls_require_client_certificate = 0
tls_handshake_timeout=30
tls_send_timeout=30
tls_method = TLSv1
tls_ciphers_list="NULL"
tls_certificate = "/usr/local/opensips/etc/opensips//tls/user/user-
cert.pem"
tls_private_key = "/usr/local/opensips/etc/opensips//tls/user/user-
privkey.pem"
tls_ca_list   =   "/usr/local/opensips/etc/opensips//tls/user/user-
calist.pem"

tls_server_domain [127.0.0.1:5061]
{
tls_method = SSLv23
tls_certificate = "/usr/local/etc/opensips//tls/user/user-cert.pem"
tls_private_key     =     "/usr/local/etc/opensips//tls/user/user-
privkey.pem"
tls_ca_list = "/usr/local/etc/opensips/tls//user/user-calist.pem"
}
```

5.4 Adresářová struktura OpenSIPS

5.4.1 Konfigurační soubory

Složka `/usr/local/opensips/etc/opensips/` obsahuje hlavní konfigurační soubor opensips.cfg, běžící konfigurační soubor a a také zdrojové soubory pro skripty opensipsctl, osipsconsolerc.

Výpis adresáře `/usr/local/opensips/etc/opensips/`:

```
-rw------- 1 root root 6023 lis 17 23:42 opensips.cfg
-rw-r--r-- 1 root root 3659 lis 17 23:52 opensipsctlrc
```

```
-rw------- 1 root root 8172 lis 18 01:13 opensips_residential_2013-
1118_0:2:12.cfg
-rw-r--r-- 1 root root 2967 lis 17 23:42 osipsconsolerc
```

5.4.2 Moduly

Všechny moduly, které dokáží spolupracovat s jádrem OpenSIPS lze najít v adresáři `/usr/local/opensips/lib/opensips/modules`.

Výpis adresáře `/usr/local/opensips/lib/opensips/modules`:

```
acc.so              diversion.so        nathelper.so            siptrace.so
alias_db.so         dns_cache.so        nat_traversal.so              sl.so
auth_aaa.so         domainpolicy.so     options.so                  sms.so
auth_db.so          domain.so           path.so                speeddial.so
auth_diameter.so    drouting.so         pdt.so                      sst.so
auth.so             enum.so             peering.so            statistics.so
avpops.so           event_datagram.so   permissions.so              stun.so
benchmark.so        exec.so             pike.so                  textops.so
b2b_entities.so     gflags.so           presence_callinfo.so          tm.so
cachedb_local.so    group.so            presence_xcapdiff.so    uac_auth.so
call_control.so     imc.so              qos.so               uac_redirect.so
cfgutils.so         load_balancer.so    ratelimit.so      uac_registrant.so
closeddial.so       mangler.so          registrar.so                uac.so
db_flatstore.so     maxfwd.so           rr.so                       uri.so
db_mysql.so         mediaproxy.so       rtpproxy.so        userblacklist.so
db_text.so          mi_datagram.so      seas.so                   usrloc.so
db_virtual.so       mi_fifo.so          signaling.so
dialog.so           mi_http.so          sipcapture.so
dispatcher.so       msilo.so            sipmsgops.so
```

5.4.3 Binární soubory

Binární soubory skriptů jsou uloženy v adresáři `/usr/local/opensips/sbin`. Pro libovolné spouštění skriptů lze adresář přidat do systémové proměnné PATH, nebo skripty zkopírovat do adresáře, který již v proměnné PATH uložený je.

Výpis adresáře `/usr/local/opensips/sbin`:

```
-rw-r--r-- 1 root root      24 lis 18 00:04 curses.out
-rwxr-xr-x 1 root root 4787991 lis 17 23:42 opensips
-rwxr-xr-x 1 root root   57484 lis 17 23:42 opensipsctl
-rwxr-xr-x 1 root root    6480 lis 17 23:42 opensipsdbctl
-rwxr-xr-x 1 root root   18181 lis 17 23:42 opensipsunix
-rwxr-xr-x 1 root root   60314 lis 17 23:42 osipsconfig
-rwxr-xr-x 1 root root  203012 lis 17 23:42 osipsconsole
```

5.4.4 Logovací soubor

Soubor opensips.log je uložen mezi ostatními logovacími soubory jiných aplikací ve složce `/var/log`.

6 VÝKONOVÉ POROVNÁNÍ ÚSTŘEDEN

Následující kapitola je věnována popisu fungování a výkonovému srovnání ústředen Kamailio a OpenSIPS. Srovnání ústředen, jež proběhlo pomocí různých druhů měření, bylo rozděleno do 5 logických celků. Každý celek se věnoval jedné konkrétní problematice a to:

- **Popis funkce ústředen Kamailio a OpenSIPS (kapitola 6.2)** – V kapitole je popsán konfigurační soubour ústředen jimž se určuje požadovaná funkcionalita ústředen. Také se věnuje popisu zpracování SIP požadavku.
- **Paměťová náročnost ústředen (kapitola 6.3)** – Kapitola se věnuje popisu využití operační paměti, jež ústředny potřebují ke svému chodu. Měření pak kromě porovnání rozdílů mezi ústřednami popisuje chování ústředen při nedostatku volné paměti.
- **Rychlost odezvy ústředen na SIP dotazy (kapitola 6.4)** – Měření v kapitole popisuje, jakým způsobem ovlivňuje druh zapisování dat o uživateli rychlost odpovědi na daný SIP dotaz.
- **Výkonové porovnání ústředen (kapitola 6.5)** – Kapitola se kromě porovnání schopnosti obsloužení maximálního počtu registrací a sestavení spojení u ústředen Kamailio a OpenSIPS, věnuje i ukázce rozdílu výkonnosti mezi páteřními ústřednami, reprezentovanými Kamailio a OpenSIPS, a pobočkovými ústřednami reprezentovanými ústřednou Asterisk.
- **Ochrana ústředen proti napadení (kapitola 6.6)** – Kapitola se věnuje praktické ukázce rozdílu výkonnosti při výměně dat pomocí zabezpečeného protokolu TLS.

6.1 Testovací aplikace

K provedení výkonového porovnání ústředen bylo potřeba generovat a vyhodnocovat SIP požadavky. K těmto účelům byly využity aplikace SIPp, Wireshark a Spirent.

6.1.1 Aplikace SIPp

Jedná se o aplikaci volně šiřitelnou pod licencí GNU/GPL určenou pro generování a vyhodnocování SIP požadavků. Ty jsou generovány ze souborů formátu XML, ve kterých je definována přesná struktura odesílaných SIP požadavků, odpovědi které má aplikace zachytit a případně jak má na ně reagovat. Pro samotné testování lze využít předem definovaných xml scénářů, nebo lze vytvořit vlastní. Aplikace také vytváří statistiky o provedených testech, ze kterých lze vyčíst potřebné informace. Jako například počet generovaných dotazů za sekundu, počet správně přenesených nebo přijatých dotazů, rychlost odpovědí na dotazy atd. [15]

Níže je uvedený vzorový příkaz pro spuštění aplikace se všemi dodatečnými parametry, které byly pro samotné výkonové testování využity [16]:

```
./sipp [IP adresa:port] -sn/sf [scénář] -inf [soubor] -m [číslo] -r
[číslo] -fd [číslo] -rate_increase [číslo]  -rate_max [číslo] -
no_rate_quit -l [číslo] -rsa [IP adresa:port] -tls_cert [soubor] -
tls_key [soubor] -max_socket [číslo] -trace_stat -trace_screen
```

Podrobné vysvětlení příkazů:

`./sipp [IP adresa:port]` – Inicializace aplikace a určení koncového uzlu pomocí IP adresy a portu.

`-sn [scénář]` – Určení předem nadefinovaného scénáře. Příkazem lze například spustit UAC nebo UAS klienta.

`-sf [scénář]` – Určení uživatelem definovaného scénáře, kde se zadává cesta k xml souboru.

`-inf [soubor]` – Určení souboru obsahujícího měnící se informace vkládané do SIP požadavků, jako například název odesilatele nebo uživatelské jméno heslo potřebné pro registraci.

`-m [číslo]` – Určení maximálního počtu vyslaných SIP požadavků. Po jeho dovršení se aplikace ukončí.

`-r [číslo]` – Určení maximálního počtu SIP požadavků vyslaných za 1 sekundu.

`-fd [číslo]` – Určuje časovou periodu, při které se má zvýšit maximální vysílaný počet SIP požadavků.

`-rate_increase [číslo]` – Udává, o kolik SIP požadavků bude navýšeno zasílání při nové inkrementaci.

`-rate_max [číslo]` – Udává maximální počet vyslaných SIP požadavků.

`-no_rate_quit` – Příkaz zabraňuje ukončení aplikace při dosažení limitu udávajícím hodnota `-rate_max`

`-l [číslo]` – Určení maximálního počtu souběžně vyslaných SIP požadavků.

`-rsa [IP adresa:port]` – Příkaz se využívá pro scénáře s proxy serverem, kde se musí definovat IP adresa a port uzlu, který požadavky dále spracovává.

`-tls_cert [soubor]` – Udává cestu k certifikačnímu souboru potřebnému pro navázání TLS komunikace.

`-tls_key [soubor]` – Udává cestu ke klíči potřebnému pro navázání TLS komunikace.

`-max_socket [číslo]` – Příkaz se udává při navazování TCP komunikace, která je potřebná pro navázání TLS spojení. Příkaz udává maximální počet souběžně vytvořených komunikací

`-trace_stat` – Příkaz pro uložení vygenerovaných statistik ukončeného testování.

`-trace_screen` – Příkaz pro uložení výstupní informační tabulky, jež se v průběhu testu pravidelně zobrazuje

6.1.2 Wireshark

Wireshark je multiplatformní, volně šířitelný odchytávač síťové komunikace, jež je určený k analýze a ladění.

Program, jež vytvořil Gerald Combs v roce 1998 pod názvem Ethereal se v roce 2006 přejmenoval na Wireshark. Je volně šířitelný pod hlavičkou GNU/GPL licencí [17].

Mezi jeho základní funkce patří [18]:

- Odchytávání packetů ze síťových rozhranní (např. ethernet, wifi).
- Analýza dat odchycených různými aplikacemi jako například tcpdump/WinDump.
- Zobrazení detailních informací o protokolech nesených v packetech.
- Hledání a filtrování packetů podle zadaných kritérií.
- Tvorba statistik.

6.1.3 Spirent Test Center

Jedná se o modulární testovací platformu pro komplexní měření konvergovaných sítí určenou k testování cloudových služeb, mobilního páteřního spojeni a vysokorychlostního ethernetového připojení. Generuje a analyzuje data 2. až 7. vrstvy modelu ISO-OSI [19].

Mezi výhody testovací platformy patří:

- Možnost testování rychlostí v řádu 100 GBit/s.
- Testování různých přenosových prostředí (Ethernet, Fiber Channel, ATM…).
- Podpora IPv4 i IPv6.
- Možnost testování různých aplikačních protokolů (SIP, http atd.).
- Výpis výsledků v průhěbu testu.
- Ukládání datové komunikace pro možnost další analýzy (například programem Wireshark).

6.2 Popis funkce ústředen Kamailio a OpensIPS

Jelikož ústředny Kamailio a OpenSIPS fungují na podobném principu, v dalším textu bude běh obou aplikací popsán jednotně. Na případné rozdíly bude upozorněno vždy na konkrétním místě.

6.2.1 Hlavní konfigurační soubor

Konfigurace ústředen je definována v hlavním konfiguračním souboru (opensips.cfg, kamailio.cfg), jež se zadává do inicializačního souboru (kapitola 4.3 a 5.1). Soubor lze rozdělit do 3 hlavních sekcí a to:

- **Sekce globálních parametrů** – Zde se zadávají parametry ovlivňující chod celé aplikace. Jako například IP adresa a port, na kterých má aplikace naslouchat, počet dětských podprocesů, nebo debug mód.
- **Sekce modulů** – V dané sekci se definují všechny externí moduly pro inicializaci funkcionalit, které jádro nenabízí. Dále se zde definují specifické parametry modulů.
- **Směrovací logika** - Sekce, ve které je definováno chování aplikace při příchodu SIP požadavku

6.2.2 Zpracování SIP požadavků

Na hlavní konfigurační soubor lze také nahlížet jako na skript, který se vždy spustí pro každý příchozí SIP požadavek. Ten je pak vždy zpracován v sekci směrovací logiky a po vykonání všech definovaných úkonů je pak skript ukončen. Pro další příchozí SIP požadavek je pak vždy spuštěna nová instance konfiguračního souboru. Z toho důvodu jsou definovány 2 módy, ve kterém aplikace může fungovat. A to tzv. bezestavový a stavový mód.

Bezestavový mód

V bezestavovém módu ústředna zpracovává požadavky, aniž by znala jejich kontext. Pokud se například ústředna využívá pouze jako Registrar server bez nutnosti autentizace a přijde na ní SIP požadavek typu REGISTER, je spuštěn skript, který jej zpracuje podle směrovací logiky (pošle zpátky adresátovi potvrzovací odpověď, nebo například požadavek zahodí). Po té se skript ukončí a všechny interní informace o požadavku jsou vymazány. Pokud by ale registrar server využíval autentizaci, byla by adresátovi zaslána zpráva s hodnotou nonce potřebnou pro výpočet hash kódu. Po odeslání by ale hodnota nonce byla smazána a tudíž by při dalším přijetí ústředna nedokázala ověřit správnost výpočteného hashe

Stavový mód

Ve chvíli kdy jsou potřeba funkce jako účtování, či směrování hovorů, je potřeba aby ústředna znala kontext přijatých SIP zpráv a například dokázala správnému požadavku INVITE přiřadit odpovídající odpověď 200 OK, musí pracovat v tzv. stavovém módu. Což znamená, že stejně jako pro bezestavový mód je pro každý SIP požadavek vyvolán skript, ale po jeho provedení nejsou interní informace zahozeny.

Funkce stavového módu lze popsat 5 kroky [20]:

1. Validace požadavku – Kontrola pole max-forwards pro zabránění smyček, kontrola syntaxe, nebo kontrola velikosti, kvůli zabránění přeplnění zásobníku
2. Předzpracování směrovacích informací – Pokud existuje, zpracovává informace z pole záhlaví record-route.
3. Vyhledání cíle požadavku – Prohledávání lokační databáze, případně hledání výstupního směru.
4. Vybavení požadavku – Je v plné režii ústředny. Modifikace hlavičky SIP dotazu (např. přidání via pole, inkrementace max-forwards).

5. Zpracování odpovědí – V plné režii ústředny. Například při příchodu odpovědi 487 Busy přesměrování na voicemail server.

6.3 Pameťová náročnost ústředen

Ústředny Kamailio a OpenSIPS využívají 2 druhy paměti. A to paměť rezervovanou a paměť sdílenou. V následující podkapitole jsou popsány oba druhy pamětí a dále byla provedena měření, kde:

- **Měření č.1** – Zkoumá velikost využité sdílené paměti při zápisu účastníka do lokační tabulky.
- **Měření č.2** – Zkoumá chování stanic při překročení zvolené velikost sdílené paměti bez využití zápisu do database.
- **Měření č.3** – Zkoumá chování stanic při překročení zvolené velikosti sdílené paměti při využití kombinovaného zápisu do databáze.
- **Měření č.4** – Zkoumá chování stanic při překročení zvolené velikost sdílené paměti pri využití ikamžitého zápisu do database.
- **Měření č. 5** – Zkoumá paměťové nároky ústředen při sestavování telefonního spojení.

6.3.1 Paměť rezervovaná

Jedná se o velikost paměti, kterou ústředna využívá pro svůj chod. Její velikost není konstantní a v průběhu doby se mění. Po startu aplikace je velikost využité paměti závislá hlavně na počtu načtených modulů, jež jsou definované v konfiguračním souboru. Pro názornost byly spuštěny ústředny s konfigurací bez načtených externích modulů, s upravenou konfigurací pro test lokačního serveru a se základní konfigurací vytvořenou při instalaci systému. Výsledek měření je vypsán v tabulce č. 6.1. Pro názornost je uvedena i hodnota rezervované paměti ústředny Asterisk po instalaci bez zásahu do konfiguračního souboru.

Jak lze vidět, ústředny Kamailio a OpenSIPS nepotřebují velkou rezervovanou paměť. Jako základní byly vybrány vzorové konfigurace, jež byly obsaženy v adresáři ústředen po jejich instalaci. Rozdíl ve velikosti u základní konfigurace a nastavením pro lokační server ústředny OpenSIPS je dán faktem, že v základu konfigurace neobsahovala modul pro práci s databázemi db_mysql.so.

Tab. 6.1: Velikost využité paměti při startu ústředen

	Kamailio	OpenSIPS	Asterisk
Minimální konfigurace	1,2 MB	0,96 MB	-
Lokační server	4,5 MB	4 MB	-
Základní konfigurace	4,6 MB	3 MB	22,3 MB

6.3.2 Paměť sdílená

Sdílenou paměť využívají ústředny pro data, které využívají i jiné aplikaci, hlavně databáze. Maximální velikost paměti se určuje v inicializačním souboru a to:

Ústředna Kamailio v souboru `/etc/init.d/kamailio`, kde lze modifikovat řádek:

```
OPTIONS="-P $PIDFILE -m $S_MEMORY -M -u $USER -g $GROUP -f
/usr/local/etc/kamailio/kamailio.cfg"
```

Ústředna OpenSIPS v sobouru /etc/init.d/opensips, kde lze modifikovat řádek:

```
OPTIONS="-P $PIDFILE -m $S_MEMORY -M -u $USER -g $GROUP -f
/usr/local/etc/opensips/opensips.cfg"
```

Kde změnou hodnoty `$S_MEMORY` za parametrem –m určíme velikost sdílené paměti v MB.

Stejně jako paměť rezervovaná není od startu aplikace paměť plně využita, ale narůstá podle počtu uložených dat. Velikost sdílené paměti ovlivňuje chod ústředny, a proto musí být zvolena správná velikost. Podrobněji je sdílená paměť vysvětlena na příkladu funkce lokačního serveru.

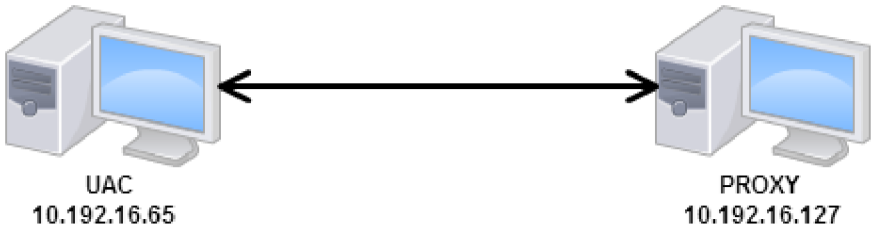
Jak je uvedeno v kapitole 1.3.1 spravuje lokační server databázi připojených uživatelů dané domény a v případě příchozí SIP zprávy, jež patří uživateli dané domény, vyhledává jeho polohu. Záznamy o těchto uživatelích jsou uloženy v databázové tabulce location. Při inicializaci ústředen lze zvolit 4 druhy zápisu pomocí změny parametru db_mode modulu usrloc. A to:

- **Mód 0** – Modul ukládá data pouze do sdílené paměti a zápis do databáze je zakázaný. Při ukončení chodu ústředny jsou všechna uložená data vymazána
- **Mód 1** – Modul všechna data ihned ukládá do databáze. Při ukončení chodu ústředny nejsou data ztracena.
- **Mód 2** – Ústředna všechna data určená pro lokační modul nejdříve ukládá do sdílené paměti a pak postupně data kopíruje do lokační tabulky databáze.
- **Mód 3** – V daném módu ústředna využívá data pouze uložená v tabulce databáze. Všechny příchozí požadavky jsou zakázány k uložení.

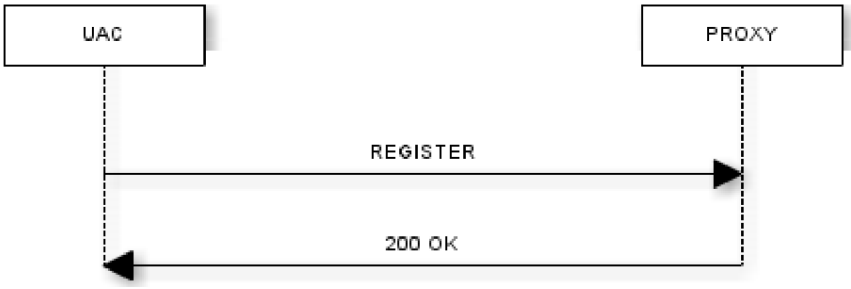
6.3.3 Měření č.1

Zapojení a základní konfigurace

Pro potřeby měření kromě PC s nainstalovanými ústřednami bylo využito další PC (viz. Obr. 6.1) kde se pomocí programu SIPp jako User Agent Client (UAC) zasílaly SIP požadavky REGISTER (Příloha D.3), na které ústředna po zapsání do tabulky reagovala kladnou odpovědí 200 OK (viz. Obr. 6.2). Pro zápis do lokační tabulky bylo zvolen mód 0. Hodnota sdílené paměti byla nastavena na velikost 128 MB. Měřila se velikost využité paměti pro 1000 až 100000 registrujících se uživatelů. Byl využita konfigurace s minimálním kódem (Přílohy A.1 a B.1).



Obr. 6.1: Zapojení pro testování paměťové náročnosti ústředen



Obr. 6.2: Diagram zasílaných SIP požadavků a odpovědí

Výsledky

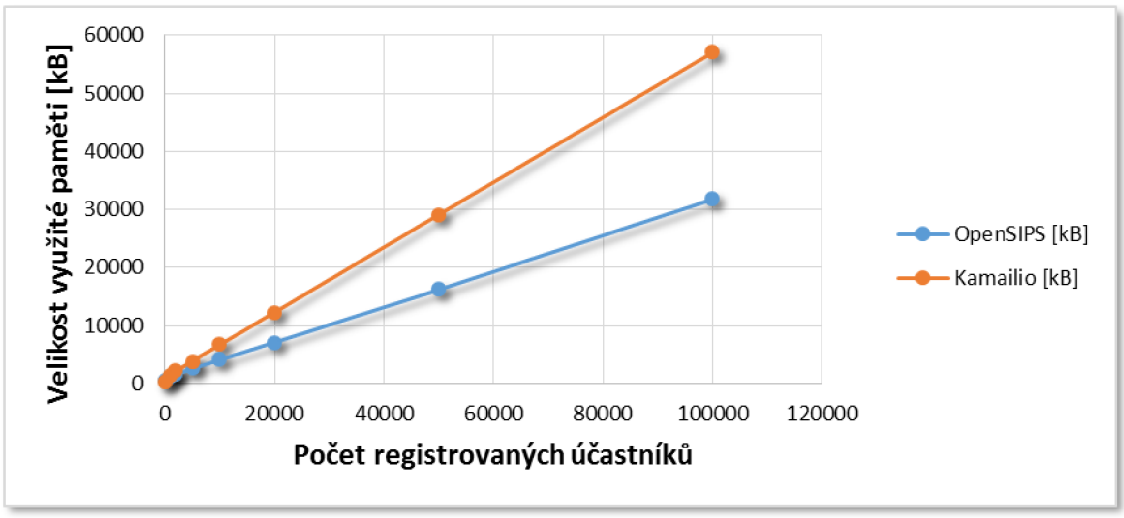
Z hodnoty výsledků dat uložených ve sdílené paměti pro různý počet požadavků lze vyčíst z tabulky č. 6.2, že pro 100000 uživatelů potřebuje ústředna OpenSIPS velikost paměti minimálně 31 MB a ústředna Kamailio velikost 57 MB.

Tab. 6.2: Velikost využité paměti na počtu záznamů v lokační tabulce

Počet uživatelů	0	1000	2000	5000	10000	20000	50000	100000
OpenSIPS [kB]	508	1244	1508	2564	4140	7048	16288	31860
Kamailio [kB]	200	1428	2220	3804	6708	12248	29140	57124

Z grafu 6.1 lze vypozorovat, že čím větší počet uživatelů je registrováno, lineárně narůstá využití sdílené paměti. Velikost využité paměti je závislá na počtu záznamů, ale také na velikosti samotného záznamu, která se může měnit podle počtu ukládaných hodnot. Na obr. 6.3 lze vidět výpis z databáze lokační tabulky obou ústředen. Z ní je patrné, že ústředna Kamailio (vrchní tabulka) ukládá více hodnot než OpenSIPS (spodní tabulka), z čehož vyplívá větší nárok na velikost paměti (viz. Tab. 6.2).

Graf 6.1: Využitá paměť ústředny při registraci účastníků do lokační databáze

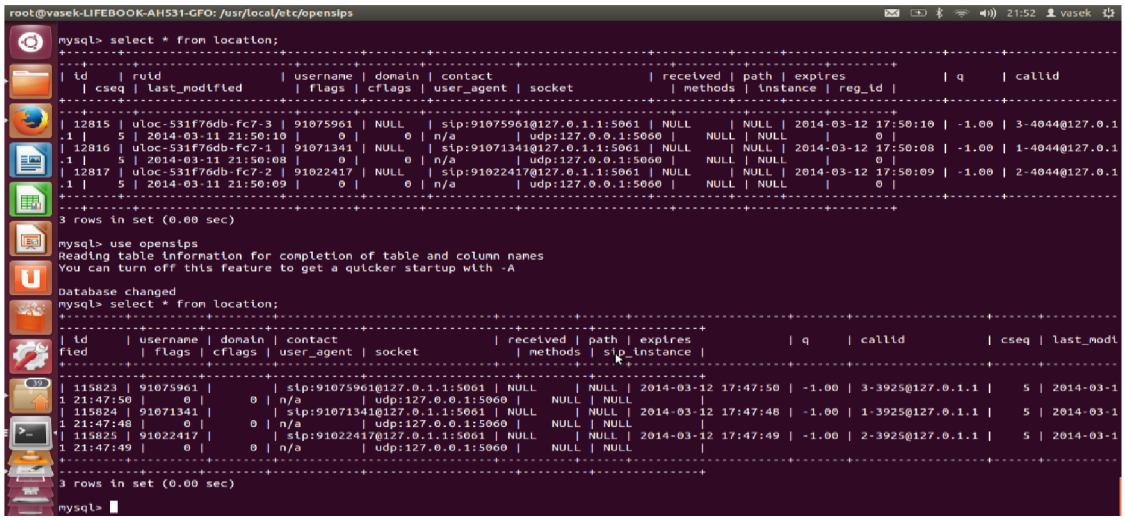


6.3.4 Měření č. 2

Z tabulky lze vyčíst, že pro nastavenou velikost paměti 12 MB by ústředna kamailio měla schopná vést kolem 20000 záznamů a OpenSIPS 35000. Proto v dalším měření se zkouší chování ústředen při příchodu většího počtu požadavků na zaregistrování do lokační tabulky.

Zapojení a základní konfigurace

Zapojení zůstalo stejné jako v měření č. 1 (viz. obr. 6.1), změněno bylo nastavení sdílené paměti na hodnotu 12 MB.



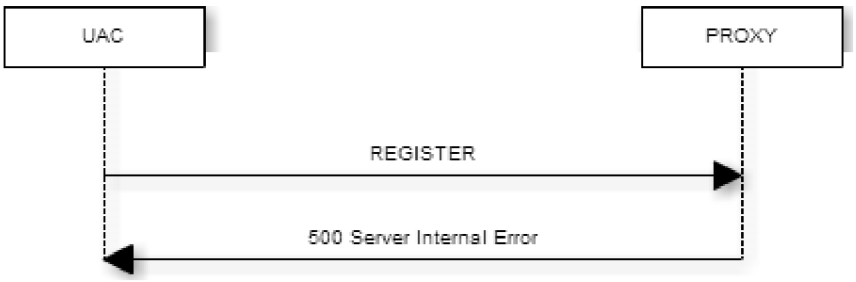
Obr. 6.3: Výpis z lokačních tabulek ústředen MySQL databáze

Výsledky

Do chvíle naplnění sdílené paměti se ústředny chovaly stejně jako v měření č. 1. Tudíž velikost sdílené paměti lineárně rostla. Ve chvíli naplnění tabulky se začali příchozí

požadavky odmítat.

A to tak, že zdroji požadavku nebyla poslána odpověď 200 ok (viz obr. 6.2) ale odpověď 500 Server Internal Error (viz. Obr. 6.4). Zároveň bylo odmítnutí požadavku zapsáno do logovacího souboru, jako je tomu na obr. 6.5 kde je výpis z logovacího souboru ústředny OpenSIPS.



Obr. 6.4:Odmítnutí registrace při zaplněné sdílené paměti ústředny

```
Mar 12 14:34:38 vasek-LIFEBOOK-AH531-GFO /usr/local/sbin/opensips[5818]: ERROR:usrloc:new_urecord: no more share memory
Mar 12 14:34:38 vasek-LIFEBOOK-AH531-GFO /usr/local/sbin/opensips[5818]: ERROR:usrloc:mem_insert_urecord: creating urecord failed
Mar 12 14:34:38 vasek-LIFEBOOK-AH531-GFO /usr/local/sbin/opensips[5818]: ERROR:usrloc:insert_urecord: inserting record failed
Mar 12 14:34:38 vasek-LIFEBOOK-AH531-GFO /usr/local/sbin/opensips[5818]: ERROR:registrar:insert_contacts: failed to insert new record structure
Mar 12 14:34:38 vasek-LIFEBOOK-AH531-GFO /usr/local/sbin/opensips[5818]: WARNING:core:fm_malloc: Not enough free memory, will atempt defragmenation
```

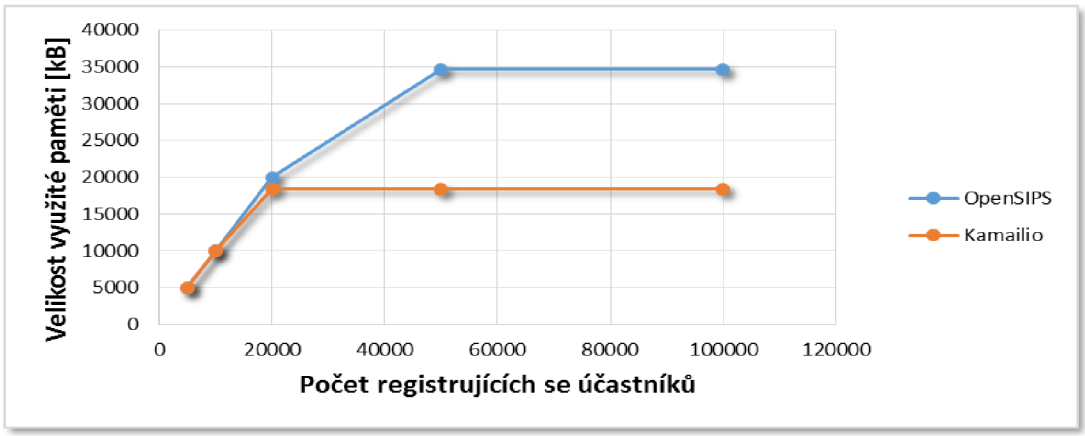
Obr. 6.5: Výpis chybové hlášky při odmítnutí registrace z důvodu zaplněné paměti u OpenSIPS

Pak je patrné, že velikost záznamu v lokační tabulce je pevně daná a ve chvíli dosažení maximální hodnoty sdílené paměti, jsou všechny další požadavky na zapsání odmítnuté (viz Tab. 6.3 a Graf 6.2)

Tab. 6.3: Počet uložených registrací při velikosti sdílené paměti 12 MB

Počet uživatelů	5000	10000	20000	50000	100000
OpenSIPS	5000	10000	20000	34688	34688
Kamailio	5000	10000	18419	18419	18419

Graf 6.2: Počet uložených registrací při velikosti sdílené paměti 12 MB



6.3.5 Měření č. 3

Měření bylo provedeno při rychlém příchodu velkého počtu požadavků a při pomalém příchodu a zkoumal se vliv zaregistrovaných uživatelů.

Zapojení a základní konfigurace

Zapojení zůstalo stejné jako při měření č.1. Hodnota sdílené paměti byla nastavena na 12 MB a pro zápis do lokační tabulky byl zvolen mód 2.

Výsledky

Schéma 2 kombinuje přednosti zápisu přímo do databáze a pouze do sdílené paměti. A to tak, že data načte do sdílené paměti, se kterou dále pracuje a podle nastaveného časovače zkopíruje všechna data do tabulky v MySQL databázi. Kopírování do databáze není spuštěno okamžitě a proto při rychlém příchodu velkého počtu požadavků se sdílení paměť ústředen naplní a další požadavky jsou odmítány stejně jako při měření č.2. Úkolem měření bylo zjistit, jestli se po zapsání dat do databáze informace ze sdílené paměti odstraní a uvolní se paměť pro nově přicházející požadavky. Jak je na tabulce 6.4 vidět, záznamy z paměti odstraněny nejsou. To se děje pouze ve chvíli, kdy záznamům vyprší doba registrace. Z toho důvodu jsou výsledky obou měření stejné. U ústředny OpenSIPS proběhlo pouze jedno měření, protože při naplnění sdílené paměti a posílání negativních odpovědí rapidně klesl počet přijmutých SIP požadavků za sekundu a tudíž měření probíhalo i ve chvíli, kdy již byla data kopírována do databáze.

Tab. 6.4: Počet uložených záznamů do lokační tabulky při ukládání v módu 2

Počet požadavků	Kamailio - rychlý příchod	Kanailio - pomalý příchod	OpenSIPS
100000	18419	18418	34687

6.3.6 Měření č. 4

V posledním ze sady měření byl zkoumán vliv okamžitého zápisu dat do tabulky databáze na velikost sdílené paměti a počtu zaregistrovaných účastníků.

Zapojení a základní konfigurace

Zapojení bylo stejné jako v předešlých měřeních a hodnota sdílené paměti byla nastavena na velikost 12 MB a bylo provedeno jedno měření, při kterém bylo vysláno 100000 požadavků na zapsání do lokační tabulky databáze.

Výsledky

Měření mělo dokázat, zda-li stejně jako v předešlých měřeních, je určujícím faktorem pro počet uložených registrací velikost sdílené paměti, nebo zda-li ústředny pracují přímo s daty uloženými v databázi. Podle výsledku měření v tabulce 6.5 je patrné, že i při okamžitém zápisu do tabulky databáze lze uložit pouze takový počet registrací, co povolí velikost sdílené paměti.

Tab. 6.5: Počet uložených záznamů do lokační tabulky při ukládání v módu 1

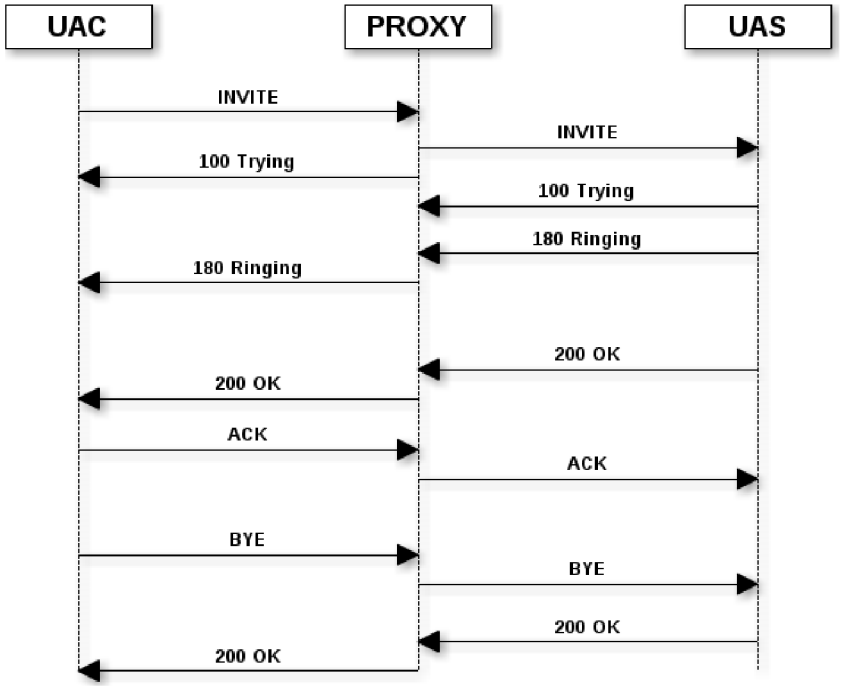
Počet požadavků	Kanailio	OpenSIPS
100000	18418	34687

6.3.7 Měření č. 5

Nejdůležitější funkcí ústředen je spojování hovorů. Tenhle proces je prováděn ve stavovém módu, při kterém musí být informace po dobu sestaveného hovoru uloženy v paměti programu. Z toho důvodu lze předpokládat, že největší paměťové nároky budou kladeny při sestavování hovorů.

Zapojení a základní konfigurace

K měření byl kromě PC s ústřednami využit další PC na kterém běžel jak UAC (Příloha D.1), který zasílal SIP zprávy, tak User Agent Server (UAS), jež na ně odpovídal (Příloha D.2). Schéma sestavení spojení lze vidět na obrázku 6.2. Počet simulovaných hovorů byl stanoven na 100, 900, 1800 a 2500 hovorů za 1 vteřinu, kdy měření bylo po 100000 spojeních ukončeno. Sekvence SIP požadavků užitých v měření je vidět na obrázku 6.6. Velikost sdílené paměti byla nastavena na velikost 512 MB a konfigurační soubor byl využit s minimálním kódem (Přílohy A.2 a B.2).



Obr. 6.6: Diagram spojení hovoru pomocí proxy serveru

Výsledky

Z výsledného tabulky 6.6 a grafu 6.3 jde vypozorovat, že ústředna Kamailio spotřebuje výrazně méně společné paměti, než ústředna OpenSIPS.

A čím větší počet obsloužených hovorů, tím je paměťový rozdíl výraznější. Pro samotné měření bylo využito minimálního kódu a doplňkové funkce jako účtování hovorů, které by výslednou paměť ovlivnily, nebyly využity.

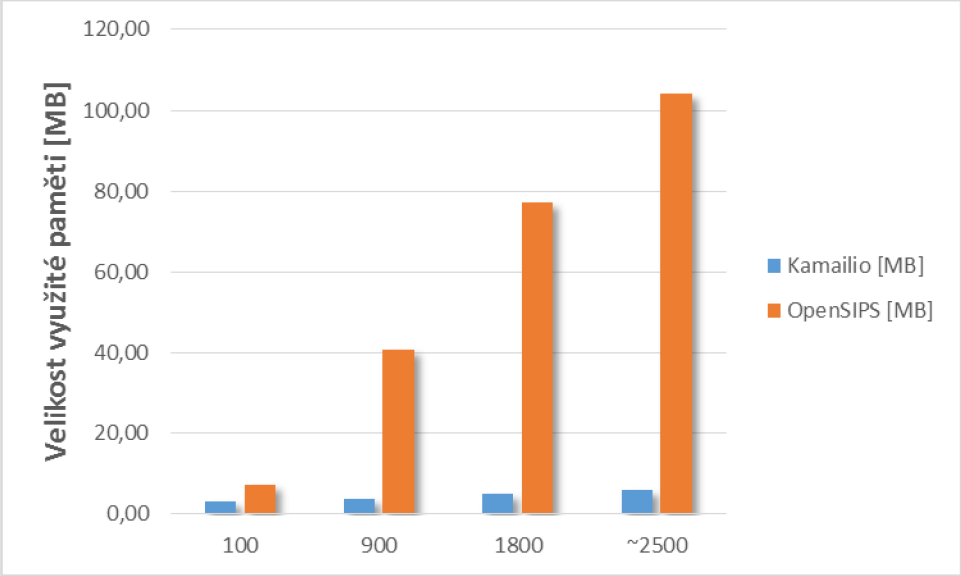
Tab. 6.6: Velikost využité paměti na počet hovorů za vteřinu

Počet hovorů za vteřinu	Kamailio [MB]	OpenSIPS [MB]
100	3,00	7,30
900	3,80	40,80
1800	5,10	77,20
~2500	5,90	104,00

6.3.8 Shrnutí

Jak je z měření patrné, velikost zvolené sdílené paměti hraje zásadní roli při chodu ústředen, jelikož veškerá pracovní data jsou v ní uložena. Databáze pak slouží jako záloha pro ztrátu dat při výpadku ústředny. Pokud by se při návrhu ústředny hledělo pouze na paměťovou náročnost, pak se jako nejideálnější volba jeví mód 1, při kterém se změny okamžitě zapisují do databáze. Měření dále ukázaly, že samotná funkce lokačního serveru není paměťově tolik náročná, jelikož pro 18 respektivě 35 tisíc uživatelů na doménu spadající pod lokační server je parametr 12 MB paměti splnitelný pro všechen dostupný hardware. Při daném počtu uživatelů je výpočetní výkon důležitější než velikost paměti.

Graf 6.3: Velikost využité paměti na počet hovorů za vteřinu



Měření paměťové náročnosti sestavení telefonního spojení již ukázaly větší rozdíl mezi ústřednami, kdy Kamailio na stejný počet sestavených hovorů potřebovala výrazně méně paměti. Nepoměr se ale projevil až při vysokém počtu hovorů.

Měření byla navrhnuta pro možnosti srovnání ústředen a pro popis jejich chování. Proto konfigurační soubory obsahovaly minimální počet funkcí. V reálném provozu by velikost sdílené paměti v poměru k přihlášeným uživatelům byla větší a to z důvodu většího počtu potřebných informací, které by pro svou práci další moduly potřebovaly ukládat.

6.4 Rychlost odezvy ústředen na SIP dotazy

Rychlost reakce na SIP dotazy patří mezi důležité parametry ústředen. Jelikož ústředny zpracovávají pouze signalizaci a nikoliv samotný mediální tok, nemá větší doba reakce vliv na kvalitu hovoru.

Při přijmutí požadavku musí ústředna provést předem definovanou posloupnost kroků podle druhu požadavku. Z toho důvodu je doba odezvy na každý požadavek individuální a záleží na samotné implementaci použitých metod. Následující měření mají dokázat rozdíl latence při různých implementacích a různém výkonovém zatížení ústředen.

6.4.1 Měření č. 6

V minulé kapitole byla provedena měření za účelem zjištění paměťových potřeb ústředen. Z měření vyplynulo, že mezi okamžitým zápisem do databáze a zápisem pouze do sdílené paměti není žádný rozdíl a tudíž by se dalo předpokládat, že bude výhodnější využít přímého zápisu do databáze. Při přímém zápisu do databáze je ale potřeba většího počtu operací (vytvoření SQL dotazu, zpracován dotazu databází…), než při přímém přístupu do sdílené paměti. Z toho důvodu by se dalo předpokládat, že rychlost odezvy bude vyšší.

Zapojení a základní konfigurace

Zapojení bylo využito jako v měření č. 4. V prvním kroku měření bylo na ústřednu vysláno 100 000 SIP požadavků REGISTER (Příloha D.3) a byla naplněna lokační tabulka. Velikost sdílené paměti byla nastavena na hodnotu 128 MB a mód ukládání nastaven na hodnotu 2. Po zkopírování všech dat do databáze bylo provedeno první měření, kdy na ústřednu bylo zasláno 30 000 požadavků REGISTER, které byly náhodně zvoleny z množiny uložených registračních záznamů. Po provedení měření byl změněn mód ukládání na hodnotu 1, restartovala se ústředna a měření se provedlo znovu.

Výsledky

Průběh měření u ústředny OpenSIPS proběhl bez problémů a výsledky na něž odkazují obrázek 6.7 a tabulka 6.7 dokazují, že odpověď ústředny při práci se sdílenou pamětí je výrazně rychlejší, než s přímou práci s databází.

```
Average Response Time Repartition 1
         0 ms <= n <        10 ms :      29717
        10 ms <= n <        50 ms :         92
        50 ms <= n <       100 ms :          0
       100 ms <= n <       150 ms :          0
       150 ms <= n <       200 ms :          0
       200 ms <= n <       500 ms :          0
       500 ms <= n <      1000 ms :        179
              n >=      1000 ms :         12
Average Call Length Repartition
```

a

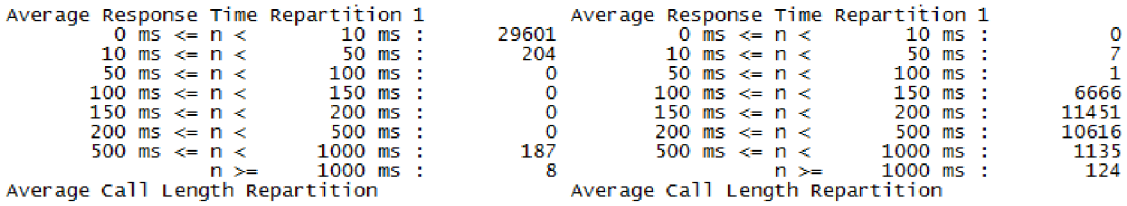
```
Average Response Time Repartition 1
         0 ms <= n <        10 ms :         75
        10 ms <= n <        50 ms :      20254
        50 ms <= n <       100 ms :       2395
       100 ms <= n <       150 ms :       1849
       150 ms <= n <       200 ms :       1386
       200 ms <= n <       500 ms :       3460
       500 ms <= n <      1000 ms :        466
              n >=      1000 ms :        115
Average Call Length Repartition
```

b

Obr. 6.7: Doba odpovědi ústředny OpenSIPS při zpracování požadavku v módu 2 (a) a v módu 1 (b)

Měření u ústředny Kamailio ukázalo (obr. 6.8), že zatím co rychlost odpovědí při práci se sdílenou databází byla nepatrně rychlejší, než u OpenSIPS (viz. Tab. 6.7), při

práci s databází nastal extrémní pokles výkonu systému (viz obr. 6.9), kde hodnota call per second (cps) ukazuje počet vyřízených požadavků za sekunudu, pro který se musela snížit rychlost zasílání požadavků, protože ústředna nebyla schopná obsloužit víc jak 30 požadavků za sekundu a maximálně 5 shodně příchozích požadavků (výkonová porovnání jsou řešeny v kapitole 6.5). Díky tomu nastalo i rapidní zpomalení odezvy (viz. Tabulka 6.7).



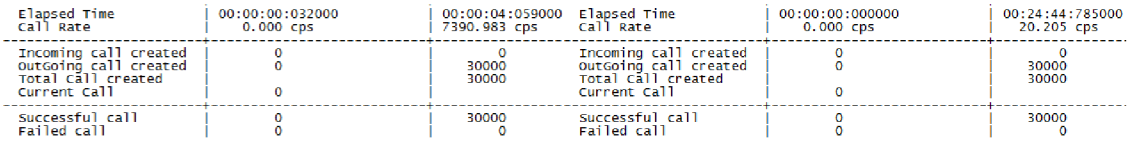
```
Average Response Time Repartition 1
        0 ms <= n <        10 ms :       29601
       10 ms <= n <        50 ms :         204
       50 ms <= n <       100 ms :           0
      100 ms <= n <       150 ms :           0
      150 ms <= n <       200 ms :           0
      200 ms <= n <       500 ms :           0
      500 ms <= n <      1000 ms :         187
            n >=      1000 ms :           8
Average Call Length Repartition
```

a

```
Average Response Time Repartition 1
        0 ms <= n <        10 ms :           0
       10 ms <= n <        50 ms :           7
       50 ms <= n <       100 ms :           1
      100 ms <= n <       150 ms :        6666
      150 ms <= n <       200 ms :       11451
      200 ms <= n <       500 ms :       10616
      500 ms <= n <      1000 ms :        1135
            n >=      1000 ms :         124
Average Call Length Repartition
```

b

Obr. 6.8: Doba odpovědi ústředny Kamailio při zpracování požadavku v módu 2 (a) a v módu 1 (b)



Elapsed Time Call Rate	00:00:00:032000 0.000 cps	00:00:04:059000 7390.983 cps
Incoming call created	0	0
OutGoing call created	0	30000
Total Call created		30000
Current Call	0	
Successful call	0	30000
Failed call	0	0

a

Elapsed Time Call Rate	00:00:00:000000 0.000 cps	00:24:44:785000 20.205 cps
Incoming call created	0	0
OutGoing call created	0	30000
Total Call created		30000
Current Call	0	
Successful call	0	30000
Failed call	0	0

b

Obr. 6.9: Rychlost registrace při ukládání dat do lokační tabulky při módu 2 (a) a při módu 1 (b)

Tab. 6.7: Rychlosti odezvy ústředen při různých způsobech ukládání do lokační tabulky

	OpenSIPS	Kamailio
Mód 2	9,1 ms	9,03 ms
Mód 1	95,55 ms	229,2 ms

Výsledky jako takové neukazují přesnou dobu odezvy ústředen, jelikož jsou v celkové době započítány i vlivy přenosového prostředí, jako délka kabelu, čas zpracování síťových prvků a další. Jelikož ale vlivy byly konstantní pro všechna měření, nezkreslují porovnání výsledků a ukazují dobu odezvy v reálnějším prostředí, než kdybychom počítali pouze samostatnou dobu odpovědi.

6.5 Výkonové porovnání ústředen

Z obrázku 6.8 je patrné, že rychlost obsloužených požadavků patří mezi nejdůležitější parametry ústředen. Ústředna může držet záznamy velkého počtu registrovaných účastníků při poměrně malé využité paměti, ale bez chybějícího výkonu, který zajistí dostatečný počet obsloužených požadavků, je velký počet účastníků v podstatě zbytečný údaj.

Následující sada měření se zabývá zjištěním výkonových potřeb ústředen a porovnává je s výkonem ústředny Asterisk, kde:

- **Měření č. 7** – Zkoumá maximální počet generovaných SIP dotazů ústřednami.
- **Měření č. 8** - Zkoumá maximální počet autentizovaných registrací za sekundu.
- **Měření č. 9** – Zkoumá vliv velikosti konfiguračního souboru na výkon ústředen.
- **Měření č. 10** – Zkoumá maximální počet spojitelných hovorů ústředen bez autentizované registrace.
- **Měření č. 11** – Zkoumá maximální počet spojitelných hovorů ústředen s autorizovanou registrací.

6.5.1 Měření č. 7

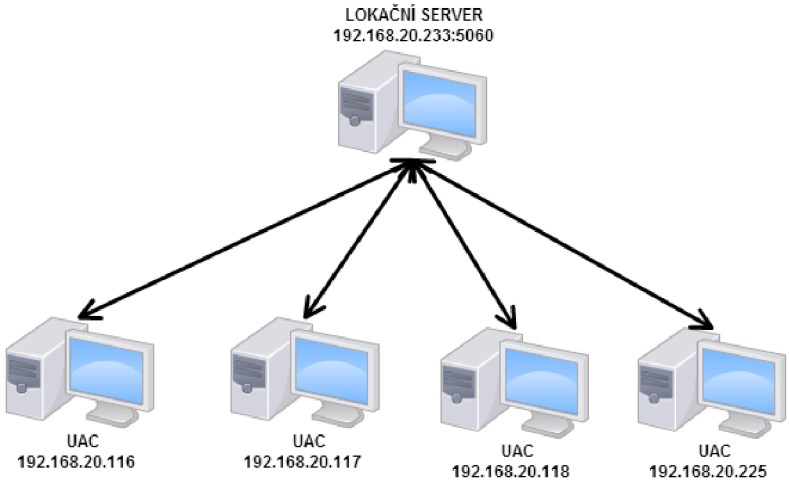
Prvním výkonové měření bylo navrhnuté na zjištění maximálního počtu odpovědí generovaných na příchozí SIP dotazy. V kapitole 6.2 je popsáno, že pro každý příchozí SIP požadavek je spuštěn skript, jež vykoná odpovídající funkci, na kterou v drtivé většině vygeneruje SIP zprávu s danou odpovědí. V následujícím měření byl použit dotaz REGISTER (Příloha D.3), pro aktualizaci lokační tabulky, na který byla generována odpověď 200 OK. Jelikož se jedná o bezstavovou operaci, jež patří mezi základní, bude počet obsloužených zpráv nejvyšší.

Zapojení a základní konfigurace

Pro generování SIP požadavků byly využité až 4 PC (viz obr. 6.10) na kterých byly spuštěny až 3 instance programu sipp, jež pomocí SIP požadavku REGISTER (viz obr. 6.2) aktualizovali svoji registraci v lokační tabulce. Pro měření byly využity konfigurační soubory s minimálním kódem. Lokační tabulka byla naplněna a pro práci s daty byl zvolen mód 0. Velikost sdílené paměti byla nastavena na hodnotu 512 MB. Počet child procesů byl postupně nastaven na hodnoty 1, 2 a 4.

Výsledky

Jako první byly měřeny ústředna Kamailio a OpenSIPS s nastaveným 1 child procesemem. Ten udává, kolik podprocesů bude obsluhovat příchozí požadavky. Využití 2 jádrohového procesoru v operačním systému Ubuntu je rozděleno díky technologii hyper-threading na 4 vlákna. Každý proces je obsluhován maximálně 1 vláknem, což znamená, že při maximálním využití 1 child procesu bude celkově využito pouze čtvrtina celkového výkonu. Měření bylo provedeno na 2 PC, kde generovaly data 2 instance programu sipp. Jak lze vyčíst z tabulky, ústředna OpenSIPS dokázala obsloužit jednou tolik příchozích požadavků.



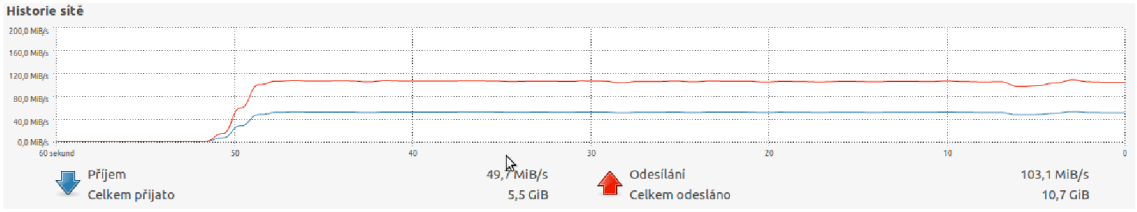
Obr. 6.10: Zapojení měření pro zjištění maximálního počtu odpovědí

Tab. 6.8: Počet registrací za sekundu v závislosti na počtu child procesů

Child Procesy	Kamailio	OpenSIPS
1	25067	52431
2	31801	92066
4	60504	115621

Pro 2 child procesy se dá předpokládat, že bude výkon vyšší, než při využití 1 child procesu. A to díky využití 2 vláken procesoru. Jak lze ale z tabulky vyčíst, nárust obsloužených procesů není lineární. Menší nárust je zapříčiněn režií při rozdělování zátěže mezi procesy ústředen. Zatím co u ústředny OpenSIPS není snížení počtu obsloužených požadavků tak výrazný, u ústředny Kamailio není nárust počtu obsloužených požadavků vysoký. Pro měření byly opět využity 2 PC. Pro ústřednu Kamailio byly na PC puštěny 2 instance sipp. Z důvodu vysokého počtu obsloužených požadavků bylo pro měření výkonu ústředny OpenSIPS využity 3 instance programu sipp.

Pro měření maximálního počtu obsoužitelných požadavků bylo potřeba spustit 4 child procesy, kdy každý proces je obsluhován jedním vláknem procesoru. Nárust požadavků u ústředny Kamailio byl očekávaný u ústředny OpenSIPS již pouze třetinový nárust ne. Menší počet obsloužených požadavků může být zapříčiněný zahlcením počítačové sítě. Na obr. 6.11 Je zachycený výpis zatížení systému ústředny při testu ústředny se spuštěným 4 child procesy. Jak lze vidět na síťovém zatížění, hlavně v odchozím směru je generovaná zátěž 103,1 MB/s, což při přepočtu je přibližně 845 Mbit/s. Díky tak vysoké zátěži již může vznikat spoždění zapříčiněné čekáním zpráv ve frontě vyrovnávací paměti ať už síťové karty ústředny, prvku generujícím SIP požadavky, nebo v přepínači.



Obr. 6.11: Zatížení sítě při měření ústředny OpenSIPS se 4 child procesy

Rozdíl mezi přijímanými a odesálanými daty je zapříčiněn velikostí odpovědí. Z obr. 6.12 lze vyčíst, že odchozí odpověď generovaná ústřednou OpenSIPS je jednou tak velká jako příchozí požadavek. Je to z toho důvodu, že ve správě 200 OK se přenáší informace na kterých IP adresách a portech byl uživatel s daným telefoním číslem zaregistrován a jak dlouho registrace ještě poběží. Díky testu, kdy se 12 různých instancí SIPp přihlašovalo pod stejným uživatelským jménem a heslem je pak v takové zprávě záznamů 12 (12 bindings v obr. 6.12). Čím víc záznamů, tím větší je pak samotná zpráva.

No.	Time	Source	Destination	Protocol	Length	Info
30	15.468282	192.168.20.118	192.168.20.233	SIP	455	Request: REGISTER sip:192.168.20.233:5060
31	15.468531	192.168.20.233	192.168.20.118	SIP	984	Status: 200 OK (12 bindings)

Frame 30: 455 bytes on wire (3640 bits), 455 bytes captured (3640 bits)
Frame 31: 984 bytes on wire (7872 bits), 984 bytes captured (7872 bits)

Obr. 6.12: Velikosti rámců při registraci do lokační tabulky ústředny

Zatím co pro měření výkonu u ústředny Kamailio byly využity 2 PC se spuštěnými 3 instancemi programu SIPp, pro měření ústředny OpenSIPS byly využity 4 PC.

Jelikož při měření při 4 child procesech bylo u ústředny Kamailio vytížení procesu maximální, existoval předpoklad, že při využití více child procesů již nebude počet obsloužených SIP požadavků stoupat. Z toho důvodu bylo pro potvrzení hypotézy provedeno měření, kdy pro ústřednu Kamailio bylo nastaveno počet child procesů na hodnotu 16. Jak lze vidět v tabulce 6.9, byl předpoklad správný.

Tab. 6.9: Počet registrací za sekundu ústředny Kamailio

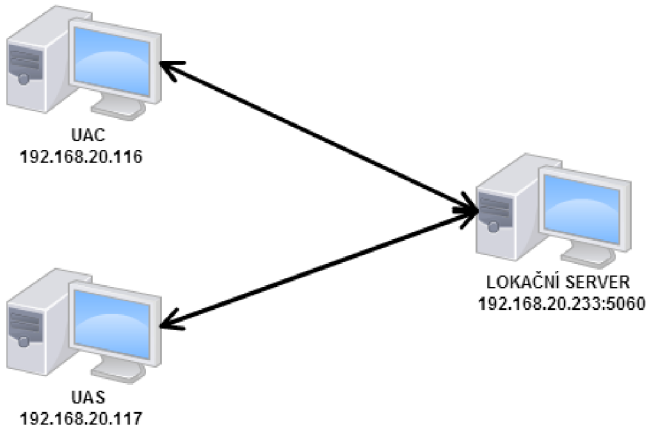
	4 Child procesy	16 Child procesů
Počet SIP požadavků	60504	60199

6.5.2 Měření č. 8

Měření č. 7 mělo ukázat maximální možný počet vygenerovaných zpráv. Další měření již prezentuje reálnější příklad, kdy ústředny se chovají jako Registrar servery na které se přihlašují účastníci. V tabulce účastníků jsou uložené přihlašovací údaje s heslem, z něhož a náhodně vygenerované posloupnosti nonce je vypočítán kontrolní hash řetězec, jež je porovnán s řetězcem, který při přihlášení zašle ústředně účastník. Aby mohl účastník poslat správný řetězec, musí znát posloupnost nonce. Ta je mu zaslána prostřednictvím odpovědi unauthorized, jež je vygenerována při prvním SIP požadavku REGISTER (viz. Obr 6.14). Jelikož si po odeslání odpovědi musí ústředna hodnotu nonce pamatovat i po ukončení skriptu, pracuje ústředna ve stavovém módu.

Zapojení a základní konfigurace

Pro generování SIP požadavků byly využité 2 PC (viz obr. 6.13) jež pomocí SIP požadavku REGISTER (Příloha D.4), se přihlašovali k registrar serveru. Pro měření byly využity konfigurační soubory s minimálním kódem. Registrar tabulka byla naplněná 100 uživateli, jež byly k registraci vybírání náhodně. Velikost sdílené paměti byla nastavena na hodnotu 512 MB. Počet child procesů byl nastaven na hodnotu 1. Zápis do lokační tabulky byl proveden v módu 0. Pro měření byl využit konfigurační soubor s minimálním kódem (Přílohy A.4 a B.4, C.1)



Obr. 6.13: Zapojení měření autentizované registrace

Výsledky

Minulé měření ukázalo, že co se týká zápisu do lokační tabulky a generování SIP odpovědí, je ústředna OpenSIPS mnohem výkonnější, než Kamailio. Při registraci účastníků je již ale prováděno více úkonů, než při samotném zápisu do lokační tabulky.

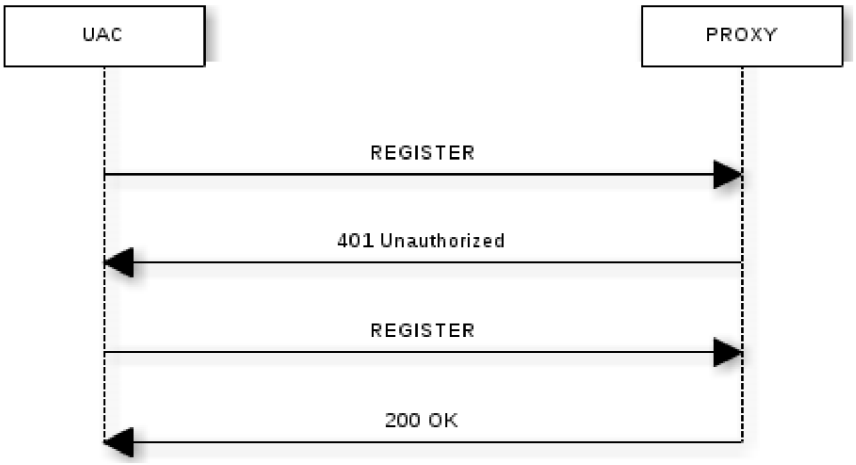
Kromě aktualizace lokační tabulky, která se provede po úspěšné registraci, je potřeba vygenerovat hodnotu nonce a realm, vygenerovat požadavek pro autorizaci ve kterém se zašlou obě hodnoty, v MySQL databázi zjistit údaje o uživateli a následně vypočítat kontrolní hash a porovnat ho s příchozí hodnotou. Z uvedeného je jasné, že výsledná rychlost odpovědi bude záležet na více faktorech, jako je výpočetní výkon hash posloupnosti, nebo rychlost spolupráce s databází.

Jak je z tabulky vidět, zatěžuje registrace ústřednu mnohem více, než samotný zápis do lokační tabulky 6.10.

Tab. 6.10: Počet autorizovaných registrací za sekundu

	Kamailio	OpenSIPS	Asterisk
Počet registrací za vteřinu	6507	4212	258

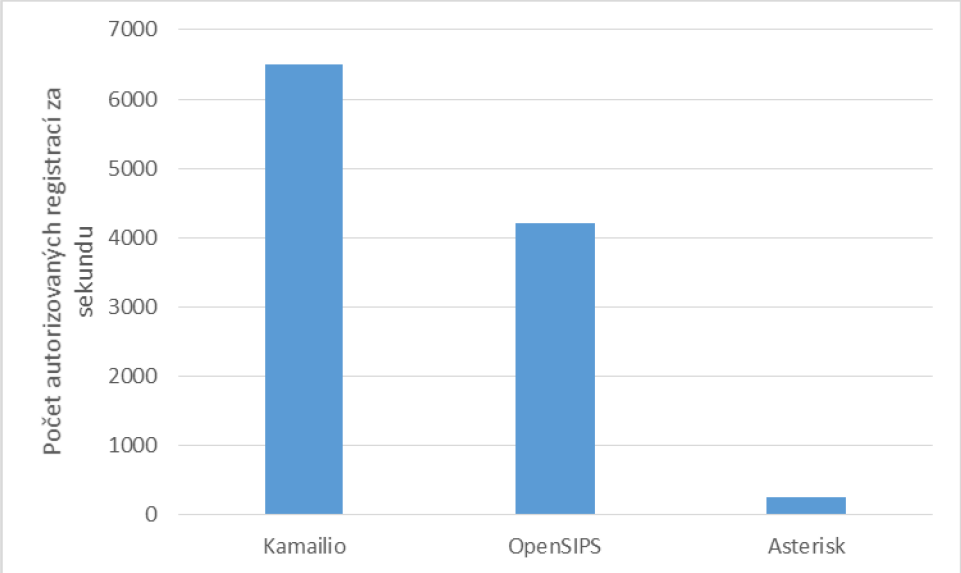
Pro ústřednu Kamailio se jedná přibližně o čtvrtinový výkon oproti minulému měření. U ústředny OpenSIPS dokonce je výkon registrace kolem 8 procent oproti výkonu aktualizace lokační tabulky.



Obr. 6.14: Diagram registrace s pomocí autentizace uživatele

Pokud se výkony registrace porovnají s výkonem ústředny Asterisk (viz. Graf 6.4), je i nižší počet odbavených registrací ústředny OpenSIPS výrazně lepší a více než postačující.

Graf 6.4: Počet autorizovaných registrací za sekundu



6.5.3 Měření č. 9

Pro každou příchozí zprávu je vyvolán prováděcí skript. Velikost a délka skriptu je závislá na počtu požadovaných funkcí ústředny, jež jsou definovány v kódu vytvořeným správcem ústředny. Další měření porovnává velikost skriptu na výkon registrar serveru.

Zapojení a základní konfigurace

Základní zapojení zůstalo stejné jako v měření č.8, jen místo konfiguračních souborů s minimálním kódem byly využity konfigurace základní (A.2 a B.2).

Výsledky

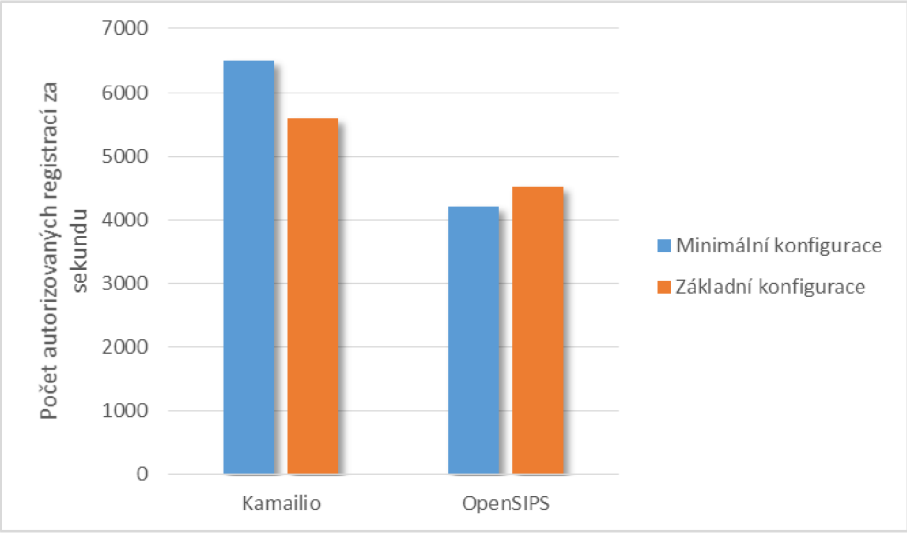
V měření č. 8 dokázalo, že ústředna Kamailio dokáže registrovat uživatele rychleji, než ústředna OpenSIPS. Tenhle fakt mohlo zapříčinit více faktorů. A to například uživatelsky špatně implementovaný kód v konfiguračním souboru, nebo jiná práce jádra ústředny s příchozími informacemi. V tabulce 6.11 a grafu 6.5 kde jsou vynesené výsledky měření je u ústředny Kamailio patrný předpokládat, že větší počet funkcí v konfiguračním souboru (větší obsah kódu) zapříčiní zpomalení registrace.

Tab. 6.11: Počet autorizovaných registrací za sekundu při využití základní konfiguračního souboru

	Kamailio	OpenSIPS
Minimální konfigurace	6507	4212
Základní konfigurace	5589	4521

Zatím co u ústředny Kamailio byla pro registraci v souboru využita implementace napsaná přímo vývojaří a pro její aktivaci stačilo přidat řádek na začátek souboru #!define WITH_AUTH, ústředna OpenSIPS tuhle možnost nenabízela a tudíž se nutnost autentizace musela připsat ručně, což mohlo zapříčinit použití nevhodné implementace a tudíž zpomalení registračního procesu.

Graf 6.5: Počet autorizovaných registrací za sekundu při využití základní konfiguračního souboru



6.5.4 Měření č. 10

Všechna dosavadní měření zjišťovala možnosti ústředen pro registraci účastníků. Nejdůležitější funkcí ústředny je ale sestavení hovoru mezi účastníky. Následující měření zjišťují počet uskutečnitelných hovorů schopných pomocí ústředen sestavit.

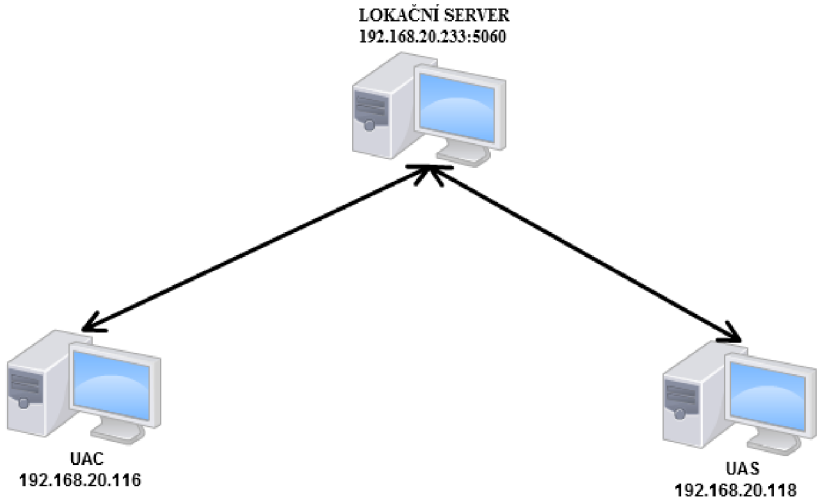
Zapojení a základní konfigurace

K následujícímu měření byly použito PC, na kterém běžely 3 instance programu SIPp

jako klienti UAC (Příloha D.1), jež generovali SIP požadavky se spojením a PC s 3 instancemi programu SIPp jako servery UAS (Příloha D.2), jež na žádosti odpovídali (viz. Obr. 6.15). Ústředny byly použity jako proxy servery, jež přijaté zprávy přeposílaly správným adresátům (viz obr. 6.6). Pro měření byly využity konfigurační soubory s minimálním kódem (Přílohy A.3 a B.3). Velikost sdílené paměti byla nastavena na hodnotu 1024 MB. Počet child procesů byl nastaven na hodnotu 1.

Výsledky

Program SIPp neukončuje testy do příchodu všech vyslaných požadavků, což ve chvíli zahlcení ústředny požadavky na sestavení spojení a ztrátě některého vyslaného požadavku by zkreslovalo výsledný maximální přenos, musely se najít takové hodnoty zasílání, které se přiblížili maximálnímu zatížení procesoru, ale ještě se nezačali ztrácet zprávy. I přes tuhle nejednoznačnost lze z tabulky 6.12 vyčíst, že ústředna OpenSIPS dokázala spojit více hovorů, než ústředna Kamailio.



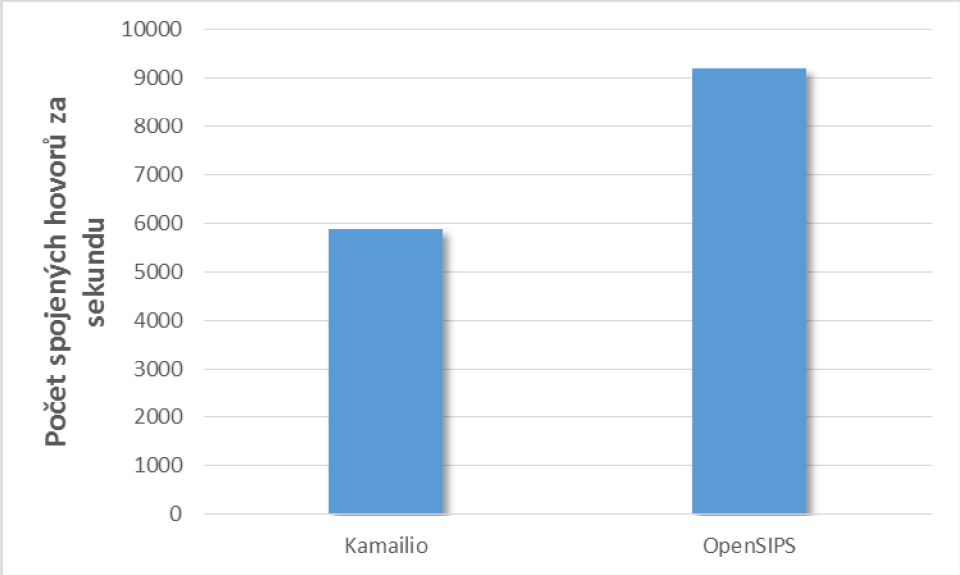
Obr. 6.15: Zapojení pro zjištění maximálního počtu sestavených hovorů

Tab. 6.12: Počet spojených hovorů za sekundu

	Kamailio	OpenSIPS
Počet spojených hovorů za sekundu	5871	9206

Jak lze vyčíst z grafu 6.6, dokáže ústředna OpenSIPS při stejném zatížení procesoru obsloužit o víc jak polovinu hovorů víc než ústředna Kamailio. Je nutné si ale uvědomit, že daný typ měření byl navržen pro laboratnorní odzkoušení a mnohé běžné funkce, jako autentizace účastníka, nebo vyhledání účastníka v lokační tabulce byly na ústřednách vypnuty. Daný typ měření nebyl vhodný pro porovnání výkonu s ústřednou Asterisk. Z toho důvodu bylo potřeba provést měření, které by více odpovídalo realitě.

Graf 6.6: Počet spojených hovorů za sekundu

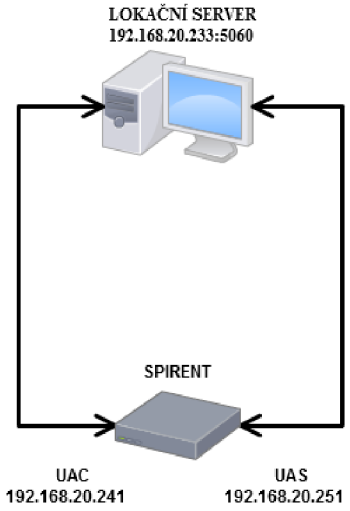


6.5.5 Měření č. 11

Jak bylo již výše zmíněno, Asterisk je využívána hlavně jako pobočková ústředna, zatím co Kamailio a OpenSIPS se využívají jako páteřní ústředny. Z toho důvodu jsou vždy kladeny jiné požadavky na nabízené služby v místě implementace. Na pobočkové ústředny se uživatelé pomocí autentizace přihlašují, což již není potřeba na páteřních prvcích. Naopak páteřní ústředny nemají implementovány mnéhо funkce nabízející pobočkové ústředny. To se týká hlavně oblasti práce s RTP daty. Jelikož ale všechny ústředny dokáží kromě spojení hovorů i uživatele autentizovat a najít v lokační tabulce, lze provést porovnávací měření.

Zapojení a základní konfigurace

Pro měření (viz. Obr. 6.16) bylo využité testovacího zařízení SPIRENT, jež sloužilo jako volající (UAC), tak volaný účastník (UAS). Scénář měření byl nastaven tak, že po registraci účastníka UAS k ústředně z důvodu zapsání dat do lokační tabulky, byl po půl minutě postupně zvyšován počet aktivních hovorů z 0 do 1000. Hovory byly nastavené tak, aby je UAS po sekundě hovoru samy ukončili. Pro samotné měření byly využity základní konfigurační soubory všech tří ústředen (Přílohy A.2, B.2 a C.2-3), kde byly vytvoření 2 uživatelé s heslem. Hodnota sdílené paměti byla u ústředen Kamailio a OpenSIPS nastavena na hodnotu 512 MB. Zápis do lokační tabulky byl nastaven do módu 0 a počet child procesů byl nastaven na hodnotu 1.



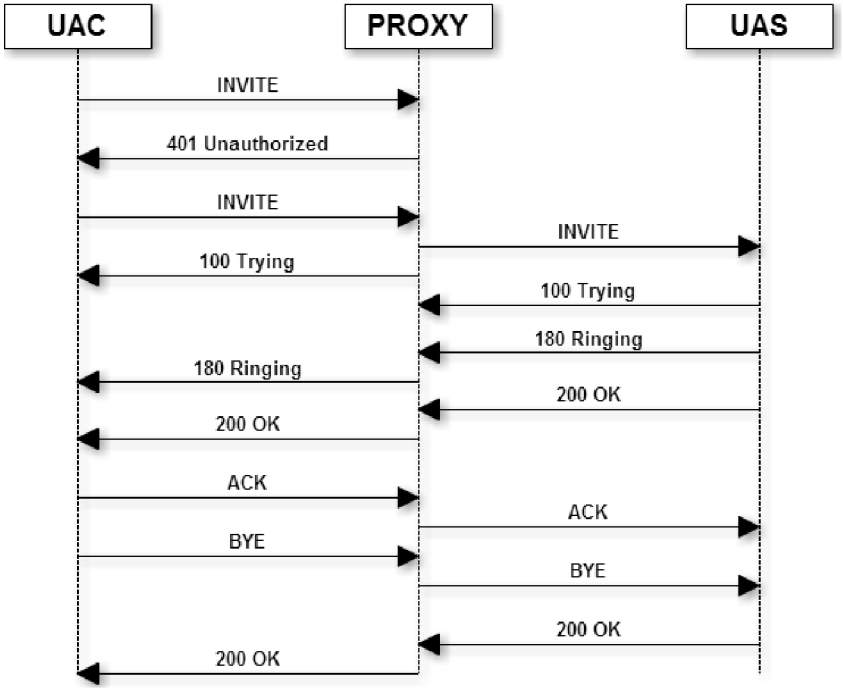
Obr. 6.16: Zapojení měření spojení hovorů pomocí testeru Spirent

Výsledky

Měření, jehož diagram lze vidět na obr. 6.17 u ústředen Kamailio a OpenSIPS proběhl bez problémů. V měření byly využity všechny dosavadní praktiky, autentizace uživatele, vyhledání v lokační tabulce a sestavení spojení. Z toho důvodu se dalo předpokládat, že hodnoty spojení budou výrazně nižší než u předešlého měření. Počet sestavených hovorů postupně konstatně rostl, tak jak zvyšoval tester jejich vyslaný počet. Až po určité době, kdy se ústředna dostala na své maximum, se ustálil.

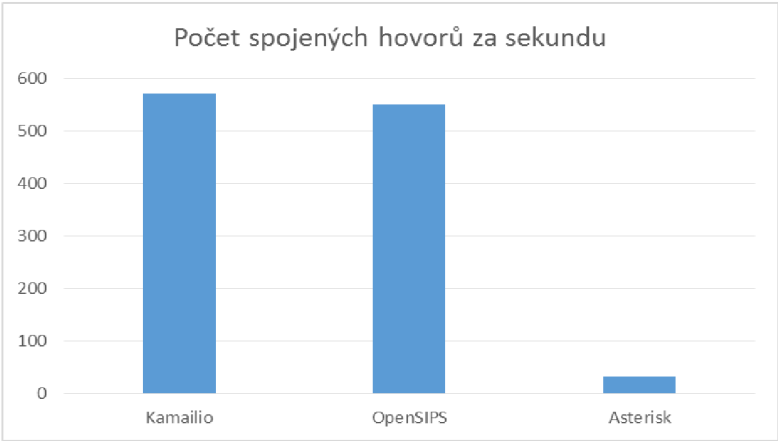
Zatím co ústředna OpenSIPS držela konstatní hodnotu kolem 550 hovorů za sekundu, ústředna Kamailio měla výkyvy větší, ale střední hodnota odbavených hovorů se pohybovala kolem 570 hovorů (viz. Graf 6.7). Při nastavení ukončení hovoru po 1 sekundě pak hodnoty udávají počet sestavený hovorů za sekundu. Pokud by hovor trval déle než jedno sekundu, rostl by počet aktivních hovorů, jež by ústředna dokázal obsloužit a to z důvodu toho, že po sestavení není ústředna daným hovorem více zatěžována do chvíle, než přijde jeho rozpojení. Pokud bychom určili, že průměrný hovor má dobu trvání 1 minutu a 42 sekund [21], pak při agregaci 1:8 a faktu, že telefonní spojení je vedeno mezi 2 účastníky vychází pro OpenSIPS 897 600 a pro Kamailio 930 240 obslužitelných uživatelů.

Z tabulky 6.10, ve které jsou hodnoty počtu registrací za sekundu, je patrné, že ústředna Asterisk při plném vytížení dokáže registrovat 258 uživatelů za sekundu. Z toho důvodu nelze předpokládat, že dokáže spojit více hovorů. Proto bylo měření pro ústřednu upraveno, kdy maximální počet hovorů byl nastavena 300. Oproti ústřednám Kamailio a OpenSIPS jež fungovaly jako klasické Proxy servery a nevedly přes sebe žádný RTP provoz, Asterisk jako zástupce B2BUA nevede spojení RTP provoz až po sestavení spojení. Možnost nastavení, kdy je RTP vedeno již od začátku přímo sice u ústředny existuje, ale není zcela stabilní a proto nemohlo být pro měření využito. Při měření byl počet spojení velice nestabilní a pohyboval se kolem 33 hovorů za sekundu (viz. Graf 6.7). I při takovém nízkém čísle je počet obsloužitelných uživatelů 49920, což je pro podmínky pobočkové ústředny více než dostačující.



Obr. 6.17: Diagram zasílání zpráv při sestavování spojení pomocí testeru Spirent

Graf 6.7: Počet registrovaných hovorů za sekundu



6.5.6 Shrnutí

Výkonnové porovnání ukázalo zásadní rozdíl mezi schopnostmi ústředen pobočkových a páteřních. Díky jednomstrannému zaměření ústředen Kamailio a OpenSIPS na práci se SIP protokolem dokáží obsloužit o hodně větší počet SIP požadavků a hovorů, než ústředny pobočkové reprezentované ústřednou Asterisk.

Samotné testy byly zaměřeny na zjištění maximálních možnosti a kromě měření č.

11 nereprezentují reálné stavy. Testy prokázaly, že výkonnost ústředen Kamailio a OpenSIPS záleží na implementaci kódu a ve velké míře i na výkonnosti spolupracujících aplikací (např. databáze MySQL). Při maximálním vytížení ústředen se také musí myslet na propustnost počítačové sítě, jejíž limity mohou výkon ovlivnit.

Testy prokázaly, že i na běžně dostupném počítačovém vybavení lze provozovat VoIP ústřednu, jež v drtivé většině plný výkon nikdy nevyužije.

6.6 Ochrana ústředen proti napadení

Napadení ústředen lze rozdělit na 2 hlavní části. A to:

- Odposlouchávání komunikace
- Útok na dostupnost ústředny

Útok na dostupnost ústředny je veden pomocí tzv. Denial of Service (DoS) útoků, kdy je na ústředny zasílán vysoký počet SIP požadavků, na které ústředna odpovídá. Mezi nejčastěji používané zprávy se řadí například INVITE, OPTIONS nebo REGISTER. Ústředny Kamailio a OpenSIPS dokáží bojovat proti takovým útokům různými způsoby. A to úpravy firewallu pomocí aplikace fail2ban, nebo využitím modulu pike, který po zvolenou periodu sleduje počet SIP zpráv od jednoho zdroje a je schopný případně komunikaci ze zdroje na určitý čas zakázat. Jelikož problematika přesahuje rámec diplomové práce, nebude se jí v další části textu věnovat pozornost.

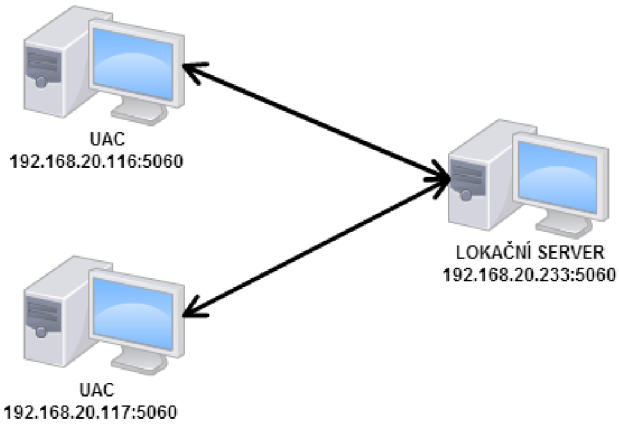
Odposlouchávání komunikace lze zabránit pomocí zabezpečeného přenosu dat. Zabezpečený přenos dat probíhá u ústředen pomocí protokolu Transport Layer Security (TLS). Ten pomocí vytvořených certifikátů a klíčů dokáže mezi 2 koncovými subjekty šifrovat přenášená data a tudíž skrýt všechny informace před nežádoucím odposlechem. Oproti dosavadnímu průběhu komunikace, který probíhal pomocí protokolu UDP, je spojení sestavováno protokolem TCP. Pro sestavení spojení je potřeba většího počtu vyměněných zpráv (viz obr. 6.19), kvůli tomu je výkonová náročnost mnohem vyšší, než při klasickém nezabezpečeném spojení.

6.6.1 Měření č. 12

Všechna předešlá měření by se dala uskutečnit i v zabezpečeném režimu pomocí TLS. Pro demonstraci náročnosti, ale nebylo třeba všechna měření opakovat. Pro jasné porovnání byl vybrán úkol registrace uživatele do lokačního serveru.

Zapojení a základní konfigurace

K měření výkonové náročnosti zabezpečené komunikace bylo využito lehce pozměněné měření č. 6, kdy pro generování dat stačilo využít pouze 2 PC (viz. Obr. 6.18). Pro měření byly využity konfigurační soubory s minimálním kódem (Přílohy A.5 a B.5). Lokační tabulka byla naplněna a pro práci s daty byl zvolen mód 0. Velikost sdílené paměti byla nastavena na hodnotu 512 MB.



Obr. 6.18: Zapojení měření zebezpečeného TLS spojení

Výsledky

Na obr. 6.19 je pomocí programu Wireshark zachycena jedna registrace účastníka pomocí šifrování TLS. Jak lze vidět, oproti klasické registraci u které jsou vygenerovány pouze 2 pakety (viz obr. 6.9), k registraci pomocí TLS je jich potřeba 13. Z toho pakety 1-9 a 12-13 jsou určeny k sestavení zabezpečeného spojení. Pakety 10 a 11 je samotná registrace do ústředny. Připočte-li se nutnost po sestavení zabezpečeného spojení všechna data šifrovat, bude počet registrovaných uživatelů výrazně nižší než v měření č. 6. Což dokládá i výsledná tabulka a graf.

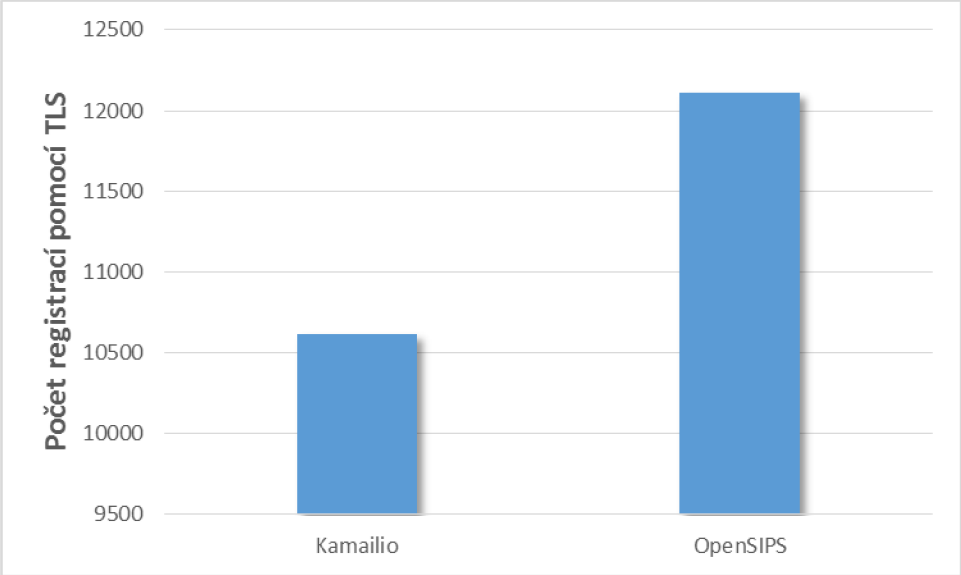
1	0.000000	127.0.0.1	127.0.0.1	TCP	74 40176 > sip-tls [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=60298 TSecr
2	0.000036	127.0.0.1	127.0.0.1	TCP	74 sip-tls > 40176 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=
3	0.000057	127.0.0.1	127.0.0.1	TCP	66 40176 > sip-tls [ACK] Seq=1 Ack=1 Win=65495 Len=0 TSval=60298 TSecr=60298
4	0.000370	127.0.0.1	127.0.0.1	TLSv1	291 Client Hello
5	0.000403	127.0.0.1	127.0.0.1	TCP	66 sip-tls > 40176 [ACK] Seq=1 Ack=226 Win=44800 Len=0 TSval=60298 TSecr=60298
6	0.000931	127.0.0.1	127.0.0.1	TLSv1	826 Server Hello, Certificate, Server Hello Done
7	0.000991	127.0.0.1	127.0.0.1	TCP	66 40176 > sip-tls [ACK] Seq=226 Ack=761 Win=64735 Len=0 TSval=60298 TSecr=60298
8	0.002470	127.0.0.1	127.0.0.1	TLSv1	264 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
9	0.006882	127.0.0.1	127.0.0.1	TLSv1	316 Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
10	0.007411	127.0.0.1	127.0.0.1	TLSv1	524 Application Data, Application Data
11	0.007814	127.0.0.1	127.0.0.1	TLSv1	471 Application Data
12	0.008193	127.0.0.1	127.0.0.1	TCP	66 40176 > sip-tls [FIN, ACK] Seq=882 Ack=1416 Win=65360 Len=0 TSval=60300 TSecr=60300
13	0.008250	127.0.0.1	127.0.0.1	TCP	66 40176 > sip-tls [RST, ACK] Seq=883 Ack=1416 Win=65360 Len=0 TSval=60300 TSecr=60300

Obr. 6.19: Registrace do lokační tabulky pomocí šifrované TLS komunikace

Tab. 6.13: Počet registrací do lokační tabulky pomocí TLS spojení

	Kamailio	OpenSIPS
Počet registrací pomocí TLS	10614	12108

Graf 6.8: Počet registrací do lokační tabulky pomocí TLS spojení



Jelikož celý proces sestavení zabezpečeného spojení byl proveden pouze pro zašifrování 2 zpráv, dá se předpokládat, že při komunikaci s větším počtem vyměněných zpráv, bude rozdíl mezi zabezpečenou a nezabezpečenou variantou klesat.

7 ZÁVĚR

Se stále se zvětšující dostupností připojení k rychlému internetu a to pomocí pevného, wi-fi a stále častěji i mobilního připojení 3. generace (CDMA a LTE), roste i počet uživatelů využívající služeb VoIP. Díky digitalizaci hlasu a přechodu ze spojování hovorů pomoci přepínání okruhů na přepínání paketů, prochází odvětví zabývající se telefonními ústřednami velkými změnami. Ty jsou patrné hned na první pohled, kdy kromě ucelených proprietárních řešení velkých výrobců lze díky open-source řešením vystavět ústředny na běžném kancelářském počítači. Popis 2 takových ústředen Kamailio a OpenSIPS je tématem téhle semestrální práce.

Obě ústředny pojí stejná historie, kdy při odchodu vývojářů z projektu SIP Express Router nejdřív vznikla ústředna Kamailio a po čase dalším rozštěpením vznikla i ústředna OpenSIPS. Díky několika letům společného vývoje a i po rozdělení nezměnění základních funkcí obou ústředen nelze mezi nimi najít výraznější rozdíl. Obě ústředny v základu plní funkci registrar/proxy serverů, kdy jsou schopné pracovat pouze se SIP signalizací. Dodatečné funkce jsou pak realizované pomocí modulů, jež si volí uživatel libovolně podle vlastní potřeby. Díky tomu dokáží ústředny plnit různé role v rámci komunikační sítě. Nejčastěji se využívají jako páteřní ústředny, nebo slouží jako SBC pro ryzí pobočkové ústředny nabízející média služby nebo připojení k jiným komunikačním technologiím.

V diplomové práci je popsána instalace a základní nastavení obou ústředen na operačním systému Ubuntu 12.04. Kromě toho je popsána i tvorba bezpečnostního certifikátu a klíče a jejich implementace pro případnou možnost zabezpečené komunikace pomoctí metody TLS. I přes fakt, že popis je velice podrobný, měl by ho provádět uživatel, jež má aspoň elementární znalosti operačních systémů založených na principech UNIX systému.

Velká část diplomové práce se věnuje výkonovému porovnání ústředen jak mezi sebou, tak u vybraných měření i s pobočkovou ústřednou Asterisk. Zatím co pro Asterisk je SIP jeden z více prokolů, které musí umět obsluhovat, jsou ústředny Kamailio a OpenSIPS vytvořeny pouze pro protokol SIP na který jsou i optimalizovány. Díky tomuto faktu dokáží obsloužit výrazně větší počet požadavků nebo hovorů i při menším zatížení procesoru, nebo menším využití paměti.

První část měření byla zaměřena na zjištění paměťových požadavků ústředen při registraci účastníků do lokační tabulky. V rámci měření se zjistilo, že velikost využití paměti záleží na počtu uložených informací o účastníkovi. Naměřené hodnoty pro 100 000 účastníků pohybující se v řádu desítek MB dokazují, že pro registraci uživatelů nejsou paměťové nároky výrazné. I přes tento fakt je nutné zvolit dostatečně velkou sdílenou paměť, protože měření prokázala, že po registraci zůstávají data o účastníkovi po celou dobu uloženy ve sdílené paměti ústředny a to i ve chvíli, kdy jsou zapsána do databáze. Při přetečení paměti jsou pak další požadavky na registraci odmítány. Test paměťové náročnosti při sestavování hovorů pak dokázal, že ústředna Kamailio potřebuje alokovat výrazně méně paměti pro stejný počet hovorů. Pro vzrůstající počet hovorů je rozdíl čím dál větší.

Měření rychlosti odpovědí na SIP požadavky prokázalo rozdíl rychlosti odpovědi, pokud ústředna využívá zápisu pouze do své interní, sdílené paměti, nebo zda-li čeká na zápis do databáze, kdy v tomhle případě byla doba reakce na požadavky výrazně vyšší.

Při výkonnostím porovnání ústředen, byla měření navrhnuta tak aby se projevili maximální schopnosti ústředen při odbavování požadavků. Aby bylo dosaženo požadovaných vlastností, museli se použít konfigurační soubory s minimálním kódem, který nebyl ošetřen proti různým chybám, případně příchodu jiného druhu SIP požadavku. V reálném provozu by byly výkony nižší. I přes tento fakt měření dokazují výrazný výkonnostní rozdíl oproti pobočkovým ústřednám jako Asterisk a to jak při odbavování jak registračních požadavků, tak požadavků na sestavení hovorů. A to i ve chvíli kdy je pro chod ústředny zvolen pouze jeden proces, který u moderních procesorů dokáže vytížit pouze poměrnou část jeho výkonu a zbylý lze věnovat jiným procesům. Díky tomuto faktu lze bezproblémů provozovat ústředny na zařízeních na kterém jsou spuštěné i jiné serverové aplikace.

Ústředny Kamailio a OpenSIPS lze chápat jako vývojová prostředí, které jsou upraveny tak, aby důležité informace z příchozích SIP požadavků byly předány jako systémové proměnné, se kterými uživatel aplikace může pracovat a libovolně tak může měnit chování ústředny podle vlastní potřeby. Co se na jednu stranu může zdát jako veliká výhoda, přináší i komplikace v podobě špatné implementace kódu, které se může uživatel dopustit. Díky tomu pak může výkon ústředny hodně degradovat, nebo přímo ústřednu zahltit na tolik, že i při malém příchodu požadavků přestane ústředna plnit svoji funkci. Pak samotná výkonnost nemusí záležet na vlastnostech ústředny, ale na šikovnosti programujícího uživatele.

Vysoký výkon odbavování příchozích požadavků se projevuje i při navázání zabezpečeného spojení pomocí technologie TLS, kdy u měření registrace do lokační tabulky i snížený počet odbavených požadavků, jež byl zapříčiněný navázáním zabezpečeného spojení a šifrováním dat byl i při využití pouze jednoho procesu se pohyboval kolem 4 tisíc registrací za sekundu.

Z naměřených výsledků nelze jednoznačně určit, jestli je lepší ústředna Kamailio nebo OpenSIPS. Zatím co v registraci pouze do lokační tabulky je výrazně lepší ústředna OpenSIPS, při registraci s autentizací je lepší ústředna Kamailio. Zatím co při registraci účastníků pomocí lokačního serveru méně paměti potřebuje OpenSIPS, při sestavování spojení méně paměti využívá Kamailio atd. Z toho vyplívá, že každá z ústředen je v některých aspektech silnější a v některých slabší, než ta druhá. Stejně tak špatnou, nebo naopak dobrou implementací kódu může uživatel výkon ústředen výrazně ovlivnit.

LITERATURA

[1] Telefonní ústředna. Wikipedia, otevřená encyklopedie [online]. [cit. 19-12-2013]. Dostupné z: http://cs.wikipedia.org/wiki/Telefonní_ústředna

[2] Goncalves, F.E. *Building Telephony Systems with OpenSER*. Birmingham: Packt Publishing Ltd., 2008. ISBN: 978-1849510745

[3] SIP servers Explained. Smartvox [online]. [cit. 23-10-2013]. Dostupné z: http://kb.smartvox.co.uk/voip-sip/sip-servers-explained/

[4] SIP Router history. Sip-router [online]. [cit. 28-12-2013]. Dostupné z: http://sip-router.org/history/

[5] SIP Express Router. Wikipedia, otevřená encyklopedie [online]. [cit. 14-10-2013]. Dostupné z: http://en.wikipedia.org/wiki/SIP_Express_Router

[6] Johansson, O.E. A Quick Introduction to Kamailio. Edvina AB [online]. [cit. 23-1-2013]. Dostupné z: http://www.slideshare.net/oej/kamailio-a-quick-introduction

[7] Mierla, D.C., Modroiu E.R. Kamailio SIP Server v3.2.0 Development Guide. Asipto, 2011. [cit 28-12-2013]. Dostupné z: http://www.asipto.com/pub/kamailio-devel-guide/

[8] Get OpenSIPS. OpenSIPS [online]. [cit. 11-9-2013]. Dostupné z http://www.opensips.org/Downloads/Downloads

[9] OpenSIPS Webinar – Introduction to OpenSIPS. OpenSIPS [online]. [cit. 28-12-2013]. Dostupné z: http://www.opensips.org/html/docs/video/webinar001/

[10] Asterisk vs OpenSIPS. VOIP Today [online]. [cit. 28-12-2013]. Dostupné z: http://www.voiptoday.org/index.php?option=com_content&view=article&id=231:asterisk-vs-opensips&catid=55:pbx-comparison&Itemid=103

[11] Mierla D.C., Install and mantain Kamailio v4.1.x from GIT. Kamailio [online].[cit. 28-12-2013]. Dostupné z: http://www.kamailio.org/wiki/install/4.1.x/git

[12] Create certificate to be used with Kamailio. Kamailio [online]. [cit. 30-9-2010]. Dostupné z: http://www.kamailio.org/dokuwiki/doku.php/tls:create-certificates

[13] OpenSIPS Webinar – Getting Started. OpenSIPS [online]. [cit. 28-12-2013]. Dostupné z: http://www.opensips.org/Documentation/Tutorials-GettingStarted

[14] Griffiths, P. TLS Support [online]. OpenSIPS [cit. 31-3-2014]. Dostupné z: http://www.opensips.org/html/docs/tutorials/tls-1.4.x.html

[15] SIPp [online]. [cit. 10-11-2013]. Dostupné z: http://sipp.sourceforge.net/

[16] SIPp reference documentation. SIPp [online]. [cit. 25-10-2010]. Dostupné z: http://sipp.sourceforge.net/doc3.0/reference.html

[17] Wireshark. Wikipedia, otevřená encyklopedie [online]. [cit. 22-03-2014]. Dostupné z : http://en.wikipedia.org/wiki/Wireshark

[18] Lamping, U., Sharp, R., Warnicke, E. Wireshark User's Guide. Wireshark [online]. [cit. 31-3-2014]. Dostupné z: http://www.wireshark.org/docs/wsug_html/

[19] Spirent Test Center. Spirent [online]. [cit. 5-5-2014]. Dostupné z: http://www.spirent.com/Ethernet_Testing/Software/TestCenter

[20] Goncalves, F.E. *Building Telephony Systems with OpenSIPS*. Birmingham: Packt

Publishing Ltd., 2010. ISBN 978-1-849510-74-5

[21] Fišer, F. Rozdílná tarifikační perioda má na cenu volání zásadní vliv. Lupa.cz [online]. [cit. 27-1-2009]. Dostupné z: http://www.lupa.cz/clanky/tarifikacni-perioda-ma-na-cenu-volani-vliv/

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AAA	Authentication,Authorization and Accounting protocol
API	Application Programming Interface
B2BUA	Back To Back User Agent
CDMA	Code Division Multiple Access
CPS	Calls Per Seconds
DNS	Domain Name System
DTMF	Dual-Tone Multi-Frequency
ENUM	tElephone NUmber Mapping
FDM	Frexuency Division Multiplex
GmBH	Společnost s ručením omezeným
GPL	GNU General Public License
HTTP	Hypertext Transfer Protocol
IAX	Inter-Asterisk eXchange
IP	Internet Protocol
IVR	Interactive Voice Response
LDAP	Lightweight Directory Access Protocol
LTE	3GPP Long Term Evolution
MGCP	Media Gateway Protocol
MI	Module Interface
NAT	Network Address Translation
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RADIUS	Remote Authentication Dial In User Service
RPC	Remote Procedure Call
RTP	Real-time Transfer Protocol
SDP	Session Description Protocol
SER	SIP Express Router
SIP	Session Initiation Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol

TDM	Time Division Multiplex
TLS	Transport Layer Security
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
VoIP	Voice over Internet Procotol
VPN	Virtual Private Network
XML	eXtensible Markup Language

SEZNAM PŘÍLOH

A KONFIGURAČNÍ SOUBORY ÚSTŘEDNY KAMAILIO

A.1 Usrloc konfigurace

```
#!KAMAILIO
####### Global Parameters #########
log_facility=LOG_LOCAL2
fork=yes
children=1
disable_tcp=yes
listen=10.192.16.65:5060
auto_aliases=no
####### Modules Section ########
mpath="/usr/local/lib/kamailio/modules/"
loadmodule "db_mysql.so"
loadmodule "tm.so"
loadmodule "sl.so"
loadmodule "usrloc.so"
loadmodule "registrar.so"
loadmodule "textops.so"
# ----- auth_db params -----
modparam("usrloc", "db_url",
"mysql://kamailio:kamailiorw@localhost/kamailio")
modparam("usrloc", "db_mode", 0)
####### Routing Logic ########
route{

      if(is_method("REGISTER"))
      {
            save("location");
            exit;
      }
}
```

A.2 Základní konfigurace

```
#!KAMAILIO
####### Global Parameters #########

log_facility=LOG_LOCAL2
fork=yes
children=1
disable_tcp=yes
port=5060
```

```
####### Modules Section ########

mpath="/usr/local/lib/kamailio/modules/"

loadmodule "db_mysql.so"
loadmodule "sl.so"
loadmodule "pv.so"
loadmodule "usrloc.so"
loadmodule "registrar.so"
loadmodule "textops.so"
loadmodule "auth.so"
loadmodule "auth_db.so"

# ----------------- setting module-specific parameters ---------------

# ----- registrar params -----
modparam("registrar", "method_filtering", 1)
/* uncomment the next line to disable parallel forking via location */
# modparam("registrar", "append_branches", 0)
/* uncomment the next line not to allow more than 10 contacts per AOR
*/
modparam("registrar", "max_contacts", 10000)
# max value for expires of registrations
modparam("registrar", "max_expires", 3600)
# set it to 1 to enable GRUU
modparam("registrar", "gruu_enabled", 0)


# ----- usrloc params -----
modparam("usrloc", "db_url",
"mysql://kamailio:kamailiorw@localhost/kamailio")
modparam("usrloc", "db_mode", 2)

# ----- auth_db params -----
modparam("auth_db", "db_url",
"mysql://kamailio:kamailiorw@localhost/kamailio")
modparam("auth_db", "calculate_ha1", yes)
modparam("auth_db", "password_column", "password")
modparam("auth_db", "load_credentials", "")

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

request_route {

      # authentication
      route(AUTH);

      # handle registrations
      route(REGISTRAR);
}
```

```
# Handle SIP registrations
route[REGISTRAR] {
     if (is_method("REGISTER"))
     {
          if (!save("location"))
               sl_reply_error();
          exit;
     }
}


# Authentication route
route[AUTH] {

     if (is_method("REGISTER") || from_uri==myself)
     {
          # authenticate requests
          if (!auth_check("$fd", "subscriber", "1")) {
               auth_challenge("$fd", "0");
               exit;
          }
          # user authenticated - remove auth header
          if(!is_method("REGISTER"))
               consume_credentials();
     }
     return;
}
```

A.3 Relay Konfigurace

```
#!KAMAILIO

log_facility=LOG_LOCAL2

mpath="/usr/local/lib/kamailio/modules/"

fork=yes
children=1
listen=192.168.20.233:5060
disable_tcp=yes
auto_aliases=no

loadmodule "tm.so"
modparam("tm", "wt_timer", 2)

route{
     t_relay();
}
```

A.4 Registrar Konfigurace

```
#!KAMAILIO
####### Global Parameters #########

log_facility=LOG_LOCAL2
fork=yes
children=1
disable_tcp=yes
port=5060

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

mpath="/usr/local/lib/kamailio/modules/"

loadmodule "db_mysql.so"
loadmodule "sl.so"
loadmodule "pv.so"
loadmodule "usrloc.so"
loadmodule "registrar.so"
loadmodule "textops.so"
loadmodule "auth.so"
loadmodule "auth_db.so"

# ----------------- setting module-specific parameters ---------------

# ----- registrar params -----
modparam("registrar", "method_filtering", 1)
/* uncomment the next line to disable parallel forking via location */
# modparam("registrar", "append_branches", 0)
/* uncomment the next line not to allow more than 10 contacts per AOR
*/
modparam("registrar", "max_contacts", 10000)
# max value for expires of registrations
modparam("registrar", "max_expires", 3600)
# set it to 1 to enable GRUU
modparam("registrar", "gruu_enabled", 0)


# ----- usrloc params -----
modparam("usrloc", "db_url",
"mysql://kamailio:kamailiorw@localhost/kamailio")
modparam("usrloc", "db_mode", 2)

# ----- auth_db params -----
modparam("auth_db", "db_url",
"mysql://kamailio:kamailiorw@localhost/kamailio")
modparam("auth_db", "calculate_ha1", yes)
modparam("auth_db", "password_column", "password")
modparam("auth_db", "load_credentials", "")
```

```
####### Routing Logic ########

request_route {

      # authentication
      route(AUTH);

      # handle registrations
      route(REGISTRAR);
}


# Handle SIP registrations
route[REGISTRAR] {
      if (is_method("REGISTER"))
      {
            if (!save("location"))
                  sl_reply_error();
            exit;
      }
}


# Authentication route
route[AUTH] {

      if (is_method("REGISTER") || from_uri==myself)
      {
            # authenticate requests
            if (!auth_check("$fd", "subscriber", "1")) {
                  auth_challenge("$fd", "0");
                  exit;
            }
            # user authenticated - remove auth header
            if(!is_method("REGISTER"))
                  consume_credentials();
      }
      return;
}
```

A.5 TLS Konfigurace

kamailio.cfg

```
#!KAMAILIO

####### Global Parameters #########
log_facility=LOG_LOCAL2
```

```
fork=yes
children=1
disable_tcp=yes
listen=127.0.0.1:5060
listen=tls:127.0.0.1:5061
auto_aliases=no
enable_tls=yes


####### Modules Section ########
mpath="/usr/local/lib/kamailio/modules/"

loadmodule "db_mysql.so"
loadmodule "tm.so"
loadmodule "sl.so"
loadmodule "usrloc.so"
loadmodule "registrar.so"
loadmodule "textops.so"
loadmodule "tls.so"

# ----- auth_db params -----
modparam("usrloc", "db_url",
"mysql://kamailio:kamailiorw@localhost/kamailio")
modparam("usrloc", "db_mode", 0)

# ----- tls params -----
modparam("tls", "config", "/usr/local/etc/kamailio/tls.cfg")

####### Routing Logic ########
route{

     if(is_method("REGISTER"))
     {
          save("location");
          exit;
     }
}
```

Tls.cfg

```
[server:default]
method = TLSv1
verify_certificate = yes
require_certificate = no
private_key = /etc/certs/pbx/key.pem
certificate = /etc/certs/pbx/cert.pem
ca_list = /etc/certs/CA/cert.pem

[server:127.0.0.1:5061]
method = SSLv23
verify_certificate = no
require_certificate = no
```

```
private_key = /etc/certs/pbx/key.pem
certificate = /etc/certs/pbx/cert.pem
ca_list = /etc/certs/CA/cert.pem

[client:default]
verify_certificate = no
require_certificate = no
```

B KONFIGURAČNÍ SOUBORY ÚSTŘEDNY OPENSIPS

B.1 Usrloc konfigurace

```
log_facility=LOG_LOCAL1
fork=yes
children=1
auto_aliases=no
listen=udp:10.192.16.65:5060
disable_tcp=yes

mpath="/usr/local//lib/opensips/modules/"

loadmodule "db_mysql.so"
loadmodule "signaling.so"
loadmodule "sl.so"
loadmodule "tm.so"
loadmodule "rr.so"
modparam("rr", "append_fromtag", 0)

loadmodule "sipmsgops.so"
loadmodule "uri.so"
modparam("uri", "use_uri_table", 0)
modparam("uri", "db_url",
"mysql://opensips:opensipsrw@localhost/opensips")

loadmodule "usrloc.so"
modparam("usrloc", "db_mode", 1)
modparam("usrloc", "db_url",
"mysql://opensips:opensipsrw@localhost/opensips")

#### REGISTRAR module
loadmodule "registrar.so"

route{
      # record routing
      if (!is_method("REGISTER"))
            record_route();

      if (is_method("REGISTER"))
      {
            if (!save("location"))
                  sl_reply_error();
            exit;
      }
```

```
}
```

B.2 Základní konfigurace

```
####### Global Parameters #########

debug=3
log_stderror=no
log_facility=LOG_LOCAL0
fork=yes
children=1

/* uncomment the following lines to enable debugging */
#debug=6
#fork=no
#log_stderror=yes

/* uncomment the next line to enable the auto temporary blacklisting of
   not available destinations (default disabled) */
#disable_dns_blacklist=no

/* uncomment the next line to enable IPv6 lookup after IPv4 dns
   lookup failures (default disabled) */
#dns_try_ipv6=yes

/* comment the next line to enable the auto discovery of local aliases
   based on revers DNS on IPs */
auto_aliases=no

listen=udp:192.168.20.233:5060   # CUSTOMIZE ME
disable_tcp=yes
disable_tls=yes

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

#set module path
mpath="/usr/local//lib/opensips/modules/"

#### SIGNALING module
loadmodule "signaling.so"

#### StateLess module
loadmodule "sl.so"

#### Transaction Module
loadmodule "tm.so"
modparam("tm", "fr_timer", 5)
modparam("tm", "fr_inv_timer", 30)
```

```
modparam("tm", "restart_fr_on_each_reply", 0)
modparam("tm", "onreply_avp_mode", 1)

#### Record Route Module
loadmodule "rr.so"
/* do not append from tag to the RR (no need for this script) */
modparam("rr", "append_fromtag", 0)

#### MAX ForWarD module
loadmodule "maxfwd.so"

#### SIP MSG OPerationS module
loadmodule "sipmsgops.so"

#### FIFO Management Interface
loadmodule "mi_fifo.so"
modparam("mi_fifo", "fifo_name", "/tmp/opensips_fifo")


#### URI module
loadmodule "uri.so"
modparam("uri", "use_uri_table", 0)
modparam("uri", "db_url",
"mysql://opensips:opensipsrw@localhost/opensips")

loadmodule "db_mysql.so"

#### USeR LOCation module
loadmodule "usrloc.so"
modparam("usrloc", "nat_bflag", 10)
modparam("usrloc", "db_mode",   0)

#### REGISTRAR module
loadmodule "registrar.so"
modparam("registrar", "tcp_persistent_flag", 7)

/* uncomment the next line not to allow more than 10 contacts per AOR
*/
#modparam("registrar", "max_contacts", 10)

#### ACCounting module
loadmodule "acc.so"
/* what special events should be accounted ? */
modparam("acc", "early_media", 0)
modparam("acc", "report_cancels", 0)
/* by default we do not adjust the direct of the sequential requests.
   if you enable this parameter, be sure the enable "append_fromtag"
   in "rr" module */
modparam("acc", "detect_direction", 0)
modparam("acc", "failed_transaction_flag", 3)
/* account triggers (flags) */
modparam("acc", "log_flag", 1)
```

```
modparam("acc", "log_missed_flag", 2)

#### AUTHentication modules
loadmodule "auth.so"
loadmodule "auth_db.so"
modparam("auth_db", "calculate_ha1", yes)
modparam("auth_db", "password_column", "password")
modparam("auth_db", "db_url",
      "mysql://opensips:opensipsrw@localhost/opensips") # CUSTOMIZE ME
modparam("auth_db", "load_credentials", "")

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

# main request routing logic

route{
      if (!mf_process_maxfwd_header("10")) {
            sl_send_reply("483","Too Many Hops");
            exit;
      }
      if (has_totag()) {
            # sequential requests within a dialog should
            # take the path determined by record-routing
            if (loose_route()) {

                  if (is_method("BYE")) {
                        setflag(1); # do accounting ...
                        setflag(3); # ... even if the transaction fails
                  } else if (is_method("INVITE")) {
                        # even if in most of the cases is useless, do RR
for
                        # re-INVITEs alos, as some buggy clients do
change route set
                        # during the dialog.
                        record_route();
                  }

                  # route it out to whatever destination was set by
loose_route()
                  # in $du (destination URI).
                  route(1);
            } else {

                  if ( is_method("ACK") ) {
                        if ( t_check_trans() ) {
                              # non loose-route, but stateful ACK; must be
an ACK after
                              # a 487 or e.g. 404 from upstream server
                              t_relay();
                              exit;
                        } else {
                              # ACK without matching transaction ->
```

```
                        # ignore and discard
                        exit;
                    }
                }
                sl_send_reply("404","Not here");
        }
        exit;
    }

    # CANCEL processing
    if (is_method("CANCEL"))
    {
        if (t_check_trans())
            t_relay();
        exit;
    }

    t_check_trans();

    if (is_method("REGISTER"))
    {

        # authenticate the REGISTER requests
        if (!www_authorize("", "subscriber"))
        {
            www_challenge("", "0");
            exit;
        }

        if (!db_check_to())
        {
            sl_send_reply("403","Forbidden auth ID");
            exit;
        }

        if (!save("location"))
            sl_reply_error();

        exit;
    }

    # preloaded route checking
    if (loose_route()) {
        xlog("L_ERR",
        "Attempt to route with preloaded Route's
[$fu/$tu/$ru/$ci]");
        if (!is_method("ACK"))
            sl_send_reply("403","Preload Route denied");
        exit;
    }

    # record routing
```

```
if (!is_method("REGISTER|MESSAGE"))
      record_route();

# account only INVITEs
if (is_method("INVITE")) {

      setflag(1); # do accounting
}

if (!uri==myself) {
      append_hf("P-hint: outbound\r\n");

      route(1);
}

# requests for my domain

if (is_method("PUBLISH|SUBSCRIBE"))
{
      sl_send_reply("503", "Service Unavailable");
      exit;
}

if (is_method("REGISTER"))
{

      if (   0 ) setflag(7);

      if (!save("location"))
            sl_reply_error();

      exit;
}

if ($rU==NULL) {
      # request with no Username in RURI
      sl_send_reply("484","Address Incomplete");
      exit;
}

# do lookup with method filtering
if (!lookup("location","m")) {


      t_newtran();
      t_reply("404", "Not Found");
      exit;
}

# when routing via usrloc, log the missed calls also
setflag(2);
route(1);
```

```
}


route[1] {
      # for INVITEs enable some additional helper routes
      if (is_method("INVITE")) {

            t_on_branch("2");
            t_on_reply("2");
            t_on_failure("1");
      }

      if (!t_relay()) {
            send_reply("500","Internal Error");
      };
      exit;
}

branch_route[2] {
      xlog("new branch at $ru\n");
}

onreply_route[2] {

      xlog("incoming reply\n");
}

failure_route[1] {
      if (t_was_cancelled()) {
            exit;
      }

      # uncomment the following lines if you want to block client
      # redirect based on 3xx replies.
      ##if (t_check_status("3[0-9][0-9]")) {
      ##t_reply("404","Not found");
      ##    exit;
      ##}


}
```

B.3 Relay Konfigurace

```
fork=yes
children=1
log_facility=LOG_LOCAL2
listen=192.168.20.233:5060
disable_tcp=yes
auto_aliases=no
```

```
mpath="/usr/local/lib/opensips/modules/"

loadmodule "tm.so"
modparam("tm", "wt_timer", 2)

route{
     t_relay();
}
```

B.4 Registrar Konfigurace

```
####### Global Parameters #########

log_facility=LOG_LOCAL1
fork=yes
children=1
disable_tcp=yes
listen=udp:127.0.0.1:5060

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

mpath="/usr/local/opensips/lib/opensips/modules/"

loadmodule "db_mysql.so"
loadmodule "sl.so"
loadmodule "pv.so"
loadmodule "usrloc.so"
loadmodule "registrar.so"
loadmodule "textops.so"
loadmodule "auth.so"
loadmodule "auth_db.so"

# ----------------- setting module-specific parameters ---------------

# ----- registrar params -----
modparam("registrar", "method_filtering", 1)
/* uncomment the next line to disable parallel forking via location */
# modparam("registrar", "append_branches", 0)
/* uncomment the next line not to allow more than 10 contacts per AOR
*/
modparam("registrar", "max_contacts", 10000)
# max value for expires of registrations
modparam("registrar", "max_expires", 3600)
# set it to 1 to enable GRUU
modparam("registrar", "gruu_enabled", 0)


# ----- usrloc params -----
```

```
modparam("usrloc", "db_url",
"mysql://opensips:opensipsrw@localhost/opensips")
modparam("usrloc", "db_mode", 2)

# ----- auth_db params -----
modparam("auth_db", "db_url",
"mysql://opensips:opensipsrw@localhost/opensips")
modparam("auth_db", "calculate_ha1", yes)
modparam("auth_db", "password_column", "password")
modparam("auth_db", "load_credentials", "")

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

route {

      # authentication
      route(AUTH);

      # handle registrations
      route(RELAY);
}


# Handle SIP registrations
route[RELAY] {
      if (is_method("REGISTER"))
      {
            if (!save("location"))
                  sl_reply_error();
            exit;
      }
}


# Authentication route
route[AUTH] {

      if (is_method("REGISTER") || from_uri==myself)
      {
            # authenticate requests
            if (!auth_check("$fd", "subscriber", "1")) {
                  auth_challenge("$fd", "0");
                  exit;
            }
            # user authenticated - remove auth header
            if(!is_method("REGISTER"))
                  consume_credentials();
      }
}
```

B.5 TLS Konfigurace

```
debug=2
fork=yes
log_stderror=no

disable_tls = no
listen = tls:127.0.0.1:5061
tls_verify_server = 0
tls_verify_client = 0
tls_require_client_certificate = 0
tls_handshake_timeout=30
tls_send_timeout=30
tls_method = TLSv1
tls_ciphers_list="NULL"
tls_certificate = "/usr/local/etc/opensips//tls/user/user-cert.pem"
tls_private_key = "/usr/local/etc/opensips//tls/user/user-privkey.pem"
tls_ca_list = "/usr/local/etc/opensips//tls/user/user-calist.pem"
tls_server_domain [127.0.0.1:5061]
{
tls_certificate = "/usr/local/etc/opensips//tls/user/user-cert.pem"
tls_private_key = "/usr/local/etc/opensips//tls/user/user-privkey.pem"
tls_ca_list = "/usr/local/etc/opensips/tls//user/user-calist.pem"
tls_method = SSLv23
}


  # ------------------ module loading ----------------------------------
-

mpath="/usr/local//lib/opensips/modules/"

loadmodule "db_mysql.so"
loadmodule "signaling.so"
loadmodule "sl.so"
loadmodule "tm.so"
loadmodule "rr.so"
modparam("rr", "append_fromtag", 0)

loadmodule "sipmsgops.so"
loadmodule "uri.so"
modparam("uri", "use_uri_table", 0)
modparam("uri", "db_url",
"mysql://opensips:opensipsrw@localhost/opensips")

loadmodule "usrloc.so"
modparam("usrloc", "db_mode", 1)
modparam("usrloc", "db_url",
"mysql://opensips:opensipsrw@localhost/opensips")

#### REGISTRAR module
```

```
loadmodule "registrar.so"

  # -------------------------  request routing logic ------------------
-

  # main routing logic

  route{
     # record routing
     if (!is_method("REGISTER"))
          record_route();

     if (is_method("REGISTER"))
     {
          if (!save("location"))
               sl_reply_error();
          exit;
     }

}
```

C KONFIGURAČNÍ SOUBORY ÚSTŘEDNY ASTERISK

C.1 SIP.conf

```
[general]
bindport=5060
dislallow=all
allow=alaw
allow=gsm

[1001]
context=sipp
type=friend
secret=1001
username=1001
host=dynamic
auth=md5
callerid=1001

[1002]
context=sipp
type=friend
secret=1002
username=1002
host=dynamic
auth=md5
callerid=1002
.
.
.
.
[1100]
context=sipp
type=friend
secret=1100
username=1100
host=dynamic
auth=md5
callerid=1100
```

C.2 SIP2.conf

```
    [general]
```

```
bindport=5060
dislallow=all
allow=alaw
allow=gsm

[80000]
context=sipp
type=friend
secret=test
username=80000
host=dynamic
auth=md5
callerid=80000

[80001]
context=sipp
type=friend
secret=test
username=80001
host=dynamic
auth=md5
callerid=80001
```

C.3 EXTENSIONS.conf

```
[sipp]
exten => 80000,1,Dial(SIP/80000)
exten => 80001,1,Dial(SIP/80001)
```

D XML SOUBORY SCÉNÁŘŮ SIPP

D.1 UAC.xml

```
<?xml version="1.0" encoding="windows-1252"?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">
<scenario name="Basic Sipstone UAC">
  <send retrans="500">
    <![CDATA[

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

      v=0
      o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
      s=-
      c=IN IP[media_ip_type] [media_ip]
      t=0 0
      m=audio [media_port] RTP/AVP 0
      a=rtpmap:0 PCMU/8000

    ]]>
  </send>

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

  <recv response="180" optional="true">
  </recv>
  <recv response="200" rtd="true">
  </recv>
  <send>
    <![CDATA[

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

```
      Subject: Performance Test
      Content-Length: 0

    ]]>
  </send>
  <pause/>

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

      BYE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[call_number]
      To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
      Call-ID: [call_id]
      CSeq: 2 BYE
      Contact: sip:sipp@[local_ip]:[local_port]
      Max-Forwards: 70
      Subject: Performance Test
      Content-Length: 0

    ]]>
  </send>

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

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

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

</scenario>
```

D.2 UAS.xml

```
<?xml version="1.0" encoding="windows-1252"?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">
<scenario name="Basic UAS responder">
  <recv request="INVITE" crlf="true">
  </recv>

  <send>
    <![CDATA[

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

    ]]>
```

```
  </send>

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

      SIP/2.0 200 OK
      [last_Via:]
      [last_From:]
      [last_To:];tag=[call_number]
      [last_Call-ID:]
      [last_CSeq:]
      Contact: <sip:[local_ip]:[local_port];transport=[transport]>
      Content-Type: application/sdp
      Content-Length: [len]

      v=0
      o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
      s=-
      c=IN IP[media_ip_type] [media_ip]
      t=0 0
      m=audio [media_port] RTP/AVP 0
      a=rtpmap:0 PCMU/8000

    ]]>
  </send>

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

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

  <send>
    <![CDATA[

      SIP/2.0 200 OK
      [last_Via:]
      [last_From:]
      [last_To:]
      [last_Call-ID:]
      [last_CSeq:]
      Contact: <sip:[local_ip]:[local_port];transport=[transport]>
      Content-Length: 0

    ]]>
  </send>

  <pause milliseconds="4000"/>

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

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

</scenario>
```

D.3 USRLOC.xml

```
<?xml version="1.0" encoding="windows-1252"?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">
<scenario name="Basic Sipstone UAC">
  <!-- In client mode (sipp placing calls), the Call-ID MUST be         --
>
  <!--   generated   by   sipp.   To   do   so,   use   [call_id]   keyword.
-->
  <send retrans="500" start_rtd="true">
    <![CDATA[

      REGISTER sip:[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: sipp <sip:[field0]@[remote_ip]:[local_port]>;tag=[call_number]
      To: sut <sip:[field0]@[remote_ip]:[remote_port]>
      Call-ID: [call_id]
      CSeq: 5 REGISTER
      Contact: sip:[field0]@[local_ip]:[local_port]
      Expires: 72000
      Max-Forwards: 70
      Subject: Performance Test
      Content-Type: application/sdp
      Content-Length: [len]

    ]]>
  </send>
  <recv response="400" optional="true" rtd="true">
    <action>
          <exec int_cmd="stop_call"/>
    </action>
  </recv>

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

  <ResponseTimeRepartition value="10, 50, 100, 150, 200, 500, 1000"/>

</scenario>
```

D.4 REGISTER.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<scenario name="register_client">
  <send retrans="500">
    <![CDATA[

      REGISTER sip:[remote_ip] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: <sip:[field0]@[field1]>;tag=[call_number]
      To: <sip:[field0]@[field1]>
      Call-ID: [call_id]
      CSeq: 1 REGISTER
```

```
      Contact: sip:[field0]@[local_ip]:[local_port]
      Max-Forwards: 5
      Expires: 1800
      User-Agent: SIPp/Linux
      Content-Length: 0

    ]]>
  </send>

  <recv response="401" auth="true">
  </recv>

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

      REGISTER sip:[remote_ip] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: <sip:[field0]@[field1]>;tag=[call_number]
      To: <sip:[field0]@[field1]>
      Call-ID: [call_id]
      CSeq: 2 REGISTER
      Contact: sip:[field0]@[local_ip]:[local_port]
      [field2]
      Max-Forwards: 5
      Expires: 1800
      User-Agent: SIPp/Linux
      Content-Length: 0

    ]]>
  </send>

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

E PARAMETRY TESTOVANÉHO PC

Procesor:	Intel Dual Core i3-2330M – 2,2 Ghz
Paměť:	4 GB
Operační systém:	Ubuntu 12.04 LTS 32-bit

F OBSAH PŘILOŽENÉHO CD

Konfiguracni_soubory	Složka s použitými konfiguračními soubory
Mereni	Složka s výslekdy jednotlivých měření
PBX_Kamailio_OpenSIPS.pdf	Diplomová práce