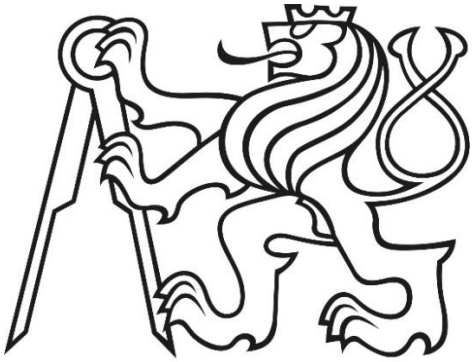
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ
V PRAZE

Fakulta elektrotechnická

Katedra telekomunikační techniky



Vyvažovač zátěže pro protokol SIP

Bakalářská práce

Květen 2018

Bakalant: Pont Martin

Vedoucí práce: Ing. Pavel Troller, Csc.,

Čestné prohlášení

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

Datum: 25. 5. 2018

……..……

podpis bakalanta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Pont** Jméno: **Martin** Osobní číslo: **434667**

Fakulta/ústav: **Fakulta elektrotechnická**

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

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

Studijní obor: **Síťové a informační technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Vyvažovač zátěže pro protokol SIP

Název bakalářské práce anglicky:

SIP Load Balancer

Pokyny pro vypracování:

Vyhledejte existující a nabízená řešení vhodná pro vyvažování zátěže (Load Balancing) protokolu SIP. Tabularizujte jejich základní technické parametry, jakož i dostupnost, licenční podmínky a případnou cenu. Naleznete-li řešení na bázi otevřeného kódu, vyberte nejvhodnější a proveďte jeho instalaci do prostředí laboratorní virtualizované VoIP sítě. Následně řešení otestujte a zhodnoťte dosažené výsledky. V případě nenalezení vhodného řešení navrhněte postup pro jeho implementaci.

Seznam doporučené literatury:

[1] Membrey, P.; Pludge, E.; Hows, D.: Practical Load Balancing. Apress, 2012. ISBN: 978-1-4302-3680-1.
[2] Dokumentace RFC dostupná na https://www.rfc-editor.org/info/rfc3261 [on-line]

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Troller, CSc., katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **06.02.2018** Termín odevzdání bakalářské práce: **25.05.2018**

Platnost zadání bakalářské práce: **30.09.2019**

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

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání	Podpis studenta

Anotace:

V dnešní době se stále více používá technologie VOIP. Pro sestavení relací se zde také používá protokol SIP. S nárůstem provozu této technologie se nabízí otázka jakým způsobem tento provoz kontrolovat a vyvažovat pro zařízení, která s tímto protokolem pracují. Tato práce se věnuje detailnímu rozboru vyvažování zátěže protokolu SIP. Cílem této bakalářské práce je provést analýzu dostupných produktů a tuto funkcionalitu ověřit na konkrétním řešení v reálném provozu protokolu SIP.

Klíčová slova: SIP, VOIP, vyvažování zátěže, Kamailio

Summary:

Nowadays, VOIP technology is still more used. In VOIP is also used SIP protocol to build sessions between users. As technology traffic grows, there is a question of how to control and balance this traffic for the devices that work with this protocol. The theme of this work is to do a detailed analysis of SIP load balancing. The aim of this bachelor's thesis is to analyze the available products and verify load balancing functionality on one of the solutions in real traffic of the SIP protocol.

Index Terms: SIP, VOIP, load balancing, Kamailio

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

V dnešní době stále více slýcháváme pojem VOIP (Voice over Internet Protocol). V několika posledních letech, nebo spíše desetiletích, se tato technologie stále více rozšiřuje. Přímo to souvisí s lepší dostupností připojení k internetu a navyšování rychlosti u koncových zákazníků.

Pokud se zaměřím pouze na telefonii je důvod použití VOIP řešení oproti klasické telefonii jasný, a tím je cena hovoru. Sám jsem se setkal s tím, že ve spoustě firemních prostředí je používáno toto řešení již několik let. Nemusí se jednat jen o firemní prostředí ale i běžné domácnosti. Existuje velké množství operátorů, kteří tyto služby v dnešní době nabízejí.

A proč jsou tedy jejich nabídky výhodné? Hlavním důvodem je nízká režie provozu a také není potřeba zřizovat připojení k telefonní síti nebo vlastnit nějakou větší telefonní infrastrukturu. Vše funguje v rámci protokolu IP a stačí mít připojení k internetu.

V posledních letech komunikační a sdělovací technologie směřuje ke světu IP. Můžeme to pozorovat i na rozšíření použití IP televizí nebo technologii VoLTE či rozvoj technologie Internetu věcí. Tato fakta mě jen utvrzují v názoru, že budoucnost patří světu IP včetně VOIP.

V technologii VOIP je jedním z nejrozšířenějších protokolů SIP protokol. Jedná se o protokol pro sestavování relací. Nejedná se pouze o telefonní hovory ale i konferenční hovory, video hovory nebo služby pro Instant Messeging. Pokud se na problematiku podíváme z druhé strany tedy například od poskytovatele, nebo z pohledu nějakého většího podniku, je třeba mít zařízení, která zvládnou velký provoz komunikačních dat s tímto protokolem. Zde může nastat situace, kdy bude zapotřebí vyvažovat zátěž mezi více serverů anebo mezi více lokalit.
A tím se dostávám k náplni této práce a tím je analyzovat problematiku vyvažování zátěže pro protokol SIP.

Tato bakalářská práce je rozdělena na čtyři hlavní části. První část je zaměřena na detailní teoretický rozbor protokolu SIP. Druhá část detailně rozebírá pojem vyvažování zátěže neboli Load Balancing, jeho algoritmy a technické možnosti.
Třetí část spojuje předchozí dvě v jeden celek. Zabývá se totiž analýzou trhu pro řešení vyvažování zátěže protokolu SIP. Je zde uvedena charakteristika dostupných řešení a to jak softwarového charakteru tak hardwarového a volně dostupného či řešení komerčního. V závěru této kapitoly jsou všechna řešení porovnána a vybráno je jedno z nich, které je volně dostupné. Tímto přímo přecházím k části poslední a tou analýza tohoto vybraného produktu. Ve virtualizované VOIP infrastruktuře je provedena instalace této aplikace a její následná konfigurace. Poté je ověřena jeho funkčnost při vyvažování reálného provozu protokolu SIP.

2. Protokol SIP

SIP (Session Initiation Protocol - protokol pro inicializaci relací) se začal vyvíjet v roce 1996 skupinou Multi-Party Multimedia Session Control Working Group MMUSIC v rámci IETF (Internet Engineering Task Force). Tento protokol byl standardizován v březnu roku 1999 v rámci normy RFC 2543 [1]. Byla založena skupina, která začala vyvíjet hlavní jádro protokolu. V roce 2002 byla vydána nová norma RFC 3261 [2], která specifikuje i aktuálně používanou verzi protokolu SIP 2.0.Jeho předchůdcem a později konkurentem je starší signalizační protokol H.323 ale je oproti SIP protokolu výrazně složitější. Hlavním rozdílem je, že H.323 používá binární formát založený ASN.1 [3], což představuje větší složitost při konfiguraci a ladění provozu. Protokol SIP proto vznikl jako reakce na tento standard, a snaží se být co nejjednodušší a založený na Internetem dobře prověřených principech. Protokol SIP je oproti H.323 textově orientovaný. Protokol SIP přejímá základní principy ze dvou obecně používaných internetových protokolů [4]. Prvním je protokol HTTP (Hyper Text Transport Protocol) pro webové služby a druhým je protokol SMTP pro elektronickou poštu. Z těchto protokolů přejímá hlavně to, že je právě textově založený, přebírá schéma klient-server a používání URL a URI. Dále se jedná o převzetí struktury protokolu a to strukturu používání hlavičky protokolu. SIP používá hlavičky ze SMTP jako To, From, Date, Subject.

SIP je signalizačním protokolem na aplikační vrstvě závisející na protokolech vrstev nižších. Protokol SIP tedy neurčuje, jaký bude použit transportní protokol, jak bude řešen přenos multimediálních dat a další řízení hovoru. Má za úkol řešit sestavení spojení dvou a více účastníků a následný dohled na toto spojení a také rušení tohoto spojení. Jako transportní protokol se nejčastěji používá nespojovaný protokol UDP (User Datagram Protocol), který sice nemusí být spolehlivý, protože není zaručeno 100% doručení paketů, ale má menší zpoždění oproti protokolu TCP (Transmission Control Protocol). Jedná se o tzv. end-to-end protokol, což znamená, že koncová zařízení mají uloženu veškerou logiku a znají i jednotlivé stavy komunikace, tímto je zvýšena odolnost komunikace před vzniklými chybami. Výjimkou je směrování SIP zpráv.

O řízení přenosu se stará protokol SDP (Session Description Protocol), který je zapouzdřený v těle SIP zprávy. Tento protokol dojednává parametry přenosu mediálního obsahu a schopnosti zařízení pro tento přenos. Například vyřizuje jaké porty a IP adresy pro vysílání a přijímání dat se mají použít, jaké kodeky strany podporují, metody šifrování, jakou šířku pásma využít atd. [5].

Pro přenos multimediální zprávy se používá protokol RTP standardizovaný v RFC 3550 [6]:

Přenosový protokol v reálném čase (RTP, Real-time Transport Protocol) je protokol, který se používá k paketovému přenosu mediálního obsahu ke koncovým uživatelům. Zakládá se na synchronizaci časového přenosu a zjištění ztráty nebo nesprávného pořadí dat. RTP nejčastěji používá protokol UDP, ale může využít i jiné protokoly. Bezpečnou variantu RTP představuje protokol SRTP (Secure Real-time Transport Protocol) [4].

Pokud je třeba SIP zabezpečit, lze využít IPSec či TLS (Transport Layer Security), pro zabezpečený přenos hlasu lze využít SRTP (Secure Real-time Transport Protocol).

2.1 Struktura sítě s protokolem SIP

V síťové infrastruktuře protokolu SIP se vyskytuje několik zařízení, které jsou schopná s ním spolupracovat. Lze je rozdělit na několik typů, které jsou uvedeny v této kapitole.

2.1.1 Server Proxy

Přijímá SIP žádosti od uživatelů UA nebo od jiného Proxy serveru a jedná za uživatele při předávání nebo odpovídání na žádost. Lze to přirovnat k routeru pracujícímu s pakety na třetí vrstvě modelu OSI [7], s tím rozdílem, že zde se pracuje se SIP zprávami na vrstvě aplikační. Pokud nezná volaného, předává žádost jinému Proxy serveru, dokud některý neodpoví. Poté volanému posílá žádost o spojení.

Proxy servery se dají rozdělit na dvě základní skupiny:

- Stateful (informace o stavech) - Po přijetí požadavku vytvoří stav a ten drží, dokud spojení není ukončeno.
- Stateless (bez informace o stavu) Přeposílají zprávy nezávisle na vzájemných vazbách.

Speciální příklad Proxy je tzv. **Forking Proxy**.
Tato funkcionalita nastává v případě, pokud proxy zjistí, že uživatel je registrovaný na více místech. Proto pošle obdrženou zprávu s metodou INVITE na více zařízení a podle toho, jaké odpovědi přijdou zpět (200 OK nebo 404 Not Found), tak postupně maže ze své paměti jednotlivé dialogy.

2.1.2 Registrar server (Registrační server)

Na registrar server se posílají uživatelské požadavky na registraci. Přijímá tedy pouze požadavky typu REGISTER a na všechny ostatní odpovídá zprávou 501 Not implemented. Server tím získává údaje o uživateli, jako je IP adresa, port nebo uživatelské jméno. Může se specifikovat i doba pro registraci, jinak je nastavena defaultní a registraci je nutné neustále obnovovat.

2.1.3 Redirect server

Redirect server již podle názvu slouží k přesměrování na jiné cílové místo. Pracuje podobně jako Proxy server s tím rozdílem, že žádosti nikam nepřeposílá ale sdělí odesílateli jím hledanou adresu a ten se poté spojí sám nezávisle na redirect serveru. Stejně jako proxy server využívá databázi a lokalizační služby k nalezení cílového uživatele. Odesílateli vrací odpověď pomocí několika typů zpráv ze skupiny 3xx , které jsou detailně uvedeny v kapitole 2.2.4.

V odpovědi samozřejmě nemusí být žádná adresa, pokud ji redirect server nenalezne, ale může jich tam být naopak i více než jedna.

2.1.4 Location server

Nazval bych ho spíše lokalizační službou. Není přímo specifikována jeho podoba a většinou záleží na konkrétním řešení daného softwaru. Já bych ho definoval jako službu, která poskytuje informace o současné poloze uživatele, tedy o IP adrese, pro servery Proxy a Redirect. Údaje získává za pomoci Registrar serveru. Jedná se tedy o databázi s umístěním uživatelů a přístup k ní.

2.1.5 Sip Gateway

Neboli brána protokolu SIP slouží jako rozhraní mezi protokolem SIP a nějakým jiným protokolem. Například se může jednat o bránu mezi standartní telefonní PSTN sítí a SIP sítí. Gateway zde podporuje oba standardy a překládá jak signalizaci, tak média z jednoho formátu do druhého.

2.1.6 UA – User Agent

Koncový uživatel SIP sítě, který se stará o inicializaci hovorů například. Nejčastěji jsou reprezentovány právě SIP telefony. A to jak hardwarovými telefonu tak jejich softwarovými ekvivalenty.

Tito uživatelé se dají rozdělit na dvě skupiny, podle toho v jakém režimu zrovna pracují.

- User Agent Client (UAC) - má na starost inicializaci spojení.
- User Agent Server (UAS) - reaguje na příchozí žádosti a odesílá odpovědi.

Protože koncové zařízení téměř vždy obsahuje UAC a UAS, tak je vhodné používat jen zkratku UA. Například, volající UA funguje jako UAC, když odesílá zprávu INVITE a přijímá odpověď na požadavek. Volaný UA se chová jako UAS, když obdrží zprávu INVITE a odesílá odpověď. Ale tato situace se obrátí, když se volaný rozhodne zavěsit a ukončuje se spojení odesláním zprávy BYE.

Presence Agent

Tento speciální typ SIP zařízení zpracovává odběry pomocí metod SUBSCRIBE a NOTIFY případně informuje ostatní UA. Lze použít například k automatické kontrole stavu ostatních SIP zařízení.

2.1.7 Back to Back User Agent - B2BUA

Back to back uživatel funguje jako zprostředkovatel. Obdržené SIP požadavky upraví a posílá je dále. Stejně tak pracuje i s odpovědí. Toto se dá využít na isolaci některého UA a skrýt veškeré informace o něm, tedy bude vystupovat jako anonymní. Dále ho může využívat nějaký SIP poskytovatel služeb pro monitorování signalizace protokolu SIP a následné účtování zákazníkům.

Dalším příkladem využití B2BUA je Application Layer Gateway. B2BUA zde funguje v podstatě jako firewall pro protokol SIP. Některé firewally mají tuto funkci přímo zabudovanou. Jako B2BUA může vystupovat i ústředna Asterisk.

2.2 SIP Zpráva

Komunikace užívající SIP protokol je tvořena zprávami. Každá zpráva obsahuje hlavičku zprávy (header) a vlastní obsah (body). V prvním řádku zprávy je identifikován její typ.

Zprávy dělíme na dvě skupiny:

- Metody (Žádosti) - Jsou obvykle užívány k inicializaci procedury, například sestavení či ukončení relace.
- Odpovědi – Informují a přijetí žádosti nebo stavu zpracování.

2.2.1 Hlavička protokolu SIP

V této podkapitole se podívám na jednotlivá pole hlavičky protokolu SIP. Polí pro hlavičku je veliké množství proto uvedu a vysvětlím jen ty s nejrozšířenějším použitím. Lze je rozdělit dle použití na request and response, request only, response only a na hlavičky popisující tělo zprávy [4].

Zde jsou nejběžněji používané hlavičky:

- To - Musí ho obsahovat každá SIP zpráva a určuje, kam má být zpráva doručena
- From - Adresa uživatele který spojení realizoval
- Via - Adresa uživatele, který vysílá požadavek nebo adresa serverů přes které požadavek prošel a kudy se bude vracet odpověď
- Call-Id - Unikátní identifikace volání
- Contact - Aktuální, skutečná adresa klienta
- Record-Route - Seznam adres serverů, kteří chtějí dostávat veškerou komunikaci náležející k hovoru
- Route - Posloupnost adres serverů, přes které je požadavek směrován a klienta, ke kterému má požadavek dorazit. Každý server je přidáván pod novým polem
- Request URI - Aktuální adresát požadavku. Údaj se vyskytuje v první řádce požadavku za typem metody
- CSeq - Využívá se ke správě pořadí požadavků, pokud by se stalo, že žádosti dorazí v různém pořadí lze pomocí tohoto pole identifikovat pořadí správné.
- Contact - Obsahující IP adresu a port, na kterém odesílatel očekává další žádosti odesílané volaným.
- Expires - Například u metody REGISTER. Označuje dobu vypršení registrace
- Max forwards - Používá se pro určení maximálního počtů skoků zprávy SIP. Postupně se dekrementuje v postupu zprávy.
- Subject - Může být označena i zkratkou s. Indikuje předmět mediálního obsahu zprávy.
- Content-Length - Má význam délky těla zprávy.
- Allow - Obsahuje seznam metod, které UA podporuje.
- User-Agent - Doplňující informace o UA.
- Record-Route - Používá se například při průchodu přes více proxy. Postupným průchodem metody INVITE se přidávají hlavičky Record route.
- Date - Čas a datum.
- Supperted - Vyjmenuje podporované funkce klienta nebo serveru.

2.2.2 Ukázka hlavičky zprávy SIP

Níže je uvedena ukázka skutečné hlavičky zachycené na ústředně Asterisk. První je metoda INVITE a druhá ukázka je potvrzující zpráva OK. Na této ukázce je vidět použití hlaviček, které jsem uvedl v předchozí kapitole 1.3.1.

Příklad hlavičky zprávy s metodou INVITE:

```
INVITE sip:192.168.33.25 SIP/2.0.
Via: SIP/2.0/UDP 192.168.33.2:5060;branch=z9hG4bK39447748.
Max-Forwards: 70.
From: <sip:9@192.168.33.2:5060>;tag=as095fff68.
To: <sip:192.168.33.25>.
Contact: <sip:9@192.168.33.2:5060>.
Call-ID: 1eb8cd33615304a0788d87df65ad1a27@192.168.33.2:50609.
CSeq: 102 INVITE.
User-Agent: Asterisk PBX 14.5.0.
Date: Wed, 23 May 2018 06:16:01 GMT.
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO, PUBLISH, MESSAGE.
```

Supported: replaces, timer.
Remote-Party-ID: "9" <sip:9@192.168.33.2>;party=calling;privacy=off;screen=no.
Content-Type: application/sdp.
Content-Length: 284.

Příklad hlavičky zprávy s odpovědí 200 OK:

SIP/2.0 200 OK.
Via: SIP/2.0/UDP 192.168.33.35:5060;branch=z9hG4bK8b865504925c0fa144514ea2fa281e0a;received=192.168.33.35;rport=5060.
From: "Telefon1" <sip:1@192.168.33.2>;tag=284437547.
To: "Telefon1" <sip:1@192.168.33.2>;tag=as7d5aaed3.
Call-ID: 4257572473@192_168_33_35.
CSeq: 35655 REGISTER.
Server: Asterisk PBX 14.5.0.
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO, PUBLISH, MESSAGE.
Supported: replaces, timer.
Expires: 180.
Contact: <sip:1@192.168.33.35:5060>;expires=180.
Date: Wed, 23 May 2018 06:18:24 GMT.
Content-Length: 0.

SIP URI

K identifikaci jednotlivých SIP zařízení se používá SIP URI (Uniform Resource Identifier). Konkrétní URI je vidět v předcházející ukázce zprávy INVITE v poli from.

SIP URI má podobnou formu jako emailová adresa a má následující formát:

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

2.2.3 Metody

SIP metody jsou termíny v protokolu, při jejichž požadavku následuje specifická operace u klienta či na straně serveru. Šest základních metod, které jsou popsané přímo v RFC 3261 [2] pro protokol SIP jsou INVITE, REGISTER, BYE, ACK, CANCEL a OPTIONS.

Další metody, které existují, ale definovány v jiných standardech jsou REFER, SUBSCRIBE, NOTIFY, PUBLISH, MESSAGE, UPDATE, INFO a PRACK.

INVITE

Metoda INVITE slouží k přizvání uživatele nebo služby k podílení se na relaci. Tělo zprávy obsahuje popis relace (spojení). Lze ji použít také ke změně parametrů již probíhajícího spojení.

ACK (Acknowledge)

Metoda ACK potvrzuje, že klient v pořádku přijal odpověď na INVITE metodu. Při inicializaci se používá tzv. 3-way hand shake. Volaný periodicky opakuje odpověď OK, dokud od protistany nepřijme odpověď ACK až poté může začít hovor.

CANCEL

Metoda CANCEL ukončuje nevyřízený požadavek se stejnou identifikací, tedy položkami Call-ID, To, From a pořadovým číslem požadavku Cseq. Vyřízené požadavky již metoda CANCEL neovlivní. Požadavek je vyřízen, pokud byla odeslána konečná odpověď, např. 200 OK. Typickým příkladem použití je při sestavovaném hovoru, pokud ještě volaný nepotvrdil žádost INVITE a volající chce zrušit sestavování spojení. Pak se využije metoda CANCEL.

BYE

Klient používá metodu BYE k oznámení protistraně, že hodlá ukončit hovor. Metoda BYE může být vyslána jak volaným tak volajícím. Na rozdíl od metody CANCEL se používá již při sestaveném spojení. Metodu mohou použít pouze koncoví uživatele, nikdy proxy server nebo jiný prvek v cestě.

REGISTER

Metoda REGISTER je používána k registraci současné adresy klienta u SIP serveru, který jí předá lokalizační službě. Smyslem žádosti je sdělit aktuální polohu uživatele. V této zprávě je přenášena informace o aktuální IP adrese a portu, na které lze uživatele zastihnout. Registrace jsou omezeny časem a musí se tedy pravidelně obnovovat.

OPTIONS

Je žádost o zaslání vlastností a dostupnosti serveru nebo UA. Nikdy není generován proxy serverem ale UA.

Ostatní metody

SUBSCRIBE

Tato metoda se používá k přihlášení odběru nebo přijímání událostí. Uživatel, který chce dostávat oznámení, posílá zprávu SUBSCRIBE na server. Zpráva SUBSCRIBE zahajuje dialog a server okamžitě odpovídá 200 OK. Od této chvíle je dialog sestaven a server zasílá metodu NOTIFY pokaždé, jestliže se změní událost, kterou chce uživatel sledovat. Žádost SUBSCRIBE obsahuje pole hlavičky Expires, která definuje dobu spojení pro odběr. Je tedy nutné, jako u metody REGISTER. Obnovuje se novým odeslání metody SUBSCRIBE. Pro zrušení odběru metoda není a použije se tato metoda s parametrem Expires nastaveným na 0.

Pro použití naléhavých zpráv se používá metody MESSAGE. Metoda MESSAGE nesestavuje dialog a text naléhavé zprávy je přenášen uvnitř SIP žádosti.

NOTIFY

Metoda NOTIFY se používá k upozornění, že došlo ke změně události, která byla požadována předchozí metodou SUBSCRIBE. Metoda se odesílá na začátku přihlášení odběru a také na konci odběru. Na žádost se odpovídá zprávou 200 OK.

PUBLISH

Aktualizace stavu informace na serveru. Lze použít například pro oznámení nové hlasové zprávy na serveru.

REFER

Označuje, že příjemce (identifikován podle URI požadavku) by měl kontaktovat třetí stranu s použitím kontaktních informací uvedených v požadavku.

MESSAGE

Slouží k předání rychlé zprávy za použití protokolu SIP. Zpráva MESSAGE nesestavuje dialog, jako například metoda SUBSCRIBE ale text naléhavé zprávy je přenášen uvnitř těla SIP zprávy. Při obdržení se odpovídá zprávou 200 OK.

INFO

Metoda INFO slouží k přenosu signalizačních informací během relace.

Žádost INFO může být použita pro zaslání doplňujících informací během hovoru, například signalizace mezi bránami do veřejné telefonní sítě, informace o kvalitě signálu nebo informace o stavu účtu uživatele.

PRACK

Tato metoda má stejnou funkci jako metoda ACK, avšak s rozdílem, že se jedná o provizorní potvrzení. Jelikož pro provizorní odezvy protokol SIP nezajišťuje spolehlivé koncové doručení, lze pro tento případ metodu využít. Například pro potvrzení informačních zpráv skupiny 100. Na potvrzení odpovědí skupin 200 a výše se používá metoda ACK.

UPDATE

Metoda UPDATE slouží ke změně parametrů relace, ale nemá vliv na změnu stavu dialogu. Je to podobné jako znovu použití metody INVITE ale metoda UPDATE může být použita ještě pře dokončením počáteční metody INVITE. Používá se hlavně k aktualizaci parametrů v počátku relace. Ovlivňuje například protokol SDP, třeba přenos mediálních dat a kodeky atd.

2.2.4 SIP odpovědi

Odpovědi se dělí na šest skupin podle jejich významu.

1xx: Informační odpovědi o zpracovávání požadavku UA. Odesílají se v reakci na žádosti, které byly přijaty, ale ještě nejsou zpracovány.

- 100 – Trying – Informuje, že bude následovat nějaký druh akce či zpracování.
- 180 – Ringing – U klienta začíná vyzvánění.
- 181 – Call is being Forwarded – Upozornění o přesměrování na jiné cílové místo.
- 182 – Queued – Požadavek je vložen do fronty a bude následně zpracován.
- 183 – Session progress – Postup relace.

2xx: Odpověď úspěšně vykonána.

- 200 – OK – Potvrzuje úspěšnou žádost.
- 202 – Accepted – Žádost přijata ale nebyla splněna.
- 204 – No notification – Používá se v některých případech jako odpověď pro metodu SUBSCRIBE.

3xx: Odpovědi o přesměrování. Informují o nové poloze uživatele nebo služby.

- 300 – Multiple choises – Informuje, že v hlavičce Contact je více políček a zpět můžeme dostat více cílových adres.
- 301 – Moved permanently - Uživatel je přemístěn na jiné adrese a to trvale.
- 302 - Moved temporarily – Uživatel je přemístěn na jiné adrese, ale jen dočasně.
- 305 – Use proxy – Pro zjištění adresy kontaktujte proxy server.
- 380 – Alternative service – Vrátí, jaký typ služby nabízí.

4xx: Dává zprávu o chybě. Požadavek selhal kvůli chybě na straně uživatele. Uživatel by měl upravit svůj požadavek před následujícím pokusem.

400 - Bad Request – Zpráva měla špatný formát

401 - Unauthorized – Je třeba autorizovat uživatele.

403 - Forbidden - Tato odpověď se používá k zamítnutí žádosti a neobsloužení žádosti. Volajícímu není sděleno žádné řešení.

404 - Not Found – Uživatel nebyl nalezen na dané adrese

405 - Method Not Allowed - Metoda není povolena

406 - Not Acceptable – Informuje, že v SIP zprávě je nějaký požadavek, který nelze splnit

407 - Proxy Authentication Required – Podobné jako 401, ale je požadována autentizace u proxy serveru

408 - Request Timeout - Vypršel čas pro zpracování žádosti

410 - Gone – Uživatel existoval, ale není již k dispozici

414 - Request-URI Too Long - URI příkazu je příliš dlouhé

415 - Unsupported Media Type - Typ média není podporován

480 - Temporarily Unavailable - Dočasně není k dispozici

484 - Address Incomplete - Adresa není úplná

486 - Busy Here - Uživatel je zaneprázdněný

487 - Request Terminated - Příkaz ukončen

488 - Not Acceptable Here - Nepřijatelný

5xx: Chyba na straně serveru. Zpráva je pravděpodobně v pořádku, ale chyba nastala při jejím zpracování.

500 - Server Internal Error – Informace o vnitřní chybě serveru.

501 - Not Implemented – SIP zpráva obsahuje neznámou metodu.

502 - Bad Gateway - Nesprávná brána.

503 - Service Unavailable – Server je dočasně nedostupný.

504 - Server Time-out - Časová limit serveru.

513 - Message Too Large - Zpráva je příliš velká pro zpracování.

6xx: Všeobecná chyba. Tento kód je vysílán, pokud žádost nemůže být splněna na žádném serveru.

600 Busy Everywhere - Vše je zaneprázdněno.

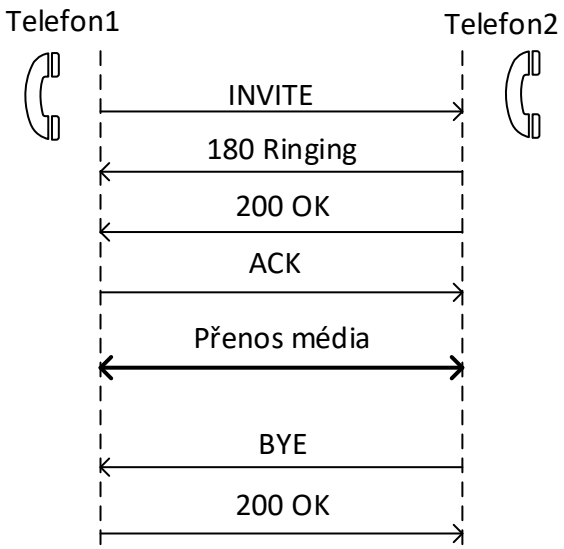
603 Decline - Odmítnutí, odpověď obsahuje důvod zamítnutí.

604 Does Not Exist Anywhere - Nikde neexistuje.

606 Not Acceptable – Uživatel byl kontaktován, ale jsou nevyhovující parametry spojení.

2.3 Sestavení relace

Po tom co jsem v předchozích kapitolách vysvětlil, co jsou to metody a odpovědi SIP, je možné si ukázat, jakým způsobem by proběhlo sestavení jednoduché relace, konkrétně hovoru. Názorné schéma je uvedeno na obrázku 2.1, kde Telefon1 volá na Telefon2.



Obr. 2.1 – Diagram SIP relace

Hovor je inicializován ze strany Telefonu1. Pošle se zpráva s metodu INVITE, která přizve účastníka Telefon2 do podílení se na relaci. Telefon2 pošle informační zprávu 180 Ringing, která znamená, že na telefonu započalo vyzvánění. Na straně Telefonu2 je hovor přijat a je poslána zpráva 200 OK, ta se bude posílat periodicky, než z protistrany dorazí odpověď ACK. Tímto může dojít k přenosu média, tedy hlasu v tomto případě. Hovor probíhá, dokud se ho strana telefon 2 nerozhodne ukončit. Při ukončení se odešle zpráva s metodou BYE a po odpovědi 200 OK od protistrany dojde ke korektnímu ukončení relace.

3. Load Balancing

3.1 Definice základních pojmů

Load balancing nebo také Server Load balancing (SLB), v češtině vyvažování zátěže, je v oblasti informačních technologií základním nástrojem pro rozložení zátěže mezi více prvků. Může se jednat například o rozložení zátěže mezi počítače, paměťová média nebo také rozložení provozu v komunikačních sítích. Cílem je dosáhnout optimálního využití zdrojů a případně i zvýšit dostupnost služby neboli odolnost proti výpadku.

V souvislosti s definicí tohoto termínu by bylo vhodné uvést několik základních pojmů, které jsou s ním spojené a několikrát je použiji v následujícím textu.

HA (High availability) - V češtině vysoká dostupnost je pojem, který představuje dostupnost nějaké služby. Je tím myšleno, zda je služba aktivní a dostupná v momentě potřeby koncového uživatele. Samozřejmě ji také lze vyjádřit v procentuální hodnotě a vyskytuje se v nabídce různých služeb.

Failover - Pojem, který lze použít také pro zajištění dostupnosti služby pro koncového uživatele. Konkrétně jde o to, že v případě výpadku prvku dojde k přesunutí či přesměrování na prvek záložní.

Scaling - V překladu škálovatelnost, bych definoval jako vlastnost dané služby či aplikace dynamicky se přizpůsobit změně systémových prostředků, dle vytížení bez dopadu na funkčnost.

Cluster - Pokud se omezíme pouze na počítače, tak se dá definovat jako seskupení počítačů, které jsou propojeny a úzce spolupracují. Navenek se mohou tvářit jako jeden stroj. Jednotlivé stroje v clusteru se nazývají nody. Clustery je možné rozdělit do několika základních skupin, nicméně nejvíce používaná je jejich kombinace:

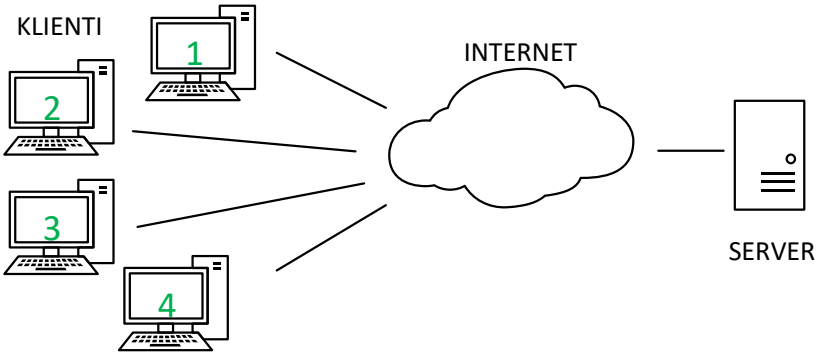
Výpočetní (High performance computing) cluster - Slouží k maximalizaci výpočetního výkonu.

HA (failover) cluster - Cluster, který zajištuje nepřetržitý provoz nějaké služby a je tím pádem odolný proti selhání některého ze členů clusteru.

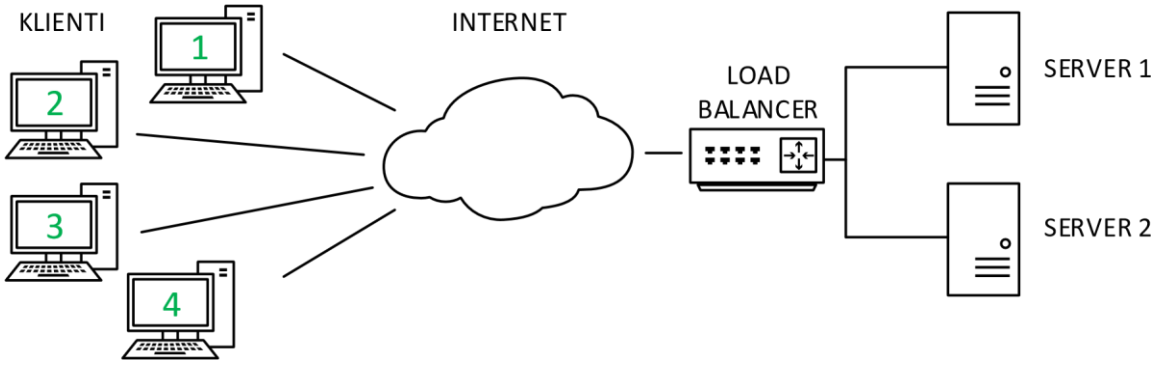
Scallable cluster - Takový cluster, který využívá právě loadbalancing pro rozložení zátěže mezi jednotlivé nody v clusteru.

Storage cluster - Slouží k zpřístupnění diskové kapacity rozložené mezi více strojů.

Na obrázcích 3.1 a 3.2 jsou uvedena základní schémata bez použití load balancingu a s jeho použitím. Jde pouze o představení problému. Z obrázku je patrné, že pro vyvažování je použité nějaké dodatečné zařízení zvané Load balancer, které nám tuto funkci obstará. Toto zařízení může být nějaké hardwarové řešení, nebo naopak softwarové. Samozřejmě také nemusí být nutné použít dodatečné zařízení a jednotlivé nody v clusteru si mohou vyvažování obstarávat sami mezi sebou.



Obr. 3.1 - Zjednodušené schéma bez použití load balanceru



Obr. 3.2 - Zjednodušené schéma s použitím load balanceru

3.1.1 Výhody použití vyvažování zátěže

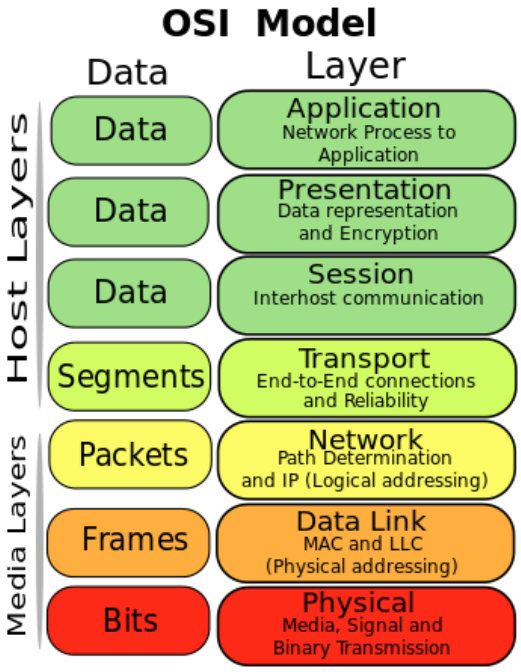
Pokud se rozhodneme, pro poskytování nějaké služby většího rozměru, bude třeba mít zajištěn dostatečný výkon pro její provoz ale také mechanismus, který bude schopen službu ochránit proti výpadkům a také zajistí automatizovanou správu. Protože v dnešní době, i krátký výpadek může znamenat finanční újmu nebo ztrátu zákazníka.

Výhody použití:

- Navýšení výkonu pro nabízenou službu v případě webové aplikaci například rychlejší odezva.
- Zajištěná nepřetržitá dostupnost aplikace a odolnost proti selhání dílčího prvku.
- Pokud používám více strojů, není problém na některém z nich provádět HW i SW úpravy bez nutnosti výpadku či odstavení služby.
- Možnost přidat do clusteru další stroj pro zvýšení celkového výkonu.
- Snížení nákladů. Myslím tím, že lze používat méně výkonné servery a zátěž rozložit mezi ně.

3.2 Pohled dle vrstev OSI modelu

Load balancing lze rozdělit na několik typů. Nejjednodušším kritériem je rozdělení problematiky dle modelu OSI/ISO [7].



Obr. 3.3 – Schéma OSI modelu [8]

3.2.1 Druhá vrstva

Load balancing na druhé vrstvě, je sdružování více fyzických spojů do jednoho logického, které bude mít vyšší propustnost než spoje jednotlivě. Pokud jsou spoje vedeny geograficky různými cestami, pak jejich agregace nám zajistí redundanci a zvýší odolnost proti výpadkům.

3.2.2 Třetí a čtvrtá vrstva

Tedy zpracovaní paketů na síťové vrstvě. Bohužel vyvažování na úrovni třetí vrstvy ve většině případů nestačí a je nutno zpracovat segmenty i z vrstvy čtvrté. Vyvažování tedy probíhá na základě IP adres, TCP a UDP portů u každého z paketů. Takže se nekontroluje obsah paketů, a proto také tyto load balancery budou mít vyšší propustnost nežli ty pracující na vrstvě aplikační. Jedná se hlavně o hardwarové load balancery, protože nutné operace jako přepočítávání kontrolních součtů, vyhledávání v tabulkách nebo překlad adres, vyžadují větší výkon.

SW řešení pracující na čtvrté vrstvě povede ke značnému snížení výkonu. Na jednoduché úlohy, které by bylo možné zpracovat na úrovni paketů, musí systém decapsulovat jednotlivé vrstvy a najít tak odpovídající data, alokovat paměť, předat je procesu, navázat spojení. Vzniknou i větší nároky na paměť. Z tohoto důvodu bude softwarové řešení několikrát pomalejší než hardwarové zařízení postavené za tímto účelem. Na první pohled se zdá, že vše mluví pro použití dedikovaných zařízení. Ovšem jejich velikou nevýhodou je cena. Naproti tomu softwarové řešení lze zprovoznit na drtivé většině dostupného hardwaru a velké množství softwaru je volně dostupného.

Z pohledu čtvrté vrstvy se dají popsat základní principy použití load balanceru:

NAT (Routed) režim

V tomto režimu load balancer směruje provoz mezi uživatelem a serverem tak, že u paketů mění cílovou IP adresu. Pokud se jedná o forwardování mění se i zdrojová adresa. TCP spojení je vytvořeno mezi klientem a serverem. Load balancer, zde musí být nastaven jako GW pro jednotlivé servery. Jedná se o rychlý způsob LB.

DSR

Direct Server Return [9] je režim load balanceru, kdy se směruje tak, že se mění pouze mac adresa. IP adresa zůstává a poté co si server požadavek zpracuje, pošle jej zpět přímo ke klientovi bez použití loadbalanceru. Opět jde o rychlý způsob LB.

3.2.3 Sedmá vrstva

Rozhodování o vyvažování na sedmé vrstvě (aplikační vrstva) probíhá na základě skutečného obsahu každé zprávy. Rozdělování provozu je tedy mnohem sofistikovanější, než je tomu u vyvažování na třetí a čtvrté vrstvě. Vyvažování na L7 nabízí spoustu výhod v tom, jakým způsobem jsou data směrována a řízena. Oproti tomu jsou zde i problémy jako pořadí paketů, u IPv4 fragmentace paketů, hlavička přes více paketů atd.

Zde se dají popsat také dva základní principy konfigurace load balanceru.

Proxy mode

Loadbalancer se nachází mezi serverem a klientem a obsluhuje všechny požadavky. Udržuje dva typy spojení. První s klientem, kde vystupuje jako server a přijímá požadavky a poté přeposílá odpovědi. Druhé spojení udržuje se serverem, kde vystupuje jako klient. Zde přeposílá požadavky a přijímá odpovědi. Nikdy tedy nedojde k přímému spojení klienta a serveru. Výhodou je že servery jsou z bezpečnostního hlediska nedostupné napřímo. Nevýhodou je pomalejší zpracování.

Transparent proxy mode

Velmi podobné předchozí metodě ale pro komunikaci se servery load balancer mění zdrojovou adresu na klientskou. Výhodou tedy je, že server pak vidí zdrojovou adresu klienta.

3.3 DNS vyvažování zátěže

Load balancing pomocí DNS:

Asi nejjednodušší způsob, jakým zařídit nějaký typ load balancingu.

Princip je takový že, pro DNS záznam máme přiděleno více IP adres pro více strojů a klientovi je pro daný záznam po dotazu přidělena IP adresa. Tímto právě získáme i jistý způsob vyvažování zátěže, protože DNS server nám bude vracet více IP adres ale v různém pořadí, a tak klienti budou přistupovat na různé servery. Zde ale také může nastat problém s tím, že se DNS záznamy mohou cachovat ať už v PC či u ISP providera. Hlavní problém nastane, pokud začne být aktuálně používaná adresa nedostupná. U klienta není nijak ošetřeno přejití na jinou adresu pro daný DNS záznam a služba se stane nedostupnou. Protože ne všechny aplikace umí využít vícero záznamů a v případě nedostupnosti jednoho, použít další v pořadí. Navíc nemáme žádnou kontrolu nad provozem. A poslední nevýhoda, kterou uvedu je provádění případných změn v záznamech DNS. Změna se totiž projeví až po delší době. Tento způsob pouze uvádím jako možnost a v následujícím odstavci uvedu jeho možné vylepšení.

Vylepšení DNS LB a Fail-over na úrovni IP:

Další možností je fail-over IP adresy. Pokud budeme mít na jednom koncovém stroji přiřazenou IP adresu a tento stroj se stane nedostupným, například pro hardwarovou chybu, adresa automaticky přejde na stroj záložní. Toto lze zajistit například pomocí protokolu VRRP [10] (Virtual Router Redundancy Protokol), který se používá zejména u routerů pro zajištění větší spolehlivosti sítě, kdy jedna IP je obsluhována několika uzly. Pro jednoduché použití v linuxovém systému se dá využít nástroj keepalived. Při konfiguraci tohoto nástroje se určí priorita pro dané stroje a tím zajistíme, kdo bude master a v případě výpadku si ten druhý převezme jeho IP adresu. V tomto stavu se ale nejedná o žádné vyvažování zátěže. Nicméně pokud bychom konfiguraci u keepalived vylepšili a přidali druhou virtuální adresu na oba stroje a obě umístili do DNS záznamu pro danou službu, dojde k situaci jako v předchozí podkapitole. S tím rozdílem, že při výpadku jednoho ze serverů přejdou obě adresy na server druhý a nedojde k výpadku [11].

Tímto se vylepší metoda vyvažování pomocí DNS a při výpadku jedné adresy nám služba zůstane dostupná. Tuto metodu lze použít pouze pro statické služby, protože nemáme zajištěnou synchronizaci dat klienta. Použití bych viděl třeba u statického webu či služby pro sdílení dat např. NFS.

3.4 GSLB

Neboli Global Server Load Balancing [9] je způsob rozložení síťového provozu mezi více serverů či datových center, která se nacházejí na různých geografických místech. Výhodou je, že pokud dojde k výpadku sítě například z důvodu výpadku elektřiny v daném regionu, tak load balancer začne provoz směrovat na jiné místo.

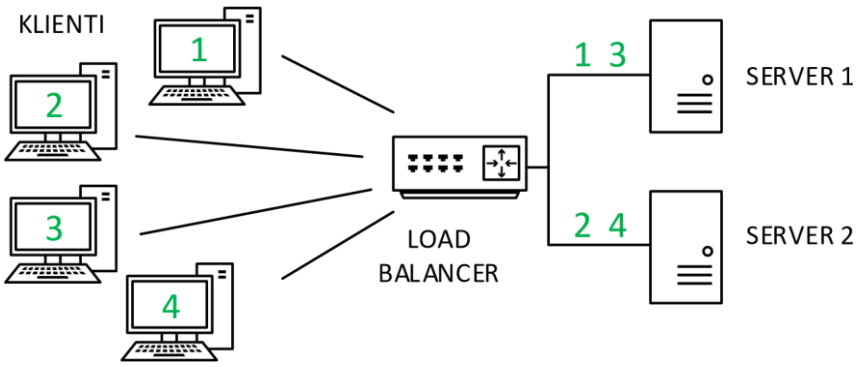
Tento způsob distribuce také může vést ke zvýšení výkonu. Lze například používat místo, které bude blíže a bude mít menší odezvu. Vzdálenost mezi uživatelem a serverem se stanoví pomocí doby odezvy a počtu skoků v cestě. Díky takto zjištěné vzdálenosti mohou být zákazníci a zaměstnanci přesměrováni na optimálně dostupný a nejblíže instalovaný server v rámci celé globální sítě. Další možnost využití může být dodávání lokalizovaného obsahu pro danou zemi, kdy load balancer začne směrovat požadavky na servery, které se nachází právě v té dané zemi. Často se využívá active-passive schématu. Toto řešení spočívá v tom, že služba je poskytována pouze z jednoho zdroje a data jsou duplikována na ostatní geografická místa. K jejich použití dojde až v případě výpadku hlavního zdroje.

3.5 Algoritmy vyvažování zátěže

V této kapitole bych rád uvedl některé základní algoritmy, které lze použít u konkrétních řešení vyvažování zátěže.

3.5.1 Round Robin

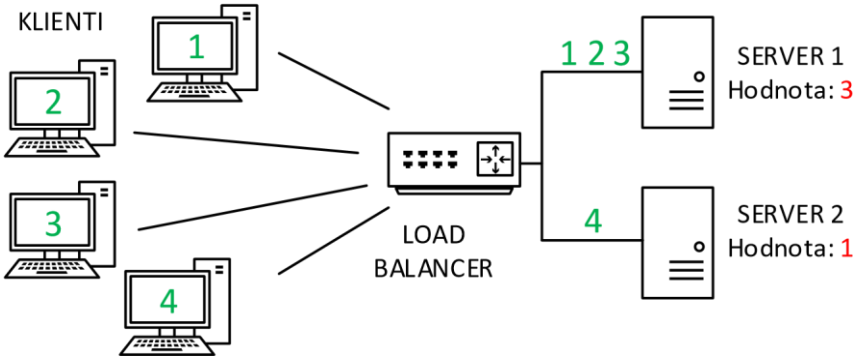
Nejjednodušší metoda, pokud pominu náhodné rozdělování. Jedná se o metodu statickou. Výhodou je nenáročná implementace. Žádost o připojení se předává postupně na další koncové servery cyklicky v pořadí. Z toho tedy vyplývá, že metoda nijak nerozlišuje výkonost jednotlivých nodů v clusteru. Proto je vhodné metodu využít tam, kde jednotlivé nody clusteru mají stejný výkon a na zpracování požadavku potřebují přibližně stejný čas. Nicméně nelze nijak kontrolovat náročnost příchozích požadavků a může nastat situace, kdy se na některém z nodů mohou hromadit úlohy, které jsou náročnější na výpočetní výkon, přestože se požadavky rozdělují rovnoměrně.



Obr. 3.4 - Zjednodušené schéma metody Round Robin

3.5.2 Round Robin s hodnotou

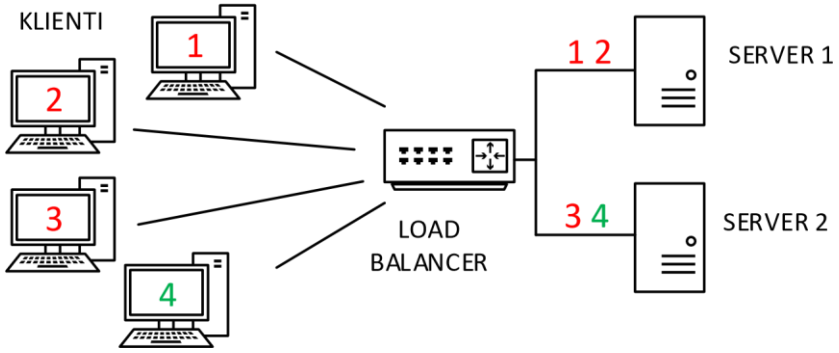
Jedná o vylepšenou verzi předchozí metody Round Robin. Jednotlivé požadavky jsou opět přiřazovány v postupném pořadí, ale rozdíl je v tom, že jednotlivé servery jsou ohodnoceny podle jejich výpočetního výkonu. Před použitím je potřeba stanovit výkon jednotlivých nodů. Metoda je tedy vhodnější, pokud servery v clusteru mají rozdílný výkon. Ale opět se zde nijak nedá zaručit, kam se budou směrovat výpočetně náročnější požadavky. Může nastat situace, kdy nejméně výkonný může být nejvíce zatížený.



Obr. 3.5 - Schéma metody Round Robin s hodnotou

3.5.3 Least Connections

Least connection, tedy nejmenší počet spojení je dynamická metoda, která počítá s okamžitou hodnotou zatížení serverů. Tato hodnota může být vyjádřena například aktivním počtem spojení. Každé nové spojení je směrováno na server s nejmenším počtem aktivních spojení. Aplikaci s malým nebo pomalu se měnícím provozem dává tato strategie velice dobré výsledky. Nevýhodou této metody je způsob ověření počtu aktivních spojení, jelikož ověřuje všechna udržovaná spojení, ale již nezjišťuje, která spojení jsou aktivně využita. Tato nevýhoda se může projevit u služeb, které využívají spojení typu klient-server, aniž by spojení využívali ke komunikaci. Při malém provozu a za předpokladu rovnoměrného rozložení takovýchto neaktivních spojení to není na závadu, ale při rostoucích zátěžích tento stav už může vést k přetížení některého serveru.

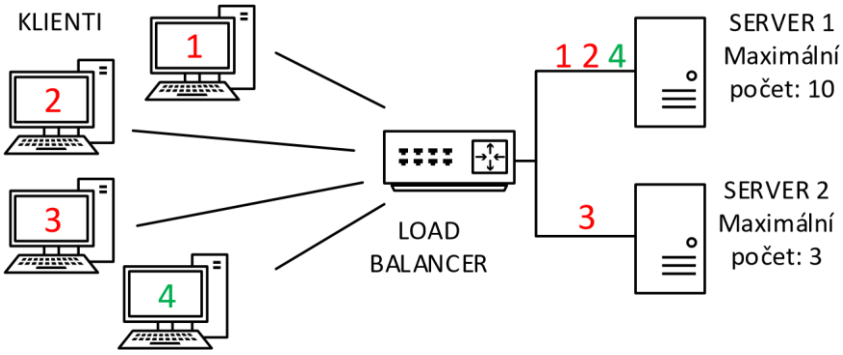


Obr. 3.6 - Schéma metody Least Connections (červení klienti, jsou ve výchozím stavu připojeni)

3.5.4 Least Connections s hodnotou

Metoda Weighted Least Connections je dynamickou metodou a modifikací metody předchozí. Vylepšení spočívá v tom, že kromě zjišťování aktuálního počtu udržovaných připojení na jednotlivých nodech, přidává také kontrolu maximálního množství připojení, která jsou jednotlivé koncové nody schopny udržovat. Při konfiguraci je tedy nezbytné nastavit tuto hodnotu pro každé zařízení individuálně. Příchozí spojení se tedy nepřidává serveru s nejmenším počtem spojení ale tomu, který má největší rezervu k maximální hodnotě.
Takto upravená metoda je efektivnější, jelikož rozlišuje výkony jednotlivých koncových serverů a dle toho přiřazuje příchozí spojení.

Nicméně problém s tím, zda aktivní spojení, je opravdu aktivně využíváno, zůstává stejný.



Obr. 3.7 - Zjednodušené schéma metody Least Connections s hodnotou

3.5.5 Observed

Jedná se o dynamickou metodu velmi podobnou předchozí metodě Weighted Least Connections. Nová spojení se distribuují stejně, tedy na základě počtu aktivních spojení a na maximálním počtu udržitelných spojení. Novinkou v metodě je parametr, který zohledňuje počet spojení odeslaných v minulosti. Tímto jsme schopni rozhodovat přesněji na základě předpovědi z nasbíraných dat z minulosti.

Tato metoda může být více užitečná pro rozdílné výkony jednotlivých serverů.

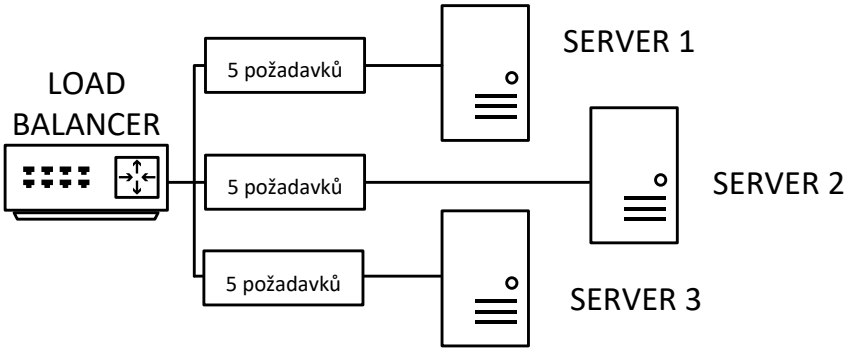
3.5.6 Hashing

Požadavky jsou rozdělovány na základě uživatelem definovaných klíčů. Klíčem může být například URL, zdrojová IP adresa, hodnota v hlavičce HTTP, SIP požadavku. Tento klíč může být zahashován a uložen do mapovací tabulky.

3.5.7 Even Size Task Queue

Metoda je založena na tom, že load balancer si udržuje údaje o tom, kolik požadavků má každý ze serverů ve své frontě. Fronta s požadavky obsahuje všechny požadavky, které jsou daným serverem aktuálně zpracovávány a ty, které teprve na zpracování čekají. Když server dokončí zpracování požadavku, je požadavek pro daný server odstraněn z fronty. Tato metoda funguje tak, že se snaží v každém okamžiku vyrovnávat velikost fronty pro každý server.

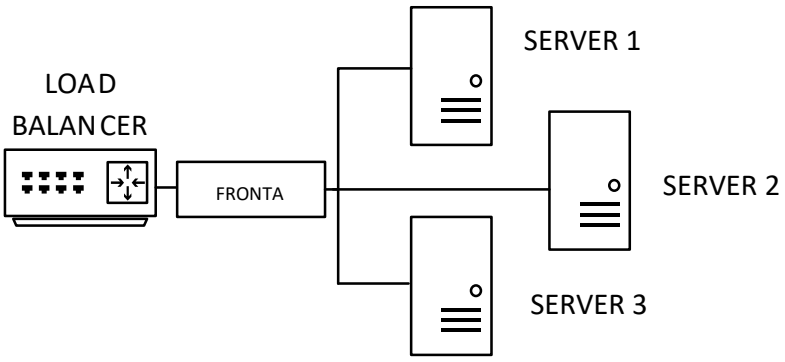
Tato metoda také defaultně zohledňuje náročnost na výpočetní čas pro jednotlivé požadavky. Nové požadavky jsou posílány na servery s nejmenším počtem požadavků ve frontě. Pokud je jeden ze serverů přetížen a tím se zvýší i velikost jeho fronty, nové požadavky se začnou rozdělovat na servery zbývající, dokud se fronty nevyrovnají. Toto schéma přináší také větší zátěž na load balancer, jelikož ten musí jednak sledovat počet požadavků ve frontě, ale také musí sledovat, které z požadavků jsou již dokončené, aby mohly být z fronty odstraněny.



Obr. 3.8 - Zjednodušené schéma metody Even Size Task Queue

3.5.8 Autonomous Queue

Veškeré příchozí požadavky jsou uloženy ve frontě. Servery v clusteru se připojují do této fronty a berou si takový počet požadavků, který jsou schopny zpracovat. V případě, kdy některý ze serverů v clusteru selže, zůstanou jeho potenciální požadavky nezpracovány ve frontě a jsou později zpracovány některými z dalších serverů v clusteru.
Výhodou tohoto řešení je to, že load balancer nemusí znát servery, které se nacházejí v clusteru. Jediné, co je potřeba zabezpečit je to, aby servery v clusteru věděly, kde se nachází fronta s úlohami. Toto schéma implicitně bere v potaz vytíženost serveru a jeho výpočetní kapacitu, neboť si server vždy vezme jen tolik, kolik je v daném okamžiku schopen zpracovat.



Obr. 3.9 - Zjednodušené schéma metody Autonomous Queue

3.5.8 Least Bandwidth, Least Response

Least Bandwith:
Metoda s nejméně využitou šířkou pásma. Metoda je založena na rozhodování podle využití pásma spoje k danému nodu. Hodnota je měřena například v Mbps.

Least Response:
Požadavky jsou rozdělovány na základě toho, který server v clusteru má nejrychlejší odezvu. Využití této metody může být například v situacích, kdy se servery nachází v různých logických sítích. Nevýhodou této metody může být to, že odezva nemusí být stejná i o pár sekund později.

3.5.9 Sticky Session

Předchozí metody jsou založeny na předpokladu, že každý z příchozích požadavků může být zpracován nezávisle na již zpracovaných požadavcích. Jako příklad bych uvedl přihlášení na webové stránce, kde se po autentizaci vytvoří session ID pro daného uživatele a je pro web autentizován. Tato session je pouze na node 1 a pokud následující dotaz bude nasměrován load balancerem na node 2, ten požadavek odmítne, protože nemá k dispozici data session z nodu 1. Řešením daného problému je metoda nazývaná jako Sticky Session. Load balancer si mapuje session ID k nodům v clusteru, a tak všechny příchozí požadavky patřící do stejné session, tedy stejnému klientovi, jsou distribuovány vždy na stejný server. Tak je zajištěno, že veškerá uložená data v session potřebná následujícími požadavky, jsou dostupná.

3.6 HW a SW vyvažování zátěže

Hardwarové řešení:
V případě hardwarového řešení se mezi uživatele a serverový cluster vloží speciální zařízení, zvané load balancer, které distribuuje provoz aplikace mezi více serverů. Výhoda tohoto řešení spočívá v tom, že server pracuje i nadále tak jako předtím, tj. bez dalšího instalovaného softwaru, a řízení provozu přebírá specializovaný hardware, tím dojde k odlehčení zátěže jednotlivých nodů. Výsledkem je zvýšená výkonnost dané jednotky, případně celého clusteru. Hardware navíc pracuje zcela nezávisle na operačním systému, takže lze vytvořit i heterogenní serverové prostředí. Zařízení typu load balancer mohou být instalována lokálně i globálně. V případě lokálního vyrovnávání výkonu je nadměrná zátěž rozdělována mezi servery instalované na jedné lokalitě. Lokální load balancer se proto zabývá sledováním aktivity systémových zdrojů a testuje jejich dostupnost.

Výhody použití hardwarového vyvažování zátěže:

- nárůst kapacity systému,
- škálovatelnost operací pro přizpůsobení více uživatelům a novým aplikacím
- zvýšený výkon aplikací
- zvýšená bezpečnost.

Softwarové řešení:

V dnešní době, kdy se výpočetní výkon dostal na mnohem vyšší úroveň, nemusíme load balancing řešit pouze pomocí specializovaného hardwaru. Vzniklo tedy několik softwarových řešení, která mají výhodu, že je lze nainstalovat na většinu dnes dostupného serverového vybavení. Softwarové řešení poskytuje mnohem větší flexibilitu a škálovatelnost. Hlavní výhodou takového řešení je eliminace požadavku na dedikovaný hardware. Tento hardware bývá zpravidla velice nákladnou položkou. I komerční, softwarové řešení bude několikrát levnější než hardwarové. Existuje také mnoho volně dostupného softwaru pro řešení load balancingu. Mají stejné funkce jako hardwarové load balancery ale nikdy nedosáhnou takového výkonu. Podrobnější informace jsou uvedeny v následující kapitole 4.

4. Dostupná řešení

V této kapitole jsem se zaměřil na analýzu trhu a uvedl zde několik dostupných produktů.

4.1 A10 Networks

A10 Networks [12] je americká společnost se sídlem v San Jose, která byla založena v roce 2004. Nabízí řadu vysoce výkonných řešení v oblasti zabezpečení aplikací a dalších síťových řešení. Známí jsou hlavně nabídkou svých zařízení ADC (Application Delivery Controller). U těchto zařízení je podporována funkce SIP proxy. Jedná se o společnost s celosvětovou působností ve více než 80 zemích, která má přes 5600 zákazníků v oblastech finančnictví, zdravotnictví, statní správy webových aplikací a mnoho dalších. Firma nabízí produkty v několika skupinách. Jmenovitě se jedná o skupiny Physical, Virtual, Bare Metal a Cloud.

Řada produktů Physical, jak již název napovídá je již hotové řešení se zakoupením hardwaru. V této skupině produktů, které jsou označovány jako Thunder se nachází přibližně tři desítky produktů, proto uvedu jen nejnižší a nejvyšší model. Modely uprostřed spektra se liší celkovou aplikační propustností, počtem síťových rozhraní, použitým procesorem atd.

Nejnižší model se jmenuje Thunder 840. Jeho základní parametry jsou 50000 spojení za vteřinu na 7. vrstvě OSI modelu a cena se pohybuje okolo 10400$ [13]

Naopak model z opačné strany nabídky je Thunder 7440. Jeho parametry jsou 2.8 milionu spojení ne 7. vrstvě. Jeho cena se pohybuje v relaci okolo 285000$. [13]

Produkty Virtual, jsou nabízené obrazy pro hypervizory KVM,Vmware a Microsoft Hyper-V. Produkty jsou označovány jako vThunder a jsou rozděleny licencí podle využitelné šířky pásma od 200 Mbps do 100 Gbps. Řada Bare Metal je takové řešení, kdy je možnost instalace přímo na zákazníkův HW. Minimálními požadavky pro instalaci jsou 4 jádrový procesor Intel a 4GB RAM, dvě síťová rozhraní. V závislosti na počtu jader je přidělena licence na software a určuje celkovou propustnost pro použití. 10 Gbps odpovídá 4 jádrům, 20 Gbps 8 jader, 40 Gbps 14 jader a 60 Gbps odpovídá 24 jádrům. A poslední skupinou je Cloud, kde je řešení dostupné pro Microsoft Azure a Amazon Web Services.

Pro vyvažování se používají metody Round Robin, Least Connections, Weighted Round Robin, Weighted Least Connections, Least Response

4.2 Citrix NetScaler ADC

Citrix NetScaler poskytuje hardwarové i softwarové řešení pro vyrovnání zátěže. Jedná se o aplikační přepínač, který inteligentně distribuuje, optimalizuje a zabezpečuje vrstvy L4 a L7. Je schopen vyvažování zátěže protokolu SIP v režimu SIP proxy. Nabízené produkty jsou v řadách NetScaler MPX, NetScaler SDX a NetScaler VPX.

NetScaler MPX/SDX je řada dedikovaných hardwarových zařízení, která jsou vhodná ke správě webových aplikací, pro různě velké datové toky (výkon 500 Mbps - 200 Gbps). Tato řada má díky vestavěnému firewallu mimořádně vysoký výkon zabezpečení webových aplikací. Modely MPX jsou určeny spíše pro menší podniky a řady s označením SDX jsou určeny pro optimalizaci datového toku v datových centrech.

Jako příklad bych uvedl zařízení MPX/SDX 25200A. Jedná se o highend zařízení s propustností 200 Gbps. Na L7 vrstvě zvládne 5,2M HTTP dotazů. Cena zařízení je přibližně 220 000$ [14].

NetScaler VPX je virtualizované softwarové zařízení, které je vhodné pro veřejné i privátní cloudové infrastruktury. Poskytuje aplikační vyvažování zátěže, bezpečný vzdálený přístup, akceleraci zpracování požadavků a zabezpečení poskytované služby. Vhodné pro telekomunikační společnosti a malé i velké podniky.

Je nabízen v řadě VPX 10 - VPX 100G, kde 10 znamená 10Mbps a 100 znamená 100Gbps propustnost. Všechny verze se také liší tím, který virtualizační nástroj je kompatibilní a lze ho

použít. Na KVM jsou dostupné všechny.

NetScaler také rozlišuje tři druhy licencí: Standard Edition, Enterprise Edition a Platinum Edition. Tyto licence přicházejí s určitými softwarovými nástroji. Základní licence přináší spolehlivou dostupnost aplikací, komplexní vyvažování zátěže L4 –L7, optimalizaci výkonu a bezpečný vzdálený přístup. Další dvě edice přicházejí s vylepšeními jako je například řízení dopravy, podpora clusterů, optimalizace provozu aplikací v cloudu, rozšířené funkce pro akceleraci aplikací atd. Citrix NetScaler využívá k vyvažování zátěže metody Least Connection a Round Robin. Dále pak využívá vyvažovací metody Least Response Time, Least Bandwidth, Least Packets nebo hashovací metody.

4.3 F5 Networks

Společnost F5 Networks je jedním z hlavních dodavatelů v oblasti ADC (Active Delivery Controller). Nabízí jak dedikovaný hardware, tak i virtualizovaný software pro vyvažování zátěže a cloudové řešení. Tyto produkty jsou pod sekcí Local Traffic manager a jsou označeny jako BIG-IP a v nové řadě BIG-IP iSeries.

HW zařízení jsou rozdělena do několika sekcí: 2000, 4000, 5000, 7000, 10000 a 12000.

Na všech řadách běží vlastní operační systém TMOS. K produktům lze dokoupit velké množství softwarových modulů jako třeba, GTM (Global Traffic Manager), který zabezpečuje DNS před DDoS útoky, ASM (Aplication Security Manager), který zabezpečuje aplikace nebo AFM (Advanced Firewall Manager), který slouží pro zabezpečení dat a mnoho dalších.

Všechny produkty ze série BIG IP, lze provozovat v režimu SIP proxy. Pro vyvažování zátěže F5 využívá metody Round robin, Ratio, Least Connections, Weighted Least Connections, Observed, Predictive, Least Session.

Pro srovnání s konkurencí uvedu parametry výkonného modelu i11800 [15]. Cena tohoto zařízení je 144 000$.

L7 žádostí za sekundu: 5.5M; L4 spojení za sekundu: 2.1M;

L4 HTTP dotazy za sekundu: 25M; Propustnost L4 160 Gbps/ L7 80 Gbps

4.4 Mizutech

MLB (Mizutech SIP Load balancer) [16] je robustní a efektivní SIP vyvažovač zátěže. Jedná se o řešení, které je součástí Mizutech VOIP Softswitch. Jedná se o nabídku několika různých služeb. Například se zde nabízí kompletní služby pobočkové ústředny, Registrar server, Proxy server, Aplikační server, SMS brána dále také nabízí vlastní softwarové a webové SIP klienty. Lze ho také využít pro potřeby IVR (Interactive Voice Response) a callcenter.

Jedná se software pro platformu Windows. Nabídkou pokrývá celé spektrum použití, variantami Free, Basic, Standart, Advanced a Enterprise. Nejlevnější varianta Basic v ceně 3900$ zvládne obsloužit 30 hovorů současně. Varianta Standart obsloučí 300 hovorů a cena je 7900$. Verze Advanced zvládne 5000 hovorů a stojí 17000$.

Verze Enterprise není licenčně omezená ale spíše hardwarovými možnosti a cena je individuální. Konkrétní cenu uvedu na příkladu níže v textu této podkapitoly. Dále je dostupná verze Free, která sice není omezena počtem hovorů, ale neobsahuje tak velké množství funkcí a je pro nekomerční použití.

Výhody použití LB řešení Mizutech: [16]

- Distribuce VOIP provozu na stávající funkční řešení SIP server nebo softswitche
- Sledovaní stavu jednotlivých koncových bodů a vyvarování se softwarové, hardwarové nebo síťové chybě.
- Ochrana sítě. Aplikace obsahuje vestavěný firewall, dále lze nastavit limity pro síťový provoz
- Funkce SIP Proxy s podporou nastavení priority výkonosti serverů

- Funkce Failover s možností ověření a automatického vrácení do původního stavu
- Detailní konfigurace s širokými možnostmi
- Možnost spustit aplikaci jako Portable i jako službu Windows

Nyní přejdu na nejdůležitější část a tou je vyvažování zátěže. Musíme zde rozlišovat dva typy.

První je vestavěná funkce přímo ve VOIP serveru. Jedná se o nastavení routovaní SIP provozu. Pokud přídáme jako destinace další SIP servery, mezi které bychom chtěli provoz rozdělit a nastavíme jim stejnou prioritu, server začne rozdělovat rovnoměrně mezi tyto destinace. Vznikne tím Load Balancer, který vyvažuje metodou Round Robin. Tato varianta řešení je dostupná ve všech typech nabízených řešení od Free po Enterprise.

Druhým typem vyvažování zátěže je aplikace SIP Load Balancer. Tato varianta je doporučená pro aplikace s nějvětším provozem, kdy je potřeba využít větší množství serverů, které budou obsluhovat provoz SIP. Tato nabídka je dostupná pouze individuálně ve variantě Enterprise.

SIP Loadbalancer lze využít také v demoverzi na 6 měsíců, ale nesmí být použit pro komerční účely. A také má omezení na 50000 hovorů, poté je třeba software restartovat. Lze tedy s variantou Free vyzkoušet i robustnější řešení před nákupem varianty Enterprise.

Load Balancer lze využít i s jinými SIP servery než od Mizutech, například i s ústřednou Asterisk.

Orientační výkon tohoto Load Balanceru je uváden následovně:

- Sestava: Intel Xeon 8core, 8GB RAM
- 100 000 PPS – paketů za vteřinu
- 20 000 CPS – hovorů za vteřinu
- 2 000 000 CC – souběžný počet hovorů

Cena pro řešení, které by obsahovalo ten Load Balancer se pohybuje v různých relacích podle potřeb zákazníka. Z tohoto důvodu uvedu konkrétní případ konfigurace.

Požadavek 12000 jako souběžný počet hovorů, provoz na vlastních servech a trvalá licence pro SW vyžaduje následující:

- OS: Windows Server 2008/2012 Standard edition 64 bit
- CPU: 22 jader 2.6 GHz Xeon
- RAM: 31 GB

Licence pro takovouto aplikaci vychází na přibližně **35000$.** Jedná se ale pouze o licenci na použití softwaru a je potřeba mít vlastní hardware. Zde je již vhodné použít rozdělení zátěže mezi více serverů, protože procesory s kapacitou 22-24 jader se pohybují také ve vysoké cenové relaci.

4.5 Kamailio

Počátky tohoto projektu sahají do roku 2001, kdy vznikl projekt s názvem SER (SIP Express Router), který měl za úkol vytvořit výkonný proxy server a v roce 2002 byl vydán. V roce 2004 se tento projekt poprvé rozdělil a vznikl projekt OpenSER, který byl opensourcem a více spolupracoval s komunitou. V roce 2008 se projekt OpenSER musel, kvůli lcenčním podmínkám přejmenovat na Kamailio. V tento rok také vznikla další větev projektu s názvem OpenSIPS. A také v tomto roce se Kamailio spojilo s původním projektem SER [17].

V současnosti tedy existují projekty Kamailio a OpenSips, které mají stejného původce a tím je projekt SER.

Jedná se o proxy serverovou aplikaci, založenou na protokolu SIP. Jejím nejčastějším použitím je SIP-proxy, Registrar server, Location server, Redirect server a SIP aplikační server.

Kamailio je vytvořen v jazyce C pro Unixové a Linuxové architektury systémů a je optimalizován pro co nejvyšší výkon. Má modulární architekturu. Obsahuje tedy jádro, interní knihovny a rozšiřující moduly, které zastávají požadované funkce. Moduly jsou dostupné v Kamailio repozitáři a je jich více než 150. Tím, že jádro aplikace, hlavní binární soubor, je menší velikosti a všechny potřebné funkce obstarávají moduly, je skvělý pro použití i v méně výkonných strojích. Umožňuje i na slabších počítačích zvládat velké zátěže hovorů, např. na dvouprocesorovém počítači i několik tisíc hovorů za vteřinu. Je tedy velice škálovatelný a používá se na výkonných serverech i v embedded zařízeních

Cílem projektu Kamailio je spolupráce s komunitou a uživateli. A vytvořit tak volně šiřitelný a bezpečný SIP server. Jedná se o svobodný software šiřitelný pod GPL licencí.

Umožňuje i na slabších počítačích zvládat velké zátěže hovorů, např. na dvouprocesorovém počítači i několik tisíc hovorů za vteřinu.

Umožňuje dokonce podporu pro více domén a ENUM. Výborná je podpora NAT průchodů díky nathelper, rtpproxy nebo mediaproxy modulů. Umožňuje spojení s databází MySQL

Hlavním úkolem aplikace je směrování SIP signalizace. Jeho použitím lze vytvořit následující funkce [17]:

- Loadbalancing – velké množství algortimů (zmíněno v následující kapitole)
- Sériový a paralelní fork
- Podpora Least Cost směrování
- Podpora pro Fail-over
- Možnost kombinace serverů pro HA (High availability)

Mezi hlavní funkcionality patří:

- Podpora komunikaci pomocí UDP, TCP, TLS a SCTP
- Možnost IPv4 i IPv6
- WebSocket pro WebRTC
- Propracovaná autentizace uživatelů SIP, autorizace pomocí ACL
- TLS podpora pro SIP
- Autentizace a autorizace pomocí několika typů databází (MySQL, PostgreSQL, UnixODBC, BerkeleyDB, Oracle, text files), RADIUS and DIAMETER
- Podpora pro SRV (Service record) a NAPTR (Naming Autority PoinTeR) DNS záznamy
- Podpora DNSsec, ENUM a SRV DNS failover
- Interní DNS cachovací systém s podporou DNS blokace, blacklist na úrovni IP
- Rozšiřující API - Perl Programming Interface, Java SIP Servlet Application Interface, Lua Programming Interface, JavaScript Programming Interface, Managed Code (C#) Programming Interface, Python Programming Interface ,Java Programming Interface
- Interkonektivita - přímé napojení na PSTN brány, Brána pro SMS nebo XMPP (Extensible Messaging and Presence Protocol), dobrá kompatibilita se SIP zařízeními a aplikacemi jako jsou telefony či telefonní ústředny

4.6 OPENSIPS

OpenSIPS (**Open SIP S**erver) [18] je opensource implementace SIP serveru. Jeho historii jsem již zmínil v předchozí kapitole 3.5. Vznikl tedy v roce 2008 odtržením od projektu OpenSER (Kamilio) je mu tudíž velice podobný svou architekturou. Taktéž je napsán v jazyce C a zachoval si svou modulární architekturu, díky které lze server jednoduše rozšířit o požadované funkce. Obsahuje více než 120 modulů.

OpenSIPS nabízí několik možností použití a mezi nejdůležitější patří:

- SIP registrar server
- SIP router / proxy
- SIP redirect server
- SIP presence agent
- SIP B2BUA
- SIP IM server
- SIP to SMS gateway
- SIP load-balancer or dispatcher
- SIP front end for gateways/asterisk
- SIP NAT traversal unit
- SIP aplikační server

Funkcemi a architekturou systému je OpenSIPS velice podobný Kamailiu, což je zapříčiněno i společnou minulostí. Z tohoto důvodu by bylo vhodné je srovnat po stránce výkonu. Níže jsou uvedeny oficiální parametry výkonu pro počet souběžně udržitelných hovorů z oficiálních stránek daných alikací (zdroje jsou uvedeny u výsledků). Obě aplikace jsou zde nastaveny v režimu proxy mezi UAC a UAS.

U OpenSIPS byl test proveden na sestavě: CPU 4 jádra 2,6 GHz a dostupná RAM 6GB.

V této konfiguraci OpenSPIS zvládne přibližně 13000 hovorů [18].

U Kamailio byla testovací sestava s CPU-2 jádra 1.86GHz a dostupná RAM 768 MB

Kamailio v této konfiguraci zvládlo přibližně 4100 hovorů za vteřinu [17].

Výsledek je celkem nejednoznačný vzhledem k použitému hardwaru, nicméně si myslím, že za použití stejných podmínek by test dopadl podobně pro obě aplikace.

4.7 Srovnání produktů

Srovnání všech dostupných produktů jsem rozdělil na dvě části. V první části a první tabulce 4.1 porovnávám řešení od výrobců, jejichž hlavním produktem jsou hardwarová zařízení. Proto v tabulce porovnávám pouze je a řešení cloudové či v podobě virtualizace jsem neporovnával.

U těchto produktů bohužel není dostupná podrobná dokumentace z hlediska protokolu SIP. Proto jsem jako parametr porovnání výkonu uvedl výkon na sedmé vrstvě, který je uveden v oficiálních parametrech jednotlivých zařízení. Parametr je udáván pro počet příchozích připojení HTTP, které lze obsloužit za jednu sekundu.

Z hlediska metod pro vyvažování zátěže bych žádný velký rozdíl neshledal. Podle cenových kategorií, v kterých se zde pohybujeme, je zřejmě, že se jedná o produkty určené pro velké firemní zákazníky, kteří se na tuto oblast specializují.

Výrobce	**A10 Networks**	**Citrix Netscaler**	**F5 Networks**
Typ řešení	dedikovaný HW, Virtualizace, Cloud		
model HW zařízení	Thunder 7440	MPX/SDX 25200A	BIG IP i11800
Cena ($)	285000	220000	144000
Výkon L7	2.8M	5.2M	5.5M
LB metody	Round Robin, Least Connections, Weighted Round Robin, Weighted Least Connections, Fastest Response	Least Connection, Round Robin, Fastest Response, Least Bandwidth, Least Packets Hashovací metody	Round robin, Ratio, Least Connections, Weighted Least Connections, Observed, Predictive, Least Session
OS	vlastní	vlastní	vlastní

Tab 4.1 Srovnání HW produktů

Ve druhé části jsem se zaměřil na řešení softwarová. Zde jsem vybral komerční produkty ale i zástupce volně dostupných řešení.

	Mizutech	**Kamailio**	**OpenSIPS**
Dostupnost	komerční SW	open source	open source
Cena	3000 - 50000$	freeware	freeware
Typ řešení	SW	SW	SW
Výkon (Call/s)	20000	4100	13000
OS	Windows	Unix/Linux	Unix/Linux

Tab 4.2 Srovnání SW produktů

Hledisko porovnání výkonu je bohužel zkreslené vzhledem k použití rozdílného hardwaru u oficiálních testů. Při porovnání obou kategorií jsem došel k závěru, že softwarové řešení se pohybuje z hlediska výkonu daleko za řešením hardwarovým.

Cílem práce bylo po uvedení přehledu zvolit řešení na bázi otevřené kódu. Po tomto kritériu skončila volba na softwaru Kamailio a OpenSIPS. Oba projekty jsou volně dostupné a s otevřeným zdrojovým kódem. Což je asi nejdůležitější pro následnou instalaci a testování.

Obě řešení jsou velice dobře dokumentovaná a spolupracují s komunitou. Vychází ze stejného projektu, jsou si tudíž celkem podobná. Nakonec jsem zvolil projekt Kamailio, který mě oslovil více a v následující kapitole jsem se jím zabýval podrobněji.

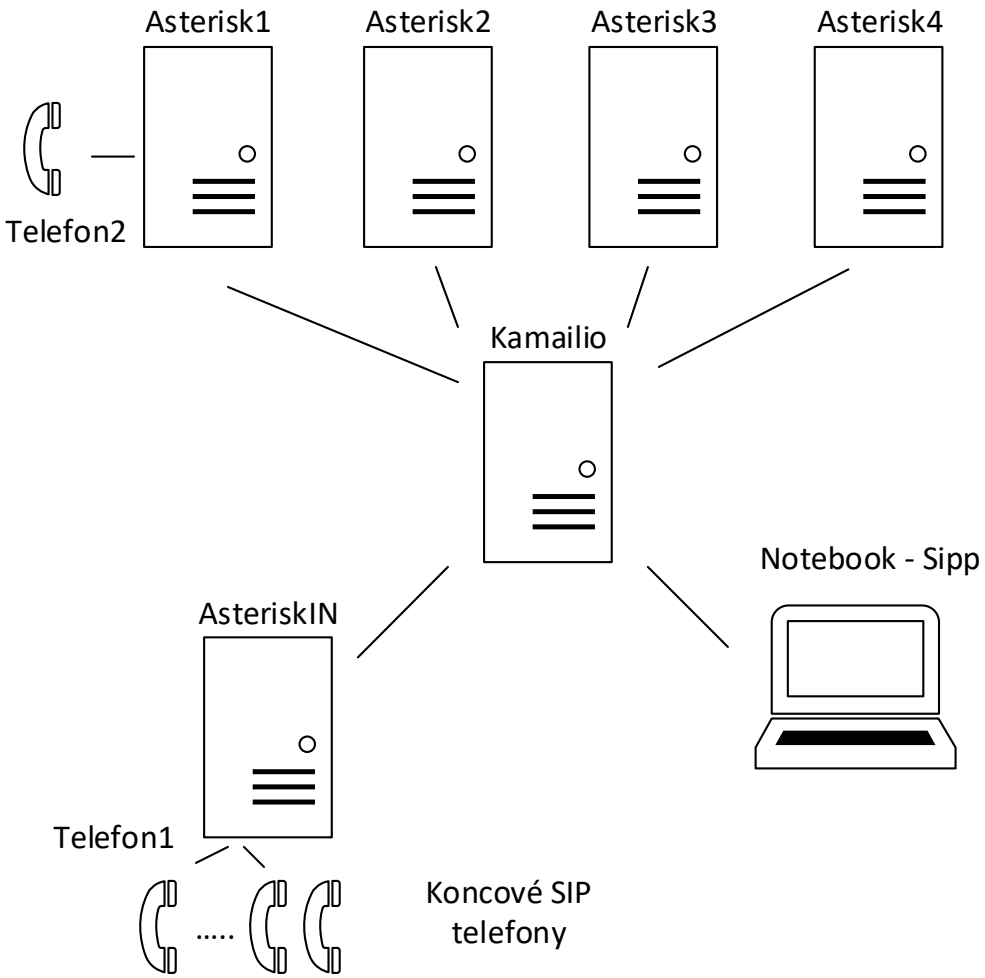
5. Instalace a konfigurace vyvažovače zátěže

5.1 Popis řešení

Dle zadání jsem jako nejvhodnější řešení pro Vyvažování zátěže zvolil projekt Kamailio. Jedná se o bezplatný software s otevřeným zdrojovým kódem. Navíc jako další pozitivní aspekt mé volby vidím modularitu tohoto softwaru a možnost přes API naprogramovat si vlastní modul, a to ve velkém množství programovacích jazyků. Také se jedná o aplikaci, které je velmi dobře zdokumentována.

SIP server Kamalio, zde bude fungovat jako vyvažovač zátěže. Abych tedy mohl vyvažovat protokol SIP, bylo třeba mít nějaké koncové body, mezi které bych zátěž rozložil. Pro tento účel jsem zvolil ústřednu Asterisk více v kapitole 5.3. Přesněji jsem zvolil čtyři koncové ústředny. Pokud se na problém podíváme z opačné strany, bylo zapotřebí i nějakého zdroje, který by protokol SIP generoval. Pro tento účel jsem zvolil aplikaci Sipp [19], která dokáže generovat SIP zprávy a chovat se jako UAC. Jako další ověření jsem na Kamailio nasměroval další ústřednu Asterisk, která disponovala několika SIP telefony a několika softwarovými klienty. Bylo tedy zapotřebí nainstalovat 5 strojů s ústřednou Asterisk a jeden se SIP proxy Kamailio. Jako nejvhodnější jsem zvolil cestu virtulizace pomocí KVM [20]. Pro všechny stroje, včetně hostitelského, jsem zvolil systém Debian [21], protože s ním mám nejvíce zkušeností.

Všechen použitý software, tedy Kamailio, Debian, KVM, Asterisk a Sipp je volně dostupným softwarem, tudíž není žádný problém s jeho použitím. Na obrázku 5.1 je uvedeno logické schéma, kterého jsem se držel při instalaci a konfiguraci softwaru.



Obr. 5.1 – Schéma pro konfiguraci aplikace

Parametry použitých fyzických a virtualizovaných zařízení:

Server pro virtualizaci:

- OS - Debian GNU/Linux 9.4 (stretch)
- CPU - Intel Xeon X3430 2.40GHz – počet jader 4
- RAM - 12GB
- HDD – 1000GB

Virtualizované stroje (Asterisk1, Asterisk2, Asterisk3, Asterisk4, Kamailio, AsteriskIN):

- OS - Debian GNU/Linux 9.4 (stretch)
- CPU - přiděleno 1 jádro z hostitelského PC – 2.4 GHz
- RAM - 1GB
- HDD – 60 GB

Notebook pro použití Sipp (Notebook - sipp):

- OS - Ubuntu 17.10 KDE
- CPU - i5-8250U 1,6/3,6 Ghz
- RAM - 8GB
- SSD – 240 GB

SIP klienti:

- Zoiper [22] pro OS Android (Telefon1, Telefon2)
- SIP telefony Snom 320, Gigaset C530 IP

5.2 Kamailio

Uvádím zde pouze instalaci aplikace Kamilio. Ostatní potřebné věci jako je instalace a konfigurace serveru pro virtualizaci, instalace operačních systémů, kompilace a instalace ústředny Asterisk [23], konfigurace softwarových klientů a telefonů nebo konfigurace ústředny Asterisk uvedeny nejsou. Všechny konfigurační soubory jsou uvedeny v příloze práce.

Jen tedy zmíním, že na stroji pro virtualizaci jsem síťovou kartu nastavil do režimu bridge, tak aby všichni klienti byli dostupní ze sítě LAN 192.168.33.0/24. Jednotlivé adresy budou uvedeny v průběhu použití.

5.2.2 Instalace Kamailio

Pro instalaci, jak jsem již uvedl, jsem zvolil systém Debian ve verzi 9.4.

Kamailio je dostupné z oficiálního repozitáře v podobě balíčku. Já jsem pro instalaci zvolil nejnovější verzi softwaru 5.1.3 a zkompiloval jsem ho ze zdrojového kódu.

Při instalaci počítám s instalací s právy uživatele root.

Prvním a nezbytným krokem je rozšířit operační systém o potřebné nástroje, knihovny a zdrojové soubory pro funkci aplikace.

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

Dále je třeba vytvořit složku pro zdrojové soubory a soubory zkopírovat z repozitáře git.

```
mkdir -p /usr/local/src/kamailio-5.1

cd /usr/local/src/kamailio-5.1

git clone --depth 1 --no-single-branch https://github.com/kamailio/kamailio kamailio
```

```
cd kamailio/
git checkout -b 5.1 origin/5.1
```

V dalším kroku je třeba vytvořit a upravit konfigurační soubory pro kompilaci.

```
make cfg
vim src/modules.lst
```

V tomto souboru se upraví řádek s proměnou *include_modules* o hodnotu *db_msql*. Tímto jsme upravili, které moduly se také zkompilují. Do proměnné je možné přidat další moduly.

```
include_modules= db_mysql
```

Provedeme kompilaci a instalaci.

```
make all
make install
```

Pokud nemáme, přidáme složku */usr/local/sbin* do systémové proměnné.

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

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

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

V souboru najít a upravit 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
```

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

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

Na začátek souboru přidáme řádky, pokud jsme změnili heslo pro uživatele MySQL je třeba změnit proměnou *dr_url*:

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

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

```
cp /usr/local/src/kamailio-devel/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
```

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-5.1/kamailio/pkg/kamailio/deb/debian/kamalio.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 souboru 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 připravená ke spuštění.

Start, stop a restart aplikace lze provézt příkazy:

```
/etc/init.d/kamailio start (restart,stop)
```

5.2.2 Konfigurace Kamailio

Po nainstalování aplikace se ve složce /usr/local/sbin budou nacházet soubory a skripty:

- **kamailio** - Kamailio SIP server, hlavní binární soubor.
- **kamctl** – Tento nástroj se používá pro správu kamailia z příkazové řádky. Lze přidávat nebo upravovat SIP účastníky nebo prohlížet registrace a interní statistiky například. Má vlastní konfigurační soubor kamctlrc, ve stejné složce jako je kamalio.cfg
- **kamdbctl** – Nástroj, který se používá pro správu a vytvářen databáze. Používá konfigurační soubor kamctlrcscript.
- **kamcmd** - CLI – Aplikace která posílá RPC (Remote Procedure Call) příkazy Kamailiu z příkazové řádky. Vyžaduje načtený modul ctl.

Kamailio.cfg

Hlavní konfiguračním souborem je kamailio.cfg [17] a nachází se ve složce /usr/local/etc/kamailio. Kamailio obsahuje všechnu hlavní konfiguraci v tomto souboru. Kompletní konfigurační soubor je uveden v příloze práce.

Tento soubor lze rozdělit na tři, respektive čtyři části.

První částí je **Global Parameters.** Zde se upravují parametry jádra aplikace. Jako například níže.

```
log_facility=LOG_LOCAL0
disable_tcp=yes
alias="sip.mydomain.com"
port=5060
```

Druhou a třetí sekcí je nastavení modulů. Nejdříve je část **Module section.** V této části se definuje, jaké moduly se mají použít při spuštění a z jaké složky.

```
mpath="/usr/lib/kamailio/modules"
loadmodule "dispatcher.so"
```

Další část je **Module specific parameters.** Zde se nastaví všechny parametry pro vybrané moduly z předchozí části.

```
modparam("dispatcher", "list_file", "/var/run/kamailio/dispatcher.list")
```

Modul se upravuje příkazem *modparam*. Kde prvním argumentem je název modulu, druhým argumentem je jméno nastavovaného parametru a tím třetím argumentem je přímo hodnota parametru. Například zde se upravuje modul *dispatcher* a jeho *parametr list_file*. Nastaví se tím cesta k souboru, z kterého bude modul dispatcher načítat jednotlivé destinace.

A poslední je nejdůležitější část, která se nazývá **Routing Logic**. Je to hlavní část, která obstarává veškerou požadovanou funkčnost aplikace.

Tato sekce funguje jako program a postupně prochází jednotlivé příkazy. Syntaxe této části je velice podobná jazyku C. Tato část je ohraničena sekcí request_route.

Zde je příklad části kódu a syntaxe:

```
request_route {
                route(REQINIT); # Request Initial
                route(NATDETECT); # NAT Detection
                if (is_method("CANCEL")) {
                  exit;
                }}
```

5.2.3 Moduly Kamailia

Jak jsem již několikrát zmínil, funkcionalita serveru Kamailio je založená na modulech. Veškeré moduly, které jsou schopné pracovat s jádrem Kamailia jsou uložené ve složce */usr/local/lib64/kamailio/modules*.

Pokud chceme aplikaci o nějaký modul rozšířit, je třeba modul nakopírovat do této složky a načíst ho v konfiguračním souboru. Jak jsem uvedl v kapitole 4.5 modulů je více než 150, proto zde uvádím jen několik z nich na ukázku.

async.so - Asynchronní zpracování SIP požadavků

registrar.so - Modul, který implementuje funkci pro SIP registrace

statistics.so - Podpora skriptů pro statistiky

sms.so - SIP-to-SMS IM Gateway modul

tls.so - Modul pro použití TLS

qos.so - QOS kontrolní API

rtpproxy.so – RTPProxy modul pro kontrolu RTP provozu

siputils.so - Nástroje pro SIP protokol

Modul Dispatcher

Pro tuto práci se jedná o nejdůležitější modul, který obstará požadovanou funkcionalitu.

Tento modul vybírá cílovou destinaci pro příchozí provoz. Nabízí funkci vyvažování zátěže a lze ho použít pro směrování provozu protokolu SIP. Používá několik standartních algoritmů pro vyvažování, konkrétní uvedu níže. Při konfiguraci souboru kamilio.cfg jsem zmínil, že jednotlivé moduly mají parametry a lze jejich funkčnost ovlivnit při načítání modulu. Dále moduly obsahují vlastní funkce pomocí, kterých se moduly používají v sekci Routing Logic.

Například funkce **ds_select_dst(set, alg[, limit])** [24].

Parametr set vybírá skupinu, z které se vybírají cílové adresy.

Metody pro vyvažování, parametr alg:

0 - Hash na základě callid
1 - Hash na základě from URI.
2 - Hash na základě to URI.
3 - Hash na základě request-URI.
4 – Round Robin
5 – Hash na základě autorizačního jména, pokud není, použije se Round Robin.
6 – Random destination – náhodný cíl
7 – Hash na základě obsahu, je třeba nastavit paramter hash_pvar.
8 – Výběr na základě priority.
9 – Vážená metoda výběru, je třeba u adresy destinace nastavit parametr weight.
10 – Vybírá destinaci s nejmenším počtem aktivních spojení
11 – Relativní vážená metoda, Je třeba nastavit parametr rweight. Hodnota se pak počítá v poměru k celkovému počtu aktivních destinací.
12 – Paralelní fork. Doručí požadavek na všechny detinace.
X – Algoritmus není implementován a použije se první dostupná destinace.

Dispatcher list:

Jedná se o soubor pomocí, kterého lze načítat cílové destinace do databáze, případně je lze přidat ručně. Jednotlivým adresám lze nastavit různé parametry, také je lze přidávat do skupin pro použití u složitějších schémat.

Do souboru jsem vložil následující adresy pro virtuální stroje Asterisk1, Asterisk2, Asterisk3 a Asterisk4:

1 sip:192.168.33.21:5060
1 sip:192.168.33.22:5060
1 sip:192.168.33.23:5060
1 sip:192.168.33.24:5060

5.3 Asterisk

Asterisk je kompletní open source softwarové řešení pobočkové telefonní ústředny, které vytvořila firma Digium. Pracuje na platformách Unix a Linux. Je šířena pod svobodnou licencí GNU GPL. Podporuje celou řadu protokolů a množství fyzických zařízení, se kterými je schopen spolupracovat. Jedná se o digitální i analogová zařízení. Poskytuje také velké množství služeb jako, výchozí brána pro SIP, IAX2, H.323, IVR služby, fronta volajících atd.

Pro mou aplikaci mi Asterisk poslouží jako koncový bod pro vyvažování zátěže. Asterisk jsem nakonfiguroval tak, že všechny příchozí hovory se vyzvednou a přehraje se přednastavená hlasová zpráva. U ústředny Asterisk1 jsem připojil softwarového klienta Zoiper.

Konfigurační soubory jsou uvedeny v příloze A.

5.4 SIPp, TCPdump

SIPp je open source generátor provozu protokolu SIP a využívá se hlavně k testovaní prostředí, kde se tento protokol používá. Je schopen pracovat jako UAC i UAS a generovat SIP signalizaci a přenášet mediální obsah (RTP).
Užitečnou funkcí je možnost importovat přednastavený scénář SIP komunikace ze souboru XML. Je tedy možné si tento soubor vytvořit přesně podle potřeby testování.
Konkrétně se dá použít pro testování skutečných SIP zařízení, jako jsou SIP proxy, B2BUA, SIP media servery, SIP brány, ústředny PBX a mnoho dalších. Tento nástroj je velice nápomocný, protože dokáže tisíce uživatelů. Je dostupný pro platformu Linux a Unix.

Použití uvedu přímo na příkazu, který jsem pro testování *použil.*

./sipp -sn uac -d 10000 -s 2 192.168.33.25 -l 100

Parametry:

-sn: použití přednastaveného scénáře s hodnotou uac

-d: délka hovoru v ms

-s: username pro request URI, v tomto případě klapka 2

-l: počet souběžně realizovaných hovorů

TCPdump je paketový analyzér pracující v příkazové řádce. Jedná se volně dostupný software určen hlavně pro TCP/IP. Ve své práci jsem ho využil pro sledování provozu protokolu SIP.

5.5 Nastavení a ověření vyvažování zátěže

Tímto jsem měl připraven veškerý software a hardware pro vyvažování zátěže i pro jeho otestování. Jako další bod jsem musel nastavit pravidlo pro směrování příchozích hovorů.

Toto je část z hlavního konfiguračního souboru ze sekce Routing Logic:

```
#Deklarace funkce
route[DISPATCH] {
        # Round robin metoda pro skupinu serveru 1
if(!ds_select_dst("1", "4"))
{
        send_reply("404", "No destination");
        exit;
}
sl_send_reply("100","Trying");
route(RELAY);
exit;
}
```

V ukázce je uvedena funkce, kterou zavoláme příkazem route(DISPATCH) z těla programu. Nejdříve se zavolá funkce *ds_select_dst*, která je podmíněna nalezením cílové destinace, kde následuje odeslání informační zprávy 404 No destination a opuštění funkce. Pokud se cílová destinace nalezne, odešle se informační zpráva 100 Trying a adresa se uloží do dst_uri pole a je předána funkci route(RELAY), která se postará o doručení na cílovou adresu. Použil jsem metodu Round Robin. Tímto je vše připraveno pro otestování.

Nyní tedy přišlo na řadu otestování tohoto nastavení. Spojení jsem ověřil na zkušebním hovoru mezi ústřednou AsteriskIN a Asterisk1. Spustil jsem si program TCPdump a to na Asterisk1, AsteriskIN a Kamailio. Na Asterisk1 je nastaven příchozí hovor pro vytočení klienta Telefon2 s aplikací Zoiper

Adresy uvedené v ukázce:
AsteriskIN – 192.168.33.2
Telefon1 – 192.168.33.56 – telefon registrován na ústředně AsteriskIN
Asterisk1 -192.168.33.21
Telefon2 192.168.33.53 - telefon registrován na ústředně Asterisk1
Kamalio – 192.168.33.25

Zde je uveden výpis komunikace zachycený programem tcpdump:

```
IP 192.168.33.2.sip > 192.168.33.25.sip: SIP: INVITE sip:192.168.33.25 SIP/2.0
```

Telefon1 pošle přes ústřednu AsteriskIN zprávu INVITE na Kamailio.

```
IP 192.168.33.25.sip > 192.168.33.2.sip: SIP: SIP/2.0 100 Trying
IP 192.168.33.25.sip > 192.168.33.21.sip: SIP: INVITE sip:192.168.33.25 SIP/2.0
```

Kamailio odpoví informační zprávou 100 a přepošle INVITE na Asterisk1.

```
IP 192.168.33.21.sip > 192.168.33.25.sip: SIP: SIP/2.0 100 Trying
IP 192.168.33.25.sip > 192.168.33.2.sip: SIP: SIP/2.0 100 Trying
IP 192.168.33.21.sip > 192.168.33.25.sip: SIP: SIP/2.0 200 OK
IP 192.168.33.25.sip > 192.168.33.2.sip: SIP: SIP/2.0 200 OK
```

Asterisk1 pošle informační zprávu 100 a poté odpoví OK.

```
IP 192.168.33.21.sip > 192.168.33.53.39779: SIP: INVITE
```

Asterisk1 lokalizuje účastníka podle konfigurace a přepošle INVITE na Telefon2.

IP 192.168.33.53.39779 > 192.168.33.21.sip: SIP: SIP/2.0 100 Trying
IP 192.168.33.53.39779 > 192.168.33.21.sip: SIP: SIP/2.0 180 Ringing
Začne vyzvánění na Telefon2.

IP 192.168.33.2.sip > 192.168.33.25.sip: SIP: SIP/2.0 200 OK
IP 192.168.33.25.sip > 192.168.33.21.sip: SIP: SIP/2.0 200 OK
IP 192.168.33.21.sip > 192.168.33.25.sip: SIP: ACK sip:9@192.168.33.2:50609 SIP/2.0
IP 192.168.33.25.sip > 192.168.33.2.sip: SIP: ACK sip:9@192.168.33.2:50609 SIP/2.0
Na telefonu 1 dojde k přijmutí hovoru, telefon 1 pošle zprávu 200 OK, která přes Kamailio. projde na Asterisk1. Telefon 2 potvrdí zprávou ACK, a dojde k zahájení RTP streamu.

IP 192.168.33.2.sip > 192.168.33.25.sip: SIP: BYE sip:s@192.168.33.21:5060 SIP/2.0
IP 192.168.33.25.sip > 192.168.33.21.sip: SIP: BYE sip:s@192.168.33.21:5060 SIP/2.0
Na telefonu 1 dojde k ukončení hovoru a pošle se metoda BYE, ta projde přes Kamailio na Asterisk 1.
IP 192.168.33.21.sip > 192.168.33.25.sip: SIP: SIP/2.0 200 OK
IP 192.168.33.25.sip > 192.168.33.2.sip: SIP: SIP/2.0 200 OK
Telefon2 akceptuje a posílá zprávu 200 OK, a tím dojde k ukončení relace.

Z vypsané komunikace je vidět, že hovor byl úspěšně realizován. Jelikož jsem analyzoval spojení mezi dvěma telefony, mohl jsem funkčnost ověřit i fyzicky.

Jako poslední bylo třeba ověřit i samotné vyvažování zátěže. Pro tento účel není testování za pomoci telefonu příliš vhodné. Zvolil jsem tedy generátor SIPp, s parametry uvedenými v kapitole 5.4. Sipp začal generovat velké množství SIP relací. Na serveru Kamailio jsem opět spustil tcpdump a sledoval provoz protokolu SIP. Konkrétněji jsem sledoval příchozí zprávy obsahující metodu INVITE a sledoval algoritmus jejich rozdělování.

Zde je uveden výpis komunikace zachycený programem tcpdump:

IP 192.168.33.54.sip > 192.168.33.25.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.25.sip > 192.168.33.21.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.54.sip > 192.168.33.25.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.25.sip > 192.168.33.24.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.54.sip > 192.168.33.25.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.25.sip > 192.168.33.23.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.54.sip > 192.168.33.25.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.25.sip > 192.168.33.22.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.54.sip > 192.168.33.25.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.25.sip > 192.168.33.21.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.54.sip > 192.168.33.25.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.25.sip > 192.168.33.24.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.54.sip > 192.168.33.25.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.25.sip > 192.168.33.23.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.54.sip > 192.168.33.25.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.25.sip > 192.168.33.22.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.54.sip > 192.168.33.25.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0
IP 192.168.33.25.sip > 192.168.33.21.sip: SIP: INVITE sip:2@192.168.33.25:5060 SIP/2.0

Zhodnocení:

V první části testování jsem ověřil schopnost sestavit SIP relaci. Jako důkaz ověření jsem uvedl podrobnou analýzu paketového provozu. Funkčnost jsem ověřil i pomocí skutečných telefonů. V druhé části testování jsem se snažil ověřit metodu vyvažování zátěže Round Robin. Z druhé ukázky je patrné, že tento algoritmus je funkční. Pokud si zachycenou komunikaci prohlédneme podrobněji je patrné že Kamailio periodicky přeposílá provoz na adresy definované pro modul dispatcher.

Pokud si první ukázku síťového provozu prohlédneme podrobněji, zjistíme, že Kamailio zde opravdu funguje v módu proxy. Veškerá komunikace v rámci relace prochází pře proxy Kamailio.

6 Závěr

Obsahem této práce je analýza dostupných řešení, která by byla schopna nabídnout funkci vyvažování zátěže protokolu SIP. Cílem bylo najít takové řešení, které by bylo možné otestovat v laboratorní síti. Výběr jsem provedl napříč celým spektrem a v počátku jsem představil řešení založená primárně na hardwarovém principu. Tato zařízení se ale pohybují svou cenou v hladině stovek tisíc dolarů a cílí na největší firemní zákazníky.

Dále jsem se přesunul do oblasti softwarových řešení. Zde jsem vybral jak zástupce pro operační systémy Linux a UNIX, tak zástupce řešení pro systém Windows. Řešení pro systém Windows ale také nebylo volně dostupné. Nabízí sice variantu Free a dočasné použití vyvažování zátěže ale toto řešení by nebylo vhodné z hlediska instalace a dostupnosti většího množství licencí pro systém Windows. Přesunul jsem se tedy do kategorie softwaru s otevřeným zdrojovým kódem. Zde jsem vybral dva velice podobné projekty a nakonec jsem se rozhodl pro SIP server Kamailio. Poté bylo třeba tuto aplikaci otestovat. Pro pochopení a návrh takového testu bylo nejdříve nutné provést teoretický rozbor. V první teoretické části jsem se věnoval samotnému protokolu SIP a v části druhé jsem podrobně rozebral termín load balancing.

Po teoretickém rozboru jsem se mohl pustit do návrhu laboratorní virtualizované VOIP sítě, která byla schopna tento typ aplikace simulovat. SIP server Kamailio , zde fungoval v režimu SIP proxy a byl nakonfigurován dle dokumentace pro vyvažování zátěže. Tuto konfigaraci jsem také musel ověřit. Zvolil jsem vhodné nástroje, kterými jsem sledoval síťový provoz a dle dostupných dat uvedených v kapitole 5.5 jsem funkci vyvažování zátěže úspěšně ověřil.

Považuji za důležité uvést, že aplikace Kamailio díky své architektuře a způsobu konfigurace je velmi bohatým nástrojem. Jelikož struktura konfigurace je podobná struktuře programu, lze vytvořit veliké množství funkcionalit, které by přesně vyhovovaly požadavkům uživatele.

Pokud zhodnotím výsledek tak vyvažovač zátěže byl funkční a šlo by ho implementovat do skutečného provozu. Jako možné rozšíření vyvažovače zátěže SIP by bylo možné použít modul rtpproxy a získat tím kontrolu nejen nad protokolem SIP ale i nad přenosem protokolu RTP, tedy přenosu multimediálních dat.

Literatura

[1] „SIP: Session Initiation Protocol RFC 2543,“ 1999. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2543.txt.

[2] „J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson SIP: Session Initiation Protocol RFC 3261,“ 2002. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3261.txt.

[3] „H.323 versus SIP: A Comparison,“ 2018. [Online]. Available: https://www.packetizer.com/ipmc/h323_vs_sip/.

[4] A. B. a. D. S. Johnston, „SIP : Understanding the Session Initiation Protocol, Third Edition, Artech House, 2009.“.

[5] „V. Jacobson, C. Perkins SDP: Session Description Protocol RFC 4566,“ 2006. [Online]. Available: https://tools.ietf.org/html/rfc4566.

[6] „H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson RTP: A Transport Protocol for Real-Time Applications RFC 3550,“ 2003. [Online].

[7] „ ISO/IEC,“ 1994. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:7498:-1:ed-1:v2:en.

[8] „Model OSI,“ [Online]. Available: https://commons.wikimedia.org/wiki/File:Osi-model-jb.svg.

[9] C. Kopparapu, Load Balancing Servers, Firewalls, and Caches, 2002.

[10] „Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6 RFC 3768,“ 2010. [Online]. Available: https://tools.ietf.org/html/rfc5798.

[11] J. Baier, „Seriál Velká řešení otevřeně,“ [Online]. Available: https://www.root.cz/serialy/velka-reseni-otevrene/.

[12] „A10 Networks,“ 2018. [Online]. Available: https://www.a10networks.com/products.

[13] „A10 Networks Datasheet,“ 2018. [Online]. Available: https://www.a10networks.com/sites/default/files/A10-DS-15100-EN.pdf.

[14] „Citrix Netscaler Products,“ [Online]. Available: https://www.citrix.cz/products/netscaler-adc/.

[15] „F5 Products,“ 2018. [Online]. Available: https://f5.com/products.

[16] „Mizutech Professional VOIP Solutions,“ 2018. [Online]. Available: https://www.mizu-voip.com/Software/SIPLoadBalancer.aspx.

[17] „Kamailio,“ 2018. [Online]. Available: http://www.kamailio.org/w/documentation/.

[18] „OpenSIPS,“ 2018. [Online]. Available: https://www.opensips.org/Documentation/Manuals.

[19] „The SIPp testing tool,“ [Online]. Available: http://sipp-wip.readthedocs.io/en/latest/index.html.

[20] „KVM - Kernel Virtual Machine,“ 2018. [Online]. Available: https://www.linux-kvm.org/page/Main_Page.

[21] „Debian GNU,“ 2018. [Online]. Available: https://www.debian.org/.

[22] „Zoiper - Voip-softphone,“ 2018. [Online]. Available: https://www.zoiper.com/.

[23] „Installing Asterisk From Source,“ 2018. [Online]. Available: https://wiki.asterisk.org/wiki/display/AST/Installing+Asterisk+From+Source.

[24] „DISPATCHER Module,“ 2018. [Online]. Available: https://kamailio.org/docs/modules/4.3.x/modules/dispatcher.html.

Přílohy
Příloha A CD

Součástí přiloženého CD jsou následující položky:

- Text této práce *Bakalarska-prace-2018-Pont-Martin.pdf*
- Konfigurační soubory ústředen Asterisk ve složce *config/nazev_ustredny/*
- Konfigurační soubory SIP Proxy Kamailio ve složce */config/Kamailio/*.