Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Ilkovičova 3, 842 16 Bratislava 4

Manažment VoIP relácií

Projektová dokumentácia

Študijný program: Počítačové a komunikačné systémy a siete

Tím č.3: Bc. Jozef Baláž, Bc. Tomáš Boros, Bc. Adam Močkoř, Bc. Martin Pivarník, Bc. Matej Rybár, Bc. Timotej Tkáč

Vedúci tímového projektu: Ing. Ján Murányi

Ak. rok: 2013/14

Obsah

Manažment VoIP relácií 1
Obsah 1
Zoznam obrázkov 3
Zoznam grafov 3
1 Úvod 4
2 Zadanie 5
3 Analýza 6
3.1 Manažment 6
3.2 Kontext 7
3.3 Existujúce riešenia 7
3.3.1 SIP Single Port 7
3.3.2 Single a Dual Proxy režim 7
3.3.3 Funkcionalita SIRUP - u 9
3.3.4 Prepínanie medzi cestami 9
3.3.5 SIRUP Master 10
3.4 Manažment Server 10
3.4.1 Implementačné nástroje 11
3.5 Manažment API 13
3.5.1 API 13
3.5.2 JSON 13
3.5.3 Implementácia API 13
3.6 Manažment užívateľského rozhrania 14
3.7 Simulácia siete 18
3.7.1 Network Emulation (NetEm) 18
3.7.2 Informácie o kvalite linky 19
4 Špecifikácia 21
4.1 Funkcie systému 21
4.2 Údaje 21
4.3 Manažment API správy 21
4.4 Správanie systému 22
4.4.1 Manažment 22
4.4.2 SIRUP komunikácia 22
5 Návrh 24

5.1 Architektúra ... 24
5.2 Manažment API ... 24
5.3 Databáza ... 24
5.4 Údaje ... 25
5.5 Simulácia siete NetEm ... 25
5.6 SIRUP ... 27
5.7 Klient ... 27
5.8 Server ... 28
5.9 SIRUP Master ... 28
6 Prototyp ... 30
6.1 SIRUP ... 30
6.2 SIRUP Master ... 30
6.3 Simulácia siete NetEm ... 30
6.4 Manažment API ... 31
7 Implementácia ... 34
7.1 Manažment ... 34
7.1.1 Revízia manažment API správ ... 34
7.2 Zobrazovacia / rozhodovacia jednotka ... 35
7.3 SIRUP server a klient ... 36
7.4 SIRUP Master ... 37
7.5 Simulácia siete NetEm ... 38
8 Testovanie ... 41
8.1 1. scenár testovania ... 42
8.2 2. scenár testovania ... 43
3. scenár testovania ... 45
4. scenár testovania ... 46
8.3 5. scenár testovania ... 47
8.4 Testovanie manažmentu API ... 47
9 Záznam o používaní systému ... 48
10 Čo sme sa naučili ... 49
11 Zhodnotenie ... 50
12 Literatúra ... 51
13 Prílohy ... 52

Zoznam obrázkov

Obrázok 3-1 Single proxy režim 8
Obrázok 3-2 Dual proxy režim 8
Obrázok 3-3 SIRUP značkovanie 8
Obrázok 3-4 Porovnanie knižníc na zobrazenie grafov[14] 16
Obrázok 3-5 Graf HTML JS Gauge Widgets[13] 16
Obrázok 3-6 Graf pre zobrazovanie aktuálnych štatistických informácii[13] 17
Obrázok 3-7 Graf pre zobrazovanie historických štatistických informácii[13] 17
Obrázok 3-8 Monitorovací panel[13] 18
Obrázok 3-9 Nástroj TCP Ping 19
Obrázok 4-1 Tok správ medzi uzlami systému 21
Obrázok 5-1 Architektúra systému 24
Obrázok 5-2 Štruktúra radov, tried a filtrov 26
Obrázok 5-3 Preregistrácia klienta počas aktívneho hovoru 28
Obrázok 6-1 Grafické používateľské rozhranie nástroja NetEm 31
Obrázok 6-2 Zobrazovacia/rozhodovacia jednotka 33
Obrázok 7-1 Schéma API správ 34
Obrázok 7-2 Rozhranie zobrazovacej / rozhodovacej jednotky 36
Obrázok 7-3 Diagram súčastí SIRUP Master 37
Obrázok 7-4 Grafické používateľské rozhranie nástroja NetEm 39
Obrázok 8-1 Schéma testovania 41
Obrázok 8-2 Testovacia schéma s reálnymi telefónmi 45
Obrázok 8-3 Testovacia schéma s reálnymi telefónmi a SIPp 46

Zoznam grafov

Graf 7-1 Testovanie zmeny liniek pri prebiehajúcom hovore 43
Graf 7-2 Testovanie zmeny liniek pri prebiehajúcom hovore 44

1 Úvod

Predkladaný dokument obsahuje dokumentáciu k projektu Manažment VoIP relácií. Tento projekt sa rieši v rámci predmetu Tímový projekt.

V úvode dokumentu je uvedená motivácia pre riešenie manažmentu VoIP, podobné existujúce riešenia a zhrnutie výsledkov bakalárskej práce, z ktorej sa pri riešení vychádza.

Dokument ďalej obsahuje podrobnú analýzu jednotlivých funkčných prvkov a spôsobu komunikácie medzi nimi. V tejto časti je tiež opísaný spôsob využitia SIRUP - u pri riešení problému a analýza možností testovania výsledného systému v sieti.

Ďalšou časťou dokumentu je špecifikácia riešenia. Táto časť opisuje požiadavky na výsledný systém.

Opisuje jeho celkovú funkcionalitu , úlohu jednotlivých prvkov systému a tok dát medzi nimi.

Časť návrhu riešenia podrobnejšie špecifikuje funkcionalitu prvkov systému. Konkrétne sú opísané SIRUP zariadenia, ich rozmiestnenie, spôsob komunikácie a správy, ktoré sa pri komunikácií posielajú. V návrhu riešenia je tiež opísaný nástroj, ktorý umožní testovanie systému.

Ďalším prvkom projektovej dokumentácie je prototyp a implementácia. V tejto časti opisujeme čiastočne implementovanú funkcionalitu jednotlivých prvkov systému, ktorá bude neskôr v rámci kapitoly Implementácia doplnená o kompletnú funkcionalitu.

Jedna z najdôležitejších častí celého projektu je testovanie. V rámci tejto kapitoly preveríme systém rôznymi testovacími scenármi, aby sme dosiahli čo najlepšie testovacie výsledky a dostatočne overili funkcionalitu systému.

Kapitolami Záznam o používaní systému, Čo sme sa naučili a Zhodnotením ukončíme prácu na projekte a popíšeme si, aký prínos mal pre nás celý projekt, a čo sme ním dosiahli.

Na konci celého dokumentu nájdeme projektovú dokumentáciu riadenia.

2 Zadanie

Multimediálne relácie založené na protokole SIP (Session Initiation Protocol) sa väčšinou skladajú z troch rôznych komunikačných kanálov (SIP, RTP a RTCP), pričom každý z týchto kanálov vyžaduje komunikáciu na osobitnom porte.

Táto skutočnosť sa odzrkadľuje na zložitom manažmente VoIP (Voice over IP) relácií.

Analyzujte architektúru „SIP Single Port“ a porovnajte ju s inými existujúcimi architektúrami zameranými na manažment VoIP relácií. SIP Single Port architektúra pre multimediálnu komunikáciu využíva na rozdiel od protokolu SIP len jeden port, čo je hlavnou motiváciou pre návrh optimalizácie manažmentu VoIP relácií.

Navrhnite aplikáciu, ktorá umožní riadiť aktívny manažment relácií. Výsledná aplikácia musí byť schopná adaptovať sa na náhle zmeny v sieti ako napríklad zahltenie, zhoršenie kvality, pád linky, zmena IP adresy klienta a podobne.

Na základe analýzy a návrhu implementujte aplikáciu a výsledok práce zhodnoťte.

3 Analýza

3.1 Manažment

Multimediálne relácie sa stali súčasťou nášho každodenného života a ich možnosti a kvalita pomáhajú medziľudskej komunikácii prekonávať veľké vzdialenosti s cieľom poskytnúť komunikujúcim stranám zážitok v najväčšej možnej miere podobný „živému“ kontaktu. Je žiaduce, aby zážitok z takejto komunikácie bol vždy na vysokej úrovni bez ohľadu na počet zúčastnených strán.

Na umožnenie účasti čo najvyššiemu počtu potenciálnych účastníkov v multimediálnej relácii je vhodné používať existujúce technológie so širokou dostupnosťou a podporou v koncových zariadeniach. Vhodným kandidátom je SIP protokol, ktorý je považovaný za štandard v danej oblasti.

Manažment multimediálnych relácií založených na SIP protokole pozostáva okrem iného aj z manažmentu kvality multimediálnych relácií, pričom pod týmto pojmom chápeme zabezpečenie potrebných zdrojov a parametrov v časti sieťovej infraštruktúry, ktorou prechádza dátový tok multimediálnej relácie. Ďalšou úlohou je správa sieťových filtrov (firewall), ktoré by v záujme bezpečnosti mali prepúšťať len požadovanú sieťovú komunikáciu a všetky ostatné komunikácie zahadzovať. Manažment multimediálnych relácií pokrýva aj profilovanie poskytovaných služieb, ktoré na základe identifikácie príjemcu poskytovanej služby sú poskytované v dohodnutom rozsahu a kvalite.

Vyššie popísané činnosti je jednoduché vykonávať nad reláciami medzi koncovými bodmi reprezentovanými jedinečnými IP adresami. Keďže množstvo IP adries je obmedzené, bolo nutné zaviesť koncept verejných a privátnych IP adries a preklad medzi nimi. Tento krok síce oddialil potrebu rozšírenia 32-bitového IP adresného priestoru, ale protihodnotou bolo znefunkčnenie niektorých sieťových protokolov vyšších vrstiev referenčného modelu ISO OSI a potreba zavedenia opravných mechanizmov, ktoré svoju úlohu plnia s čiastočným úspechom alebo naopak komunikáciu koncových bodov skomplikujú. Problémy nastávajú z dôvodu prekladu sieťových portov bez toho, aby na tento fakt boli komunikujúce koncové body upozornené.

Zabezpečenie kvalitatívnych parametrov prenosu dátového toku multimediálnej relácie je možné uskutočňovať na základe informácií protokolov nižších vrstiev referenčného modelu ISO OSI, ale tento prístup bez použitia výkonovo-náročnej hĺbkovej analýzy paketov nemusí vždy citlivo reagovať na požiadavky multimediálnej relácie a prípadné zlyhanie má za následok zlyhanie aplikácií využívajúcich čiastočne alebo úplne nefunkčnú časť siete. Preto je vhodné, aby aplikácia prevádzkujúca samotnú multimediálnu reláciu mala prehľad o stave siete a vedela sa prispôsobiť prípadným výpadkom spojenia.

3.2 Kontext

Produkt tohto tímového projektu má za cieľ zjednodušiť manažment multimediálnych relácií založených na protokole SIP zjednodušením správy sieťových filtrov vyplývajúcej z použitia architektúry SIRUP popísanej v ďalších častiach dokumentácie. Rozšírením tejto architektúry o sledovanie vybraných metrík jednotlivých ciest v sieti a umožnením presunu multimediálnych relácií na inú dostupnú sieťovú cestu chceme predviesť schopnosť adaptovania aplikácie na náhle zmeny v sieti.

Aplikácia bude vhodná pre používateľov multimediálnych relácií založených na protokole SIP, konkrétne organizácie s redundantnou konektivitou do siete Internet, prípadne poskytovateľov internetového pripojenia a IP telefónie s redundantnou infraštruktúrou.

3.3 Existujúce riešenia

Relevantným príkladom existujúceho riešenia manažmentu multimediálnych relácií založených na SIP protokole je modul programu Kamailio s názvom Mediaproxy, ktorý umožňuje prechod multimediálnych relácií cez smerovače s prekladom sieťových adries a obmedzené profilovanie prichádzajúcich multimediálnych dátových tokov prostredníctvom manipulácie INVITE správ.

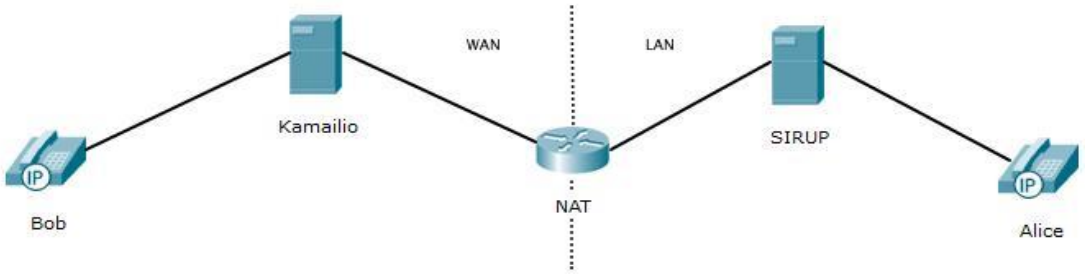
3.3.1 SIP Single Port

Téma tímového projektu vychádza z pôvodnej práce SIP Single Port. Účelom SIP Single Portu bolo umožniť prechod RTP a RTCP paketom v sieti s implementovaným prekladačom adries NAT. Nakoľko protokoly SIP, RTP a RTCP fungujú na troch rôznych portoch, prekladač adries ich videl ako tri rôzne komunikácie, a teda snahou bolo vytvoriť určitý preposielač, ktorý by tieto tri porty spojil do jedného a vytvoril tak z pohľadu prekladača jednu spoločnú komunikáciu. Riešenie bolo implementované a je funkčné.

3.3.2 Single a Dual Proxy režim

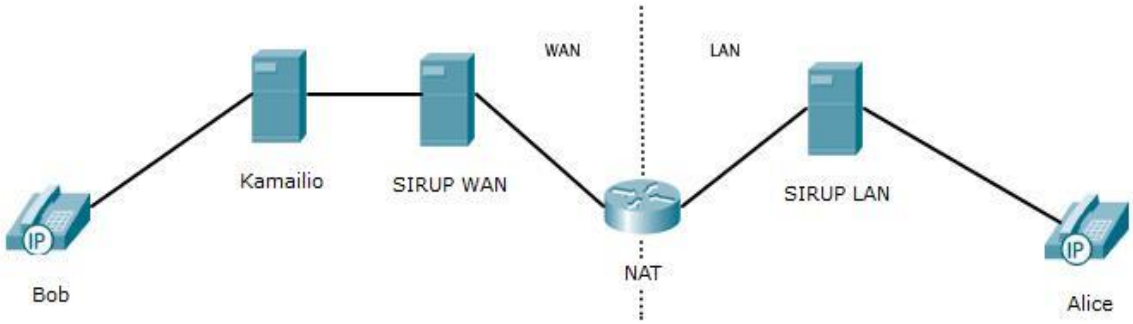
V závislosti od typu prekladu adries (preklad nezávislý od koncovej stanice, závislý od koncovej adresy, alebo závislý od koncovej adresy a portu) je používaný SIRUP (proxy server multiplexujúci pakety na jeden port) v režime single alebo dual proxy.

V režime single proxy (obrázok 3 - 1) je inštancia SIRUP - u len jedna a je umiestnená na strane klienta (režim prekladu nezávislého od koncovej stanice).



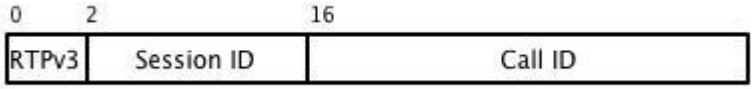
Obrázok 3-1 Single proxy režim

V prípade dual proxy (obrázok 3 - 2) režimu sú inštancie SIRUP - u dve (režimy prekladu závislého od koncovej adresy alebo závislého od koncovej adresy a portu), pričom jedna z nich je umiestnená v LAN sieti za prekladačom u klienta a druhá sa nachádza na druhej strane prekladača u poskytovateľa. Na oboch stranách sa pakety po prijatí multiplexujú na jeden port a následne sa posielajú na druhú stranu, kde sú znova demultiplexované a preposielané ďalej.



Obrázok 3-2 Dual proxy režim

Keďže sa viacero komunikácií zbieha do jedného portu medzi dvoma SIRUP - mi, nie je možné na druhej strane RTP a RTCP pakety spätne priradiť do jednotlivých komunikácií na základe portu ako zvyčajne, a teda je potrebné rámce značkovať (obrázok 3 - 3). Prvé dva bity značky sú nastavené na jednotky tak, aby paket vyzeral ako RTPv3. Za týmito dvoma bitmi sa nachádza sessionID, ktorý určuje číslo relácie v rámci hovoru. Koniec značky tvorí callID. Táto značka sa umiestňuje hneď za UDP hlavičku.



Obrázok 3-3 SIRUP značkovanie

3.3.3 Funkcionalita SIRUP - u

Na to aby sme mohli realizovať manažment relácií potrebujeme dva navzájom kompatibilné proxy servery, a preto budeme od teraz uvažovať len o SIRUP - e v dual proxy režime.

Aktuálna inicializácia SIRUP - ov po ich spustení prebieha nasledovne: Na začiatku Alica posiela REGISTER správu na server Kamailio. LAN SIRUP je pritom nastavený ako outbound proxy, a teda REGISTER zachytí. Pozmení ho a preposiela na adresu WAN SIRUP - u, ktorú sme vopred špecifikovali v konfiguračnom súbore. Ten po prijatí prvej SIP správy zistí skutočnú externú adresu LAN SIRUP - u, ktorú si uloží a všetky nasledujúce správy smeruje cez túto externú adresu. Pokým WAN SIRUP neprijal REGISTER, nemôže žiadne správy smerovať, nakoľko nepozná externú adresu LAN SIRUP - u.

Po novom predpokladáme, že každý klient bude mať u seba jednu bežiacu inštanciu SIRUP - u. Na druhej strane sa bude nachádzať poskytovateľova inštancia serverového SIRUP - u, ktorá bude bežať na viacerých portoch a pomocou MPLS tunelov na základe portov vyberať cestu pre reláciu.

Aktuálne je SIRUP riešený ako jednoduchý preposielač. Všeobecný postup spracovania paketov sa dá popísať takto: Po prijatí paketu ho najskôr prezeráme na kernelovej vrstve pomocou nástroja iptables. Kontrolujeme, či paket prišiel z očakávanej adresy a portu (zistené z SDP zo zachytených paketov) a v prípade, že paket dané pravidlo spĺňa, označkujeme ho a prepošleme ďalej na adresu druhej inštancie SIRUP - u. Ďalším pravidlom sa pozeráme do paketu za UDP hlavičku a v prípade, že sa tu nachádza očakávaná značka, odstránime ju a paket prepošleme podľa pravidla ďalej. Táto kontrola značky však nezohľadňuje zdrojovú adresu a port paketu, a preto zbytočne kontroluje reťazec za UDP aj pre pakety bez značky. Z hľadiska efektivity toto riešenie nie je úplne vhodné a zrejme bude potrebné do pravidla zakomponovať aj kontrolu adresy a portu.

Zjednodušená verzia iptables pravidiel:

- očakávaná zdrojová adresa, port (paket neprišiel od druhej inštancie SIRUP - u) -> vlož značku pre daný hovor, pošli na druhú inštanciu SIRUP - u
- očakávaná značka za UDP (paket prišiel od druhej inštancie SIRUP - u) -> odstráň značku a pošli na danú adresu a port podľa mapovania

3.3.4 Prepínanie medzi cestami

Na riešenie problému prehadzovania ciest medzi SIRUP - mi bude potrebné vyriešiť akým spôsobom budú medzi sebou komunikovať klientská a serverová inštancia SIRUP - u. Pokiaľ totiž uvažujeme najmenej vhodné mapovanie pre prechod cez NAPT, ktoré je závislé od cieľovej adresy aj portu, môžeme predpokladať, že keď klient začne komunikovať so serverom na inom porte, zmení sa mu mapovanie a bude vystupovať pod zmeneným portom alebo aj zmenenou adresou. Jedným z riešení by

mohlo byť poslanie správy REGISTER od klienta serveru, pomocou ktorej by oznámil svoju novú polohu (IP adresu a port).

Zložitejšia situácia nastáva, ak chceme zmeniť cestu počas prebiehajúceho hovoru. Táto zmena musí byť plynulá a v ideálnom prípade nezaznamenaná komunikujúcimi stranami. Je preto potrebné navrhnúť mechanizmus potvrdzovania zmeny kontaktných informácií SIRUP - ov tak, aby počas hovoru nedošlo k strate paketov. Vychádzajme z predpokladu, že klientský SIRUP posiela správu REGISTER na oznámenie svojej novej externej adresy a portu. Keď túto správu zachytí server, pripraví si potrebné pravidlá a posiela odpoveď 200 OK. Vďaka tejto odpovedi klient vie, že server je pripravený preposielať správy na inom porte a teda môže cieľový port odosielaných paketov zmeniť. Otvorená však zostáva otázka, akým spôsobom je možné potvrdiť správu 200 OK tak, aby klient vedel svoju pripravenosť oznámiť serveru. Správa ACK sa môže použiť len v rámci metódy INVITE a správa PRACK sa používa na potvrdzovanie provizionálnych odpovedí 101-199.

Nakoľko doposiaľ klient aj server boli nastavení tak, aby oba pracovali len s jednou inštanciou, je tiež potrebné zaviesť nejaký identifikátor klientov, pomocou ktorého by sa vedeli prihlásiť voči serveru. Na zabezpečenie jedinečnosti identifikátora ho bude spravovať jeden centrálny uzol.

3.3.5 SIRUP Master

Sirup Master je server ktorý primárne komunikuje s klientskymi (LAN) SIRUP - mi a tvorí riadiacu časť architektúry. Tento server bude mat statickú IP adresu a bude nevyhnutný pre zriadenie spojenia medzi klientovou a serverovou časťou SIRUP architektúry. Keďže SIRUP v súčasnej implementácií dokáže pracovať s protokolom SIP, najrozumnejším riešením bude, aby tento server rozumel tomuto protokolu tiež. Počas štúdia na bakalárskom stupni na predmete Konvergencia mobilných a pevných sietí zadaním bolo naprogramovať vlastné SIP proxy, ktoré ovláda základné SIP správy. Tento proxy server môžeme využiť v tejto architektúre. Server bude tvoriť rozhranie medzi manažmentom a SIRUP architektúrou, bude zbierať informácie od klientov a oznamovať im, čo robiť v prípade, keď dôjde k nepredvídanej situácie na základe pokynov manažment servera.

Server s klientmi SIRUP - u bude komunikovať pomocou protokolu SIP, manažérskym serverom pomocou websocket - ov v JSON objektoch. Pre rýchlejšiu komunikáciu si bude udržiavať vlastnú databázu klientov a serverov architektúry SIRUP.

3.4 Manažment Server

Manažmentový server je ústredný bod celej architektúry. Mal by zabezpečovať komunikačné rozhranie medzi všetkými uzlami infraštruktúry. Takéto jednotné rozhranie sa nazýva API.

3.4.1 Implementačné nástroje

3.4.1.1 Programovací jazyk JavaScript

JavaScript je, ako už názov napovedá, skriptovací objektovo orientovaný programovací jazyk, ktorý sa hlavne využíva v moderných webových prehliadačoch ako nástroj interakcie s používateľom. Syntax je ovplyvnený jazykom C a veľa názvov je prebraných s jazyku Java. Navzdory svojmu názvu, má ale JavaScript veľmi málo spoločné s jazykom Java. JavaScript bol prvý krát štandardizovaný v roku 1997. Momentálne posledná verzia štandardu 5.1 je z roku 2011[1].

Väčšinou sa k jazyku JavaScript pristupuje interpretovaným spôsobom. To znamená, že zdrojový kód sa priamo vykonáva bez predchádzajúceho prekladu do strojového kódu. Engine V8 od Google si ale zvolil iný prístup. Zdrojový kód sa kompiluje priamo do natívneho kódu použitej architektúry. Podporované sú 32 a 64 bitové verzie x86, ARM a MIPS, čo zahrňuje majoritnú väčšinu bežne používaných systémov. Takýto prístup dovoľuje výrazne optimalizovať vykonávajúci sa kód a zvyšuje výkon aplikácií vykonávaných na tomto engine [2][3].

Veľa ľudí jazyk JavaScript škatuľkuje len do sveta webových prehliadačov, s tým že to nie je plnohodnotný jazyk. To nie je tak úplne pravda, pretože existuje viacero implementácií mimo webových prehliadačov, napríklad platforma Node.js, určená pre vývoj webových aplikácií.

3.4.1.2 Platforma Node.js

Node.js je platforma postavená na Chrome JavaScript engine pre jednoduché vytváranie rýchlych a škálovateľných sieťových aplikácií. Platforma Node.js vznikla v roku 2009 a relatívne rýchlo nadobudla značnú popularitu, nielen v komunite vývojárov, ale aj medzi veľkými softvérovými hráčmi, ako napríklad Microsoft. Je často nazývaná ako serverový JavaScript. Základom tejto platformy je JavaScript engine V8 od spoločnosti Google, ktorý je voľne dostupný pod BSD licenciou s otvoreným zdrojovým kódom. Tento engine je zodpovedný za vykonávanie JavaScript kódu v prehliadači Chrome od spoločnosti Google [2].

Node.js implementuje programovací model udalostí postavený na neblokujúcich vstupno-výstupných operáciách. Väčšina bežných serverových implementácií vytvára pre každého klienta samostatné vlákno v systéme. Node.js serverová aplikácia beží a obsluhuje klientov v jednom vlákne, čo znamená, že odpadáva záťaž systému pri prepínaní medzi množstvom procesov. Tieto vlastnosti robia túto platformu ideálnu pre aplikácie v reálnom čase, ktoré využívajú veľké množstvo vstupno-výstupných operácií od množstva klientov [2].

Z hľadiska programátora je Node.js taktiež zaujímavá platforma, pretože aplikácie sa píšu v jazyku JavaScript, ktorého aspoň základy ovláda veľké množstvo webových vývojárov. Jednou z nevýhod je, že väčšinu kódu tvoria asynchrónne JavaScript funkcie, ktoré pri bežnom zápise vytvárajú relatívne neprehľadnú štruktúru kódu a sťažujú jeho čitateľnosť [4].

Výhodou je modulový systém, ktorý zjednodušuje znovupoužitelnosť jednotlivých častí a výrazným spôsobom urýchľuje vývoj. Napríklad pri použití modulu s názvom Express, je implementácia webového servera záležitosť niekoľko málo riadkov kódu. Takýto typ systému dovoľuje programátorovi sústrediť sa na samotné správanie sa vyvíjanej aplikácie. Na manažment použitých modulov sa používa program npm, ktorý sa stará napríklad o inštaláciu aktualizáciu modulov [5].

Node.js je možné inštalovať a prevádzkovať na systémoch s Windows, Linux alebo OS X. Niektoré moduly ale nemusia podporovať všetky tri platformy, no multiplatformovosť je značnou výhodou [2].

Všetky tieto vlastnosti Node.js kvalifikujú túto platformu ako ideálnu pre implementáciu manažmentového servera a pre vytvorenie API rozhrania medzi jednotlivými časťami navrhovaného systému. Počíta sa s obsluhou veľkého množstva požiadaviek, ktoré ale sú nenáročné na výpočtový výkon, čo je dôležité z výkonnostného hľadiska [6].

3.4.1.3 Databáza

Pri použití Node.js ako platformy pre manažérsky systém, je okruh možných databáz veľmi široký. Je možné použitie klasických SQL databáz ako napríklad MySQL alebo PostgreSQL. No lepšie výsledky je možné dosiahnuť pri použití NoSQL databáz, ktoré sú svojou podstatou bližšie k tejto platforme, pretože poväčšine dokážu pracovať priamo s JavaScript objektmi vo formáte JSON. Vhodné sú hlavne dve databázy s názvom MongoDB a CouchDB. Obe tieto databázy môžu bežať pod systémami Windows, Linux alebo OS X [7].

3.4.1.4 Mongo Database

MongoDB sa svojím prístupom podobá na tradičné SQL databázy ako MySQL alebo PostgreSQL. Prístupy k dátam sú definované cez indexy, podobne ako v SQL databázach. Použitie tejto databázy je vhodné, ak je potrebné mať dynamické dopyty a vysoká rýchlosť je zásadnou podmienkou. Pre komunikáciu s touto databázou sa používa vlastný binárny protokol. Databáza je dostupná pod AGPL licenciou [8].

3.4.1.5 Couch Database

Existuje aj NoSQL databáza, ktorá ale na rozdiel od MongoDB používa namiesto dynamických dopytov MapReduce funkcie. Je zameraná na jednoduché použitie a zaručenie konzistencie dát. Použitie tejto databázy je vhodné, ak sa dáta v nej akumulujú a príliš často nemenia. Využívajú sa dopredu definované dopyty. Komunikáciu je zabezpečená cez REST API, a teda protokol HTTP. Databáza je dostupná pod Apache licenciou [9].

3.5 Manažment API

3.5.1 API

Aplication programming interface (API) je špecifikácia prístupu k jednotlivým funkciám programových celkov. Väčšinou je to zbierka procedúr a funkcií, ktoré sú poskytnuté používateľovi pre prístup ku knižnici alebo službe a definujú jej vonkajšie správanie sa.

3.5.2 JSON

JavaScript Object Notation (JSON) je otvorený štandardizovaný formát používaný primárne na prenos dát. JSON je definovaný v štandarde RFC 4627. Formát je odvodený z JavaScript, no je jazykovo nezávislý a dostupný v množstve iných jazykov.

Podobným štandardom ako JSON je XML. Oproti XML má JSON dátovo úspornejšiu notáciu. Je natívne podporovaný jazykom JavaScript, čo je výhodné v spolupráci s Node.js. Pre JSON sú charakteristické tieto dve vlastnosti:

- Notácia je tvorená kolekciami kľúč - hodnota podobne ako v iných jazykoch sú objekt, štruktúry alebo hash tabuľky.
- Hodnoty tvoria usporiadaný zoznam podobne ako polia, alebo usporiadané listy v iných jazykoch.

Tieto dva kľúčové prvky zabezpečujú vysokú prenosnosť medzi rôznymi jazykmi, rýchlosť spracovania a nenáročnosť na veľkosť a výkon. Pre všetky tieto vlastnosti je JSON ideálny formát na prenos dát medzi klientmi a serverom v budovanej manažment API [10].

3.5.3 Implementácia API

K vytváraniu API je možné pristupovať rôznymi spôsobmi. Najtradičnejšie je použitie Representational state transfer (REST) architektúry. Pre aplikácie v reálnom čase sa stáva obľúbená relatívne nová platforma Socket.io, ktorej serverová časť je implementovaná v prostredí Node.js.

3.5.3.1 REST architektúra

REST je architektúra, nie samotná implementácia. To znamená, že je nezávislá od platformy a protokolov a sústredí sa na samotné prvky tejto architektúry. Vo svete Internetu je to momentálne dominantná API architektúra a väčšinou využíva protokol HTTP, s ktorým má veľa spoločného. Pri využití HTTP sa implementácia nazýva RESTful [11].

Architektúra je postavená na princípe klient - server a využíva sa metóda požiadavka - odpoveď. Podobne ako v protokole HTTP, najviac používané požiadavky sú:

- **GET** - Požiadanie o dáta

- **POST** - Požiadavka o modifikáciu dát
- **PUT** - Požiadavka na vytvorenie alebo prepísanie dát
- **DELETE** - Požiadavka na vymazanie dát

Samotné dáta v požiadavkách a odpovediach bývajú často objekty typu XML alebo JSON.

3.5.3.2 Socket.io knižnica

Socket.io je JavaScript knižnica implementovaná na platforme Node.js. Používa sa pre komunikáciu medzi serverom a klientmi v reálnom čase. Primárna komunikačná technológia je WebSocket protokol, no táto knižnica od neho nie je závislá a v prípade jeho nedostupnosti na používanej platforme používa iné protokoly a ich implementácie ako Flash Sockets, JSONP pooling alebo AJAX long pooling.

Na serverovej strane spolupracuje s jednoduchým HTTP serverom v prostredí Node.js, či už vstavaným alebo o funkcie bohatší modulom Express. Serverová časť ponúka asynchrónne posielanie správ klientom, a takisto asynchrónne prijatie správ od klientov. Takéto správy môžu obsahovať dátové objekty. Zaujímavá je aj schopnosť jednoduchého rozosielania správ všetkým klientom naraz.

Klientska časť býva často webový prehliadač a klient teda býva napísaný v jazyku JavaScript. No nemusí to tak byť a klientska aplikácia je nezávislá na platforme a existujú implementácie pre väčšine bežne rozšírených jazykoch ako C, C++, C#, Java atď. Aplikačné rozhranie u klienta je väčšinou totožné ako u serveru, no záleží od konkrétnej implementácie [12].

3.6 Manažment užívateľského rozhrania

Dôležitým prvkom každej aplikácie je ako pôsobí a vyzerá navonok. Ako sa s ňou narába a aké je rozhranie medzi používateľom a aplikáciou. Rozloženie jednotlivých prvkov na obrazovke by malo byť intuitívne a prehľadné. V tejto časti sa budeme zaoberať užívateľským rozhraním - frontendom pre zobrazenie informácií potrebných na riadenie systému.

Frontend je rozhranie medzi používateľom a vnútornou funkčnou časťou aplikácie. Slúži na zber vstupných údajov v rôznych formách od používateľa alebo na zobrazenie výstupných informácií systému.

Naša aplikácia má umožniť **aktívny manažment** relácií a mala by byť schopná adaptovať sa na náhle zmeny v sieti. Aby táto adaptácia bola prezentovaná klientovi je potrebné zobrazovať niekoľko dôležitých informácií. Tieto informácie upozornia používateľa na zmeny, ktoré aplikácia vykonala a prečo.

Aby sme dosiahli výnimočné zobrazenie všetkých relevantných informácií je potrebné analyzovať dostupné nástroje na zobrazovanie. Či už formou grafu alebo pomocou dynamických prvkov, ktoré poskytujú jednotlivé nástroje rozhrania API. V celom manažmente budeme používať **skriptovací programovací jazyk JavaScript** s rôznymi platformami a knižnicami, či už štandard JSON alebo Socket.io prípadne Node.js, je preto vhodné použiť aj na tvorbu rozhrania medzi klientom a manažmentom

JavaScript. Zaručuje nám to nielen kompatibilitu jednotlivých programových celkov, ale aj zjednodušenie práce z dôvodu lepších znalostí daného jazyka. Ďalšou výhodou použitia je **podpora väčšiny prehliadačov** bez potreby inštalovať zložité a rozsiahle dodatočné nástroje.

Pri tvorbe webových stránok či aplikácií netreba zabúdať aj na **hypertextový značkovací jazyk HTML**. Je určený na tvorbu webových stránok a zobrazovanie informácií vo webových prehliadačoch. Poslednou verziou je HTML 5, ktorá kladie dôraz na rýchlejší a jednoduchší zápis značiek a zároveň na ich účinnosť. Podporuje tiež vytváranie aplikácií, ktoré fungujú bez internetového pripojenia a ukladajú dáta do lokálneho úložiska na počítači užívateľa. Obsahuje množstvo ďalších užitočných prvkov a značiek ako je podpora 2D kreslenia alebo prehrávanie audio a video súborov.

Teraz sa pozrieme na informácie, ktoré budeme zobrazovať:

- **RTT – round-trip delay time** – čas potrebný na prenesenie paketu zo zdrojového uzla do cieľovej siete
- **PL – packet loss** – strata paketov počas prenosu
- **Jitter** – kolísanie veľkosti oneskorenia paketu pri prechode sieťou
- **Bandwith** – aktuálna šírka pásma
- **Štatistické informácie** – počet hovorov na danej linke, počet registrovaných klientov, počet relácii na jedného klienta, počet prenesených dát na jedného klienta, počet pádov linky
- **Informácie o klientoch** – IP adresa a port klienta, užívateľské meno
- **Informácie o prepínaní liniek** – počet prepnutí, ktorá relácia bola prepnutá a kam
- **Indikácia pádu linky**

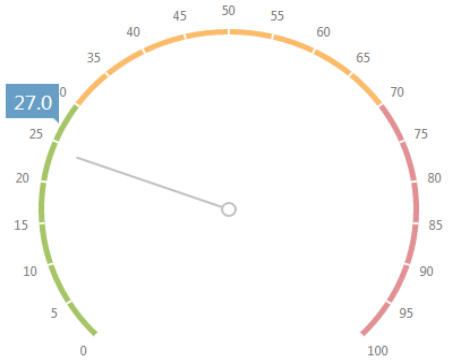
Všetky informácie, ktoré sme vymenovali by bolo najvhodnejšie zobrazovať formou grafov využitím knižníc jazyka JavaScript alebo použitím jazyka HTML. Kombináciou týchto dvoch programovacích jazykov na tvorbu webových aplikácií dosiahneme, aby aplikácia manažmentu spĺňala a zobrazovala všetko potrebné v dokonalom grafickom prevedení. Aby sme dosiahli želaný efekt, pozrieme sa bližšie na jednotlivé knižnice v programovacom jazyku JavaScript pre zobrazovanie informácií.

Hlavným prvkom webového rozhrania budú grafy. Existuje veľké množstvo knižníc v jazyku JavaScript, ktoré podporujú tvorbu grafov. Rozdiely medzi knižnicami sú len v typoch grafov, ktoré knižnice dokážu zobraziť. Základné porovnanie tých najlepších knižníc na zobrazenie grafov je na obrázku 3 - 4.

	Supported Chart Types											
	Line	Timeline	Scatter	Area	Pie	Rectangular Pie	Donut	Bullet	Radar	Funnel	Gantt	Grouped
amCharts	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes
CanvasJS	Yes	No	Yes	Yes	Yes	No	Yes	No	No	No	No	No
ChartJS	Yes	No	Yes	Yes	Yes	No	Yes	No	No	No	No	Yes
FusionCharts	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes

Obrázok 3-4 Porovnanie knižníc na zobrazenie grafov[14]

Podľa nasledujúceho porovnania by sa mohlo zdať, že najlepšou knižnicou na použitie je *FusionCharts*. No najväčším hendikepom tejto knižnice je, že **nepodporuje Gauge Widgets**. Ukážku takéhoto grafu môžeme vidieť na obrázku 3 - 5. Preto sme sa rozhodli, že budeme používať knižnicu **ChartJS Dev Express**, ktorá má množstvo ďalších výhod, ktoré popíšem v priebehu tejto kapitoly.

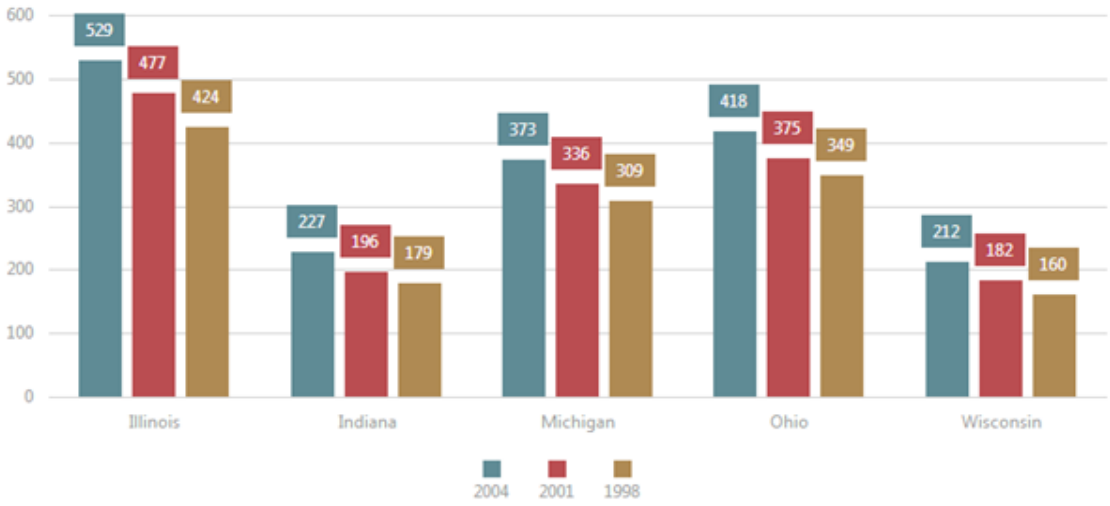


Obrázok 3-5 Graf HTML JS Gauge Widgets[13]

Tento graf **je najlepší** na zobrazovanie informácií, ktoré potrebujeme. Jedná sa hlavne o oneskorenie prenosu paketov (RTT), strata paketov (PL), odchýlka oneskorenia paketu (Jitter) a šírka pásma (Bandwith). Po nadviazaní spojenia s manažmentom, ktorý bude mať všetky potrebné informácie k dispozícii, budú posielané do webovej aplikácie periodicky. To nám zabezpečí, že graf bude mať vždy aktuálne informácie, ktoré bude na takomto grafe zobrazovať.

Zobrazovanie štatistických informácií by bolo vhodné rozdeliť. A to konkrétne na **historické informácie**, teda tie ktoré boli zaznamenané niekoľko dní dozadu **a na aktuálne**, ktoré sú v danej chvíli. Na zobrazenie aktuálnych informácií napríklad o počte klientov na danej linke môžeme použiť graf na obrázku 3 - 6, pre zobrazenie historických informácií by postačoval graf na obrázku 3 - 7.

Obrázok 3-6 Graf pre zobrazovanie aktuálnych štatistických informácii[13]



Obrázok 3-7 Graf pre zobrazovanie historických štatistických informácii[13]

Podobné grafy by mohli byť použité aj na zobrazovanie ďalších informácií. Implementácia je pomerne jednoduchá a veľkou výhodou je, že knižnicu ChartJS podporuje mnoho mobilných platforiem ako Android, iOS, Windows Phone a taktiež všetky mobilné aj desktopové webové prehliadače. Medzi ďalšie výhody tejto knižnice môžeme zaradiť[13]:

- *interakcia grafov – pri označení určitej časti grafu sa aktualizujú aj grafy s ním spojené,*
- *jednoduchá možnosť priamej konfigurácie,*
- *prispôsobenie sa väčšine mobilných zariadení,*
- *obrovské množstvo rôznych druhov grafov a nastavení,*
- *možnosť spájať rôzne typy grafov do jedného,*
- *podpora grafov s výberom úseku.*

Ďalšou veľkou výhodou tejto knižnice je *podpora monitorovacích funkcií.* Obsahuje možnosť vytvoriť si monitorovací panel, kde pomocou kombinácie rôznych grafov a vhodného prostredia dokážeme monitorovať rôzne prvky našej siete. Ukážka panelu je na obrázku 3 - 8.

Obrázok 3-8 Monitorovací panel[13]

Ako už bolo spomenuté, knižnica ktorú sme opisovali spolu s jazykom HTML 5 sú najvhodnejšou voľbou pre potreby monitoringu našej VoIP siete a dokonale sedia na požiadavky zobrazovania potrebných informácií.

3.7 Simulácia siete

Aby bolo možné overiť funkčnosť implementovaného riešenia, je potrebné vytvoriť prostredie pre jeho testovanie. Vytvorenie sieťových tunelov na reálnych alebo virtuálnych zariadeniach by prestavovalo zvýšené nároky na počet a výkon použitých zariadení, a tiež by obmedzilo možnosti nútenej zmeny charakteristiky tunelov. Z pohľadu manažéra relácií sú sieťové tunely opísané určitými parametrami a ich skutočná existencia nie je potrebná. Cieľom je možnosť meniť parametre opisujúce tunely tak, aby výsledná aplikácia preukázala schopnosť adaptovať sa na tieto zmeny.

3.7.1 Network Emulation (NetEm)

NetEm (Network Emulation) poskytuje potrebnú funkcionalitu pre testovanie protokolov imitovaním vlastností skutočných sietí. Aktuálna verzia umožňuje vytvorenie oneskorenia, straty, duplicity, chybovosti a preusporiadania paketov. Na obmedzenie šírky pásma možno použiť Token Bucket Filter (TBF). Súčasné verzie Linuxového jadra (od verzie 2.6) obsahujú podporu pre tieto nástroje dostupné v rámci balíčka iproute2. NetEm sa ovláda pomocou konzolového programu ‘tc’. Syntax príkazov je pomerne komplikovaná a konfigurácia si vyžaduje hlbšie pochopenie problematiky. Preto je vhodné navrhnúť rozhranie, ktoré poskytne používateľsky prívetivejšie prostredie s podporou transformovania vstupov do formy vyžadovanej programom tc[15].

Pakety je možné klasifikovať pomocou filtrov. Na účely zadania je vhodné použiť filtrovanie podľa čísla UDP portov paketov, keďže tieto charakterizujú, ktorým tunelom má prebiehať komunikácia. Rozdelenie do skupín umožňuje definovať odlišné parametre preposielania pre rôzne tunely.

3.7.2 Informácie o kvalite linky

1. **Round – Trip Time (RTT)**

Inak nazývaný aj **round-trip delay time (RTD)** je čas potrebný na cestovanie paketu medzi zdrojovým uzlom v sieti a konečnou stanicou. V kontexte počítačových sietí je možné tento čas zistiť zadaním príkazu *ping* na zdrojovom počítači s IP adresou cieľového. Nazýva sa aj *ping time*. Jeho veľkosť závisí od rôznych faktorov:

- rýchlosť internetového prenosu dát zdrojového zariadenia,
- typ prenosového média,
- fyzická vzdialenosť medzi zdrojom a cieľom,
- počet uzlov v sieti,
- množstvo dopravy na sieti atď.

RTT je aj jedným z mnohých prvkov, ktoré ovplyvňujú oneskorenie siete, čo je čas medzi vytvorením požiadavky na dáta a úplným vrátením alebo zobrazením týchto súborov. Čas na prenesenie paketu je možné aj teoreticky veľmi ľahko vypočítať, no bude to len orientačný výsledok a my potrebujeme pracovať s presnými číslami.

Meranie RTT pomocou ICMP paketov alebo tzv. pingu, nie vždy dáva relevantné výsledky v reálnych sieťach. Existuje ešte jeden spôsob merania oneskorenia paketov prostredníctvom nástroja s názvom **„TCP Ping“.** Myšlienka tohto nástroja je rovnaká ako pri normálnom pingu, ale namiesto používania ICMP paketov, odosiela TCP SYN pakety do cieľového zariadenia a meria čas medzi subsekvenciou prijatia **SYN/ACK alebo RST**. Nástroj beží pod operačným systémom Linux a obsahuje niekoľko prepínačov na nastavenie portu alebo rôznych módov. Obrázok 3 - 9 ukazuje fungovanie takéhoto nástroja na reálnom stroji.

```
[6:22pm] luthien:~/proj/tcpping-% sudo ./tcpping sumatra
TCP PING sumatra.internal.kehlet.cx (10.16.74.2:80) on en1
SYN/ACK from 10.16.74.2: seq=1 ttl=64 time=1.047ms
SYN/ACK from 10.16.74.2: seq=2 ttl=64 time=0.965ms
SYN/ACK from 10.16.74.2: seq=3 ttl=64 time=1.081ms
SYN/ACK from 10.16.74.2: seq=4 ttl=64 time=1.245ms
^C
--- sumatra.internal.kehlet.cx TCP ping statistics ---
4 SYN packets transmitted, 4 SYN/ACKs and 0 RSTs received, 0.0% packet loss
round-trip min/avg/max = 0.965/1.084/1.245 ms
```

Obrázok 3-9 Nástroj TCP Ping

Zdroj: http://www.kehlet.cx/articles/77.html

2. **Packet Loss (PL)**

K strate paketov v počítačovej sieti dochádza v prípade, že prenášaný paket nedosiahne svoj cieľ. Táto vlastnosť je taktiež v počítačových sieťach dôležitá a je jednou z hlavných faktorov, ktoré sa zisťujú pri každej prevádzke. Stratu paketu môže spôsobiť niekoľko faktorov, no takým najčastejším je degradácie signálu pri prechode cez sieťové

médium, resp. zahltenie linky. Potom to môžu byť poškodené pakety, chybný sieťový hardvér, chyba sieťového ovládača alebo chyby v smerovaní.

3. **Jitter**

V počítačových sieťach vyjadruje kolísanie oneskorenia príchodu paketov. Vzniká na smerovačoch ako dôsledok funkcie interných prioritných radov smerovačov a iných sieťových zariadení, a tiež smerovacích zmien. Hodnotu kolísania oneskorenia je možné zistiť opakovaným meraním oneskorenia.

4. **Bandwidth**

Prenosová rýchlosť udáva, aký objem informácií sa prenesie za jednotku času. Základná jednotka prenosovej rýchlosti je bit za sekundu. Meranie maximálnej prenosovej rýchlosti sa vykonáva meraním času potrebného na prenesenie daného objemu dát. Možno tiež merať aktuálnu prenosovú rýchlosť. Tá je vyjadrená súčtom veľkostí prenesených rámcov za jednotku času.

4 Špecifikácia

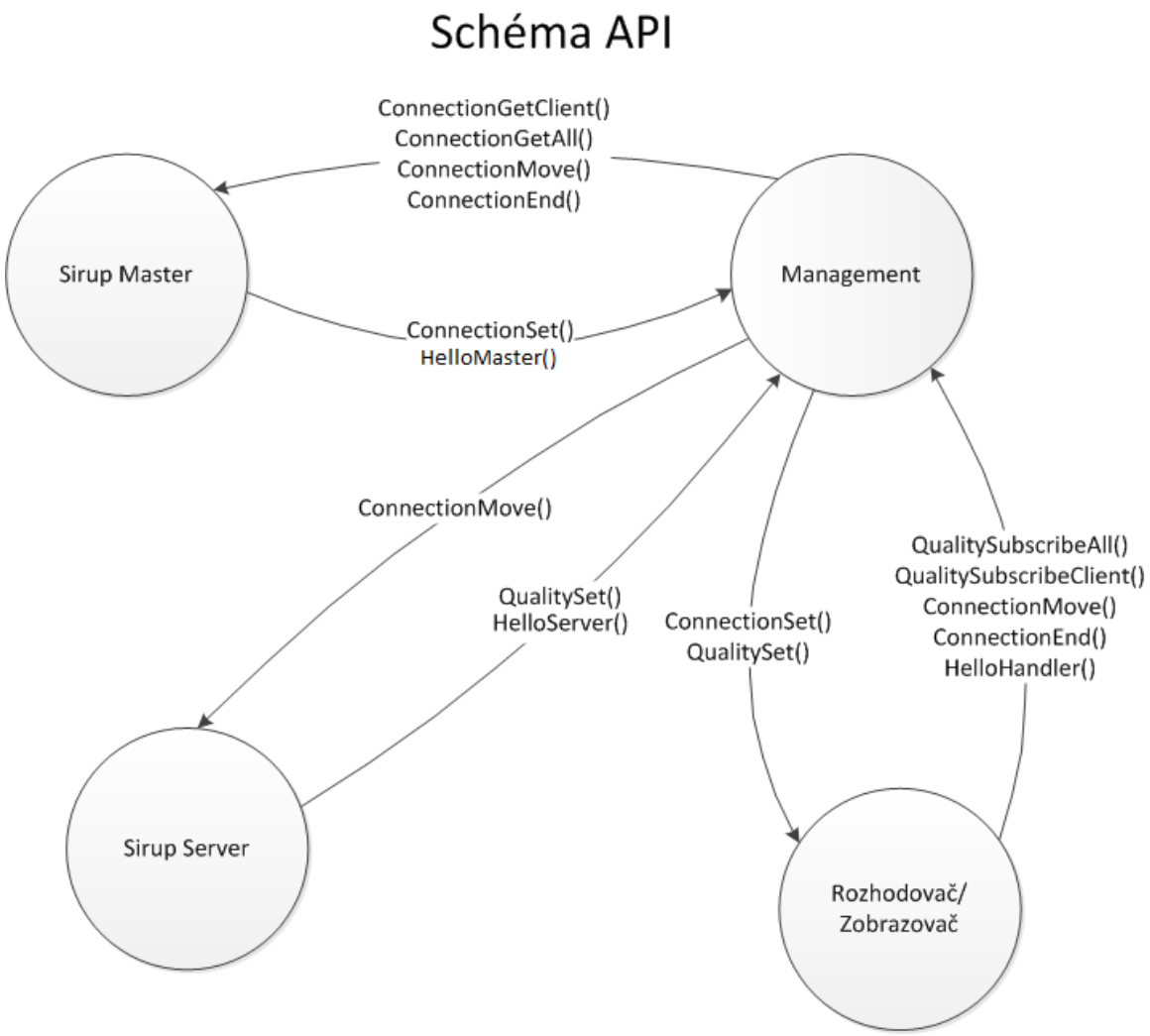
4.1 Funkcie systému

1. Rozkladanie záťaže
2. Monitorovanie multimediálnych relácií
3. Presun multimediálnych relácií medzi serverovými uzlami
4. Vykresľovanie štatistických informácií
5. Rušenie multimediálnych relácií
6. Zabezpečovanie kvality relácií - agresívnejšie prepínanie liniek - “platinum / VIP smerovanie”

4.2 Údaje

Dáta prenášané a archivované v databáze budú reprezentované ako JSON objekty.

4.3 Manažment API správy



Obrázok 4-1 Tok správ medzi uzlami systému

Správy o kvalite linky:

- **QualitySet()**
  Správa obsahujúca informácie o kvalite linky.
- **QualitySubscribeClient()**
  Správa žiadajúca o pravidelné preposielanie informácií o kvalite linky jedného klienta.
- **QualitySubscribeAll()**
  Správa žiadajúca o pravidelné preposielanie informácií o kvalite linky všetkých klientov.

Správy spojené so správou klientov

- **ConnectionSet()**
  Vytvorenie nového spojenia.
- **ConnectionGetClient()**
  Získanie informácií o spojení jedného klienta.
- **ConnectionGetAll()**
  Získanie informácií o spojení všetkých klientov.
- **ConnectionMove()**
  Zmena parametrov spojenia.
- **ConnectionEnd()**
  Ukončenie spojenia.

Správy spojene s pripojením uzlov architektúry

- **HelloMaster()**
  Prihlásenie Sirup mastra.
- **HelloServer()**
  Prihlásenie nového server Sirupu.
- **HelloHandler()**
  Prihlásenie zobrazovacej/rozhodovacej jednotky.

4.4 Správanie systému

4.4.1 Manažment

Úlohou manažmentu je poskytovať aplikačné rozhranie pre komunikáciu uzlov architektúry. To znamená, že musí vedieť posielať a prijímať správy, ktoré sú špecifikované v pasáži manažment API správy.

Ďalšou úlohou manažmentu je archivovať relevantné údaje v správach do databázy, a tiež si ich z tejto databázy vedieť vyžiadať.

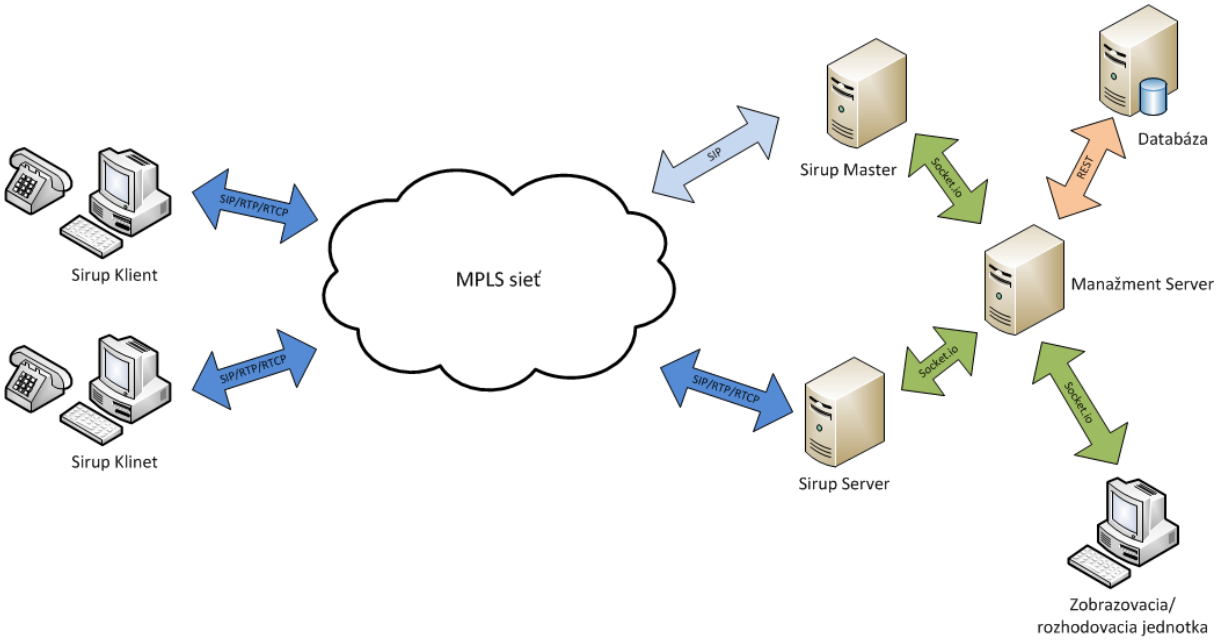
4.4.2 SIRUP komunikácia

- SIRUP - y komunikujú navzájom výlučne pomocou SIP správ
- Manažment nekomunikuje s klientmi priamo

- Klientske SIRUP - y majú pridelený jedinečný identifikátor
- Klientska inštancia SIRUP - u oznamuje svoju polohu voči serverovej pomocou správy REGISTER, obsahujúcej jej identifikátor
- Klientska inštancia SIRUP - u komunikuje s riadiacou inštanciou SIRUP - u pomocou správy SUBSCRIBE
- Riadiaca inštancia SIRUP - u udržiava s klientskou inštanciou reláciu pomocou správ NOTIFY

5 Návrh

5.1 Architektúra



Obrázok 5-1 Architektúra systému

5.2 Manažment API

Navrhované riešenie využije platformu Socket.io, ktorá bola v časti analýzy porovnávaná s bežne používanou REST architektúrou. Oproti REST architektúre sa správy Socket.io nemusia spoliehať na schému požiadavka - odpoveď. Správy môžu byť posielané kontinuálne bez nutnosti požiadavky na tieto správy. Táto schopnosť pomáha pri implementácií architektúry, ktorá má pracovať v reálnom čase a je pre ňu dôležitý faktor čas odozvy.

Ďalšou výhodou oproti REST architektúre je, že vzťahy medzi uzlami nemusia byť striktne klient - server. To dáva istú flexibilitu pri návrhu riešenia aplikačného rozhrania, keďže v REST architektúre server nemá možnosť žiadať dáta od klienta.

Všetky tieto fakty hovoria v prospech platformy Socket.io, ktorá by mala byť jednoducho implementovateľná na všetkých uzloch architektúry pomocou dostupných voľne šíriteľných knižníc.

Webové rozhranie medzi klientom a manažmentom bude mať na starosti knižnica ChartJS založená na platforme JavaScript. Zobrazenie potrebných informácii, bude vo forme grafov v spolupráci s hypertextovým značkovacím jazykom HTML.

5.3 Databáza

V časti analýzy sme porovnávali dve databázy MongoDB a CouchDB. Rozhodli sme sa použiť druhú menovanú, pretože predpokladáme, že dáta budeme hlavne archivovať a vykonávať nad nimi len dopredu definované operácie. Dáta by sa nemali

príliš často meniť a nebudeme potrebovať ani dynamické dopyty. MongoDB má síce výhodu vysokej rýchlosti, no konzistencia dát nie je zaručená. Obidve databázy ukladajú dáta v tvare JSON objektov, čo je vyhovujúce.

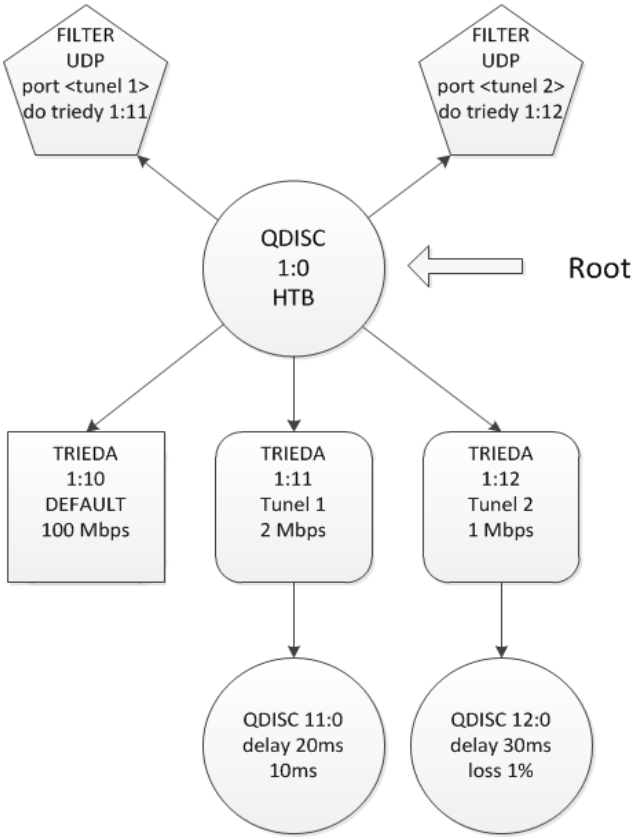
5.4 Údaje

Architektúra nášho systému sa skladá s rôznych systémov a je vhodné použiť spoločný formát pre reprezentáciu údajov. Dáta prenášané a archivované v databáze budú reprezentované ako JSON objekty, ktoré sú podporované na použitých systémoch a natívne podporované manažmentovým systémom a databázou.

5.5 Simulácia siete NetEm

Simulátor potrebný pre testovanie funkčnosti systému predstavuje oddelený a samostatne funkčný celok. Poskytuje možnosť nastavenia sieťových parametrov pre jednotlivé tunely a tiež možnosť zobrazenia aktuálne platných nastavení. Zmenou nastavení ovplyvňuje správanie systému, avšak žiadnym spôsobom s ním priamo nekomunikuje. Preto je voľba prostredia a jazyku na implementáciu nezávislá od spôsobu implementácie ostatných častí systému.

Samotné vykonávanie zmien v sieťovej komunikácií vykonáva nástroj NetEm. V predvolenom stave sa odchádzajúce pakety pridávajú do koreňového radu(QDISC root). Aby bolo možné nastavovať parametre pre rôzne tunely vytvorí sa zodpovedajúci počet radov (QDISC) a špecifikujú sa parametre radu, ako napríklad oneskorenie a odchýlka oneskorenia. Pre odchádzajúce pakety je zvolený zodpovedajúci rad podľa triedy, do ktorej patria. Triedy tiež umožňujú obmedzenie šírky pásma pre jednotlivé tunely. Trieda DEFAULT klasifikuje komunikáciu, ktorá nemá prechádzať cez tunely. Príslušnosť k triedam je určená pomocou filtrov. Filter hľadá zhodu v poliach paketu, ktoré označujú typ transportného protokolu a číslo portu. Ak sa nájde zhoda, paket sa zaradí do príslušnej triedy. Ak žiadny z filtrov nenájde zhodu, paket bude zaradený do triedy DEFAULT. Štruktúru radov, tried a filtrov môžeme vidieť na obrázku 5 – 2.



Obrázok 5-2 Štruktúra radov, tried a filtrov

NetEm môže byť nakonfigurovaný zrkadlovo na dvoch sieťových adaptéroch počítača, medzi ktorými bude preposielať komunikáciu. V tomto zapojení budú klientské zariadenia pripojené na jednom sieťovom adaptéri a serverové uzly pripojené na druhom sieťovom adaptéri. Počítač s úlohou preposielača bude simulovať existenciu sieťových tunelov oneskorením preposielania a zmenou ďalších sieťových parametrov. Pre testovanie klientskej aj serverovej časti na jednom zariadení sa použije sieťový adaptér Loopback. Je potrebné uvedomiť si, že pri komunikácií spôsobom otázka - odpoveď prechádzajú sieťou dve správy, pričom parametre siete ovplyvnia najprv otázku a neskôr aj odpoveď. Oneskorenie odpovede je teda z používateľského hľadiska dvojnásobné v porovnaní s údajom oneskorenia nastavenom na sieťovom adaptéri.

Pre lepšiu názornosť a používateľsky prívetivejšie ovládanie sa nástroj NetEm bude ovládať cez webové rozhranie. Úlohou rozhrania je zobraziť aktuálny počet tunelov, parametre tunelov a možnosť zmeny parametrov tunelov. Webové rozhranie získava potrebné informácie z výpisu programu ‘tc’. Tento tiež slúži na zmenu parametrov tunelu. Pomocou nasledujúcich príkazov sa vykoná konfigurácia NetEm profilu na jednom rozhraní na UDP porte 5091.

```
tc class add dev eth1 parent 1:0 classid 1:11 htb rate 1Mbit

tc qdisc add dev eth1 parent 1:11 handle 11:0 netem delay 50ms 10ms 25%
loss 11.1% 25%
```

```
tc filter add dev eth1 protocol ip parent 1:0 prio 11 u32 match ip
protocol 17 0xff match ip sport 5091 0xffff flowid 1:11

tc filter add dev eth1 protocol ip parent 1:0 prio 11 u32 match ip
protocol 17 0xff match ip dport 5091 0xffff flowid 1:11
```

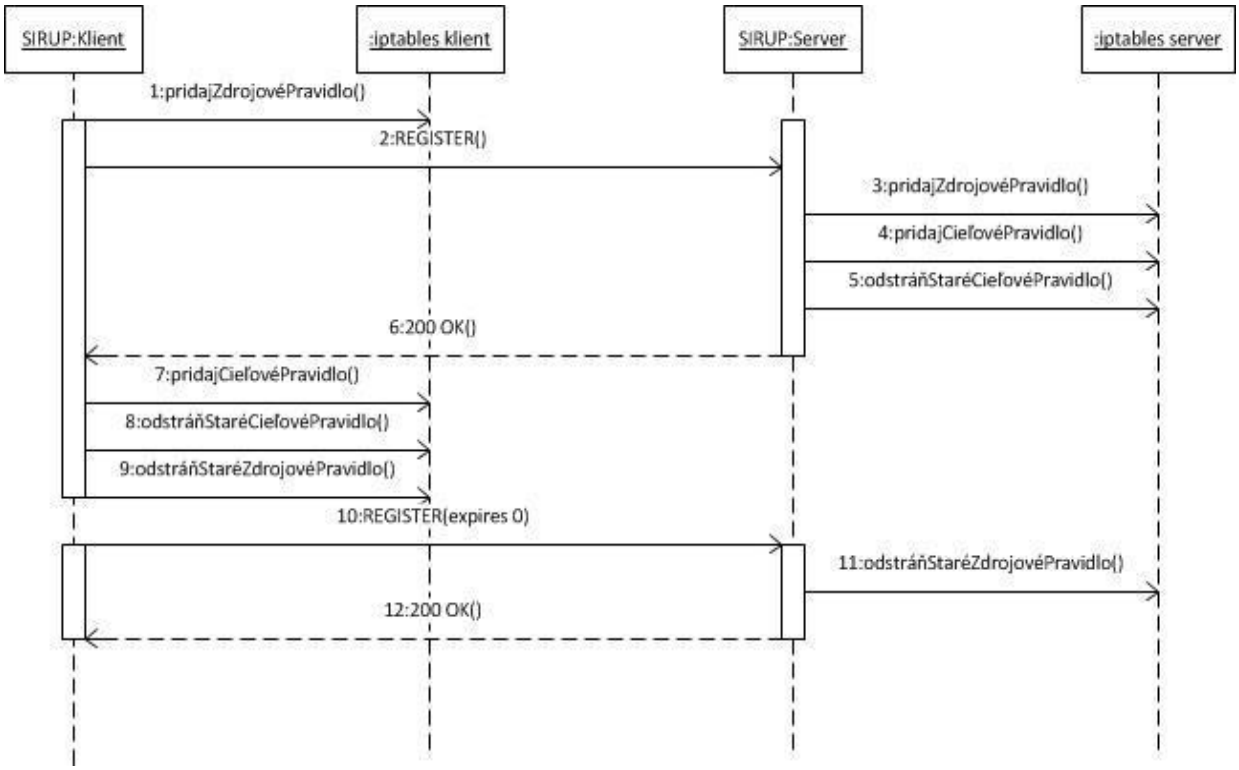
5.6 SIRUP

Ako je možné vidieť z architektúry znázornenej na obrázku 5 - 3, predpokladáme, že na našej strane sa bude nachádzať jedna serverová inštancia SIRUP - u (ďalej len server), ktorá bude fungovať na viacerých portoch. Jednotlivé porty pritom určujú MPLS tunely, cez ktoré vieme viesť jednotlivé relácie. Na opačnej strane tunelov sa budú nachádzať klientské inštancie SIRUP - u (ďalej len klienti). Komunikáciu manažmentovej časti s klientmi bude mať na starosti SIRUP Master.

5.7 Klient

Po úspešnom SUBSCRIBE by sa mal klient od mastera pomocou správy NOTIFY dozvedieť o adrese a porte servera, na ktorý sa má zaregistrovať. Tá prebieha štandardne pomocou správy REGISTER, pričom v hlavičke Contact sa musí nachádzať jeho pridelený identifikátor.

Tento jednoduchý REGISTER by postačoval v prípade, kedy aktuálne neprebieha žiaden hovor cez SIRUP - y. Úlohou projektu je však vedieť dynamicky reagovať na zmeny vo vyťažení liniek a na základe nameraných parametrov vedieť plynule presmerovať aktívne hovory cez inú linku. V prípade, že teda manažment rozhodne o jej zmene, posiela klient REGISTER na nový port servera. V tejto fáze už vie z akého portu môže očakávať po novom RTP pakety. Server po prijatí REGISTER správy nastavuje nové pravidlo s klientovou zdrojovou adresou a začína na ňu smerovať RTP pakety. Odpovedá správou 200 OK. Klient teraz môže tiež začať posielať pakety na nový port. Zároveň odstráni staré pravidlo ktoré riešilo preposielanie paketov z pôvodného serverového portu. Aby na serveri nezostávalo staré pravidlo s klientovou pôvodnou adresou a portom posielame REGISTER s Expires hodnotou nastavenou na 0 tak, aby sa z pôvodného portu odhlásil. Preregistráciu klienta počas aktívneho hovoru môžeme vidieť na obrázku 5 – 3.



Obrázok 5-3 Preregistrácia klienta počas aktívneho hovoru

5.8 Server

Aktuálne je server naprogramovaný len ako jednoduchý preposielač so statickým nastavením klienta. Po novom bude musieť vedieť pracovať na viacerých portoch a tiež si musí vedieť udržiavať asociáciu aktuálne registrovaných klientov s týmito portami.

5.9 SIRUP Master

SIRUP Master bude programovaný v jazyku Java a bude využívať knižnice z vývojovej sady verzie 1.7 s niekoľkými ďalšími knižnicami. Knižnica Jain - SIP verzie 1.2 slúži na rýchle a jednoduché spracovanie SIP správ, json - lib na spracovanie JSON objektov. Pre komunikáciu s websocketmi sme našli viaceré alternatívy ako napríklad jetty - websockets alebo socket.io - java client. Aplikácia si bude udržiavať údaje o klientoch a serverov vo vlastných poliach, ak takéto skladovanie dát nebude postačujúce, použije sa knižnica mysql - connector pre komunikáciu s databázou.

Po spustení posiela správu HelloServer(), ktorou sa registruje voči manažmentu a pomocou nej tiež udržiava spojenie. Ďalšia úloha spočíva v meraní kvality linky. Namerané parametre musí pravidelne posielať na manažment pomocou správ QualitySet(). Na základe týchto správ potom manažment rozhodne o prípadnej potrebe zmene linky. Informuje o nej server pomocou správy ConnectionMove().

Po naštartovaní, aplikácia otvorí UDP port 5060 na rozhraní, pripojí sa na manažérske API pomocou websocketov a prihlási sa do architektúry so správou HelloMaster(). Z vlastnej databázy načíta aktuálne spojenia alebo si ich vyžiada od manažmentu správou ConnectionGetAll().

Master počúva na porte 5060 a čaká na klientov, aby sa s ním spojili. Ak dostane správu SUBSCRIBE, odpovie klientovi správou 200 OK. Túto informáciu v JSON objekte pošle na API cez websocket v správe ConnectionSet() a čaká na správu ConnectionMove() od manažment serveru. V tejto správe dostane SIRUP Master informáciu, kde sa má klient pripojiť. Túto informáciu pošle klientovi v SIP správe NOTIFY, ktorá sa potvrdzuje správou 200 OK. Správa ConnectionMove() slúži aj na zmenu cieľového bodu počas hovoru, ak dôjde k zahlteniu na niektorom tuneli.

6 Prototyp

6.1 SIRUP

V rámci prototypu SIRUP - ov sa riešil predovšetkým SIRUP - server. Jeho prototyp už funguje na viacerých portoch. Klientom umožňuje registráciu, zrušenie registrácie a vo výslednom efekte preregistráciu. Táto preregistrácia však zatiaľ funguje len mimo aktívneho hovoru, nakoľko v takom prípade je potrebné riešiť aj paralelný prístup klienta na dva porty. Okrem toho bolo uskutočnené nadviazanie základného spojenia medzi klientským SIRUP - om a SIRUP - om Master pomocou správ SUBSCRIBE a NOTIFY.

6.2 SIRUP Master

Statický bod v SIRUP architektúre, pomocou ktorého sa inicializujú spojenia. Klientsky sirup sa po spustení prihlási do siete pomocou správy SUBSCRIBE k tomuto bodu. SIRUP - Master túto informáciu prepošle pomocou JSON objektov k manažment serveru. Manažment server odpovie správou, kde sa ma klient pripojiť. Túto informáciu pošle SIRUP Master klientovi v NOTIFY správe. Klienti pomocou SUBSCRIBE správ oznamujú ich dostupnosť. Správy NOTIFY slúžia na modifikáciu spojenia medzi klientom a serverom.

SIRUP - Master dokáže spracovať prichádzajúce správy SUBSCRIBE a na základe časovačov vyhodnotiť dostupnosť klienta. Taktiež dokáže vytvárať a analyzovať JSON objekty, ale nedokáže ich zatiaľ preposlať k manažment serveru. Je schopný tiež vygenerovať náhodnú NOTIFY správu, ktorá zatiaľ neplní žiadnu funkcionalitu.

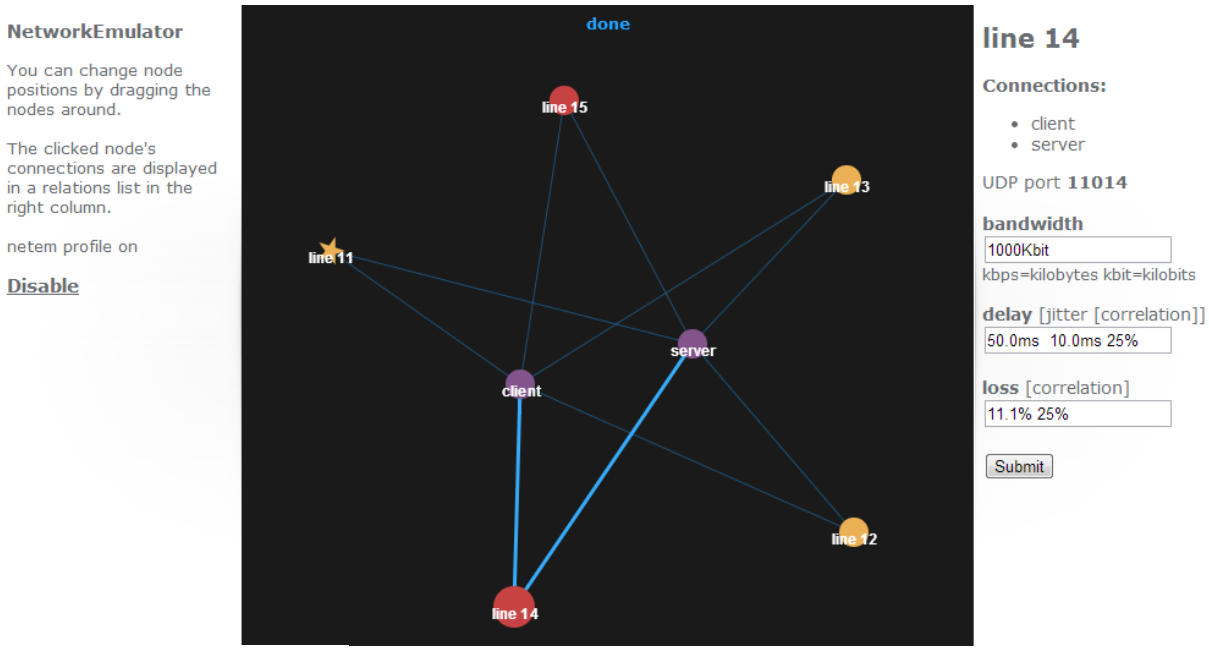
6.3 Simulácia siete NetEm

Implementácia prototypu pre simuláciu v siete vychádza z návrhu riešenia. Na zmenu parametrov tunelov sa používa nástroj NetEm, ktorý sa ovláda programom TC. Pre aktiváciu, deaktiváciu, zobrazenie a úpravu NetEm profilu sa používa skript v jazyku bash. Počet a čísla portov, ktoré sa použijú pre identifikáciu tunelov a sieťové rozhranie, na ktorom sa profil vytvorí nie je možné zmeniť pomocou vstupných argumentov. Tieto zmeny sa vykonávajú zmenou hodnôt premenných v úvodnej časti skriptu. Spustením skriptu s prepínačom „-h“ sa vypíše požadovaný tvar vstupných argumentov pre vykonanie operácií.

Pre ďalšie zvýšenie prehľadnosti a uľahčenie používania je vytvorené grafické používateľské rozhranie, ktoré môžeme vidieť na obrázku 6 - 1. Rozhranie je dostupné z webového prehliadača a vnútorne používa vyššie spomenutý skript, ktorý sa spúšťa pomocou PHP funkcie exec. Na vizualizáciu siete sa používa súbor nástrojov (toolkit) JavaScript InfoVis, ktorý sa používa pre interaktívnu vizualizáciu dát na webe. Implementácia rozhrania vychádza z ukážky dostupnej na stránke nástroja. V strednej časti rozhrania sa nachádza graf siete, ktorého veľkosť možno meniť otočením kolieska myši. Umiestnenie uzlov siete môže používateľ meniť ich ťahaním. Po kliknutí na uzol sa tento zvýrazní a zvýraznia sa aj jeho prepojenia s inými uzlami. Zároveň sa v pravej časti

zobrazia rozširujúce informácie, ako napríklad názov uzla a zoznam priamo pripojených uzlov. Pre uzly, ktoré reprezentujú sieťové tunely, sa zobrazia aj parametre tunelov. Po zobrazení parametrov možno ich hodnoty zmeniť a zmenu vykonať stlačením tlačidla v dolnej časti formuláru. Stlačením tlačidla sa všetky potrebné údaje odošlú na server, kde sa po spracovaní použijú ako vstupný argument skriptu. V ľavej časti rozhrania sa nachádza stručný popis rozhrania a tlačidlo, ktoré aktivuje alebo deaktivuje simulátor.

Obrázok 6-1 Grafické používateľské rozhranie nástroja NetEm



6.4 Manažment API

V rámci prototypu manažment serveru sa podarilo vytvoriť a sfunkčniť Node.js aplikáciu, ktorá sa podľa návrhu má starať o dostupnosť aplikačného rozhrania pre všetky časti systému, ktoré to vyžadujú. Prototyp momentálne neobsahuje všetky správy, definované v časti špecifikácia, no rozhodli sme sa implementovať len tie, ktoré potrebujeme pre simuláciu merania kvality na linkách a prenos týchto informácií k zobrazovacej/rozhodovacej časti.

Prototyp sa podľa návrhu spolieha na JavaScript knižnicu Socket.io, ktorá sa stará o komunikáciu jednotlivých častí.

V rámci správ sú implementované tieto:

- HelloServer(id,lines)

  Obsahom tejto správy je jedinečný identifikátor SIRUP - servera a objekt, ktorý popisuje stav pripojených liniek (ich počet, identifikátor linky, meno linky, IP adresy a porty koncových uzlov, maximálna šírka pásma).
- HelloHandler(id)

  Obsahom správy je jedinečný identifikátor servera.
- QualitySet(lines)

Správa obsahujúca informácie o kvalite liniek prípojných k SIRUP - serveru (celková kvalita linky, packet loss, round trip time, jitter a bandwidth).

- LinesSet(lines)

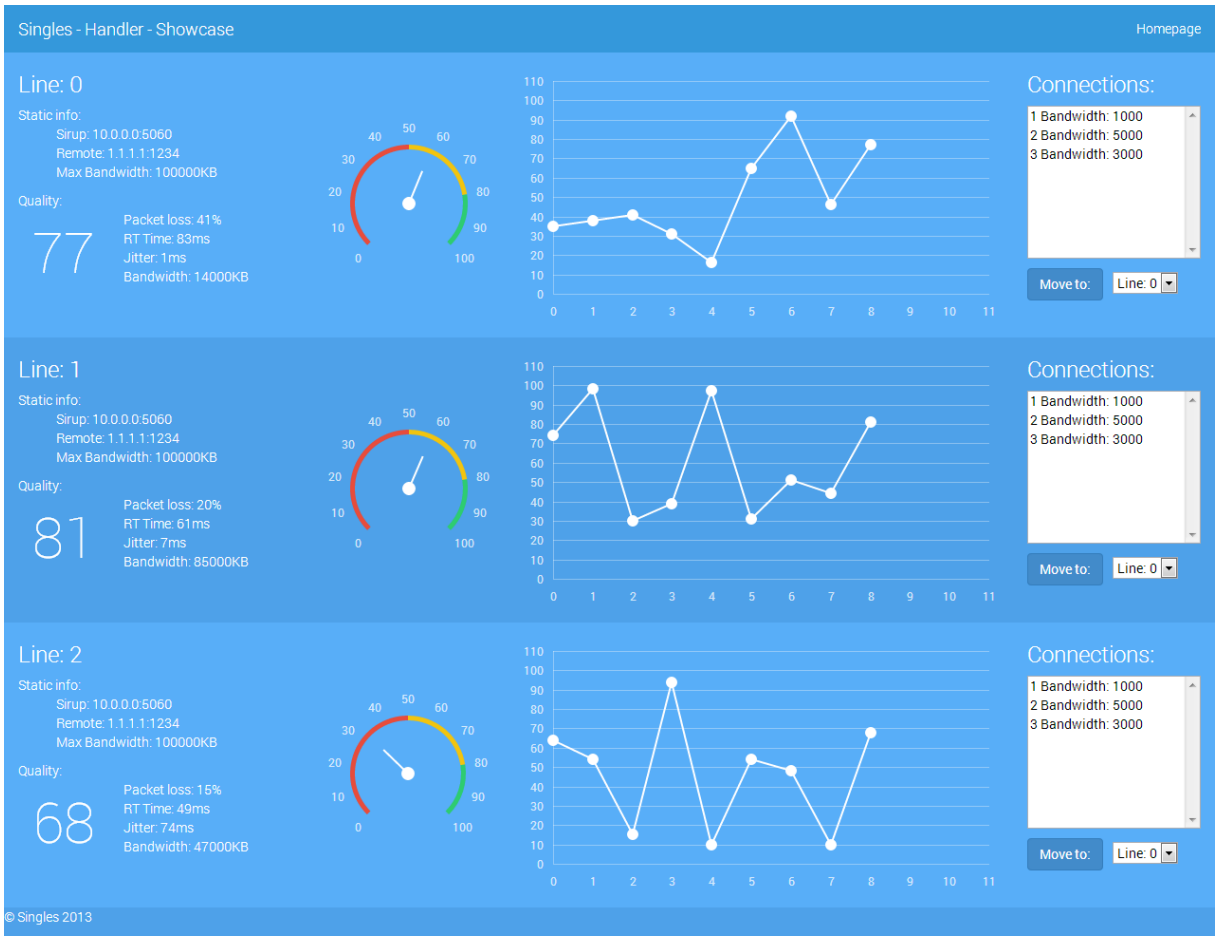
  Správa, ktorá nebola uvedená v špecifikácií, no ukázalo sa, že je vhodná. Manažment server ju posiela zobrazovacej/rozhodovacej jednotke vždy po jej prihlásení (HelloHandler), alebo keď sa zmení stav liniek u SIRUP - servera.

Zobrazovacia/rozhodovacia jednotka

Ďalej bol vytvorený prototyp zobrazovacej/rozhodovacej jednotky. Tak ako bolo špecifikované v návrhu špecifikácií, prototyp má formu webovej aplikácie. Táto aplikácia je schopná prihlásenia sa k manažment serveru a zobrazovania informácií o linkách a kvalite na týchto linkách.

Informácie sú zobrazované v dvojakej forme a to pomocou grafov a textu. V rámci návrhu sme sa rozhodli na zobrazovanie použiť knižnicu ChartJS a tá je použitá aj v prototype. Používajú sa 2 typy grafov a to tzv. „gauge“ (budík na palubnej doske), ktorý prehľadne zobrazuje informáciu o celkovej kvalite linky. Druhý je klasický dvojosový graf, kde sa zobrazuje okrem okamžitej celkovej kvality, aj niekoľko informácií historicky zaznamenaných. Ďalšie informácie o kvalite a statické informácie o linkách sa zobrazujú vo forme textu.

Nefunkčný náhľad ponúka časť „connections“, ktorá bude zobrazovať spojenia na jednotlivých linkách so šírkou pásma, ktoré zaberajú. V rámci tejto časti sa bude ovládať presúvanie spojení na jednotlivé linky. Náhľad rozhrania zobrazovacej / rozhodovacej jednotky je na obrázku 6 – 2.



Obrázok 6-2 Zobrazovacia/rozhodovacia jednotka

V rámci prototypu nie je vytvorené prepojenie medzi SIRUP - serverom a manažment serverom. Aby bolo možné simulovanie zasielania informácií o linkách a ich kvalite, bola vytvorená webová aplikácia, ktorá supluje túto časť funkcionality SIRUP - serveru. Webová aplikácie zasielané dáta pseudonáhodne vytvára a vo finálnej implementácií nebude použitá.

7 Implementácia

7.1 Manažment

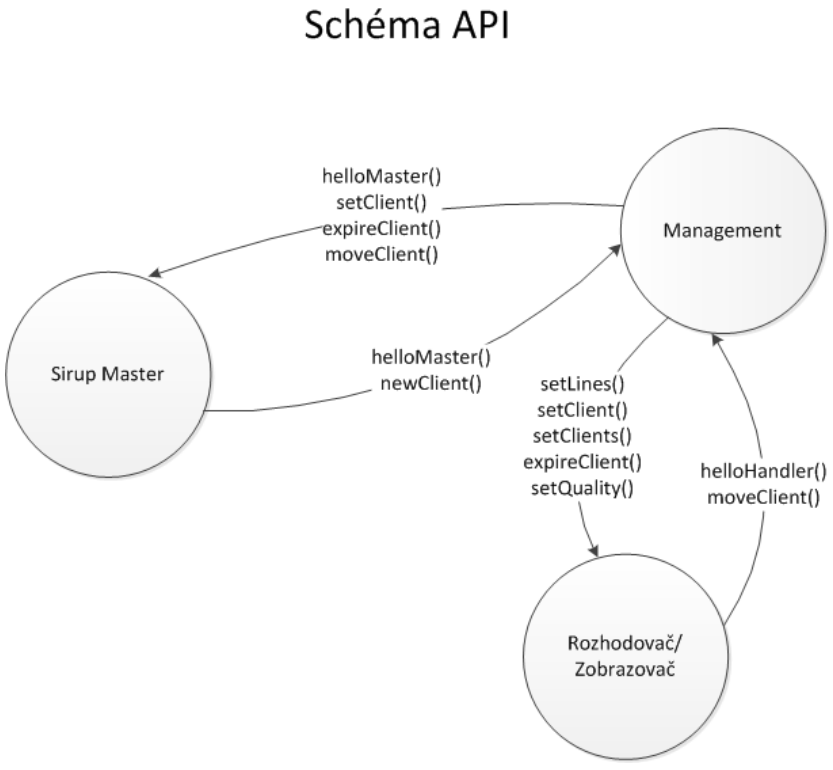
Manažment server priamo vychádza z prototypu zo zimného semestra. Prototyp bol však výrazne prepísaný, čo zvýšilo jeho výkonnosť aj robustnosť. Použitie platformy Socket.io sa ukázalo ako dobrá voľba, pretože nenastal v tomto smere žiaden problém s implementáciou požadovaných správ. Takisto nenastal žiaden problém s implementáciu v rámci ostatných častí systému. Správy obsahovali údaje vo forme JSON objektov.

Použitie platformy Node.js a jazyku Javascript sa ukázalo tiež ako dobrá voľba, pretože systém sa rýchlo a jednoducho programoval, pričom bolo možné splniť požadovanú funkcionalitu.

Oproti prototypu sa výrazne zlepšili ladiace výpisy a tak je možné rýchlejšie a jednoduchšie zistiť stav aplikácie a zároveň odhaliť prípadnú chybu. Manažment podporuje niekoľko stupňov (1 - 5) ladiacich výpisov. Pomocou týchto stupňov je možné nastaviť množstvo výpisov do konzolového okna.

7.1.1 Revízia manažment API správ

V procese implementácie sa ukázalo, že je nutná revízia správ v rámci manažment API. Implementované boli správy, ktoré sú bezprostredne potrebné pre fungujúce riešenie. Momentálna schéma správ (obrázok 7 - 1) je plne postačujúca, no dbalo sa na to, aby bolo možné túto schému jednoducho rozšíriť v prípade potreby v budúcnosti.



Obrázok 7-1 Schéma API správ

Správy spojené s pripojením uzlov architektúry:

- **helloMaster()** - Prihlásenie sa Sirup Mastera.
- **helloHandler()** - Prihlásenie sa zobrazovacej/rozhodovacej jednoty.

Správy spojené so správou klientov:

- **newClient()** - Oznámenie nového klienta.
- **setClient()** - Nastavenie parametrov klienta.
- **setClients()** - Nastavenie parametrov viacerých klientov.
- **moveClient()** - Presun klienta na inú linku.
- **expireClient()** - Odhlásenie klienta.

Ostatné správy:

- **setLines()** - Oznam parametrov liniek.
- **setQuality()** - Oznam o kvalite liniek.

Detailný popis API správ sa nachádza v súbore „models.js“, ktorý sa nachádza v adresári so zdrojovým kódom manažment serveru.

7.2 Zobrazovacia / rozhodovacia jednotka

Zobrazovacia/rozhodovacia jednotka oproti prototypu prešla výraznou zmenou. Zmenené bolo používateľské prostredie, ktoré sa sústredí na manipuláciu klientov medzi linkami. Presun klienta z jednej linky na druhú sa teraz vykonáva jednoduchým gestom myšou, potiahni a pusti (Drag and Drop). To znamená, že je možné chytiť myšou nápis klienta v zozname klientov na konkrétnej linke a presunúť ho na inú linku.

Z pohľadu implementácie zobrazovacej/rozhodovacej jednotky sa oproti prototypu veľa zmenilo. V prototype bola logická časť naprogramovaná pomocou knižnice jQuery, no vo výslednej implementácií sme sa rozhodli použiť knižnicu Angular.js. Angular.js je knižnica špecializovaná na dynamické webové aplikácie. Angular.js využíva Model – View – Controller programovací vzor a umožňuje tak vytvárať dobre štruktúrované webové aplikácie, ktoré sa dobre ladia a udržujú.

Model reprezentujú Javascript objekty, v ktorých sú uskladnené potrebné dáta o linkách atď. View je zabezpečené pomocou jazyka HTML, ktorý je rozšírený o nové štruktúry knižnice Angular.js. Logickú časť tvoria tzv. „angular controllers“, kde je naprogramovaná celá logika potrebná pre chod aplikácie.

Ďalšou zmenou oproti prototypu je výmena použitej knižnice na vykresľovanie grafov. V prototype bola použitá knižnica ChartJS, no vo výslednej aplikácia sa použila knižnica Highcharts. Bolo to z dôvodu lepšej podpory pre Angular.js, keďže funkcionalitu majú veľmi podobnú.

Rozhranie zobrazovacej / rozhodovacej jednotky môžeme vidieť na obrázku 7 – 2.

Obrázok 7-2 Rozhranie zobrazovacej / rozhodovacej jednotky

7.3 SIRUP server a klient

Implementácia SIRUP servera aj klienta realizuje vytvorený návrh. Zatiaľ čo prototyp bol schopný prehadzovať tunely len mimo prebiehajúceho hovoru, vo výslednej implementácii sa to podarilo aj počas neho. Vďaka navrhnutému postupu sa prehadzovanie podarilo implementovať bez straty paketu a bez pozorovateľného prerušenia.

Ďalej bola implementovaná komunikácia medzi SIRUP klientmi a Masterom pomocou správ SUBSCRIBE a NOTIFY. Vzhľadom na minimalistický obsah správy NOTIFY bol ako typ obsahu (Content-type) zvolený text/plain (prenáša sa len číslo portu, ktorý má klient používať pri komunikácii so serverom).

Riešenie obsahuje viacero menších vylepšení v podobe spúšťacieho skriptu, ktorý na začiatku overí aktuálne nastavenia systému potrebné pre spustenie SIRUP - u a taktiež vyčistí staré pravidlá, ktoré mohli zostať po nekorektne ukončených hovoroch.

Taktiež bol pozmenený reťazec vkladaný do RTP paketov tak, aby mal konštantnú dĺžku (z Call-ID sa vypočíta MD5 hash) a bola tak zjednodušená identifikácia RTP paketov z dôvodu merania kvality linky.

7.4 SIRUP Master

SIRUP Master tvorí rozhranie medzi svetom SIP a manažmentom. Pozostáva z rôznych častí. Vrstva SIP je zodpovedná za spracovanie správ protokolu SIP, ktoré sú prijaté a odoslané k a od klientského SIRUP - u. Vrstva API je zodpovedná za druhú stranu, ktorá si vymieňa JSON objekty medzi SIRUP Master - om a manažmentom infraštruktúry.

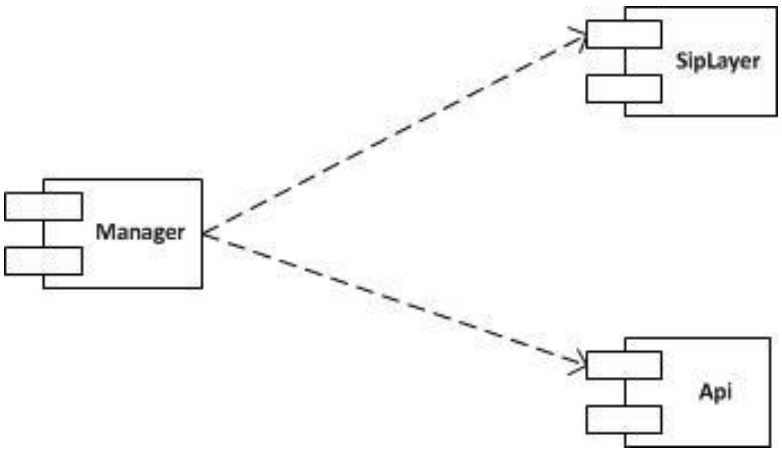
Po spustení aplikácia sa pripojí k manažmentu a otvorí socket, na ktorom počúva a očakáva SIP správy. K manažmentu sa prihlási pomocou správy helloMaster.

Ak sa klient zaregistruje pomocou správy SUBSCRIBE, pošle manažmentu správu newClient, ktorý odpovie správou setClient s uvedeným portom, na ktorý sa má pripojiť SIRUP klient. Táto správa ja klientovi poslaná v SIP správe NOTIFY. Klient a SIRUP Master komunikujú správami SUBSCRIBE a NOTIFY.

Medzi manažmentom a SIRUP Master - om sa môžu vyskytnúť tieto správy:

- **helloMaster** – inicializácia spojenia
- **newClient** – informácia o novom klientovi
- **setClient** – informácia, kde sa má klient pripojiť
- **moveClient** – informácia, kde sa má klient preregistrovať
- **clientExpired** – informácia, keď vypršal odber služby (Subscribe)

Hlavné časti aplikácie pomocou diagramu súčastí sú na obrázku 7 – 3.



Obrázok 7-3 Diagram súčastí SIRUP Master

7.5 Simulácia siete NetEm

Implementácia nástroja pre simuláciu sieti vychádza z návrhu riešenia. Na zmenu parametrov tunelov sa používa nástroj NetEm, ktorý sa ovláda programom *tc*. Pre aktiváciu, deaktiváciu, zobrazenie a úpravu NetEm profilu sa používa skript v jazyku bash. Počet a čísla portov, ktoré sa použijú pre identifikáciu tunelov pri aktivácií profilu a sieťové rozhranie, na ktorom sa profil vytvorí, nie je možné zmeniť pomocou vstupných argumentov.

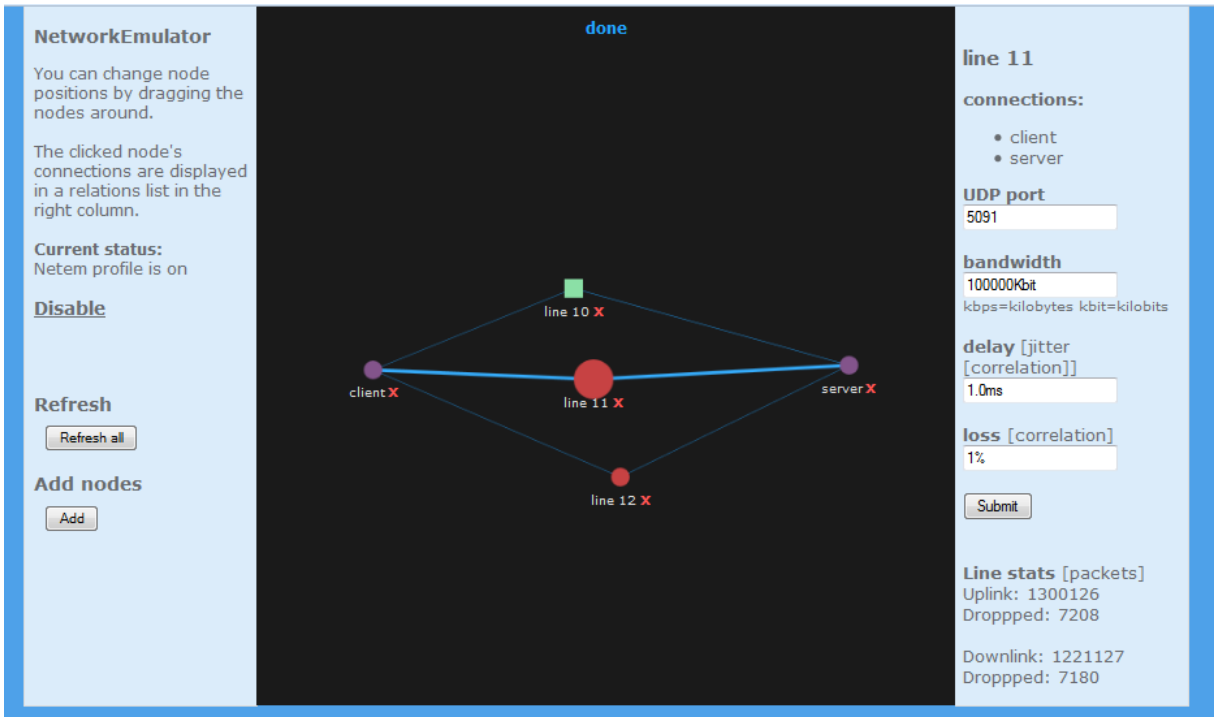
Tieto zmeny sa vykonávajú zmenou hodnôt premenných v úvodnej časti skriptu. Spustením skriptu s prepínačom „-h“ sa vypíše požadovaný tvar vstupných argumentov pre vykonanie operácií.

Pre ďalšie zvýšenie prehľadnosti a uľahčenie používania je vytvorené grafické používateľské rozhranie. Rozhranie je dostupné z webového prehliadača a vnútorne používa vyššie spomenutý skript, ktorý sa spúšťa pomocou PHP funkcie exec. Na vizualizáciu siete sa používa súbor nástrojov (toolkit) JavaScript InfoVis, ktorý sa používa pre interaktívnu vizualizáciu dát na webe. Implementácia rozhrania vychádza z ukážky ForceDirected dostupnej na stránke nástroja. V strednej časti rozhrania sa nachádza graf siete, ktorého veľkosť možno meniť otočením kolieska myši. Umiestnenie uzlov siete môže používateľ meniť ich ťahaním. Po kliknutí na uzol sa tento zvýrazní a zvýraznia sa aj jeho prepojenia s inými uzlami. Zároveň sa v pravej časti zobrazia rozširujúce informácie, ako napríklad názov uzla a zoznam priamo pripojených uzlov. Pre uzly, ktoré reprezentujú sieťové tunely, sa zobrazia aj parametre tunelov a štatistiky obsahujúce počty odoslaných paketov. Po zobrazení parametrov možno ich hodnoty zmeniť a zmenu vykonať stlačením tlačidla v dolnej časti formuláru. Stlačením tlačidla sa všetky potrebné údaje odošlú na server, kde sa po spracovaní použijú ako vstupný argument skriptu. V ľavej časti rozhrania sa nachádza stručný popis rozhrania a tlačidlo, ktoré aktivuje alebo deaktivuje simulátor. Ďalšie tlačidlá v tejto časti u slúžia na obnovenie obsahu stránky a pridanie nového uzla to topológie.

Pridávanie tunelov sa vykonáva pomocou tlačidla Add. Po stlačení sa do topológie pridá nový uzol a spustí sa pre usporiadanie uzlov, aby nedošlo k vytvoreniu nových uzlov v zákryte. Číslo UDP portu nového uzla sa určí ako nasledovník najväčšieho používaného čísla portu v aktuálnej konfigurácií.

$$port[N+1] = \max(port[1..N]) + 1$$

Odstránenie tunelov sa vykonáva kliknutím na červený krížik vedľa mena tunelu. Okrem odstránenia z grafického rozhrania sa vykoná aj zrušenie príslušnej konfigurácie na sieťových rozhraniach. Nie je možné odstrániť predvolený tunel (line 10 – Default class). Deaktivácia tvarovania tejto komunikácie sa vykoná vhodným nastavením parametrov tunelu.



Obrázok 7-4 Grafické používateľské rozhranie nástroja NetEm

Pre účely overenia a testovania riešenia boli vytvorené skripty stats.sh a statsCurrent.sh. Systém ich pre svoju funkčnosť nevyžaduje. Možno ich použiť na zobrazenie štatistických informácií o aktuálnom vyťažení tunelov. Výstup je prehľadne organizovaný, čo umožňuje získané údaje priamo použiť na vykreslenie grafov. Princíp ich fungovania je zhodný, líšia sa vo formáte výstupu.

Program *tc* umožňuje zobrazenie počtu bajtov a paketov, ktoré boli zaradené do tried na sieťovom rozhraní. Keďže tieto triedy reprezentujú tunely, súčtom hodnôt počtu odoslaných bajtov na oboch rozhraniach tunelu možno zistiť celkový objem dát prenesených sieťovým tunelom. Skripty na zobrazovanie štatistických informácií merajú celkový počet prenesených dát v pravidelných časových intervaloch (1 sekunda) a vypočítaním ich rozdielu dokážu určiť aktuálnu dátovú rýchlosť. Skripty obsahujú v úvode jednu konfigurovateľnú premennú INTERFACE, ktorou sú určené rozhrania s aktívnym NetEm profilom.

Ukážka výstupu programu `stats.sh`:

```
Stats are shown per line in KB/s (combined transmit and receive speed)
line_10     line_11     line_12
0           5091        5092
12          33          65          08:58:25,579191948
14          32          67          08:58:26,722829082
```

Výstup je organizovaný do formátu tabuľky. Stĺpce sú oddelené tabulátorom. Počet stĺpcov zodpovedá počtu tunelov a jednému stĺpcu pre uloženie časovej pečiatky

ukončenia merania. Stĺpec začína názvom tunela a číslom portu tunela. Číslo portu 0 označuje predvolený tunel (default class), čiže doplnok ostatných definovaných portov. Počet riadkov výstupnej tabuľky narastá každým meraním.

Ukážka výstupu programu statsCurrent.sh:

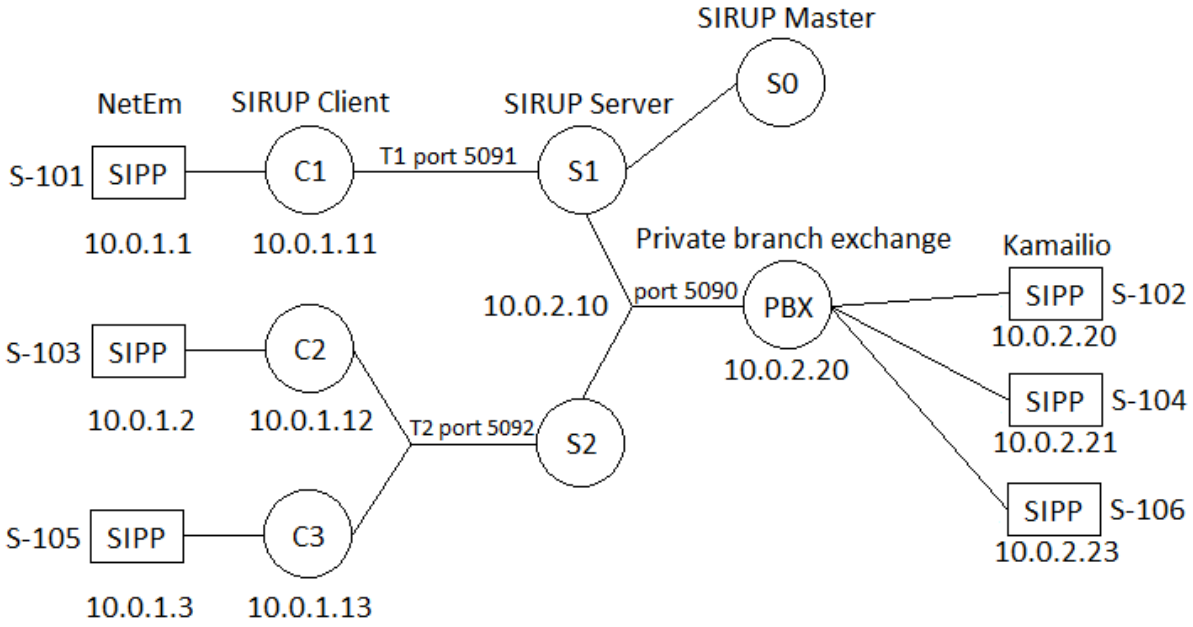
```
./statsCurrent.sh /tmp/lineStat.tmp
0               12      100000Kbit delay 3.0ms loss 1% 25%
5091            33      100000Kbit delay 1.0ms loss 1%
5092            65      8bit loss 1%
```

Výstup je organizovaný do formátu tabuľky. Stĺpce sú oddelené tabulátorom. Počet stĺpcov je pevný. Počet riadkov zodpovedá počtu definovaných tunelov. Riadok začína číslom portu tunela. Číslo portu 0 označuje predvolený tunel (default class), čiže doplnok ostatných definovaných portov. Druhý stĺpec tabuľky obsahuje aktuálnu dátovú rýchlosť komunikácie prechádzajúcej tunelom. Posledný stĺpec obsahuje aktuálnu NetEm konfiguráciu. Počet riadkov výstupnej tabuľky sa počas behu program nemení. Namerané hodnoty sa zobrazujú na štandardnom výstupe a tiež sa ukladajú do súboru, ktorý je vstupným argumentom. Súbor obsahuje v každom čase iba výsledky posledného merania.

8 Testovanie

V rámci tejto kapitoly si definujeme a popíšeme 5 základných testovacích scenárov, ktoré sú dostatočné na overenie funkčnosti systému na riadenie VoIP relácií. Výsledky zaznamenáme do prehľadných grafov a podrobne vysvetlíme postup pri testovaní.

Na začiatok testovania je potrebné sa oboznámiť s testovacou schémou, ktorú vidíme na obrázku 7-1. Schéma je rovnaká pre všetky testovacie scenáre.



Obrázok 8-1 Schéma testovania

Na zariadení, kde beží program NetEm sme umiestnili a nainštalovali nástroj na testovanie SIPp. Dokážeme ním simulovať jednotlivé SIP hovory a obsahuje mnoho nastavení a vlastností potrebných pre testovanie nášho systému. Aby sme dostatočne overili systém potrebujeme 3 inštancie nástroja SIPp, ktoré rozlíšime IP adresou z ktorej iniciujeme hovor a atribútom meno (S). Dôvodom pre inštaláciu nástroja SIPp na zariadenie, kde beží program NetEm, je nevyťažovanie zariadenia, kde sú spustení SIRUP klienti. V hovore preto vystupujeme ako SIRUP klient s danou IP adresou ako vidíme na obrázku, no reálne nástroj funguje na zariadení NetEm.

Pri testovaní potrebujeme 3 klientov C1, C2 a C3 a budeme využívať 2 linky T1, ktorá využíva port 5091 a T2 s portom 5092. Ako ukazuje obrázok 7-1, potrebujeme tiež 2 SIRUP Servery, ktoré bežia na rovnakom stroji a PBX, ktorý slúži ako telefónna ústredňa. Komunikácia medzi SIRUP servermi a PBX prebieha na porte 5090. Na ústredni je nainštalovaný tiež nástroj SIPp a SIP Server Kamailio. IP adresy klientov na strane PBX vidíme na obrázku, ako aj ich atribút meno.

Príklad príkazu na spustenie nástroja SIPp, ktorý je už pripravený na testovanie konkrétne našej topológie. Prvá časť príkazu slúži na registráciu klienta, druhá časť vytvára hovor a umožňuje tiež posielať RTP pakety:

- *./sipp -sf registration.xml -inf auth.csv -l 1 -m 1 -aa -i 10.0.1.1 -bind_local 10.0.1.1 10.0.1.11:5090 &&*
- *./sipp -sf source_caller.xml -inf auth.csv -trace_stat -trace_err -aa -r 1 -l 1 -s 102 -t u1 -mp 6001 -mi 10.0.1.1 -i 10.0.1.1 -bind_local 10.0.1.1 10.0.1.11:5090*

Xml súbory používané v príkazoch obsahujú SIP správy potrebné pre registráciu a tiež vytvorenie hovoru. Csv súbor obsahuje informácie o klientovi, konkrétne atribúty meno, heslo a IP adresu ústredne PBX.

8.1 1. scenár testovania

Schému, na ktorej budeme testovať jednotlivé scenáre, sme si popísali. Teraz sa pozrieme na fázy a jednotlivé kroky 1. testovaného scenára:

Fáza 1 – vytvorenie hovorov a priradenie linkám

- Vytvoríme hovor klienta C1 na linke T1 medzi S-101 a S-102 v čase t = 97 s
- Vytvoríme hovor klienta C2 na linke T2 medzi S-103 a S-104 v čase t = 174 s
- Vytvoríme hovor klienta C2 na linke T2 medzi S-103 a S-104 v čase t = 219 s
- Vytvoríme hovor klienta C3 na linke T2 medzi S-105 a S-106 v čase t = 271 s

Po fáze 1 máme na linke T1 celkovo 1 hovor a na linke T2 prebiehajú hovory 3.

Fáza 2 – presun hovorov

- Presunieme hovor z linky T2 klienta C3 na linku T1 v čase t = 322 s

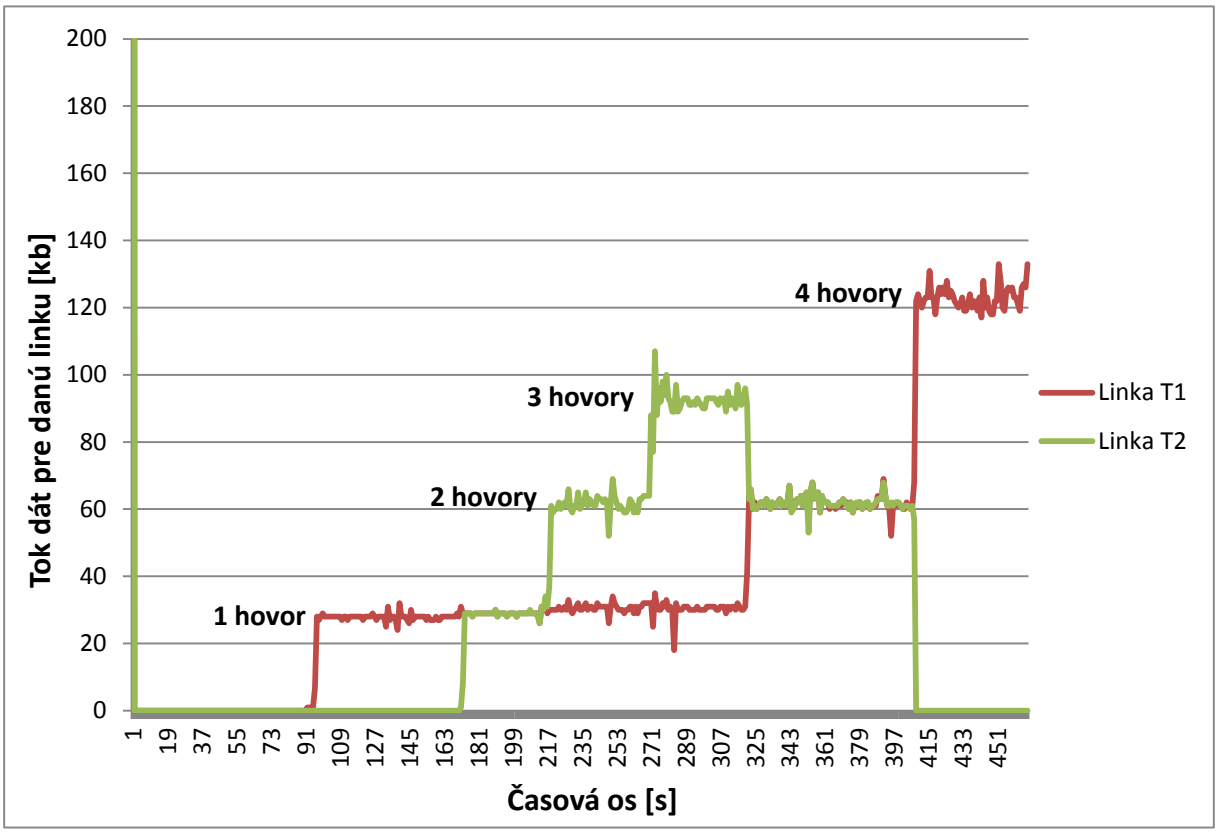
Po tomto kroku budú aktuálne na linke T1 prebiehať 2 hovory, rovnako aj na linke T2.

- Presunieme 2 hovory z linky T2 klienta C2 na linku T1 v čase t = 409 s

Na konci druhej fázy vidíme, že na linke T1 teraz máme všetky hovory, konkrétne 4 a linkou T2 neprechádza žiadny hovor.

V ďalšej fáze testovacieho scenára sa pozrieme na výsledný graf. Údaje použité v grafe boli zachytené nástrojom NetEm, ktorým sme merali veľkosť toku dát na linkách T1 a T2. Na časovej osi sú zobrazené čísla, podľa postupnosti pridávania hovorov a určuje čas plynutia pridávania. Zvislá os určuje počet hovorov na jednotlivých linkách. Keďže nástroj nevie rozlíšiť počet hovorov, vie nám ale povedať o veľkosti dátového toku a z neho vieme, že 1 hovor zaberá zhruba 30 - 33 kilobajtov. Z tohto čísla vieme ľahko odvodiť, že 2 hovory budú mať okolo 63 kb, 3 hovory okolo 95 kb a 4 hovory zhruba 128 kb. V grafe je tiež zobrazený popis k jednotlivým linkám o počte hovorov pre daný dátový tok.

Z grafu 7 - 1 vyplýva, ako sa jednotlivé hovory presúvali medzi linkami, čo je cieľom nášho systému a dosahuje požadované výsledky. V grafe je vyznačený počet hovorov na danej úrovni dátového toku, čo značí počet hovorov na danej úrovni v celom grafe.



Graf 8-1 Testovanie zmeny liniek pri prebiehajúcom hovore

8.2 2. scenár testovania

V druhom scenári testovania sme využili rovnakú schému ako vidíme na obrázku 7 – 1. Táto schéma je dostatočná aj pre potreby druhého testovacieho scenára. Základom je viacnásobné prepínanie hovorov a otočenie počtu hovorov na jednotlivých linkách. V nasledujúcej časti sa pozrieme na jednotlivé fázy a kroky testovania:

Fáza 1 – vytvorenie hovorov a priradenie linkám

- Vytvoríme hovor klienta C3 na linke T2 medzi S-105 a S-106 v čase t = 11 s
- Vytvoríme hovor klienta C3 na linke T2 medzi S-105 a S-106 v čase t = 18 s
- Vytvoríme hovor klienta C1 na linke T1 medzi S-101 a S-102 v čase t = 22 s
- Vytvoríme hovor klienta C2 na linke T1 medzi S-103 a S-104 v čase t = 26 s
- Vytvoríme hovor klienta C2 na linke T1 medzi S-103 a S-104 v čase t = 30 s
- Vytvoríme hovor klienta C2 na linke T1 medzi S-103 a S-104 v čase t = 35 s

Po fáze 1 máme na linke T1 celkovo 4 hovory a na linke T2 prebiehajú hovory 2.

Fáza 2 – presun hovorov

- Presunieme hovor z linky T1 klienta C1 na linku T2 v čase t = 40 s

Po tomto kroku budú aktuálne na linke T1 prebiehať 3 hovory, rovnako aj na linke T2.

Fáza 3 – zníženie počtu hovorov

- Počkáme, kým skončia 2 hovory na klientovi C2. Dĺžka hovoru je nastavená nástrojom SIPp na 35 sekúnd.

Na konci tretej fázy vidíme, že na linke T1 teraz máme 1 bežiaci hovor a na linke T2 sú 3 hovory.

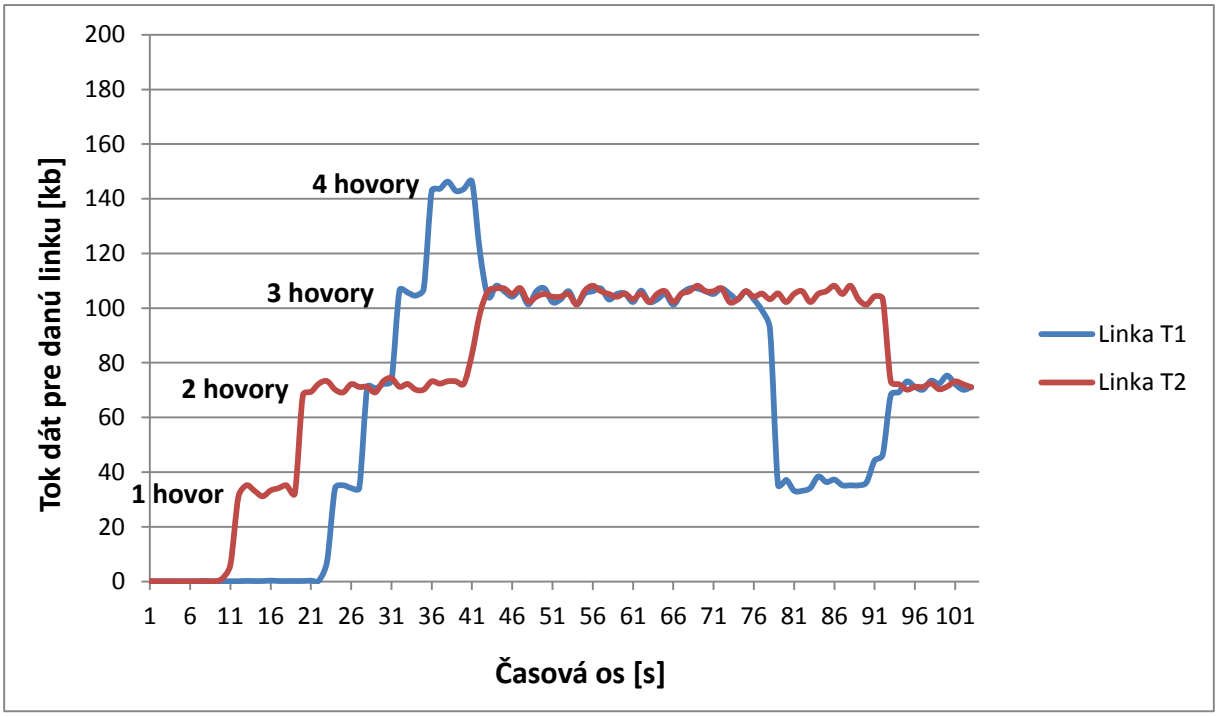
Fáza 4 – presun hovorov

- Presunieme hovor z linky T2 klienta C1 na linku T1 v čase t = 91 s

Na konci celého testovania teda dostávame na linke T1 2 hovory, rovnako aj na linke T1, čo bolo cieľom nášho testovania.

V ďalšej fáze testovacieho scenára sa pozrieme na výsledný graf. Údaje použité v grafe boli zachytené nástrojom NetEm, ktorým sme merali veľkosť toku dát na linkách T1 a T2. Všetky hodnoty a merania prebiehali rovnako ako v predchádzajúcom scenári.

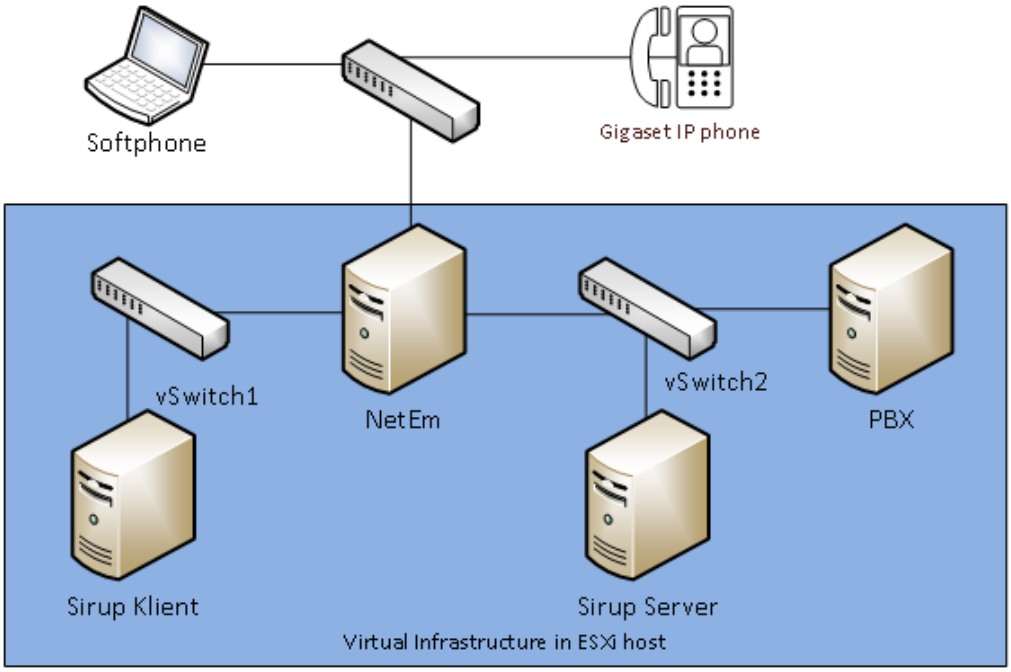
Z grafu 7 - 2 vyplýva, ako sa jednotlivé hovory presúvali medzi linkami, čo je cieľom nášho systému a dosahuje požadované výsledky. V grafe je vyznačený počet hovorov na danej úrovni dátového toku, čo značí počet hovorov na danej úrovni v celom grafe.



Graf 8-2 Testovanie zmeny liniek pri prebiehajúcom hovore

3. scenár testovania

V ďalšom testovacom scenári sme testovali náš systém s využitím reálnych telefónov. Predtým sme využívali nástroj SIPp, ktorý len simuluje reálny telefónny hovor na našich linkách. Aby sme teda lepšie overili fungovanie, a čo najlepšie simulovali reálne podmienky, vytvorili sme topológiu, kde sme pripojili Softphone a Gigaset IP phone. Testovacia schéma je na obrázku 7-2.



Obrázok 8-2 Testovacia schéma s reálnymi telefónmi

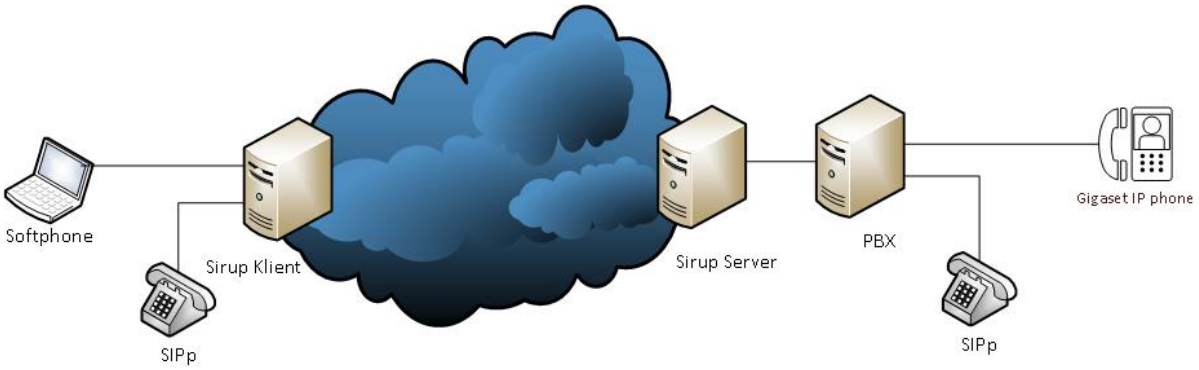
Cieľom testovacieho scenára bolo overiť fungovanie prepínania aj pri použití reálnych zariadení na telefonovanie. Celý scenár prebiehal podobne ako pri scenár popisovaných predtým. Kroky pri testovaní:

- Vytvorenie hovoru pomocou reálnych telefónov a našich zariadení
- Hovor na začiatku testovania prebiehal na linke T1
- Prepnutie reálneho hovoru na linku T2

Po prepnutí hovoru sme sledovali, či hovor bežal ďalej. Všetko fungovalo ako sme potrebovali. Taktiež sme počúvali na telefónoch, či vznikne hluché miesto pri prepnutí linky. Žiadne hluché miesto sa neobjavilo a prepínanie hovoru prebehlo bez výpadku na jednej či druhej strane.

4. scenár testovania

V štvrtom testovacom scenári sme sa zamerali hlavne na využitie nástroja NetEm a možnosti jeho nastavenia. Predtým sme nevyužívali obmedzenia na linkách, ktoré nám umožňuje NetEm nastaviť. V tomto scenári využijeme možnosť nastavenia oneskorenia na linke. A taktiež využijeme zaťaženie linky pomocou nástroja SIPp. Testovacia schéma je na obrázku 7 - 3.



Obrázok 8-3 Testovacia schéma s reálnymi telefónmi a SIPp

Cieľom testovacieho scenára bolo overiť fungovanie prepínania aj pri použití reálnych zariadení na telefonovanie spolu so simuláciou reálnych podmienok na sieti. Pri nastavení oneskorenia budeme sledovať kvalitu spojenia na reálnom hovore a čas oneskorenia slov povedaných pri hovore. Kroky pri testovaní:

- Vytvorenie hovoru pomocou reálnych telefónov a našich zariadení
- Vytvorenie 5 hovorov pomocou nástroja SIPp
- Všetky hovory na začiatku testovania prebiehajú na linke T1
- Nastavenie oneskorenia na linke T1
- Prepnutie reálneho hovoru na linku T2

Pri prvom nastavení oneskorenia sme zvolili nižšiu hodnotu, konkrétne 20 %. No oneskorenie pri reálnom hovore nebol až tak zreteľné, aby sme vyslovene počuli slová o niekoľko sekúnd neskôr. Preto sme oneskorenie nastavili na hodnotu 80 %. Táto hodnota bola dostatočná.

- Pri telefonovaní zo Softphone na Gigaset IP Phone bolo oneskorenie 8 s
- Oneskorenie v opačnom smere bolo 4 s

Kvalita hovoru bola zaznamenaná v súbore *netem_hovor_jan_pivno.pcap,* ktorý je súčasťou priloženého média. Po prepnutí hovoru na druhú linku, kde nebol žiadny hovor ani nebolo nastavené žiadne oneskorenie sa kvalita hovoru okamžite zlepšila. Nebolo počuť žiadne oneskorenie a kvality linky bola dokonalá, bez straty jediného slova.

8.3 5. scenár testovania

Posledný scenár je veľmi podobný ako predchádzajúci. Využívame všetko, čo bolo v ňom popísané no s jednou zmenou. Ňou je nastavenie parametru linky. Nenastavujeme oneskorenie, ale stratu paketov. Testovacia schéma je rovnaká ako na obrázku 7 – 3. Cieľom testovacieho scenára bolo overiť fungovanie prepínania aj pri použití reálnych zariadení na telefonovanie spolu so simuláciou reálnych podmienok na sieti. Pri nastavení straty paketov budeme sledovať kvalitu spojenia na reálnom hovore. Zopakujme si kroky pri testovaní:

- Vytvorenie hovoru pomocou reálnych telefónov a našich zariadení
- Vytvorenie 5 hovorov pomocou nástroja SIPp
- Všetky hovory na začiatku testovania prebiehajú na linke T1
- Nastavenie straty paktov na linke T1
- Prepnutie reálneho hovoru na linku T2

Pri prvom nastavení straty paketov sme zvolili nižšiu hodnotu, konkrétne 20 %. No stratovosť paketov pri reálnom hovore nebola až taká výrazná a slová pri hovore boli jasné a zrozumiteľné. Preto sme stratu paketov nastavili na hodnotu 70 %. Táto hodnota bola dostatočná a kvalita linky sa zhoršila natoľko, že nebolo možné rozpoznať slová. Bolo počuť mnoho hluchých miest a hovor bol nezrozumiteľný. Po prepnutí hovoru na druhú linku, kde nebol žiadny hovor ani nebola nastavená žiadna stratovosť paketov sa kvalita hovoru okamžite zlepšila. Hovor a slová boli okamžite zrozumiteľné a krásne sme mohli telefonovať ďalej bez problémov.

8.4 Testovanie manažmentu API

V rámci otestovania funkčnosti manažment API boli vytvorené automatické testy pomocou Node.js modulu Mocha. Mocha je Javascript framework, ktorý uľahčuje testovanie Node.js a webových aplikácií. Tieto testy sa môžu vykonávať priebežne počas vývoja a umožňujú verifikovať funkcionalitu bez nutnosti zapojiť celý systém.

Pomocou modulu Mocha bolo navrhnutých a implementovaných 5 základných testov funkcionality:

- Test pripojenia sa Sirup Master
- Test pripojenia sa nového klienta
- Test pripojenia sa zobrazovacej/rozhodovacej jednotky
- Test odosielania správ o kvalite liniek
- Test presunutia klienta na inú linku

Týchto 5 základných testov plne postačovalo na otestovanie funkčnosti manažment API počas vývoja. V prípade chyby ju bolo možné ihneď objaviť podľa výstupu testu, následne opraviť a otestovať opravu.

9 Záznam o používaní systému

V prvých krokoch tvorby celého projektu sme sa oboznamovali s jednotlivými prvkami, potrebnými na dosiahnutie cieľa. Prebiehala analýza existujúcich častí systému a možné doplnenie o požadovanú funkcionalitu. Z tejto analýzy vznikol návrh, ktorý bol opísaný v našom dokumente. Na základe skúseností jednotlivých členov sme si podelili funkcie a prácu na častiach projektu. Z tohto dôvodu vzniklo niekoľko častí, na ktorých sme pracovali samostatne. Boli to časti ako:

- SIRUP klient a server
- SIRUP Master
- Simulácia siete NetEm
- Manažment
- Práca s reálnou sieťou
- Testovací nástroj SIPp

Jednotlivé časti po individuálnej práci bolo potrebné spojiť do jedného funkčného celku. Tieto pokusy o spojenie prebiehali v mesiacoch marec a apríl. Postupne sme spájali jednotlivé časti a testovali ich vzájomnú funkčnosť. Pri spájaní sa objavilo niekoľko problémov. Hlavným bolo doladenie správ vymieňaných medzi jednotlivými časťami, na zabezpečenie komunikácie. Všetky problémy boli vyriešené.

V máji sa systém dokončoval a testovala sa funkčnosť jednotlivých častí samostatne a samozrejme aj početné testovania celého systému. Či už to bolo testovanie s využitím nástroja SIPp alebo pomocou reálnych telefónov. Testovanie prebiehalo skoro celý mesiac a funkčnosť bola dostatočne overená. Systém funguje podľa zadaných požiadaviek a spĺňa, ba až prevyšuje požadovanú funkcionalitu.

10 Čo sme sa naučili

Prácou v šesť člennom tíme sme získali cenné skúsenosti s fungovaním malej projektovej skupiny, pričom pre väčšinu členov tímu bola táto skúsenosť prvou tohto druhu. Dôležité bolo najmä dodržiavanie termínov, na rozdiel od práce na samostatných zadaniach v iných predmetoch, ak nebol dodržaný termín úlohy, od ktorej záviselo plnenie úloh ostatných členov tímu, museli sa posunúť termíny ďalších úloh. Tento fakt vytváral tlak na študentského vedúceho tímu, ktorého úlohou bolo vzniknuté problémy vyriešiť v najkratšom možnom čase. Dôraz bol kladený aj na presné špecifikovanie požiadaviek a jednoznačný návrh riešenia, aby zadefinované úlohy mohli byť riešené aj mimo stanoveného poradia v prípade, že bol niektorý z členov tímu práce neschopný. Implementačné problémy sme si odkonzultovali počas spoločných stretnutí, kde sme dospeli ku ich riešeniu v súlade s požiadavkami vedúceho tímu.

Tímový projekt rozšíril naše vedomosti o nástrojoch pre tímovú spoluprácu a ich efektívnom používaní. Aj keď na vývoj jednotlivých častí projektu nepripadal viac ako jeden vývojár, použitie nástroju Git malo zmysel v prípade potreby použitia predchádzajúcej verzie zdrojového kódu. Taktiež sme si prehĺbili poznatky o protokoloch SIP, SDP, RTP a RTCP. Ocenili sme rozhodnutie použiť formát správ JSON pre jeho jednoduchú implementáciu v časti Sirup Master a Manažment, ktoré nám uľahčilo zapracovanie revízie API do zdrojových kódov. Taktiež sme sa zdokonalili v ladení chýb v Jave a správe virtualizovanej testovacej infraštruktúry na platforme VMware ESXi. V procese testovania sme si preverili znalosti zo smerovania na 3. vrstve sieťového modelu RM OSI.

11 Zhodnotenie

Hlavným cieľom projektu bolo vytvoriť manažment VoIP relácií. Každý to možno poznáme, keď prechádzame tunelom alebo sme na koncerte, kde je obrovské množstvo ľudí, ktorý sa snažia telefonovať, tak to väčšinou nie je možné alebo trpí kvalita spojenia. Tiež pri telefonovaní cez internet pomocou protokolu VoIP, môže dôjsť k zahlteniu linky na ktorej prebieha hovor, k oneskoreniu alebo k inej situácii, ktorá má výrazný vplyv na kvalitu hovoru. Tým sa môže stať, že pri dôležitom telefonáte prídete o cenné rady. Nebudete počuť všetko, začnú vznikať hluché miesta alebo bude hovor úplne nezrozumiteľný. Toto bol základný prvok a motivácia pre náš tím na tvorbu riadenia telefónnych hovorov.

Začali sme preto analyzovať všetky potrebné nástroje, vytvorili návrh a prototyp. Implementovali sme požadovanú funkcionalitu a vytvorili testovacie scenáre na dostatočné testovanie. Výsledkom bolo dokonalé prepínanie hovorov medzi linkami. Ak na linke zbadáme vysoké zahltenie, oneskorenie alebo iné obmedzenie a kvalita hovoru je poškodená, tak pomocou nášho systému jednoducho prepneme hovor na inú, menej zaťaženú linku a hovor prebieha bez problémov ďalej s dokonalou kvalitou bez výpadku alebo straty čo i len samohlásky pri hovore.

Súčasťou riešenia je aj nástroj NetEm, ktorým sme dosiahli simuláciu reálneho prostredia v sieti. Nastavenia obmedzení sa prejavili aj pri reálnych telefónoch, kde sme počuli oneskorenie slov pri hovore alebo stratu a vypadávanie hovoru.

Tiež sme implementovali moderné, prehľadné a intuitívne používateľské rozhranie, ktoré dokonale zvýrazňuje praktickosť a jedinečnosť nášho systému.

Výsledkom projektu je dokonalé prepínanie hovorov, bez straty jediného písmena pri hovore, intuitívne a prehľadné používateľské rozhranie, reálna simulácia siete a vylepšenie programov SIRUP klient a server. Testovacie scenáre nám zase dokonale otestovali celý systém, čo môžeme vidieť aj na výsledných grafoch.

12 Literatúra

[1] About JavaScript - JavaScript | MDN. [Online] https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript.

[2] *node.js.* [Online] http://nodejs.org/.

[3] *Chrome V8 — Google Developers.* [Online] https://developers.google.com/v8/.

[4] *Callback Hell.* [Online] http://callbackhell.com/.

[5] *npm.* [Online] https://npmjs.org/.

[6] *Building Web APIs with Node.js and MongoDB.* [Online] http://www.codemag.com/Article/1210041.

[7] *DailyJS: The State of Node and Relational Databases.* [Online] http://dailyjs.com/2013/04/15/node-database-library/.

[8] *MongoDB.* [Online] http://www.mongodb.org/.

[9] *Apache CouchDB.* [Online] http://couchdb.apache.org/.

[10] *JSON.* [Online] http://www.json.org/.

[11] *Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST).* [Online] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

[12] *Socket.IO: the cross-browser WebSocket for realtime apps.* [Online] http://socket.io/.

[13] ChartJS Dev Express - JavaScript Tool 14.11.2013, [Online] www.chartjs.devexpress.com

[14] Comparison of JavaScript charting frameworks 14.11.2013 [Online] http://en.wikipedia.org/wiki/Comparison_of_JavaScript_charting_frameworks

[15] Linux Foundation - netem 19. 11. 2009 [Online] www.linuxfoundation.org/collaborate/workgroups/networking/netem

[16] RFC 3261, SIP: Session Initiation Protocol [Online] http://www.ietf.org/rfc/rfc3261.txt

[17] SIP Demystified - Gonzalo Camarillo, 2002

13 Prílohy

13.1 Používateľská príručka

13.1.1 SIRUP klient a server

Konfiguračný súbor

Pred samotným spustením SIRUP - u je potrebné upraviť konfiguračný súbor config.txt. Jeho zloženie je nasledovné:

DBhost: localhost
database: UDPproxy
DBuser: root
*DBpassword: ****

host: 192.168.1.11
port: 5090

WANProxyAddress: 10.0.0.2

clientID: 2
SIRUPmasterAddress: 192.168.1.7
SIRUPmasterPort: 5060

Prvá časť obsahuje adresu databázy, jej názov a autentifikačné údaje. Nasleduje adresa a port na ktorých má SIRUP bežať. Tieto údaje nie sú potrebné a v prípade, že neboli špecifikované, použije sa prednastavený port 5090, adresa je získaná automaticky. Nasleduje adresa SIRUP servera. Ďalším bodom je identifikačné číslo klienta. Nakoniec zadáme adresu a port na ktorých sa nachádza SIRUP Master. Adresu SIRUP servera spolu s adresou SIRUP Master a ID klienta je potrebné nastavovať len na klientovi.

Spustenie SIRUP - u

Pre spúšťanie SIRUP - u bol pripravený skript start.sh. Ten nielenže pred spustením SIRUP - u vymaže všetky záznamy v iptables, ale aj skontroluje potrebné predpoklady na správny chod ako sú napríklad zapnutie forwarding - u alebo spúšťanie SIRUP - u ako root používateľ. Po jeho spustení si vyberáme, či má fungovať v režime klienta “l” (LAN proxy), servera “w” (WAN proxy), alebo je ukončený pomocou písmena “q”.

SIRUP klienta je možné nechať ovládať výlučne SIRUP Master - om. V takomto prípade ďalej zadáme písmeno “s” dôsledkom čoho odošleme správu SUBSCRIBE. Ak potrebujeme klienta ovládať ručne, musíme ho najskôr zaregistrovať voči SIRUP serveru. Syntax je “n 5091”, pričom hodnota 5091 reprezentuje jeden z možných portov otvorených na serveri. Ak sa chceme klienta preregistrovať na iný port, použijeme podobný príkaz, ale namiesto písmena n použijeme písmeno r.

Riešenie možných problémov

V prípade nekorektne ukončeného hovoru môžu v iptables zostať staré pravidlá. Tie je najjednoduchšie možné vymazať pomocou skriptu reset.sh. Samotné pravidlá sa nachádzajú v tabuľkách raw a rawpost.

13.1.2 Manažment API

Manažment API obsahuje konfiguračný súbor, kde sa dá nastaviť IP adresa a port, na ktorej počúva. V tomto konfiguračnom súbore sa dajú taktiež nastaviť jednotlivé linky, ich identifikátor, IP adresa a port.

Manažment API sa spúšťa pomocou príkazu „node api.js“. Za týmto príkazom je možné použiť parameter --log=level, kde level je číslo od 1 do 5. Pomocou tohto príkazu sa dá ovládať množstvo výpisov z aplikácie. Základný stupeň je nastavený na 3, čo znamená informačné výpisy, varovné výpisy a chybové výpisy.

13.1.3 Zobrazovacia / rozhodovacia jednotka

Zobrazovacia rozhodovacia jednotka je webová aplikácia, na ktorú sa pristupuje cez webový prehliadač. Adresa záleží od momentálnej inštalácie. V tejto webovej aplikácií je možné presúvať klientov medzi linkami pomocou gesta „potiahni a pusti“ (drag and drop). Webová aplikácia sa pripája na manažment API automaticky.

13.1.4 SIRUP Master

V koreňovom adresári nájdeme jednoduchý skript, ktorý spustí aplikáciu. K spusteniu server musí mať nainštalované prostredie pre spúšťanie aplikácií písaných v programovacom jazyku Java. Aplikácia sa spúšťa príkazom ./stars0.sh.

13.1.5 Simulácia siete NetEm

V úvodnej časti skriptu *netem.sh sa* nachádza konfiguračná časť. Zmenou premenných v tejto časti možno upraviť správanie tejto súčasti pri priamom vykonávaní skriptu aj pri ovládaní prostredníctvom webového rozhrania.

```
## config
PORT=5091
LINE_CNT=2
INTERFACE=(eth1 eth2)
DEFAULT="netem delay 50ms 10ms 25% loss 11.1% 25%"
DEFAULT_B="1Mbit"
```

PORT – číslo UDP portu identifikujúce prvý tunel. Ak sa pri inicializácií vytvára viacero tunelov, tieto sú charakterizované portami s nasledujúcimi číslami (+1,+2,+3...). Táto hodnota sa používa iba pri aktivácií profilu.

LINE_CNT – počet tunelov vytvorených pri aktivácií profilu. Táto hodnota sa používa iba pri aktivácií profilu, nakoľko neskôr sa môže počet tunelov zmeniť ich pridaním alebo odobratím.

INTERFACE – zoznam sieťových adaptérov, na ktoré sa aplikuje NetEm profil.

DEFAULT – predvolená hodnota oneskorenia, kolísania oneskorenia a stratovosti použitá pri vytváraní tunelov

DEFAULT_B – predvolená hodnota šírky pásma použitá pri vytváraní tunelov

13.2 Systémová príručka

13.2.1 Inštalácia SIRUP servera a klienta

Prvým krokom je inštalácia prídavných modulov do iptables. Najskôr je potrebné stiahnuť správnu verziu Xtables-addons podľa verzie Kernel – u systému (http://xtables-addons.sourceforge.net). Pre Kernel verzie nižšej ako 3.7 je potrebné použiť verziu xtables 1.x, pre novšie verzie Kernel – u môžeme použiť najnovšiu verziu xtables prídavných modulov. Podrobnejšie informácie je možné nájsť v stiahnutom INSTALL súbore xtables. V čase inštalácie prídavných modulov fungovali len na 32-bitovej verzii Linuxu.

Výhodou tohto prístupu je, že si nevyžaduje prekompilovanie Kernel - u. Do priečinka ./extensions/ skopírujeme všetky iptables súbory nachádzajúce sa v zdrojových kódoch. Ďalej musíme pridať do konfiguračných súborov ./extensions/Kbuild, ./extensions/Mbuild záznamy o našich moduloch a v súbore ./mconfig pomocou m alebo n nastavujeme, ktoré moduly chceme, alebo nechceme inštalovať. V našom prípade je to okrem vlastných modulov ešte modul RAWNAT. Nasleduje štandardný postup (./configure, make, make install).

Nakoľko server pracuje s databázou MySQL je ju samozrejme potrebné tiež inštalovať a skript na vytvorenie databázy môžeme nájsť priložený pri zdrojových kódoch.

13.2.2 Inštalácia manažment API

Pre inštaláciu manažment API je nutné mať nainštalovaný Node.js a balíčkový manažér NPM. Po prekopírovaní zdrojových súborov do zvoleného priečinka treba zadať príkaz „npm install“. Po prebehnutí tohto príkazu byť inštalácia dokončená. Podporované operačné systémy sú Windows, Linux a Mac OS X.

13.2.3 Inštalácia a modifikácia zobrazovacej / rozhodovacej jednotky

Pre inštaláciu webovej stránky je nutné prekopírovať obsah priečinku „public“ do požadovaného priečinku webového servera. V prípade modifikácie je potrebné mať nainštalované Node.js a balíčkový systém NPM. Inštalácia prebehne pomocou príkazu „npm install“ v koreňovom priečinku. Po zmene zdrojových kódov sa kompilácia vykonáva pomocou príkazu „grunt“.

13.2.4 Inštalácia SIRUP Master

Nakopírujme si hotový projekt do ľubovoľného adresára v serveri. Modifikujme konfiguračný súbor ConfigurationFile v adresári s0 podľa požiadaviek architektúry.

Príklad konfiguračného súboru:

```
IP_ADDRESS 10.0.2.10        #IP adresa servera
PORT 5060                   #port na ktorom má počúvať
PROTOCOL UDP                #Transportný protokol, ktorý má požívať
MASTERID 10                 #ID Sirup Master
```

MGM_IP 147.175.145.166 #IP adresa manažmentového servera
MGM_PORT 8000 #Port manažmentového servera

Po spustení sirup Master sa pripojí na manažmentový server a pošle správu HelloMaster v JSON objekte pomocou websocketu. Od tohto momentu čaká na SUBSCRIBE správy od klientov.

13.2.5 Inštalácia nástroja NetEm

Nástroj NetEm, ktorý sa ovláda programom *tc,* je potrebné spúšťať v privilegovanom režime. Preto je pre fungovanie tejto súčasti systému potrebné zapísať do konfiguračného súboru /etc/sudoers cestu k skriptu *netem.sh*. Editácia tohto súboru sa vykonáva pomocou príkazu *visudo*. Pridaný riadok súboru môže byť zapísaný napríklad v tvare

ALL ALL=(root) NOPASSWD: /var/www/netem/netem.sh

Webové rozhranie nie je možné použiť lokálne bez nainštalovaného servera Apache a PHP. Vykonanie lokálneho prístupu možno po vložení súborov do príslušného priečinka zadaním URL napríklad v tvare *localhost/netem.*

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Ilkovičova 3, 842 16 Bratislava 4

Manažment VoIP relácií

Dokumentácia riadenia projektu

Študijný program: Počítačové a komunikačné systémy a siete

Tím č.3: Bc. Jozef Baláž, Bc. Tomáš Boros, Bc. Adam Močkoř, Bc. Martin Pivarník, Bc. Matej Rybár, Bc. Timotej Tkáč

Vedúci tímového projektu: Ing. Ján Murányi

Ak. rok: 2013/14

Obsah

1 Úvod 4
1.1 História dokumentu 4
2 Ponuka 5
2.1 Zadanie 5
2.2 Tím 5
2.3 Motivácia 6
2.4 Analýza a hrubý návrh 6
2.5 Plán projektu 8
2.5.1 Zimný semester 8
2.5.2 Letný semester 9
2.6 Zdroje pre realizáciu projektu 9
2.7 Súhrnný rozvrh členov tímu 9
2.8 Zoradenie všetkých ponúkaných tém podľa priority 9
3 Plán projektu v týždňoch 10
4 Úlohy členov tímu 11
4.1 Komunikácia členov tímu 11
5 Zápisnice 12
5.1 Zápisnica č. 1 zo stretnutia 8. 10. 2013 12
5.2 Zápisnica č. 2 zo stretnutia 15. 10. 2013 13
5.3 Zápisnica č. 3 zo stretnutia 22. 10. 2013 14
5.4 Zápisnica č. 4 zo stretnutia 5. 11. 2013 15
5.5 Zápisnica č. 5 zo stretnutia 12. 11. 2013 16
5.6 Zápisnica č. 6 zo stretnutia 19. 11. 2013 17
5.7 Zápisnica č. 7 zo stretnutia 26. 11. 2013 18
5.8 Zápisnica č. 8 zo stretnutia 3. 12. 2013 19
5.9 Zápisnica č. 9 zo stretnutia 10. 12. 2013 20
5.10 Zápisnica č. 10 zo stretnutia 18. 2. 2014 21
5.11 Zápisnica č. 11 zo stretnutia 26. 2. 2014 23
5.12 Zápisnica č. 12 zo stretnutia 12. 3. 2014 24
5.13 Zápisnica č. 13 zo stretnutia 19. 3. 2014 25
5.14 Zápisnica č. 14 zo stretnutia 26. 3. 2014 26
5.15 Zápisnica č. 15 zo stretnutia 2. 4. 2014 28

5.16 Zápisnica č. 16 zo stretnutia 9. 4. 2014 30
5.17 Zápisnica z testovania 31
6 Štandardy kódovania 32
7 Manažment verzií, konfigurácií a zmien 33
8 Posudky 2
9 Protokoly 2

Zoznam obrázkov

Obrázok 2-1 Architektúra systému 8

1 Úvod

V rámci predmetu Tímový projekt sa budeme zaoberať témou Manažment VoIP relácií. V riadiacej dokumentácii možno nájsť Ponuku, ktorej obsahom je zoznam jednotlivých členov spolu so základnými informáciami o predošlom štúdiu, znalosťami programovacích jazykov a tiež praktickými skúsenosťami s VoIP technológiami a konkrétne SIP. Následne sa pokúsime poskytnúť náš pohľad na samotnú problémovú oblasť a tiež načrtnúť našu predstavu riešenia.

V ďalších kapitolách je spôsob komunikácie medzi členmi tímu a zápisnice z jednotlivých stretnutí.

1.1 História dokumentu

Verzia dokumentu	Dátum zmeny	Opis zmeny
0.2	15.11.2013	Vytvorenie dokumentu
0.3	20.11.2013	Tvorba kapitoly 1 a 2
0.5	30.11.2013	Tvorba kapitoly 3
0.7	8.12.2013	Tvorba kapitoly 4 a 5
0.8	12.12.2013	Dokončenie dokumentu
1.0	31.1.2014	Doplnenie zápisníc
1.1	24.4.2014	Doplnenie zápisníc

2 Ponuka

2.1 Zadanie

Multimediálne relácie založené na protokole SIP (Session Initiation Protocol) sa väčšinou skladajú z troch rôznych komunikačných kanálov (SIP, RTP a RTCP), pričom každý z týchto kanálov vyžaduje komunikáciu na osobitnom porte.

Táto skutočnosť sa odzrkadľuje na zložitom manažmente VoIP (Voice over IP) relácií.

Analyzujte architektúru „SIP Single Port" a porovnajte ju s inými existujúcimi architektúrami zameranými na manažment VoIP relácií. SIP Single Port architektúra pre multimediálnu komunikáciu využíva na rozdiel od protokolu SIP len jeden port, čo je hlavnou motiváciou pre návrh optimalizácie manažmentu VoIP relácií.

Navrhnite aplikáciu, ktorá umožní riadiť aktívny manažment relácií. Výsledná aplikácia musí byť schopná adaptovať sa na náhle zmeny v sieti ako napríklad zahltenie, zhoršenie kvality, pád linky, zmena IP adresy klienta a podobne.

Na základe analýzy a návrhu implementujte aplikáciu a výsledok práce zhodnoťte.

2.2 Tím

Jozef Baláž, Bc.

- Bakalárska práca: SIP Single Port
- Cisco CCNA certifikát
- Programovanie: C, C++ (Qt), JAVA, MySQL
- Absolvovaný predmet Konvergencia mobilných a pevných sietí

Tomáš Boros, Bc.

- Bakalárska práca: Zostavenie laboratória distribučných sietí
- Cisco CCNA Certifikát
- Programovanie: C, Java, MySQL, PHP
- Absolvovaný predmet Konvergencia mobilných a pevných sietí
- Skúsenosti s administráciou Unix systémov

Adam Močkoř, Bc.

- Bakalárska práca na tému: Hraničný element VoIP architektúry (2013)
- Skúsenosti s VoIP protokolmi nadobudnuté na predmete Konvergencia mobilných a pevných sietí a pri riešení zadania bakalárskej práce.
- Skúsenosti s programovaním v C, C#, HTML, CSS, Javascript

Martin Pivarník, Bc.

- Bakalárska práca: Návrh a nasadenie protokolu IPv6 v prostredí ngnlab.eu
- Cisco CCNA certifikát

- Programovanie: C, C#, Java, HTML
- Absolvovaný predmet Konvergencia mobilných a pevných sietí

Matej Rybár, Bc.

- Cisco 3 semestre CCNA
- Absolvovaný predmet Konvergencia mobilných a pevných sietí
- Programovanie: C, C++, C#, HTML, CSS

Timotej Tkáč, Bc.

- Absolvent predmetu Konvergencia mobilných a pevných sietí
- Skúsenosti s programovaním v C, C#, Java, Python

2.3 Motivácia

VoIP, alebo hlas cez IP protokol je technológiou, ktorá sa zaoberá prenosom hlasu, resp. multimediálneho obsahu cez IP siete. Za príčinu rozmáhania sa VoIP technológie je možné považovať zvyšujúcu sa dostupnosť pripojenia do siete Internet.

Dôležitým prvkom VoIP technológii, je zabezpečenie kvality spojenia. Motiváciou pri komunikácii dvoch klientov je, aby nedochádzalo k strate alebo k oneskoreniu dát. Tieto straty by mali na svedomí zníženú kvalitu zvuku, čo je pri telefonovaní nezanedbateľný problém, a preto sa chceme venovať jeho výraznému zmierneniu. Medzi ďalšie výzvy patrí riešenie problémov s prerušovaním hovorov v dôsledku výpadku linky, zahltenie linky inou komunikáciou a pod. Tieto problémy sú hlavnou motiváciou pre riešenie tohto tímového projektu.

Členovia tímu majú už dôležité skúsenosti s týmito problémami a stretli sa s nimi aj vo svojich bakalárskych prácach. Čerpať budeme z cenných skúseností Jozefa Baláža, ktoré získal pri bakalárskej práci s názvom SIP Single Port. Podstatným prvkom je aj absolvovanie predmetu Konvergencia mobilných a pevných sietí, ktorý sa zaoberá VoIP technológiou. Tento predmet absolvovali všetci členovia tímu, a preto každý má skúsenosť s touto tematikou, čo je veľkou výhodou.

2.4 Analýza a hrubý návrh

Zabezpečenie kvality spojenia pri VoIP hovoroch je jednou z kľúčových podmienok pri poskytovaní týchto služieb. Dôraz sa preto kladie na efektívny manažment relácií. Nakoľko informácie z nižších vrstiev RM OSI modelu nie sú pre tieto potreby postačujúce, cieľom je zamerať sa predovšetkým na protokoly vyšších vrstiev vrátane aplikačnej - SIP.

Architektúra SIP Single Port navrhnutá na fakulte spája tri komunikačné kanály do jedného. Predstavuje teda zásadný rozdiel oproti pôvodným architektúram, kde signalizácia a prenos multimediálnych dát prebiehajú na troch rôznych portoch. V aktuálnej podobe však nevie zabezpečiť kvalitu spojenia. Pri návrhu riešenia sa je potrebné zamyslieť nad týmito otázkami: Akým spôsobom môžeme merať vyťaženie linky, ako zabezpečiť jej presmerovanie v prípade prílišného vyťaženia/výpadku a ako

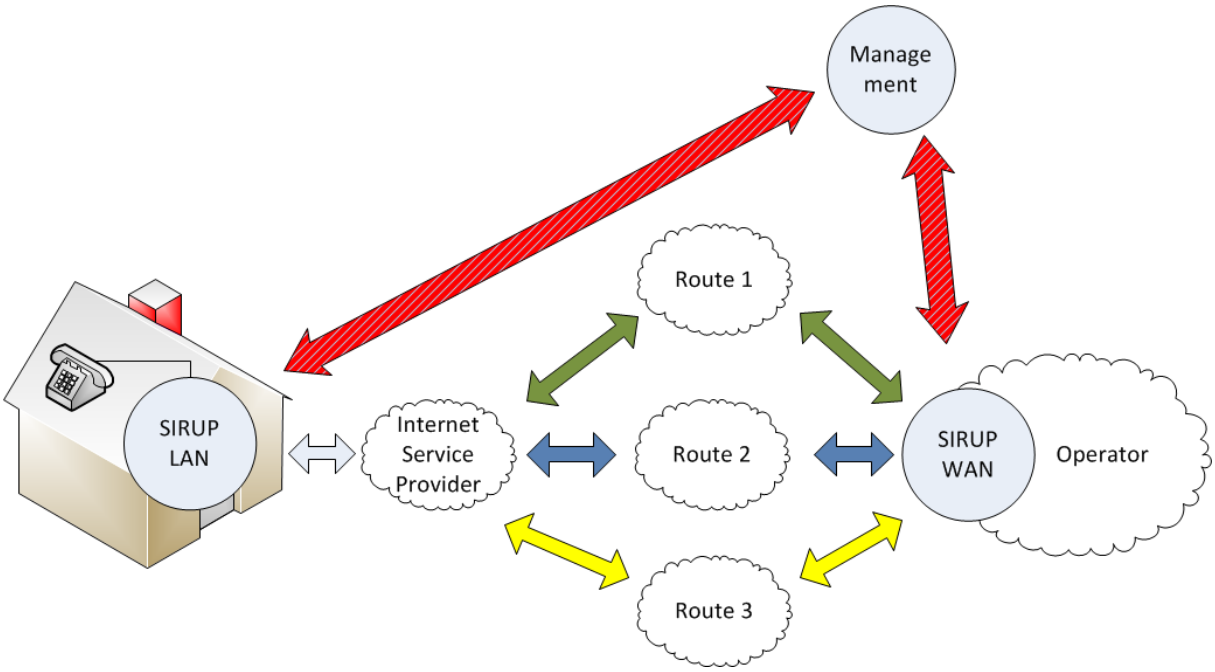
má prebiehať samotná komunikácia zabezpečujúca smerovanie medzi klientovou a poskytovateľovou inštanciou SIRUP - u.

Čo sa týka samotného presmerovania, v prípade prílišného vyťaženia linky, by mal server viacero rozhraní implementujúce rôzne cesty. Ak deteguje prílišné vyťaženie jednej z nich, môže klienta vyzvať na použitie alternatívnej cesty. Problém tiež môže nastať v prípade, keď linka ku klientovi vypadne úplne. Pre túto situáciu by klient už pri registrácii dostal od servera zoznam alternatívnych rozhraní (liniek) pod ktorými by bol pri výpadku na linke dosiahnuteľný. Tento zoznam by sa podľa potreby dynamicky menil.

Pri komunikácii medzi klientom a serverom je vhodné snažiť sa využívať už existujúce protokoly. SDP protokol nám vhodne poskytuje hlavičku "a = altc" pomocou ktorej vieme špecifikovať alternatívne trasy. [http://tools.ietf.org/html/rfc6947]

Možnosti merania záťaže liniek:

- Protocol SNMP - Simple Network Management Protocol - protokol umožňuje monitorovanie a správu zariadení v IP sieťach.
- Nástroj Nagios - v prípade detegovania výpadku linky umožní spúšťať skripty pre zotavenie komunikácie
- Monitorovanie komunikácie RTP- monitorovaním RTP správ je možné zistiť priepustnosť a detegovať stratu paketov počas spojenia pomocou poradového čísla (sequence number) v hlavičke RTP protokolu.
- Rôzne aplikácie: ifconfig, iftop, iptraf, nload



Obrázok 2-1 Architektúra systému

2.5 Plán projektu

2.5.1 Zimný semester

Analýza a návrh

Termín: do 29.10.2013

- Analýza existujúcich riešení
- Návrh a špecifikácia vlastného riešenia
- Webová prezentácia

Príprava prostredia

Termín: do 5.11.2013

- Konfigurácia virtualizačného prostredia
- Zriadenie prístupu do potrebných kolaboračných nástrojov

Implementácia - Fáza 1

Termín: do 3.12.2013

- Prototyp (monitorovanie a hlásenie zmien vlastností siete, zmena cesty sieťovej komunikácie, základné manažmentové rozhranie)

2.5.2 Letný semester

Implementácia - Fáza 2

Termín: 20. 4. 2014

- Dokončenie implementácie

Implementácia - Fáza 3

Termín: 5. 5. 2014

- Zapracovanie pripomienok k implementácii

Implementácia - Fáza 4

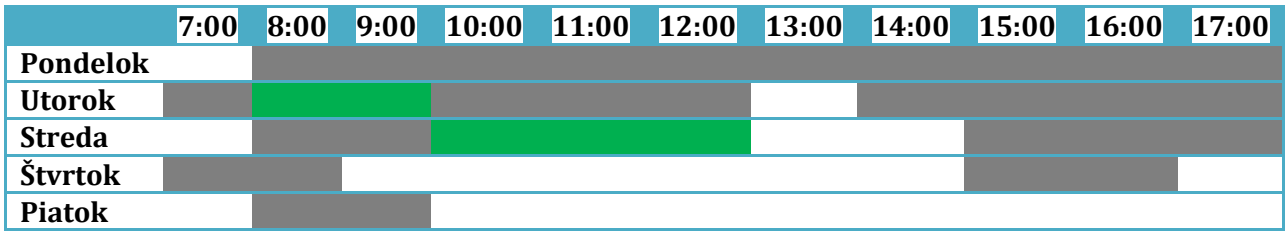
Termín: 10. 5. 2014

- Testovanie systému

2.6 Zdroje pre realizáciu projektu

Pre testovanie funkčnosti nášho riešenia budeme potrebovať virtualizované prostredie umožňujúce beh viacerých virtuálnych serverov s možnosťou vytvárania rôznych sieťových prepojení medzi virtuálnymi servermi. Softvérové požiadavky tohto projektu pokryje voľne dostupný softvér. Na stretnutia nášho tímu budú postačovať miestnosť Jobsovho softvérového štúdia.

2.7 Súhrnný rozvrh členov tímu



Preferovaný termín stretnutí:

Zimný semester - Utorok: 8:00 - 10:00

Letný semester – Streda: 10:00 - 13:00

2.8 Zoradenie všetkých ponúkaných tém podľa priority

1. Manažment VoIP relácií
2. Interaktivita mobilného zariadenia a televízie
3. Sieťový protokol IPv6
4. Aplikácia softvérového smerovania (SDN) v GPRS sieti
5. Aplikácia pre platformu Funtoro

3 Plán projektu v týždňoch

1. Výber témy
2. Prezentácia ponuky
3. Oboznámenie sa s projektom SIRUP
4. Tvorba webovej stránky tímu, analýza problematiky
5. Analýza problematiky, návrh komunikácie jednotlivých elementov
6. Analýza existujúcich riešení, simulácie a merania sieťových parametrov
7. Analýza implementačných nástrojov, špecifikácia požiadaviek
8. Doplnenie špecifikácie a návrhu, odovzdanie dokumentácie
9. Odovzdanie posudku dokumentácie iného tímu
10. Zhodnotenie posudku našej dokumentácie, práca na prototype
11. Práca na prototype
12. Príprava prezentácie prototypu
13. Príprava na implementáciu prototypu
14. Implementácia jednotlivých častí systému samostatne
15. Implementácia jednotlivých častí systému samostatne
16. Spájanie jednotlivých častí systému
17. Spájanie jednotlivých častí systému
18. Testovanie funkcionality po spojení jednotlivých častí
19. Testovanie celého systému ako celku
20. Riešenie problémov pri testovaní funkcionality
21. Testovanie celého systému ako celku
22. Testovanie celého systému ako celku
23. Testovanie celého systému ako celku
24. Tvorba dokumentácie
25. Finalizácia jednotlivých kapitol dokumentácie
26. Príprava videa a prezentácie
27. Odovzdanie projektu a prezentácia

4 Úlohy členov tímu

- **Jozef Baláž** – vývojár SIRUP časti
- **Tomáš Boros** – vývojár serveru s0
- **Adam Močkoř** – vývojár manažment serveru a rozhrania
- **Martin Pivarník** – študentský vedúci tímu
- **Matej Rybár** – hlavný dokumentarista
- **Timotej Tkáč** – vývojár sieťového prepojenia elementov

4.1 Komunikácia členov tímu

Tím Singles sa stretával s pedagogickým vedúcim na pravidelných stretnutiach v utorok o 8:30, kde sa zhodnotil stav plnenia pridelených úloh, prediskutovali sa otázky vyplývajúce z analýzy problematiky a navrhli nové úlohy.

Na evidenciu stavu plnenia úloh tím využíval kolaboračnú platformu Trello, službu Google Groups pre hromadnú komunikáciu prostredníctvom emailu, službu Google Hangouts pre videohovory s členmi nachádzajúcimi sa mimo Bratislavy a službu Google Drive pre tvorbu dokumentácie. V letnom semestri hlavne kvôli implementácii sme začali využívať BitBucket na zdieľanie zdrojových kódov projektu.

5 Zápisnice

5.1 Zápisnica č. 1 zo stretnutia 8. 10. 2013

Dátum: 8. 10. 2013

Čas: 8:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Tomáš Boros
Bc. Adam Močkoř
Bc. Timotej Tkáč

Adam Močkoř

- Riešenie rôznych možných implementácií manažment API

Timotej Tkáč

- Oboznámenie sa s projektom SIRUP

Jozef Baláž

- Implementácia serveru s jednou inštanciou na viacerých portoch vs viacero inštancií

Tomáš Boros

- Špecifikácia správ medzi S0 a klientským SIRUPOM

Náplň stretnutia:

- Oboznámenie sa s projektom SIRUP
- Bezpečnosť navrhovaného riešenia
- SIP REGISTER vs. PKI
- Spôsob komunikácia medzi manažmentom a SIRUPom
- Manažment API - platforma, dostupné knižnice, vhodné riešenie
- GUI - programové rozhranie

Úlohy do ďalšieho stretnutia:

- Rozbehnutie SIRUP - u
- Navrhnúť správy Subscribe / Notify, poprípade implementovať

5.2 Zápisnica č. 2 zo stretnutia 15. 10. 2013

Dátum: 15. 10. 2013

Čas: 8:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Tomáš Boros
Bc. Adam Močkoř
Bc. Timotej Tkáč
Bc. Martin Pivarník
Bc. Matej Rybár

Adam Močkoř

- Definovali sme si funkcionalitu manažmentovej časti systému

Timotej Tkáč

- Parametre charakterizujúce kvalitu liniek

Jozef Baláž

- Komunikácia klientského SIRUP - u so serverovým

Martin Pivarník

- Zabezpečenie prístupu na stroj labss2.fiit.stuba.sk

Tomáš Boros

- Dodatočné špecifikácie S0

Matej Rybár

- Parametre charakterizujúce kvalitu liniek
- Webové rozhranie medzi klientom a manažmentom

Náplň stretnutia:

- Komunikácia SIP Single Port medzi C (klient) a S (SIRUP)
- Default SIRUP (S0)
- Typy rozhraní medzi prvkami topológie
- Parametre charakterizujúce kvalitu liniek
- Funkcionalita manažmentu
- Typy správ medzi zariadeniami

Úlohy do ďalšieho stretnutia:

- Komunikácia medzi manažmentom a default SIRUPom (S0)
- Komunikácia medzi klientom (C) a default SIRUPom (S0)
- Publikovanie HTML prezentácie projektu na stroji labss2.fiit.stuba.sk

5.3 Zápisnica č. 3 zo stretnutia 22. 10. 2013

Dátum: 22. 10. 2013

Čas: 8:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Tomáš Boros
Bc. Adam Močkoř
Bc. Timotej Tkáč

Adam Močkoř

- Doplnenie funkcionality systému a navrhnuté riešenia pre vybudovanie API

Timotej Tkáč

- Riešenie simulácie kvality liniek pomocou nástroja NetEm

Jozef Baláž

- Presmerovanie aktívneho hovoru

Tomáš Boros

- Prezentácia stavu S0

Náplň stretnutia:

- Komunikácia SIP Single Port medzi C (klient) a S (SIRUP)
- Komunikácia medzi manažmentom a default SIRUPom (S0)
- Komunikácia medzi klientom (C) a default SIRUPom (S0)
- Možné riešenia simulácie kvality liniek
- Funkcionalita manažmentu

Úlohy do ďalšieho stretnutia:

- Špecifikácia projektu
- Presmerovanie aktívneho hovoru

5.4 Zápisnica č. 4 zo stretnutia 5. 11. 2013

Dátum: 5. 11. 2013

Čas: 8:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Tomáš Boros
Bc. Adam Močkoř
Bc. Timotej Tkáč
Bc. Martin Pivarník
Bc. Matej Rybár

Adam Močkoř

- Použitie protokolu pre návrh API (REST vs Socket.io) - výsledok Socket.io
- Výber databázového systému - výsledok CouchDB

Timotej Tkáč

- Simulácia siete - funkčný skript, zadávanie parametrov cez konzolu
- Úloha - ovládanie cez GUI

Jozef Baláž

- Potvrdenie správy 200 OK na REGISTER, implementácia SUBSCRIBE / NOTIFY
- Úloha - odskúšať pravidlo, pri ktorom sa do logu zapíše prijatie prvého RTP paketu na novom porte, podľa toho vymazať pôvodné pravidlo

Martin Pivarník

- Možnosti využitia navrhovaného riešenia - ako možných záujemcov sme identifikovali ISP s redundantnými cestami medzi dvoma SIP elementmi a organizácie s redundantnou konektivitou

Tomáš Boros

- Prezentácia stavu S0 - Default SIRUP - S0 bude implementovaný v jazyku Java
- Možnosti komunikácie s API
- Načo slúžia správy SUBSCRIBE - k zisťovaniu, či je klient aktívny
- Načo slúžia správy NOTIFY - oznamujú klientovi zmeny tunelov

Matej Rybár

- Možnosti a typy zobrazovaných informácií
- Úloha - ako zobrazovať jednotlivé informácie

Náplň stretnutia:

- Zhodnotenie plnenia úloh

Úlohy do ďalšieho stretnutia:

- Špecifikácia projektu

5.5 Zápisnica č. 5 zo stretnutia 12. 11. 2013

Dátum: 12. 11. 2013

Čas: 8:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Tomáš Boros
Bc. Adam Močkoř

Adam Močkoř

- Pripomienky k časti špecifikácie a návrhu ohľadom API

Jozef Baláž

- Použitie správy REGISTER s hodnotou Expires = 0 na odhlásenie klienta zo starého portu

Náplň stretnutia:

- Zhodnotenie plnenia úloh

Úlohy do ďalšieho stretnutia:

- Štúdium analýzy, špecifikácie a návrhu iného tímu
- Doplnenie častí Špecifikácia a Návrh

5.6 Zápisnica č. 6 zo stretnutia 19. 11. 2013

Dátum: 19. 11. 2013

Čas: 8:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Tomáš Boros
Bc. Adam Močkoř
Bc. Timotej Tkáč
Bc. Martin Pivarník
Bc. Matej Rybár

Adam Močkoř

- Spracovanie vecnej stránky kapitol 2 až 5 dokumentácie iného tímu

Timotej Tkáč

- Spracovanie formálnej stránky kapitoly 1.1 dokumentácie iného tímu

Jozef Baláž

- Spracovanie vecnej stránky kapitoly 1.1 dokumentácie iného tímu

Tomáš Boros

- Spracovanie vecnej stránky kapitol Úvod, 1.2 až 1.4 dokumentácie iného tímu

Martin Pivarník

- Spracovanie formálnej stránky kapitol Úvod, 1.2 až 1.4 dokumentácie iného tímu

Matej Rybár

- Spracovanie formálnej stránky kapitol 2 až 5 a dokumentácie riadenia iného tímu
- Skompletizovať posudok a odovzdať ho oponentskému tímu

Náplň stretnutia:

- Overenie prístupu na virtualizačný hypervízor

Úlohy do ďalšieho stretnutia:

- Spísať a odovzdať posudok oponentskému tímu

5.7 Zápisnica č. 7 zo stretnutia 26. 11. 2013

Dátum: 26. 11. 2013

Čas: 8:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Tomáš Boros
Bc. Adam Močkoř
Bc. Timotej Tkáč
Bc. Martin Pivarník
Bc. Matej Rybár

Adam Močkoř

- Návrh prototypu manažment servera

Timotej Tkáč

- Prototyp umožňujúci zmenu parametrov linky z webového rozhrania

Jozef Baláž

- Prototyp umožňujúci preregistrovanie klientského SIRUP - u na iný port

Tomáš Boros

- Prototyp s0 komunikujúci s klientskou a serverovou časťou SIRUP - u

Náplň stretnutia:

- Zhodnotenie plnenia úloh

Úlohy do ďalšieho stretnutia:

- Práca na prototype

5.8 Zápisnica č. 8 zo stretnutia 3. 12. 2013

Dátum: 3. 12. 2013

Čas: 8:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Tomáš Boros
Bc. Adam Močkoř
Bc. Martin Pivarník

Adam Močkoř

- Návrh prototypu manažment serveru a rozhrania

Timotej Tkáč

- Prezentácia prototypu webového rozhrania, zosúladenie popiskov

Jozef Baláž

- Práca na prototype

Tomáš Boros

- Práca na prototype

Martin Pivarník

- Skompletizovanie dokumentácie

Náplň stretnutia:

- Zhodnotenie stavu dokumentácie
- Sprevádzkovanie virtuálneho stroja

Úlohy do ďalšieho stretnutia:

- Práca na prototype

5.9 Zápisnica č. 9 zo stretnutia 10. 12. 2013

Dátum: 10. 12. 2013

Čas: 8:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Adam Močkoř
Bc. Timotej Tkáč
Bc. Martin Pivarník

Adam Močkoř

- Implementácia prototypu manažment serveru a rozhrania

Timotej Tkáč

- Spôsoby realizácie prepojenia SIRUP klientov a serverov

Jozef Baláž

- Príprava prezentácie komunikácie SIRUP servera, klienta a s0

Tomáš Boros

- Práca na prototype

Martin Pivarník

- Príprava prezentácie prototypu

Matej Rybár

- Doplnenie dokumentácie riešenia

Náplň stretnutia:

- Zhodnotenie stavu prototypu

Úlohy do ďalšieho stretnutia:

- Práca na prototype
- Skompletizovanie dokumentácie
- Príprava na prezentáciu prototypu

5.10Zápisnica č. 10 zo stretnutia 18. 2. 2014

Dátum: 18. 2. 2014

Čas: 10:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Adam Močkoř
Bc. Timotej Tkáč
Bc. Martin Pivarník
Bc. Tomáš Boros
Bc. Matej Rybár

Adam Močkoř

- Funkcionalita a implementácia API manažmentu

Timotej Tkáč

- Riešenie simulácie kvality liniek pomocou nástroja NetEm

Jozef Baláž

- Komunikácia klientského SIRUP – u so serverovým

Tomáš Boros

- Práca na SIRUP – e S0 komunikujúceho s klientskou a serverovou časťou SIRUP - u

Martin Pivarník

- Práca na SIPP

Matej Rybár

- Doplnenie dokumentácie riešenia + práca na zápisniciach

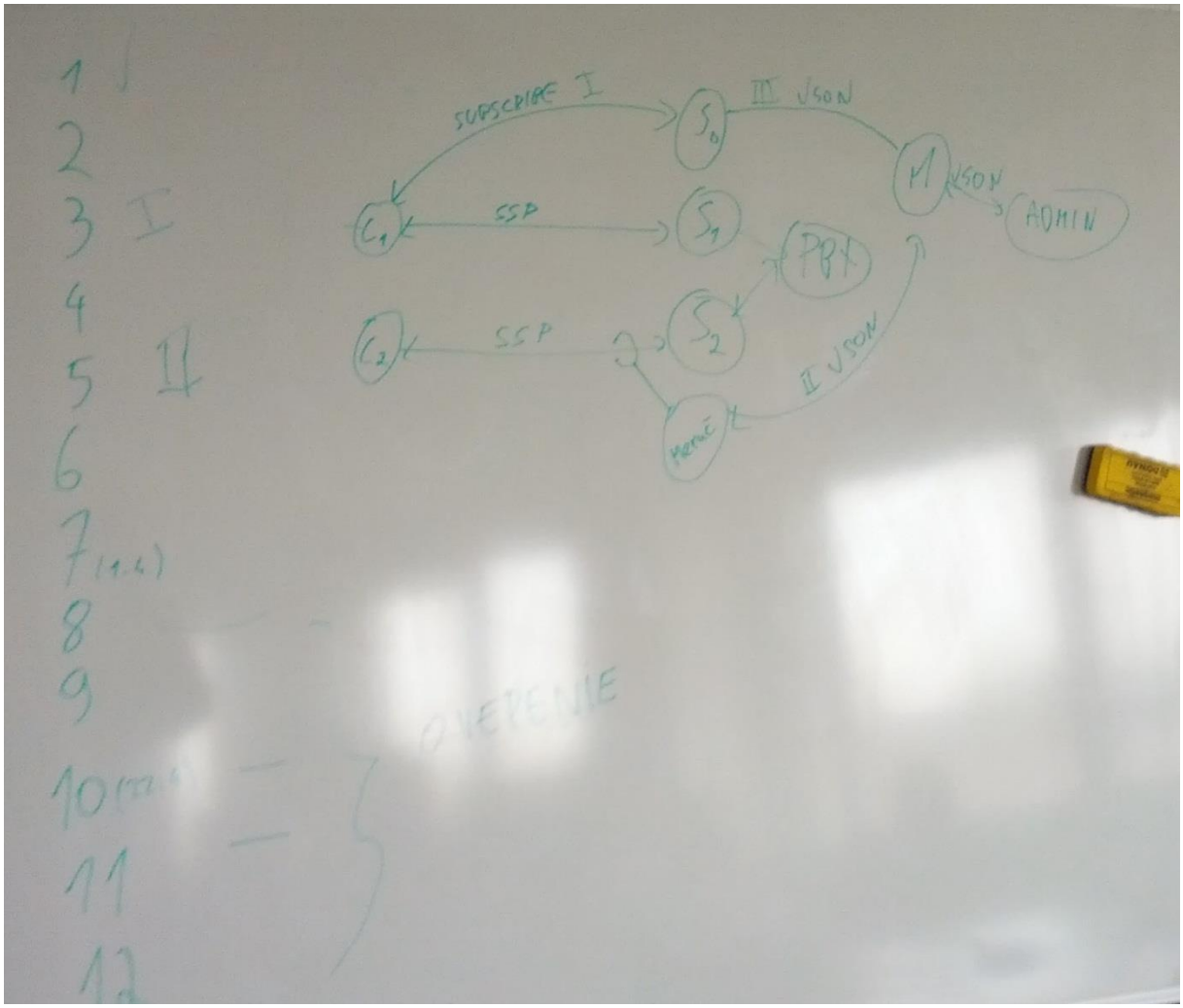
Náplň stretnutia:

- Určenie si cieľov a kontrolných bodov pre ďalšie stretnutia
- Pridelenie úloh jednotlivým členom tímu
- Komunikácia o dosiahnutých cieľoch a celkovom napredovaní projektu

Úlohy do ďalšieho stretnutia:

- Spojenie jednotlivých častí projektu do funkčného celku
- Zabezpečenie potrebných nástrojov a prvkov na spojenie a testovanie produktu
- Skompletizovanie dokumentácie

Schéma nakreslená na tabuli:



5.11 Zápisnica č. 11 zo stretnutia 26. 2. 2014

Dátum: 26. 2. 2014

Čas: 10:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Adam Močkoř
Bc. Martin Pivarník
Bc. Tomáš Boros
Bc. Matej Rybár

Adam Močkoř

- Funkcionalita a implementácia API manažmentu + prepojenie so SIRUP - om

Jozef Baláž

- Práca na spojení S0 a SIRUP servera + tvorba potrebných správ pre zabezpečenie funkcionality

Tomáš Boros

- Práca na spojení S0 a SIRUP servera + tvorba potrebných správ pre zabezpečenie funkcionality

Martin Pivarník

- Práca na SIPP

Matej Rybár

- Práca na zápisniciach + oboznámenie sa s BitBucket

Náplň stretnutia:

- Určenie si cieľov a kontrolných bodov pre ďalšie stretnutia
- Pridelenie úloh jednotlivým členom tímu
- Komunikácia o dosiahnutých cieľoch a celkovom napredovaní projektu

Úlohy do ďalšieho stretnutia:

- Spojenie jednotlivých častí projektu do funkčného celku
- Zabezpečenie potrebných nástrojov a prvkov na spojenie a testovanie produktu
- Skompletizovanie dokumentácie

5.12 Zápisnica č. 12 zo stretnutia 12. 3. 2014

Dátum: 12. 3. 2014

Čas: 10:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Adam Močkoř
Bc. Timotej Tkáč
Bc. Martin Pivarník
Bc. Tomáš Boros
Bc. Matej Rybár

Adam Močkoř

- Funkcionalita a implementácia API manažmentu + prepojenie so SIRUP - om

Jozef Baláž

- Práca na spojení S0 a SIRUP servera + tvorba potrebných správ pre zabezpečenie funkcionality

Tomáš Boros

- Práca na spojení S0 a SIRUP servera + tvorba potrebných správ pre zabezpečenie funkcionality

Timotej Tkáč

- Možnosť pridávania a odoberania liniek v nástroji NetEm

Martin Pivarník

- Práca na SIPP + oboznámenie sa s BitBucket

Matej Rybár

- Práca na zápisniciach + pomoc zvyšným členom tímu

Náplň stretnutia:

- Určenie si cieľov a kontrolných bodov pre ďalšie stretnutia
- Pridelenie úloh jednotlivým členom tímu
- Komunikácia o dosiahnutých cieľoch a celkovom napredovaní projektu

Úlohy do ďalšieho stretnutia:

- Spojenie jednotlivých častí projektu do funkčného celku
- Zabezpečenie potrebných nástrojov a prvkov na spojenie a testovanie produktu
- Skompletizovanie dokumentácie

5.13 Zápisnica č. 13 zo stretnutia 19. 3. 2014

Dátum: 19. 3. 2014

Čas: 10:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Adam Močkoř
Bc. Timotej Tkáč
Bc. Martin Pivarník
Bc. Tomáš Boros

Adam Močkoř

- Funkcionalita a implementácia API manažmentu + smerovanie JSON správ

Jozef Baláž

- Práca na spojení S0 a SIRUP servera + tvorba potrebných správ pre zabezpečenie funkcionality – správy NOTIFY

Tomáš Boros

- Práca na spojení S0 a SIRUP servera + tvorba potrebných správ pre zabezpečenie funkcionality – správy NOTIFY
- Socket.io klienta implementovať do SIRUP Master

Timotej Tkáč

- Usporiadanie topológie, pridanie základných ciest a štatistík do nástroja NetEm

Martin Pivarník

- Tvorba infraštruktúry podľa obrázka na magnetickej tabuli

Náplň stretnutia:

- Komunikácia o dosiahnutých cieľoch a celkovom napredovaní projektu
- Predvedenie aktuálneho stavu simulácie siete a implementácie SIRUP – ov a SIRUP Master
- Diskusia o sieťovej topológii pri použití samostatných virtuálnych strojov pre SIRUP klientov, NetEm a SIRUP servery

Úlohy do ďalšieho stretnutia:

- Spojenie jednotlivých častí projektu do funkčného celku
- Zabezpečenie potrebných nástrojov a prvkov na spojenie a testovanie produktu
- Skompletizovanie dokumentácie

5.14 Zápisnica č. 14 zo stretnutia 26. 3. 2014

Dátum: 26. 3. 2014

Čas: 10:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Adam Močkoř
Bc. Timotej Tkáč
Bc. Martin Pivarník
Bc. Matej Rybár
Bc. Tomáš Boros

Adam Močkoř

- Funkcionalita a implementácia API manažmentu + prepojenie so SIRUP – om
- Zistenie parametrov obsiahnutých v správe setQuality

Jozef Baláž

- Práca na spojení S0 a SIRUP servera + tvorba potrebných správ pre zabezpečenie funkcionality – správy NOTIFY + EVENT

Tomáš Boros

- Práca na spojení S0 a SIRUP servera + tvorba potrebných správ pre zabezpečenie funkcionality – správy NOTIFY + EVENT
- Socket.io klienta implementovať do SIRUP Master

Timotej Tkáč

- Možnosť pridávania a odoberania liniek v nástroji NetEm
- Pridanie default cesty + štatistiky

Martin Pivarník

- Tvorba infraštruktúry podľa obrázka na magnetickej tabuli
- Zabezpečenie funkcionality NetEm na manažment serveri

Matej Rybár

- Práca na zápisniciach + pomoc zvyšným členom tímu + SIPP

Náplň stretnutia:

- Komunikácia o dosiahnutých cieľoch a celkovom napredovaní projektu
- Predvedenie aktuálneho stavu simulácie siete a implementácie SIRUP – ov a SIRUP Master
- Diskusia o sieťovej topológii pri použití samostatných virtuálnych strojov pre SIRUP klientov, NetEm a SIRUP servery

Úlohy do ďalšieho stretnutia:

- Spojenie jednotlivých častí projektu do funkčného celku

- Zabezpečenie potrebných nástrojov a prvkov na spojenie a testovanie produktu
- Skompletizovanie dokumentácie

5.15 Zápisnica č. 15 zo stretnutia 2. 4. 2014

Dátum: 2. 4. 2014

Čas: 10:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Adam Močkoř
Bc. Timotej Tkáč
Bc. Martin Pivarník
Bc. Tomáš Boros

Adam Močkoř

- Funkcionalita a implementácia API manažmentu + prepojenie so SIRUP – om
- Formát správ NOTIFY + presunúť manažment na PBX

Jozef Baláž

- Práca na spojení S0 a SIRUP servera + tvorba potrebných správ pre zabezpečenie funkcionality
- Posielanie RTP správ

Tomáš Boros

- Práca na spojení S0 a SIRUP servera + tvorba potrebných správ pre zabezpečenie funkcionality
- Socket.io klienta implementovať do SIRUP Master
- S0 nainštalovať na SIRUP Master

Timotej Tkáč

- Simulácia JITTER

Martin Pivarník

- Inštalácia Kamailio

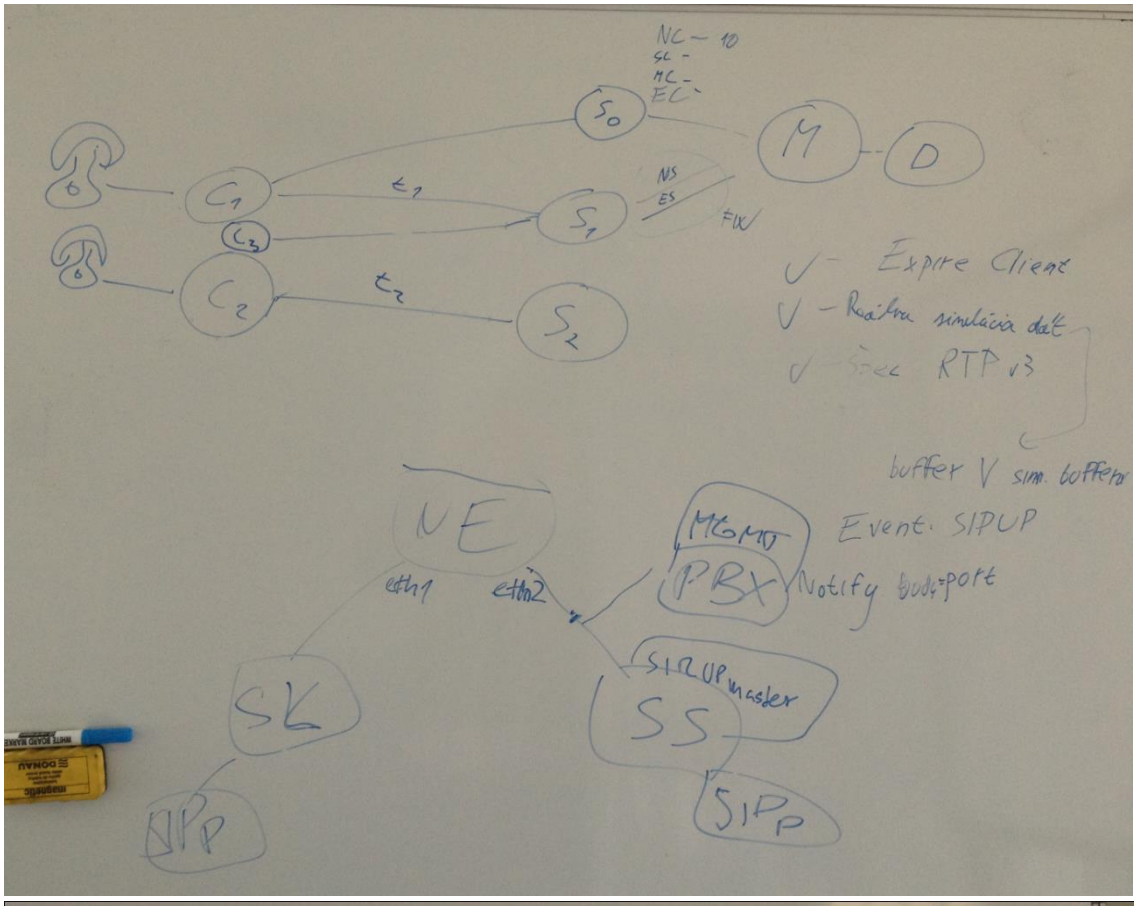
Náplň stretnutia:

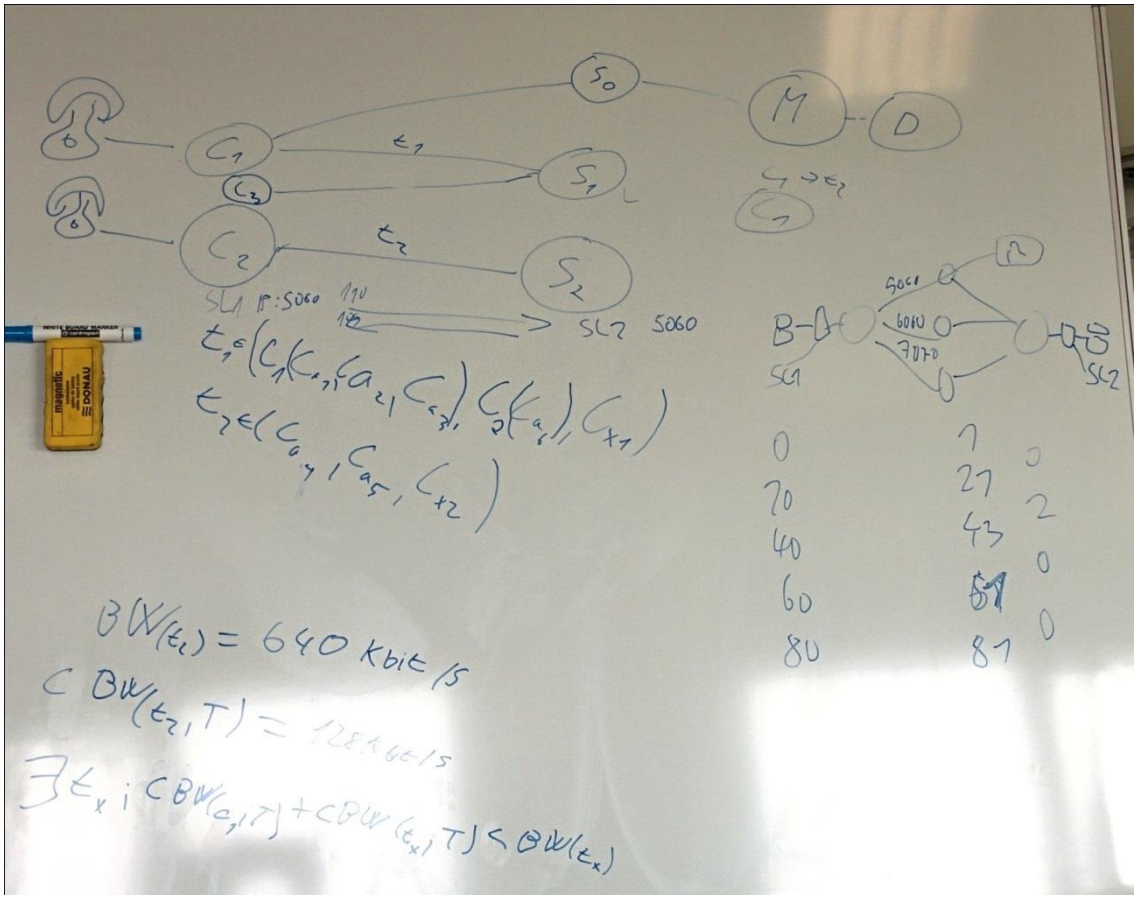
- Komunikácia o dosiahnutých cieľoch a celkovom napredovaní projektu
- Predvedenie aktuálneho stavu simulácie siete a implementácie SIRUP – ov a SIRUP Master
- Diskusia o sieťovej topológii pri použití samostatných virtuálnych strojov pre SIRUP klientov, NetEm a SIRUP servery

Úlohy do ďalšieho stretnutia:

- Spojenie jednotlivých častí projektu do funkčného celku
- Zabezpečenie potrebných nástrojov a prvkov na spojenie a testovanie produktu
- Skompletizovanie dokumentácie

Schémy nakreslené na tabuli:





5.16 Zápisnica č. 16 zo stretnutia 9. 4. 2014

Dátum: 9. 4. 2014

Čas: 10:00

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Adam Močkoř
Bc. Martin Pivarník
Bc. Matej Rybár
Bc. Tomáš Boros

Adam Močkoř

- Testovanie prepojenia so SIRUP – om a overenie správnosti formátu správ

Jozef Baláž

- Testovanie spojenia S0 a SIRUP servera + overenie správnosti formátov jednotlivých správ pre komunikáciu

Tomáš Boros

- Testovanie spojenia S0 a SIRUP servera + overenie správnosti formátov jednotlivých správ pre komunikáciu

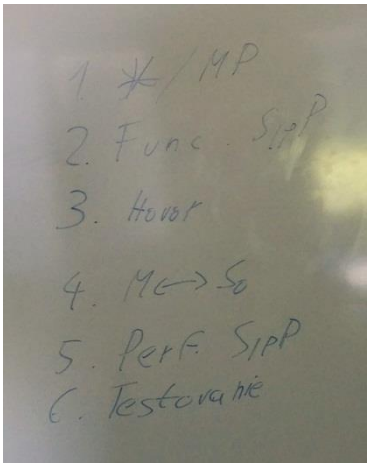
Martin Pivarník

- Inštalácia Asterisk

Matej Rybár

- Práca na zápisniciach + pomoc zvyšným členom tímu + SIPP

Náplň stretnutia:



Úlohy do ďalšieho stretnutia:

- Spojenie jednotlivých častí projektu do funkčného celku
- Zabezpečenie potrebných nástrojov a prvkov na spojenie a testovanie produktu
- Skompletizovanie dokumentácie

5.17 Zápisnica z testovania

Dátum: 16. 4. 2014, 23. 4. 2014, 30. 4. 2014, 7. 5. 2014, 14. 5. 2014, 21. 5. 2014

Čas: 08:30

Miesto: FIIT STU, 5.44

Prítomní:

Ing. Ján Murányi
Bc. Jozef Baláž
Bc. Adam Močkoř
Bc. Timotej Tkáč
Bc. Martin Pivarník
Bc. Matej Rybár
Bc. Tomáš Boros

Náplň stretnutí:

- Komunikácia o dosiahnutých cieľoch a celkovom napredovaní projektu
- Predvedenie aktuálneho stavu simulácie siete a implementácie SIRUP – ov a SIRUP Master
- Testovanie projektu po spojení jednotlivých častí

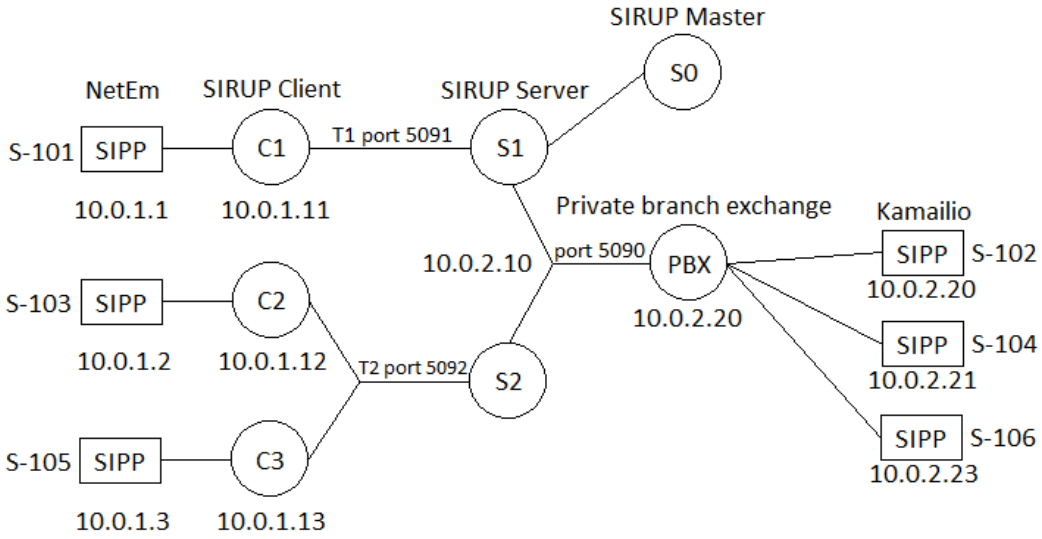
Úlohy do ďalšieho stretnutia:

- Testovanie produktu

Kroky testovania:

1. Vytvorenie hovoru
2. Zmena linky
3. Zmena linky pri viacerých hovoroch
4. Testovanie pri užívateľských agentoch s viacerými hovormi na SIRUP klientovi
5. Testovanie kvality hovoru po zhoršení parametrov linky
6. Testovanie kvality spojenia a funkčnosť zmeny linky na reálnych telefónoch

Schéma testovania:



6 Štandardy kódovania

Dokument k projektu má nasledovné formátovanie:

- okraj zhora 0,98 cm
- okraj zdola 0,79 cm
- okraj sprava 0,98 cm
- okraj zľava 0,98 cm
- kódovanie UTF8
- typ písma Cambria
- veľkosť písma 12
- farba písma čierna
- riadkovanie 1,15

Použité jazyky a nástroje pre implementáciu jednotlivých častí projektu:

- Ubuntu 12.04
- Linux 3.2.0-29
- JavaScript a HTML
- Platforma Node.js
- Mongo Database
- Couch Database
- JSON objekty
- REST architektúra
- Socket.io knižnica
- jQuery knižnica
- Angular.js knižnica
- ChartJS knižnica
- Highcharts knižnica
- Java programovací jazyk
- PHP programovací jazyk
- Testovací nástroj SIPp

7 Manažment verzií, konfigurácií a zmien

Náš tím využíval na manažment verzií a konfigurácií najmä nasledovné nástroje:

- Git - systém pre správu verzií zdrojových kódov nášho projektu. Každý vývojár aktualizoval v zdieľanom repozitári svoju časť zdrojových kódov pri vytvorení novej funkcionality alebo vylepšení tých existujúcich. Vzhľadom k obmedzeniu služby Bitbucket bolo možné poskytnúť prístup pre maximálne 5 členov tímu ku zdieľaným repozitárom.
- Trello - Kolaboračná platforma pre evidenciu úloh, ich priradenia jednotlivým členom tímu a stavu ich plnenia. Ocenili sme najmä notifikácie o zmenách prostredníctvom emailu, čo odbúralo nutnosť dodatočného vypisovania súhrnného emailu o zmenách v stave úloh.
- Fotky virtuálnych diskov - pred významnými zásahmi do konfigurácie virtuálnej infraštruktúry sme využívali možnosť vytvorenia bodov obnovenia jednotlivých virtuálnych strojov.
- Dokumenty Google - aktuálny stav konfigurácie virtuálnej infraštruktúry sme zaznamenávali do zdieľanej tabuľky najmä pre udržanie prehľadu v pridelených IP adresách v testovacej virtuálnej infraštruktúre. V zimnom semestri sme na tejto platforme písali aj projektovú dokumentáciu pretože umožňovala úpravu zdieľaného dokumentu viacerými osobami naraz. V neskorších fázach sme však narazili na obmedzenia vo formátovaní textu a preto sme prešli ku úprave projektovej dokumentácie prostredníctvom programu Microsoft Word. Ako najvhodnejšie riešenie týchto problémov sa javí použitie jazyku Latex v kombinácii s nástrojom pre vytváranie verzií, napríklad vyššie uvedený Git.

8 Posudky

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Ilkovičova 3, 842 16 Bratislava 4

Posudok tímu TIPSix

Analýza, Špecifikácia, Návrh

Študijný program: Počítačové a komunikačné systémy a siete
Tím č.3: Bc. Jozef Baláž, Bc. Tomáš Boros, Bc. Adam Močkoř, Bc. Martin Pivarník,
Bc. Matej Rybár, Bc. Timotej Tkáč
Vedúci tímového projektu: Ing. Ján Murányi
Ak. rok: 2013/14

Obsah

1. Úvod ... 3

2. Posudok z vecného hľadiska ... 3

2.1. Posudok kapitoly Úvod ... 3

2.2. Posudok kapitoly Analýza ... 3

2.2.1. Posudok kapitoly Prehľad protokolu IPv6 ... 3

2.2.2. Posudok kapitoly Možnosti nasadenia IPv6 ... 4

2.2.3. Posudok kapitoly Existujúce portály o IPv6 ... 5

2.2.4. Posudok kapitoly Simulátor GNS3 ... 5

2.3. Posudok kapitoly Špecifikácia požiadaviek ... 5

2.3.1. Posudok kapitoly Funkcionálne požiadavky ... 5

2.3.2. Posudok kapitoly Nefunkcionálne požiadavky ... 6

2.4. Posudok kapitoly Návrh ... 6

2.4.1. Posudok kapitoly Štrukturálny návrh portálu ... 6

2.4.2. Posudok kapitoly Funkcionalita z pohľadu používateľa ... 6

2.4.3. Posudok kapitoly Návrh grafického rozhrania web stránky portálu ... 6

3. Posudok z formálneho hľadiska ... 6

3.1. Gramatika ... 7

3.2. Použitie anglických výrazov ... 8

3.3. Konzistencia formátovania textu ... 9

3.4. Číslovanie strán, obrázkov, kapitol ... 9

3.5. Literatúra ... 10

4. Posudok dokumentácie riadenia projektu ... 10

5. Záver ... 11

1. Úvod

Dokument predstavuje posudok k projektovej dokumentácii projektu Sieťový protokol IPv6. Projekt je vytvorený tímom č. 6 v rámci predmetu Tímový projekt I. Posudok je členený na tri časti. Prvá časť sa zaoberá hodnotením projektovej dokumentácie z vecného hľadiska, druhá časť hodnotí dokumentáciu z formálneho hľadiska. Posledná časť posudku hodnotí dokumentáciu k riadeniu projektu.

2. Posudok z vecného hľadiska

2.1. Posudok kapitoly Úvod

Autor v krátkosti uvedie históriu vývoja protokolu IPv6, preukazuje potrebu využívania tohto protokolu. Uvádza príčiny minutia adries IPv4 a techniky, mechanizmy, ktoré pomohli odkladať potrebu nasadenia nového protokolu. Nemám žiadne výhrady k tejto kapitole.

2.2. Posudok kapitoly Analýza

2.2.1. Posudok kapitoly Prehľad protokolu IPv6

Autor uvádza, že adresný priestor protokolu IPv6 nám poskytuje "2^{128}, tj. až 10^{38} jedinečných adries", čo nie je úplne korektné nakoľko nie všetky adresy sú z globálneho hľadiska jedinečné.

2.2.1.1. Základná hlavička datagramu

Je tu uvádzané, že každý protokol je definovaný "dvojicou pravidlá - datagram". Vedel by som si predstaviť aj zrozumiteľnejšiu formuláciu ako napríklad "pravidlá používania - formát hlavičky", nakoľko táto myšlienka už v texte nie je ďalej rozvíjaná.

Tiež nepovažujem za korektné tvrdenie, že datagram vyššej vrstvy sa vkladá do datagramu nižšej. Skôr by som hovoril o enkapsulácii alebo pridávaní samotných hlavičiek a nie datagramu ako celku.

2.2.1.2. Adresácia IPv6

V kapitole sa píše, že "nikto nepredpokladá, že by používatelia museli pracovať s nejakými IPv6 adresami – vďaka autokonfigurácii". S týmto tvrdením nesúhlasím, nakoľko autokonfigurácia je do určitej miery prítomná už v IPv4 (DHCP, link-local, ...) a stále existujú prípady, kedy adresu používateľ musí zadávať ručne.

Je tu tiež opísané akým spôsobom je možné skrátiť zápis IPv6 adresy. Autor sa však zabudol zmieniť o tom, že v prípade náhrady sekvencie nulových bajtov znakom "::" môžeme túto náhradu vykonať len raz. Hexadecimálny spôsob zápisu MAC adresy tak, ako autor uvádza nie je "príčinou", prečo ju môžeme použiť na vytvorenie EUI-64 IPv6 adresy.

Označiť vygenerovanú EUI-64 adresu ako “jednoznačnú identifikáciu počítača” považujem za prisilné vyjadrenie, nakoľko MAC adresu viem zmeniť.

2.2.1.3. Typy adries

Autor sa tu zmieňuje o tom, že adresa IPv6 má niekoľko typov formátov. Z nasledujúceho textu mi však vyplýva, že mal v úmysle písať skôr o rozsahu platnosti adresy.

Tvrdiť, že globálne adresy majú celosvetový dosah sa mi zo sieťového hľadiska zdá bezpredmetné.

2.2.1.4. Autokonfigurácia

“Kvôli zložitosti zápisu adries je pri IPv6 autokonfigurácia priam nutnosťou.” Neustále zdôrazňovanie zložitosti zápisu IPv6 adresy v celej analýze považujem za prehnané a preceňované.

2.2.1.5. Doplňujúce (rozširujúce) hlavičky

Kapitola dôkladne opisuje jednotlivé hlavičky a k jej obsahu nemám výhrady.

2.2.1.6. ICMPv6

Môžeme tu nájsť vymenovanie základných úloh ICMPv6 protokolu spolu s opisom jeho zraniteľností z bezpečnostného hľadiska. V kapitole mi chýbal podrobnejší opis funkcie vyhľadávania susedných uzlov a smerovačov.

2.2.1.7. Mobilita

V kapitole sa síce dočítame o princípe fungovania, ocenil by som však aspoň základný opis prípadu použitia.

2.2.1.8. Bezpečnosť Internet Protokolu verzie 6

Ku kapitole mám jedinú výhradu a to, že skratka ESP neznamená “Encapsulation Security Header” ale Encapsulating Security Payload.

2.2.2. Posudok kapitoly Možnosti nasadenia IPv6

Autor uvádza, že protokol NAT nevyrieši problém míňajúcich sa IPv4 adries, čo nie je korektné, keďže NAT nie je protokol, ale spôsob prekladu adries z privátnych na verejné.

2.2.2.1. Dual Stack

Autor nie jasne špecifikuje pre ktoré uzly, alebo pre akú aplikáciu platí konfiguračný prepínač, ktorý dokáže zakázať IPv4 alebo IPv6 zásobník.

2.2.2.2. Tunelovanie

K tunelovaniu typu 6in4 nemáme výhrady. Podkapitola presne opisuje spôsob tunelovania. Uvádza 4 rôzne možnosti 6in4 tunelovania, pričom si myslím, že tunelovanie typu host - smerovač je také isté ako smerovač - host.

Tunelovanie 6to4 je presne opísane, avšak pri ukážkovom príklade je uvedený obrázok s inými IPv6 adresami, čo môže byť pre čitateľa mätúce. Tunely ISATAP, Teredo a statické tunely sú opísané stručne, ale správne.

2.2.2.3. Preklad

Autor neopisuje korektne fungovanie prekladu NAT64. Uvádza, že adresy z IPv6 na IPv4 sú mapované bez stavovo pri komunikáciách inicializovaných zo strany poskytovateľa (od klientov) do siete internet, pričom toto mapovanie dynamické a stavové. Okrem toho, tento preklad funguje aj v inom kontexte, IPv6 adresy sa mapujú staticky a bez stavovo na IPv4 adresy, ak chceme aby zariadenia boli dosiahnuteľné za smerovačom, ktorý prekladá pomocou NAT64, používajúce IPv4 adresy.

2.2.2.4. Jednorázový prechod na IPv6

Žiadne výhrady nemám voči tejto kapitole.

2.2.3. Posudok kapitoly Existujúce portály o IPv6

Autor vymenuje 16 rôznych internetových portálov zaoberajúce sa s problematikou IPv6. Každý portál má uvedenú URL adresu a je stručne popísaný. Nemám žiadne pripomienky k tejto kapitole.

2.2.4. Posudok kapitoly Simulátor GNS3

Okrem GNS3, kapitola opisuje aj iné simulačné programy ako OMNet++, Packet Tracer. Uvádza Dynamics ako emulátor Cisco zariadení, pričom korektný názov tejto aplikácie je Dynamips. GNS3 uvádza ako správnu voľbu pre riešenie projektu.

2.3. Posudok kapitoly Špecifikácia požiadaviek

Kapitola je správne rozdelená na funkcionálne a nefunkcionálne požiadavky.

2.3.1. Posudok kapitoly Funkcionálne požiadavky

V tejto kapitole sú špecifikované funkcionálne požiadavky v niekoľkých stručných bodoch. V požiadavke č. 5, by portál mal poskytovať testovacie úlohy a nie testové. V dokumente sa na viacerých miestach uvádza možnosť registrácie používateľa. Toto by malo byť uvedené tiež medzi funkcionálnymi požiadavkami spolu s rozdielom medzi zaregistrovaným a nezaregistrovaným používateľom. Zvyšné vymenované požiadavky nám prídu ako dostatočné vymedzenie funkcionality riešenia.

2.3.2. Posudok kapitoly Nefunkcionálne požiadavky

Táto kapitola zhŕňa nefunkcionálne požiadavky. Najdôležitejšia a dobrá požiadavka je, že portál sa bude zaoberať len a len problematikou IPv6 protokolu. Požiadavka na jednoduché používateľské prostredie je možno príliš abstraktná a ťažko sa dá odmerať miera jej splnenia.

2.4. Posudok kapitoly Návrh

Kapitola návrh obsahuje niekoľko pohľadov na implementáciu portálu. Návrh je popísaný z najmä z vonkajšieho hľadiska. V prvej kapitole sú síce vymedzené štrukturálne modely, no táto kapitola by mohla obsahovať aj návrh architektúry a implementácie jednotlivých logických celkov.

2.4.1. Posudok kapitoly Štrukturálny návrh portálu

Malý nedostatok považujeme, že v úvode kapitoly sa uvádza päť modulov a vymenované sú štyri. Ako piaty modul je zrejme myslené používateľské rozhranie, kde názov napovedá, že to nie je modul. Ostatné moduly sú určené správne, podľa funkcionálnych požiadaviek. Pridali by sme zrejme aj moduly vyhľadávania a modul fóra.

2.4.2. Posudok kapitoly Funkcionalita z pohľadu používateľa

V tejto kapitole sú uvedené dva požívateľské prípady použitia. Bolo by zaujímavé opísať aj prípad niektorý prípad použitia spojený so simulačným modulom.

2.4.2.1. Proces prístupu používateľa k portálu

Jediná výhrada je k názvu, keďže prístup nereflektuje registráciu.

2.4.2.2. Proces otestovania používateľa

Proces testovania je opísaný dobre.

2.4.3. Posudok kapitoly Návrh grafického rozhrania web stránky portálu

V tejto kapitole je opísané grafické rozhranie portálu. Podľa tohto opisu sa dá celkom presne vytvoriť predstava o výslednom rozložení prvkov.

3. Posudok z formálneho hľadiska

Dokumentácia k tímovému projektu je oficiálnym dokumentom, ktorý odzrkadľuje nielen našu vykonanú prácu v rámci tímového projektu, ale aj naše znalosti formulácie a štylizácie textu. Táto forma našej prezentácie má taktiež svoje miesto v celkovom hodnotení. V tejto časti sa preto pozrieme na gramatiku, použitie slovenských slov, konzistenciu formátovania textu, číslovanie strán, obrázkov a kapitol a formu zápisu literatúry.

3.1. Gramatika

V tejto kapitole sa zameriame na gramatické chyby v projektovej dokumentácii. Poukážeme na chýbajúce a nadbytočné čiarky prípadne chýbajúce slová a iné nezrovnalosti v texte. Posudzovaný dokument obsahuje pomerne veľa gramatických chýb.

Preklepy, ktoré sa vyskytujú v texte:

„vyskytovať len výnimočné“ (kap. 1.1.1, s. 7), “eternetových” (kap. 1.1.2, s. 9), “jednoznačné identifikovateľný” (kap. 1.1.2, s. 9), “Pri stavovej konfigurácie” (kap. 1.1.4, s. 12), „Tieto Smerovače“ (kap. 1.1.5, s. 14), „ľubovoľný smerovať“ (kap. 1.1.5, s. 14), “32 bitové” (kap. 1.1.5, s. 14), “Ipv4 protokol” (kap. 1.2.1. s.24), „nezrealizovateľný“ (kap. 1.2.4 s. 31), „textové podlohy“ (kap. 1.3, s. 32), „Známa wikipédia“ (kap. 1.3, s. 32), „reálnych zariadenia“ (kap. 1.4 s. 36), „fór“ (kap. 2.1, s. 36), „silmultačné“ (kap. 2.1, s. 36), „výber odpovede z možných” (kap. 2.1, s. 37), „po otvorené“ (kap. 3.2.1. , s. 39), „moznosť“ (kap. 3.2.1, s. 39), „na obrázku č. 3-2“ (kap. 3.2.2, s. 40), „spať“ (obr. 3-3, s. 40), „pod sekcie“ (kap. 3.3, s. 42), „videa“ (kap. 3.3, s. 42), „radu“ (kap. 3.3, s. 43), „z farebným škálou“ (kap. 3.3, s. 42)

Ďalšie pravopisné chyby:

- “prvou variantov”, “Druhou variantov” (kap 1.1.6, s.16) - správny tvar prvým variantom, druhým variantom
- “sú na sebe úplne nezávislé” (kap. 1.2.,. s.24) - správny tvar “závislý od“

Vynechanie čiarky:

- „len výnimočné a preto bola“ (kap. 1.1.1, s. 7)
- „Samozrejme môžeme“ (kap. 1.1.2, s. 8)
- „to isté a preto“ (kap. 1.1.2, s. 9)
- „prejsť kým príde“ (kap. 1.1.5, s. 14)
- “hlavičky ale“ (kap. 1.1.8.2, s. 19)
- “, respektíve NAT blokujú“ (kap. 1.2.2, s. 26)
- „požiadavka požaduje aby portál“ (kap. 2.1, s. 37)
- „v rámci učebných textov portálu ale aj v rámci RFC dokumentov“ (kap. 2.1, s. 37)
- „skupinu otázok na ktoré odpovedá “ (kap. 3.2.2, s. 40)

- „hlavných častí ktoré sú opísané“ (kap. 3.3, s. 41)
- „na inú stránku kde bude“ (kap. 3.3, s. 42)
- „prezerať edukačný portál, bez prihlásenia no po prihlásení“ (kap. 3.3, s.43) - nesprávne umiestenie čiarky v súvetí
- „ktoré sa budú striedať a takto ľahko“ (kap. 3.3, s. 43)
- „máme navrhnuté no možno“ (kap. 3.3, s. 43)
- „celej stránky aby ladilo “ (kap. 3.3, s. 43)

Nesprávne použitie čiarky:

- “zdrojovej, cieľovej adresy” (kap. 1.1.5, s. 15)

Chýbajúci text:

- “MACSec, zabezpečuje” kap. (1.1.8.1, s.17) - keďže sa skôr uvádza, že ide o bezpečnostný mechanizmus, rozširujúci popis “zabezpečuje” nie je dostatočný

3.2. Použitie anglických výrazov

Dokument v slovenskom jazyku by mal obsahovať čo najmenej cudzojazyčných slov. Namiesto nich je vhodné použiť slovo v slovenskom jazyku rovnakého významu. Ak slovenský ekvivalent slova neexistuje, cudzojazyčné slovo možno písať iba v jeho pôvodnom tvare (bez ohýbania).

- “source routingu” (kap. 1.1.5, s.14)
- “stack” (kap. 1.1.6, s.16)
- “VLANy” (kap. 1.1.8.1, s.17)
- “paket spoofingu” (kap. 1.1.8.3, s.20)
- “upgrade” (kap. 1.2.1, s.24)
- “NATko” (kap. 1.2.2, s.25)
- “IPv4-only”, ”IPv6-only” (kap. 1.2.3, s.30)
- “Worde” (kap. 1.3, s.33)

- „interface“ (kap. 2.2, s. 37)
- „ratingu“ (kap. 3.2.2, s. 40)
- „quiz“ (kap. 3.3, s. 42)
- „editovať“ (kap. 3.3, s. 42)

3.3. Konzistencia formátovania textu

Nekonzistentné odsadeniu textu rovnakého štýlu:

- odsek “Aby sa ušetril” (kap 1.1.5, s. 13)
- nadpisy “Smerovanie”, “Fragmentácia” (kap 1.1.5, s. 14) a ďalšie nadpisy rovnakej úrovne v rozpore s predchádzajúcimi nadpismi
- nadpisy “1.1.8.1. Bezpečnosť na linkovej vrstve”, “1.1.8.2.IPSec” a ďalšie nadpisy rovnakej úrovne v rozpore s predchádzajúcimi nadpismi
- zoznamu dokumentov RFC (kap 1.1.5, s.14) a (kap 1.1.8.4, s.22)

Riadkovanie v kapitole 2.2 v časti „Portál má jednoduchý interface“ a „Portál je zameraný výhradne na IPv6“ nie je zhodné s riadkovaním v celom dokumente.

Ďalšie chyby týkajúce sa konzistencie textu:

- použitie rôznych pomenovaní “domovský agent” a “domáci agent” (kap. 1.1.7. s.17)
- písanie a nepísanie dvojbodky pred nečíslovaným zoznamom (kap 1.1.8.4, s.21) a (kap 1.2, s.23)
- obrázky bez popisov (kap. 1.2.11 s.25)

Určitá nekonzistencia sa vyskytuje aj pri zvýrazňovaní textu tučným písmom. V texte sme našli miesta, kde sú body zvýraznené tučným a miesta, kde zase naopak, nie sú.

Ďalšie chyby, ktoré sa v texte objavili:

- „Niektoré z položiek majú sekcie obsahujú pod sekcie“ (kap. 3.3, s. 42) - nelogická veta
- Obrázkom prevzatým z Internetu chýba referencia

3.4. Číslovanie strán, obrázkov, kapitol

Zásadným problémom je chýbajúce číslovanie strán. Dokument síce obsahuje kapitoly Obsah, Zoznam tabuliek, Zoznam obrázkov, ktoré určujú umiestnenie jednotlivých častí dokumentu podľa strán, ich význam je bez označenia strán minimálny. V elektronickej verzií

dokumentu je možné číslo stránky určiť pomocou príslušnej funkcie prehliadača dokumentov. Pri počítaní strán týmto spôsobom však nie je číslovanie jednoznačné, keďže obvykle sa napríklad prvá strana do číslovania nezahŕňa.

Označenie častí kapitol „4. Prílohy“ a „4. Literatúra“.

3.5. Literatúra

Zápis Literatúry je v poriadku, no je tam niekoľko nezrovnalostí.

V Literatúre od 1 - 4, 14, 16, 20, 21 sú URL odkazy spolu s textom a nie sú oddelené a vyčlenené na novom riadku. Ostatné odkazy sú na novom riadku, čo by malo byť konzistentné a aj v tých prvých, by mal byť URL odkaz na samostatnom novom riadku. Pri niektorých odkazoch je pred nimi text “Available at:” a niekde nie. Ďalej sme si všimli, že bod [24] nemá URL odkaz a ani označenie, či sa jedná o knihu alebo nejaký iný knižný zdroj.

Veľkým mínusom je, že sa v texte nevyskytujú odkazy, teda označenie daných zdrojov. Preto nevieme, v ktorej časti dokumentácie bol použitý ktorý zdroj.

4. Posudok dokumentácie riadenia projektu

Dokumentácia riadenia projektu je neoddeliteľnou súčasťou každého projektu, ktorý si vyžadujú tímovú spoluprácu. Tento dokument má niekoľko častí, ktoré je potrebné popísať a vypracovať. Dokumentácia je vhodne rozdelená medzi kapitoly.

Úvod obsahuje základné informácie o riadiacej dokumentácii k tímovému projektu. A taktiež je tam prehľadná tabuľka s históriou vytvárania dokumentu. Mal by som k tejto kapitole len jednu pripomienku a to je značenie verzii dokumentu, kde som sa ešte nestretol s označením napr. 0.1. Verzie by mali začínať číslom 1.

Ďalšou kapitolou je Ponuka, kde sú predstavený jednotlivý členovia tímu, znenie primárneho a sekundárneho zadania, ktorý si tím vybral. Obsahuje tiež motiváciu, plán hrubého návrhu riešenia, plán projektu, realizovateľnosť a predpokladané zdroje pre primárne aj sekundárne zadanie. Po formálnej stránke som nenašiel v tejto kapitole závažné chyby a text je vhodne štruktúrovaný. Čo by som pripomienkoval sú dlhé vety. Vhodnejšie a lepšie na čítanie by boli kratšie vety a text by sa tým pádom dal lepšie pochopiť. Taktiež sa tam vyskytujú anglické slová a gramatické chyby, no nie vo veľkom počte. Na konci kapitoly ešte nájdeme Prílohy. V prílohe Aktuálny rozvrh členov tímu s návrhom preferovaných časov stretávania sa celého tímu som nenašiel výsledný čas, kedy budú stretnutia prebiehať.

Kapitola Komunikácia členov tímu nie je povinná, ale obsahuje informácie, ktoré nám pomáhajú pochopiť ako prebieha kooperácia medzi členmi tímu. Je vhodne členená a neobsahuje gramatické chyby.

Poslednou časťou dokumentu riadenia projektu sú Zápisnice. Každá zápisnica obsahuje prehľadnú tabuľku, kde sú všetky dôležité informácie o tom, kto sa na stretnutí zúčastnil, dátum stretnutia, čas a miesto. Trošku nevhodnou voľbou sú krížiky pri určovaní prítomných členov, čo nie je jednoznačný identifikátor, či mená s krížikom sa na stretnutiach zúčastnili alebo nie. Ďalej zápisnice obsahujú Priebeh stretnutia, Rozdelenie úloh jednotlivým členov tímu a Záver. Text je vhodne členený, štruktúrovaný, prehľadný a obsahuje požadované informácie.

Za hlavný nedostatok dokumentácie riadenia projektu považujem absenciu číslovania strán.

5. Záver

Úlohou práce oponentského tímu bolo vytvoriť edukačný portál zameraný na IPv6 protokol a na jeho nasadenie a bezpečnosť. Vzhľadom na toto zadanie je pochopiteľné, že bolo potrebné spísať dôkladnú analýzu problematiky. Tá je až na niektoré nedostatky spomenuté vyššie napísaná korektne a neobsahuje chyby závažnejšieho charakteru. V časti špecifikácia neboli všetky požiadavky dostatočne konkrétne a teda merateľné.

O niečo horšie je na tom návrh. Zatiaľ čo predmetom analýzy je získať prehľad o niečom už vytvorenom, úlohou návrhu je vytvoriť niečo nové. Z tohto dôvodu mal byť dôraz kladený na túto časť a teda na vlastný prínos tímu. Keď sa však pozrieme na dokument, návrh je poňatý príliš všeobecne a aj po rozsahovej stránke tu môžeme vidieť nepomer medzi analýzou a návrhom.

Čo sa týka dokumentácie riadenia, mala niektoré formálne nedostatky, ktoré však nepredstavujú väčší problém.

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Posudok dokumentov tímu č. 3

vypracovaný tímom č. 6

Študijný program: Počítačové a komunikačné systémy a siete
Študijný odbor: 9.2.4 Počítačové inžinierstvo
Ak. rok, semester: 2013/2014, zimný semester
Členovia tímu 6: Bc. Lukáš Danielovič
Bc. Anton Pôbiš
Bc. Lukáš Lenčéš
Bc. Marek Dukát

Obsah

Úvod 3
1 Posudok k formálnej stránke dokumentácie projektu 4
1.1 Projektová dokumentácia 4
2 Posudok k obsahovej stránke Dokumentácie riešenia 5
2.1 Projektová dokumentácia 5
2.1.1 Analýza 5
2.1.2 Špecifikácia 7
2.1.3 Návrh 8
3 Posudok k Dokumentácie riadenia projektu 9
3.1 Formálna stránka dokumentácie 9
3.2 Obsahová stránka dokumentácie 9
4 Zhodnotenie 10

Úvod

Tento dokument obsahuje posudok prvej časti práce tímu č. 3 v predmete Tímový projekt. Tím č. 3 rieši zadanie s názvom „Manažment VoIP relácií“.

Posudzovali sme dva odovzdané dokumenty: Dokumentáciu riadenia a Dokumentáciu riešenia. Tieto dokumenty boli posudzované z obsahovej stránky a z formálnej stránky.

V prvej časti tohto dokumentu sa zameriavame na formálnu úpravu Dokumentácie riešenia.

V druhej časti sa zaoberáme obsahovou stránkou Dokumentácie riešenia.

Tretia časť posudku je venovaná formálnej aj obsahovej stránke Dokumentácie riadenia.

1 Posudok k formálnej stránke Dokumentácie riešenia

V tejto kapitole sú zhrnuté formálne nedostatky posudzovaných dokumentov.

1.1 Projektová dokumentácia

Hodnotený dokument neobsahuje číslovanie kapitol a má konzistentné farby názvov kapitol. Začiatok nových kapitol nezačína na novej strane a pred začatím každej kapitoly (a tiež u mnohých podkapitolách) chýba text uvádzajúci kapitolu.

Kapitola analýzy je nesystematicky rozdelená na príliš mnoho podkapitol rovnakej úrovne. Toto zvyšuje neprehľadnosť dokumentu. Podkapitoly nie sú spojené textovými asociáciami a vytvárajú dojem „z každého rožku trošku do jednej mysky“.

Obrázky nie sú primerane popísané a číslované. Popis obrázkov sa vyskytuje viac krát. Chýba zoznam obrázkov, čo má pravdepodobne za následok chyby ako duplicitné očíslovanie obrázkov (napr. označenie „obr. 5.“ sa v dokumente nachádza 3krát, podobne aj *„obr. 4“*). Príklad ďalšieho nedostatku je v kapitole návrh, podkapitola architektúra: odkaz na obrázok 4 je nevysvetliteľne viacnásobne popísaný a obrázok sa nachádza v predošlej kapitole a zároveň pod obrázkom je menovka obrázku už číslo 5.

Vo veľkom počte odsekov chýba riadkovanie a tabulátor na začiatku odseku.

Z hľadiska citovania literatúry sa v texte vyskytuje dostatok citácií a odkazov na zdroje. Uvedené zdroje však nie sú podľa normy STN ISO 690 a mnoho z nich sú len odkazy (a nemali by byť zahrnuté medzi citovanú literatúru) na domovské stránky istých informačných techník.

Z gramatického hľadiska obsahuje dokument enormne veľa chýb, nespisovných slov, neúplných viet alebo bezvýznamných viet'. Uvedieme len niektoré nedostatky:

„*bez nutnosti žiadať o ne.“* – nedokončené slovo

„REST architektúre je že vzťahy“ – chýba tu čiarka

„...čo opäť nahráva CouchDB databáze.“ – použiť iné slová vo vete namiesto slova „nahráva“

„*...znázornenej na diagrame*“ – žiadny odkaz na diagram (v celom dokumente sa ani nevyskytuje obrázok nazvaný „diagram...“)

2 Posudok k obsahovej stránke Dokumentácie riešenia

V tejto časti dokumentu je posudok zameraný na obsahovú časť Dokumentácie riešenia. Je tu hodnotená zrozumiteľnosť a relevantnosť textu ako aj dostatok ukážok preberanej témy (obrázky, odkazy).

2.1 Projektová dokumentácia

Podkapitola sa zameriava na posudzovanie projektovej dokumentácie. Jednotlivé časti dokumentu sú rozoberané zvlášť a sú stručne popísané ich nedostatky ako aj pozitíva.

2.1.1 Analýza

V tejto časti je zhodnotená analýza – jej dokumentovanie a použité pramene.

Podkapitola *Manažment* stručne opisuje „manažment multimediálnych relácií“ a poskytuje náznak motivácie riešenia novej techniky manažmentu multimediálnych relácií. Názov kapitoly nie je jasný, mal by viac naznačovať, o čo v kapitole ide.

Obsah podkapitoly *Kontext* nesúvisí so svojim nadpisom, ale možno z neho vyčítať spresnenie zadania a tým poskytuje aj pohľad na jednu z požiadaviek navrhovanej aplikácie.

Ďalše kapitoly *Existujúce riešenia, Prepínanie medzi cestami, SIRUP master* v obraze popisujú techniku SIP Single Port a SIRUP. Celkový proces komunikácie na jednom porte nie je dostatočne opísaný a vyvoláva mnohé otázky. Z hľadiska, že v zadaní projektu je aj analýza architektúry techniky SIP Single Port, je táto analýza na nedostatočnej úrovni. Veľmi veľkým nedostatkom je aj literatúra a odkazy na túto tému. Chýba tu porovnanie s podobnými architektúrami na manažmenť multimediálnych relácií.

Kapitola *Manažment Server* obsahuje opäť mnohé podkapitoly opistujúce techniky, ktoré plánuje tím implementovať do Manažment servra. Tím zvolil ako platformu pre skripty bežiace na servery Node.js. Dobre sú spísané vlastnosti tohto výberu, ale chýba tu porovnanie s ostatnými scriptovacími jazykmi a dôvody, prečo ich nerealizovať.

Kapitola *Manažment API* opisuje tiež techniky použiteľné v serveri. Tím sa tu zameriava na rozhranie postavené na štandardoch Socket.io a JSON. Chýba tu kontextová súvislosť s predchádzajúcou kapitolou, keďže intuitívne sa jedná o jej pokračovanie, resp. doplnenie informácií.

Kapitola *Front-end manažment* popisuje, že grafické rozhranie sa tím rozhodol implementovať v HTML 5. Veľmi dobre sú tu popísané grafy, ktoré z tohto jazyka môžu využiť v aplikácií.

V kapitole *Simulácia siete* tím stručne opísal možnosť simulácie navrhnutej aplikácie pomocou emulátora NetEm. Oceňujeme dobrý zdroj informácií k tomuto nástroju, keďže v dokumente sa viac o tomto nástroji čitetel nedozvie. Praktickejšie by však bolo uviesť odkaz v texte a nie ako citovanú literatúru. Podkapitou tohto je aj časť Informácie o kvalite linky, kde tím podáva informácie o vlastnostiach linky.

2.1.2 Špecifikácia

V špecifikácií sa očakávalo rozdelenie na funkcionálne a nefunkcionálne požiadavky, ktoré však autori neurobili. Charakterizovali však hlavné funkcie pre toto riešenie, z ktorých je jasné, čo budú navrhovať a implementovať. Ako veľké pozitívum sa berie schéma zobrazujúca Manažment API správ, v ktorej je vidieť komunikáciu jednotlivých komponentov systému. Oceňuje sa popis jednotlivých správ medzi uvedenými komponentmi a význam týchto správ.

Taktiež autori neuviedli nefunkcionálne požiadavky , ktoré sa očakávajú od riešenia. Mohli tu uviesť napríklad hardvérové požiadavky, požiadavky na systém, na ktorom bude finálne riešenie spustené a taktiež potrebné softvérové nástroje na tvorbu tohto systému.

2.1.3 Návrh

Celkový návrh je popísaný pomerne dobre, no v niektorých častiach pripomína analýzu. Text je zrozumiteľný aj pre človeka, ktorý sa nezaoberá MPLS sieťami a VoIP technológii. Obrázky sú prehľadné a text poskytuje množstvo odkazov na tieto ukážky, ktoré takto zjednodušujú pochopenie problematiky.

Na začiatku kapitoly je obrázok opisujúci topológiu siete. Okrem toho by mal na začiatku byť uvedený obrázok celkovej architektúry riešenia. Postupne sú opísané jednotlivé komponenty na kvalitnej úrovni. Autori taktiež uvádzajú porovnanie alternatívnych možností, rozdiel medzi nimi a následne sa vyjadrujú, prečo si zvolili práve danú možnosť, čo sa hodnotí veľmi pozitívne. Vítané by bolo použiť viac obrázkov, napríklad grafické znázornenie databázy poprípade tabuliek, štruktúru dokumentov,...

Časť „Simulácia siete“ by sa dala chápať ako analýza, nakoľko v návrhu nie je potrebné opisovať ako funguje spomínaný simulátor.

Problematika návrhu riešenia zo strany klienta je napísaná podrobne a vyjadruje podstatu riešenia. Okrem toho je k nej vypracovaný aj sekvenčný diagram čo sa berie ako veľké pozitívum.

Časť SIRUP master opisuje návrh pomerne kvalitne a dáva obraz o tom ako v budúcnosti autori budú implementovať túto časť.

Všetky skratky a cudzie slová sú vysvetlené a dodatočne popísané v neskrátenom tvare ako aj odkázané na iné zdroje.

Pre jednotlivé návrhy riešenia je popísaný postup riešenia ako aj spomenuté pracovné prostredie pre implementáciu.

Ďalej by bolo vhodné doplniť prípady použitia (Use Case diagramy), čím by znázornili správanie sa systému v rôznych situáciách, konkrétne mohli znázorniť prípady identifikované v SIRUP komunikácií.

3 Posudok k Dokumentácie riadenia projektu

Táto kapitola je obsahuje posudok dokumentácie k riadeniu projektu tímu č. 3. Tento posudok je delený na formálnu a obsahovú stránka dokumentácie. Projektová dokumentácia obsahuje tieto posudzované kapitoly: Úvod, Ponuka a komunikácia členov tímu.

3.1 Formálna stránka dokumentácie

Dokumentácia je po formálnej stránke na dobrej úrovni až na zopár malých chýb. Obsah je správne štruktúrovaný. Dokumentácia obsahuje minimum gramatických chýb. Veľkosti textov a nadpisov sú správne, taktiež odseky a zarovnanie textu sú v poriadku. Jedinou z výhrad k formálnej stránke je začínanie kapitol na novej stránke. Taktiež chýba číslovanie stránok.

3.2 Obsahová stránka dokumentácie

Štruktúru stránky popíšem po jednotlivých kapitolách. Úvod obsahuje úvodný text, ktorý popisuje zadanie, a taktiež naznačuje ako bude projekt podľa zadania riešení. Úvod obsahuje taktiež tabuľku história dokumentu, ktorá však nie je vyplnená.

Kapitola *Ponuka* predstavuje celý dokument, ktorý bol odovzdávaný a prezentovaný vedúcim projektu, v tejto kapitole je prepísané celé zadanie projektu čo samozrejme nemohlo chýbať. Sú tu taktiež opísaný jednotliví členovia tímu, ich bakalárske práce a ich programovacie znalosti, ktoré môžu využiť pre vyriešenie projektu. Takýto opis členov tímu je veľmi prehľadné. Motivácia tímu je správne opísaná vo viacero odsekoch, k tejto časti nemáme žiadne výhrady.

Analýza a hrubý návrh je podrobne opísaná vo viacero odsekoch a odrážkach s taktiež grafickým návrhom architektúry systému, takýto návrh je veľmi prehľadný a správne spracovaný. Plán projektu je jednoducho rozdelený cez zarážky čo zabezpečuje rýchle zorientovanie sa v celom pláne.

Zvyšné časti ponuky sú spracované úplne v poriadku – sú to zdroje pre realizáciu projektu a súhrnný rozvrh členov tímu.

Kapitola *Komunikácia členov tímu* neobsahuje žiadny text, z ďalšej kapitoly zápisnice vyplýva, že komunikácia tu určite bola, no bohužiaľ nebola opísaná.

Zápisnice majú určitú štruktúru, ktorá bola vždy dodržaná. Táto štruktúra je jednoduchá a veľmi prehľadná, sú v nej spomenuté tie najpodstatnejšie veci aká bola náplň stretnutia a aké úlohy vyplynuli z daného stretnutia, ktoré museli následne členovia tímu riešiť.

4 Zhodnotenie

Formálna stránka projektovej dokumentácie je podpriemerná. dokument neobsahuje základnú štruktúru používanú na vytváranie dokumentácii. v texte je veľa formálnych chýb a celkový dojem z dokumentu naznačuje absenciu kompletizácie a vynechanie spätnej kontroly pred odovzdaním. príkladom je odkazovanie na obrázok, ktorý sa tam nenachádza a dokonca je prekopírovaný viacnásobne za sebou.

Obsahová stránka dokumentácie je v akceptovateľnejšom stave. písaný text je zrozumiteľný, názorný, problémové časti témy sú špeciálne vysvetlene a dokument obsahuje dostatok citátov. úroveň textu napovedá skúsenosť autorov v danej oblasti a celkovo rozumenie vykladanej problematiky. niektoré časti špecifikácie a návrhu aplikácie patria skôr do analýzy.

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Ilkovičova 3, 842 16 Bratislava 4

Posudok prototypu tímu TIPSix

Študijný program: Počítačové a komunikačné systémy a siete
Tím č.3: Bc. Jozef Baláž, Bc. Tomáš Boros, Bc. Adam Močkoř, Bc. Martin Pivarník,
Bc.Matej Rybár, Bc. Timotej Tkáč
Vedúci tímového projektu:Ing. Ján Murányi
Ak. rok: 2013/14

Obsah

- Úvod
- Funkčnosť
- Dokumentácia
- Záver

1. Úvod

Tento dokument predstavuje posudok k projektovej dokumentácii a protoypu projektu Sieťový protokol IPv6 doménach vytvorenej tímom č. 6 v rámci predmetu Tímový projekt I. Posudok je členený na dve časti. Prvá časť sa zaoberá hodnotením prototypu po stránke funkčnosti, druhá časť hodnotí dokumentáciu z formálneho a vecného hľadiska.

2. Funkčnosť

Tím č. 6 nám prezentoval prototyp projektu vo webovom prehliadači. Stránka je naprogramovaná v jazyku PHP s použitím prídavných technológií ako Ajax, JavaScript. Stránka je na prijateľnej úrovni, umožní používateľom registrovať sa k stránke. Iba zaregistrovaní používatelia majú možnosť prezerať si obsah stránky

Stránka je dočasne uložená na komerčnom webovom hostingu, ktorá ponúka iba limitované možnosti pre používateľov. Tím nemá pripravený presný plán, ako budú implementovať simuláciu zariadení podporujúce IPv6. Základom projektu pravdepodobne bude virtualizačná platforma GNS3.

3. Dokumentácia

K prezentovanému prototypu bola dodaná aj projektová dokumentácia. Jednotlivé prvky, ktoré sme posudzovali v predchádzajúcom posudku boli čiastočne opravené. Stále však obsahuje niektoré nezrovnalosti. Doplnená bola len kapitola Implementácia prototypu.

V časti, kde sa hovorí o dostupných súčastiach na stránke, nie sú jednotlivé časti vysvetlené a charakterizované. Nie každý čitateľ dokumentácie musí vedieť, čo znamená napr. Cron Jobs. Nenašli sme popis ani v analýze. Náročnosť, ktorú si užívateľ môže vybrať, by mohla mať viac úrovní.

Obrázky v celej kapitole Implementácie prototypu nie sú očíslované. Taktiež jednotlivé zdrojové kódy, ktoré sú v kapitole nemajú dostatočný popis jednotlivých prvkov. Gramatické chyby sa nevyskytujú v neúnosnej frekvencii. Chýba taktiež obrázok vyhodnotenia testu, ktorý sa spomína v implementácii prototypu.

4. Záver

Implementácia prezentovaná Tímom 6 je síce funkčná avšak rozsah implementovaných častí mohol byť aj väčší. Zatiaľ totiž portál obsahuje len niektoré vybrané učebné materiály, umožňuje registráciu používateľa a jeho testovanie. Z návrhu je zrejme najzaujímavejší simulátor siete. Ako už však bolo spomenuté, tím ešte nemá jasnú predstavu ako ho chce implementovať a toto považujem za najväčší nedostatok prototypu, ktorý ho už v aspoň v zjednodušenej verzii mohol obsahovať. Takto totiž vyvoláva otázky o jeho zrealizovateľnosti.

Počas testovania sme narazili na drobné nedostatky v podobe chybového hlásenia na podstránke 6. Prakticke ukazky alebo nedodržiavanie slovenského pravopisu v používateľskom rozhraní a v dokumentácii. Kapitola Implementácia prototypu v dokumentácii obsahovala relevantné podkapitoly, avšak ich hĺbka odráža rozsah implementácie prototypu.

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Posudok ku prototypu tímu č. 3

vypracovaný tímom č. 6

Študijný program: Počítačové a komunikačné systémy a siete
Študijný odbor: 9.2.4 Počítačové inžinierstvo
Ak. rok, semester: 2013/2014, zimný semester
Členovia tímu 6: Bc. Lukáš Danielovič
Bc. Anton Pôbiš
Bc. Lukáš Lenčéš
Bc. Marek Dukát

Prezentácia prototypu

Prezentácia prototypu navrhovanej aplikácie umožňujúcej riadenie multimediálnych relácií cez protokol IP bola vykonaná s prijateľným a pochopitelným vysvetlením. Počas prezentácie boli tiež ukázané názorné simulácie prototypu aplikácie. Je vidieť praktický výsledok práce tímu, zodpovedajúci po semestri činnosti, a rozloženia úloh členov tímu na základe povinností a skúseností z praxe členov v tíme.

Technické a implemetačné prostriedky prototypu súhlasia s analyzovanými a navrhnutými prostriedkami v dokumentácii. Tím implementuje prototyp (a plánuje implementovať aj výsledné riešenie) na aplikácii Kamailio, konkrétne jeho module *MEDIAPROXY*, ktorý umožňuje tok multimediálnych dát prostredníctvom SIP protokolu a okrem priamych riešení povoľuje aj mechanizmy NAT.

Prototyp bol obhájený na prijatelnej úrovni. Tím zodpovedal na položené otázky a objasnil problematiku oblasti implementácie.

Námety ku prototypu

Najdôležitejšou poznámkou činnosti navrhovanej aplikácie pri prezentácii prototypu bola impementácia bezpečnosti jediného a napadnuteľného serveru Master. Keďže tento prvok bude v sieti pridaný a bude centralizovať všetky servery SIRUP, môže byť najčastejšie napádaný rôznymi útokmi. Jedným zo základných útokov je napríklad útok DoS. Ak sa útočníkovi podarí odstaviť server Master, je automatické riadenie prechodu multimediálnych relácií zrušené, sieť je funkčná a teda napadnuteľná.

Dokumentácia prototypu

Dokumentácia prototypu je súčasťou záverečnej dokumentácie po prvom semestri a táto je pokračovaním dokumentu tvoreného v priebehu semestra. Na rozdiel od priebežnej dokumentácie, formálna stránka záverečnej dokumentácie posunula do pokročilého stavu. Okrem upravenej formálnej stránky bola doplnená aj obsahová forma dokumentácie a boli detailnejšie opísané isté funkcionality či už v časti analýzy alebo návrhu.

V dokumentácií však chýba príloha technickej dokumentácie.

Formálna stránka

Formálna stránka časti dokumentácie týkajúcej sa prototypu je dostatočná a prehľadná. Obsahuje opis použitých riešení a techník. V tejto čati je prínosnou hodnotou uvedenie niektorých snímok prototypu. Dobre sú tu opísané vlastnosti a funkcionalita prototypu.

Formálna stránka celkovej dokumentácie je obsiahlejšia a poskytuje detailnejšie a prehľadnejšie informácie ako priebežná dokumentácia hodnotená počas semestra. Prehľadnejšie sú rozdelenia kapitol, texty a obrázky. Veľmi dobrým výsledkom pozorovania je, že tím pridal nové diagramy a schémy aplikácie. Toto hodnotíme ako prínosný bod k pochopeniu riešenej problematiky.

Jedinou kritikou z formálnej stránky celkovej dokumentácie je, že sa prekrývajú oblasti analýzy, návrhu, špecifikácie požiadaviek a implementácie. Kapitoly analýzy *3.3.5 SIRUP Master* a *3.4 Manažment Server* by mali byť prehodnotené a prerozdelené. Jedná sa v nich síce o analýze ich vlastností, ale obsahujú aj informácie o návrhu ich samotnej iplementácie. Kapitola špecifikácie obsahuje informácie patriace pod implementačnú kapitolu (prípradne až prílohu implementačnej kapitoly).

Formálna stránka dokumentácie venujúcej sa implementácie prototypu je prehľadne napísaná a opisuje techniky riešené v prototype. Pochopiteľne sa tím v implementácii priebežného programu zameriava na SIRUP server. V podkapitolách sú písomne prezentované súčasti prototypu. Taktiež aj táto kapitola je obohatená o snímky obrazoviek.

Obsahová stránka

Z pohľadu na obsahovú stránku celkovej dokumentácie sa javí jednoznačná myšlienka kompletnosti informácií.

Dokumentácia obsahuje množstvo informácií a taktiež poskytuje 17 rôznych zdrojov pre ďalšie informácie.

Obsahová stránka časti dokumentácie venujúcej sa prototypu je písaná so všetkými potrebnými informáciami a poskytuje tiež ukážky obrazoviek aplikácie pri istých testovaniach prototypu.

Záver

Dosiahnutý výsledok po prvom semestri zodpovedá očakávanému výstupu tímového projektu. Členovia tímu si zodpovedne rozdelili úlohy na riešení ako dokumentácie tak aj prototypu. Tím predviedol čiastočné funkcionality navrhovanej aplikácie na prvom prototype. Prezentované vlastnosti taktiež členovia tímu zaznamenali v celkovej dokumentácií. Ich dosiahnuté riešenie je pôvodné, vychádzajúce z dostupnej literatúry. Vytvorený prototyp je schopný simulovať prepnutie kanálu z jedného portu na iný.

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Ilkovičova 3, 842 16 Bratislava 4

Správa o testovaní systému a posudok dokumentácie tímu č. 6 TIPSix

Tímový projekt

Študijný program: Počítačové a komunikačné systémy a siete

Tím č.3: Bc. Jozef Baláž, Bc. Tomáš Boros, Bc. Adam Močkoř, Bc. Martin Pivarník, Bc. Matej Rybár, Bc. Timotej Tkáč

Vedúci tímového projektu: Ing. Ján Murányi

Ak. rok: 2013/14

1 Úvod

Tento dokument obsahuje posudok práce tímu č. 6 TIPSix na predmete Tímový projekt 1 a 2. Úlohou tímu bolo vytvorenie vzdelávacieho portálu na tému „Sieťový protokol IPv6“. Pri hodnotení sa zameriavame na testovanie výsledného produktu, obsahovú a formálnu stránku doplnených častí projektovej a riadiacej dokumentácie. Cieľom tohto dokumentu je objektívne zhodnotiť vytvorený produkt s ohľadom na špecifikované požiadavky a uviesť kladné stránky a prípadné nedostatky vo výsledkoch práce tímu č. 6.

2 Správa o testovaní systému

Funkčnosť implementovaného systému nám tím predviedol počas krátkej prezentácie. Počas nej sa zamerali hlavne na simulátor NS-3, čo hodnotíme pozitívne, keďže implementácia webového portálu s možnosťou registrácie a diskutovania vo fóre nepovažujeme za príliš tvorivú úlohu, na ktorej by tím predviedol nové riešenia či vlastný prínos. Samotný simulátor hodnotíme pozitívne. Je dostupných viacero možností vloženia vstupného súboru a všetky relevantné výstupy simulácie. Pozitívne tiež hodnotíme možnosť vytvorenia topológie na simuláciu pomocou sprievodcu. Táto možnosť značne uľahčuje použitie simulátora pre používateľov, ktorý nemajú skúsenosti so zápisom topológie v požadovanom formáte.

Z používateľského hľadiska bolo najvážnejším problémom zisteným počas testovania nesprávne zobrazovanie textu spôsobené nekompatibilným kódovaním. Dôvodom vytvorenia systému vo webovom rozhraní je podľa špecifikácie požiadaviek jeho široká dostupnosť. Tím dostatočne neoveril splnenie tejto požiadavky. Odstránenie tohto problému je však jednoduché a nevyžaduje zmeny v aktuálnom návrhu alebo ostatnej časti implementácií riešenia.

Požiadavka na poskytovanie vyhľadávania na základne kľúčových slov nebola splnená. Absencia tejto funkcionality nepredstavuje problém pri použití systému, ktorý je dostatočne prehľadný, avšak bolo by vhodné vysvetliť dôvod nesplnenia tejto požiadavky. Riešenie spĺňa ostatné požiadavky.

Pri testovaní na vlastných zariadeniach sa nám nepodarilo získať výstup simulácie, hoci táto úloha údajne prebehla úspešne. Konkrétnym problémom sú odkazy na chýbajúce výstupné súbory. Nesprávne odkazy tiež spôsobili presmerovanie na predchádzajúce umiestnenie portálu. Staršia verzia sa však odlišuje napríklad nedostupnosťou simulátora.

Pred nasadením do reálnej prevádzky je nutnosťou otestovať čas vykonávania simulácie a celkovú odozvu systému počas vykonávania viacerých simulácií súčasne.

Zhodnotili sme, že posudzovaný systém má predpoklady na úspešné nasadenie po dôkladnejšom testovaní a odstránení zistených problémov.

3 Posudok dokumentácie z vecného hľadiska

Projektová dokumentácia obsahuje relevantné kapitoly, ktoré sú v texte nižšie posudzované po obsahovej stránke. Pro posudzovaní sme sa zamerali len na pridané kapitoly.

3.1 Dokumentácia riešenia

Dokumentácia riešenia bola oproti verzii v zimnom semestri rozšírená o kapitoly Implementácia výsledného produktu, Testovanie, Záver, Zhodnotenie, Inštalačná príručka, Používateľská príručka a Obsah elektronického média. Kapitola Implementácia popisuje v primeranom rozsahu ako tím implementoval produkt, avšak uvítali by sme prítomnosť opisu fyzického modelu údajov systému. Kapitola Testovanie stručne opisuje proces testovania základných funkcionalít produktu a testovanie simulácie. Prínosom by mohol byť výstup z testovania znalostí malej vzorky používateľov na základe ich vedomostí získaných z materiálov na stránke. Ku ostatným pridaným kapitolám nemáme žiadne pripomienky po obsahovej stránke.

3.2 Dokumentácia riadenia

Riadiaca dokumentácia neobsahuje kapitolu Posudky(jej obsah tím TIPSix pravdepodobne pripojí ako externú prílohu), kapitolu Štandardy kódovania a kapitolu Manažment verzií, avšak informácie o manažmente verzií možno nájsť v časti Ponuka, podkapitola Komunikácia členov tímu. Iné pripomienky ku dokumentácii riadenia nemáme.

4 Posudok dokumentácie z formálneho hľadiska

4.1 Dokumentácia riešenia

Celkovo možno konštatovať, že formálna stránka dokumentácie je na požadovanej úrovni, zodpovedá požiadavkám čitateľnosti, prehľadnosti a jednoduchosti orientácie. V práci bolo využité členenie do viacerých kapitol a podkapitol, ktoré logicky spolu súvisia a na ich označenie boli využité jednotné štýly. Rovnako boli označené aj všetky použité tabuľky a obrázky, podľa ich príslušnosti k jednotlivým kapitolám práce. Ako nedostatočné však hodnotíme samotný formát označenia, nakoľko možno identifikovať viacero nejednotných označení, ktoré sú viditeľné už v samotnom obsahu, ako aj rôzne vzdialenosti medzi samotnou tabuľkou alebo obrázkom a označením (napr. viď. s. 8).

Príklad odlišných označení:

- Tabuľka **1-1.: Oblasti** skupinových adries (Obsah)
- Tabuľka **1-2Rozvrhnutie** IPv6 adresného priestoru (Obsah)
- Obr. **1-13.:** GNS 3 (Obsah)
- Obr. **1-14:** NS-3 moduly podla vrstvy spracovania (Obsah)

Dokumentácia sa vyznačuje členením textu do kratších odsekov a číslovaním strán, čo uľahčuje orientáciu. Nebolo však využité jednotné riadkovanie, nakoľko možno pozorovať odlišné vzdialenosti medzi riadkami vo viacerých kapitolách a podkapitolách (napr. kapitola 1.4 a 1.5)

V práci možno identifikovať aj niekoľko gramatických chýb a preklepov.

Príklad:

- ***Multicast***adresuje (s. 7)
- ***Anycast***je (s. 7)
- ... systémom Debian**,** a preto je ... m na vysokú prevádzku**,** keďže ... (s. 39)
- Na obr. č. **X** je zobrazený ... (s. 48)
- ...vyhľadávanie na stránke a na **Googli**. (s. 50)
- Štruktúra fóra sa bude **počas** meniť počas chodu ... (s. 51)

V zozname literatúry možno nájsť rozdiely v označení internetových odkazov, nakoľko niektoré sú modré a podčiarknuté a iné sú čierne a bez podčiarknutia. V celom dokumente absentujú odkazy na použité zdroje, a tak nie je možné identifikovať, v ktorých častiach boli použité.

4.2 Dokumentácia riadenia

V úvode samotnej riadiacej dokumentácie absentuje jej obsah, ktorý by uľahčil orientáciu a hľadanie potrebných informácií. S tým súvisí aj absencia číslovania jednotlivých častí. Na označenie hlavných nadpisov bol využitý rovnaký formát a bolo zachované jednotné zarovnanie textu na jednotlivých stranách. Avšak, v závere časti Projekt: Sieťový protokol IPv6 (s. xxv - xxviii) bolo zvolené rozdielne riadkovanie, čo pri

čítaní pôsobí ako rušivý prvok a celá časť je tak značne neprehľadná. Pri kompletizovaní jednotlivých zápisníc bola dodržaná jednotná štruktúra, ale bola zvolená rozdielna veľkosť písma (viď s. lxiii a ixiv).

V dokumentácii možno identifikovať niekoľko gramatických chýb a preklepov. Ako príklad uvádzame nasledovné:

- Dokument je určený na oboznámenie sa **(s)** nasadzovaním... (s. xx)
- ... ako prejsť **ku** novej ére ... (s. xxv)
- ... od zberu prvotných dát**,** cez spracovanie výsledkov ... (s. xxiv)
- Multimediálne grafické prvky ... **je** dobrým predpokladom ... (s. xxiv)
- ... pretože by pri nesprávnej komunikácií **môže** dôjsť ... (s. xxxv)
- ... pri tvorbe **náždo** edukačného ... (s. xxxiv)

5 Záver

Výsledný produkt predstavuje použiteľné riešenie vzdelávania a testovania znalostí o IPv6. Kladne hodnotíme vytvorenie jednoduchého sprievodcu pre prácu so simulátorom NS-3, ktorý je v takejto forme použiteľný aj pre začiatočníkov. Otázne je však jeho použitie viacerými používateľmi v rovnakom čase. Ostatné funkcie vzdelávacieho portálu sú na akceptovateľnej úrovni.

Napriek gramatickým a štylistickým chybám hodnotíme projektovú dokumentáciu ako dobre spracovanú a dostatočne popisujúcu pokrok tímu počas uplynulého akademického roku.

Celkovo je výsledok práce tímu č. 6 TIPSix na viac ako dobrej úrovni, a to najmä vďaka vytvoreniu jednoduchého sprievodcu pre simulátor NS-3 a celkovej úrovni projektovej dokumentácie aj napriek nižšiemu počtu členov tímu.

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Posudok výsledku projektu tímu č. 3

vypracovaný tímom č. 6

Študijný program: Počítačové a komunikačné systémy a siete
Študijný odbor: 9.2.4 Počítačové inžinierstvo
Ak. rok: 2013/2014
Členovia tímu 6: Bc. Lukáš Danielovič
Bc. Anton Pôbiš
Bc. Lukáš Lenčéš
Bc. Marek Dukát

Obsah

Úvod 3
1. Posudok projektovej dokumentácie 4
   1.1. Úvod a analýza 4
   1.2. Špecifikácia požiadaviek 4
   1.3. Návrh 4
   1.4. Prototyp 5
   1.5. Implementácia 5
   1.6. Testovanie 5
   1.7. Čo sme sa naučili, Zhodnotenie a Literatúra 5
   1.8. Ďalšie kapitoly 5
2. Posudok riadiacej dokumentácie 6
   2.1. Obsahová úroveň 6
   2.2. Formálna úroveň 6
3. Posudok výsledného softvérového systému ..... **Error! Bookmark not defined.**
4. Záver 8

Úvod

Tento dokument obsahuje posudok ku výsledku práce tímu č. 3 v dvoj-semestrálnom predmete Tímový projekt. Dokument je vypracovaný tímom č. 6.

Tím č. 3 mal riešiť zadanie s názvom *Manažment VoIP relácií*. Ich prácou bolo navrhnúť a implementovať sofistikovanejšie riadenie multimediálnych komunikácií založených na protokole SIP.

Vypracovaný posudok v tomto dokumente sa zameriava na výsledok práce tímu č. 3 (samotný softvér), dokumentácie k projektu a dokumentácie riadenia.

Dokumentácie posudzujeme z dvoch hľadísk:

- z obsahovej stránky – či dokumentácie obsahujú všetky potrebné informácie k téme spracovávaného projektu a v akej kvalite sú tieto informácie.
- z formálnej stránky – zhodnotíme štruktúru textu, čitateľnosť, prehľadnosť, jazykovú kvalitu a pod.

1. Posudok projektovej dokumentácie

V tejto časti posudku sa vyjadríme k jednotlivým častiam projektovej dokumentácie tímu č. 3.

1.1. Úvod a analýza

Kapitola úvodu obsahuje všetko, čo by mala obsahovať. Stručne je tu opísaný kontext celého dokumentu.

Kapitola analýza obsahuje množstvo informácií, avšak má niekoľko nedostatkov:

- časť *3.2 Kontext* sa javí ako príliš úzky a nejasný pohľad na konfrontáciu analýzy požiadaviek (čo výsledný softvér vykoná?) a návrhom riešenia (ako to vykoná?). Podobne mimo analýzu siahajú kapitoly *3.3.4, 3.3.5* a *3.4*, ku ktorým, ak autori čerpali informácie, by mali byť uvedené zdroje.
- časť *3.3.1 SIP Single Port* – v tejto časti (ako aj pri ďalších iných), ktorá sa odvoláva na implementovaný a funkčný produkt, nám chýba odkaz na nejaký dokument, prameň, z ktorého autori čerpali.

Z formálneho hľadiska sú kapitoly Úvod a Analýza napísané spisovne a jasne. Štruktúra textu je systematicky členená na podkapitoly a odseky, čo zjednodušuje orientáciu v dokumente.

1.2. Špecifikácia požiadaviek

Kapitola špecifikácie obsahuje požadované informácie: čo má systém vykonávať, aký typ riadiacich údajov sa bude prenášať, ktorý uzol systému bude za akú činnosť zodpovedný a čo ktorá riadiaca správa má zabezpečovať. Obsahovo sme s touto časťou dokumentácie spokojný.

Z formálneho stránky však je táto kapitola veľmi nedopracovaná. Kapitole chýba úvodný text, chýbajú celé vety, vysvetlenia funkcionálnych vlastností systému ako aj jeho nefunkcionálnej štruktúry. Rozdelenie textu do podkapitol je tu nesystematické.

1.3. Návrh

Chválime zrozumiteľný nákres celkovej architektúry systému. Chýba tu však ďalšia dekompozícia systému. V kapitole *5.4 Údaje* sa po tretíkrát spomína, že prenášané údaje budú objekty JSON. V tejto stručnej kapitole je spomenuté, že celkový systém sa skladá z dielčích podsystémov. Minimálne v tejto časti chýba ďalšia systematická dekompozícia a konkretizácia systému (ktorý modul/podsystém/súčiastka a ako pracuje s týmito objektami?)

Dostatočne sú spracované aj podkapitoly *5.7 Klient* a *5.9 SIRUP Master*. V rámci návrhu by sme však uvítali viac informácií o činnostiach algoritmov ako o tom, v akom jazyku budú implementované.

Formálna stránka je slabá. Opäť chýba úvodný text do kapitoly ako aj širší popis obráku v kapitole *5.1 Architektúra* (celá podkapitola obsahuje len obrázok). Štrukturalizácia do podkapitol nedáva logický zmysel (napr. *5.7 Klient, 5.8 Server* a *5.9 SIRUP Master* by mali byť podľa textu v *5.6 SIRUP* jeho podkapitolami).

1.4. Prototyp

V tejto kapitole sa autori venovali opisu implementovaného prototypu. Jasne napísali, čo všetko do prototypu začlenili a čo nie. Bližšie opísali typy prenášaných správ (podali o nich viac informácií ako v špecifikácií požiadaviek). Obsahovo je táto kapitola postačujúca.

Štruktúrou textu a čitateľnosťou je opis prototypu v omnoho lepšom stave ako predchádzajúce kapitoly, text je dobre štruktúrovaný a obsahuje názorné obrázky zo simulácií.

1.5. Implementácia

V implementácií autori opísali, čo všetko zmenili a čo vylepšili oproti prototypu. Niektoré informácie o nových nástrojoch (ktoré sa rozhodli použiť ku konečnému výsledku) by mali byť presunuté a doplnené do analýzy (*Angular.js, Highcharts*...).

Čitateľ tu však len nájde pár riadkov textu, do ktorých bolo vtesnaných mnoho informácií (modifikácia RTP paketov, registrácia a preregistrácia komunikácie...). Chýba tu aspoň grafické znázornenie toku informácií a správ.

Veľmi dobre je spracovaná kapitola *5.7 Simulácia siete NetEm*. Kapitola obsahuje dosť informácií, čo je aj vidieť, keďže zaberá 3/7 kapacity celej kapitoly implementácie.

Z formálneho hľadiska je text dobre štruktúrovaný a prehľadný. Kapitola pôsobí dobrým dojmom na čitateľa, nenúti ho preskakovať z časti dokumentácie do inej časti a je príjemná na čítanie.

1.6. Testovanie

Chválime členov tímu č. 3 za precízne a prehľadné spracovanie testovacích scenárov. Je tu veľmi pekne načrtnutá testovacia schéma, podrobne opísané testovacie scenáre a zároveň sú tu aj odchytené výstupy komunikácií v grafoch.

Druhú pochvalu udeľujeme za testovacie scenáre na reálnych zariadeniach. Kapitola testovania je obsahovo kompletná a nemáme ku nej najmenšiu kritiku.

Formálne je táto kapitola výborne spracovaná. Členenie textu je prehľadné a logicky systematické. Jedinou chybičkou je zalamovanie – 3. scenár, 4. scenár a 5.scenár začínajú pred koncom strany.

1.7. Čo sme sa naučili, Zhodnotenie a Literatúra

Obsahovo kapitoly zahŕňajú všetko, čo by mali zahŕňať.

Z formálnej stránky vytkneme len syntax literatúry – nespĺňa normu STN ISO 690.

1.8. Ďalšie kapitoly

Ďalšie kapitoly, ako *Používateľská príručka* a *Systémová príručka* by nemali byť súčasťou projektovej dokumentácie, ale mali by byť uvedené ako prílohy. Posudok im je vyjadrený v časti posudzovania výsledného projektu v tomto dokumente.

2. Posudok riadiacej dokumentácie

V tejto časti posudku sa vyjadríme k jednotlivým častiam riadiacej dokumentácie tímu č. 3. Posudzovanie rozdelíme do dvoch kategórií podľa pohľadu: z formálneho hľadiska a z obsahového hľadiska. Na riadiacu dokumentáciu nebudeme hľadieť po kapitolách ako tomu bolo pri projektovej dokumentácií, ale ako na celok.

2.1. Obsahová úroveň

Z obsahovej stránky riadiaca dokumentácia obsahuje všetky požadované náležitosti. Sú tu stručne opísaní členovia tímu, je tu spomenuté zadanie spolu s jeho hrubým návrhom riešenia. Súčasťou dokumentácie je samozrejme aj ponuka a motivácia tímu.

Plán projektu tím rozdelil na 27 týždňov, počas ktorých sa im podľa priložených zápisníc darilo pokračovať v rozvrhnutom pláne.

2.2. Formálna úroveň

Formálna stránka riadiacej dokumentácie je prehľadná. Text je vhodne členený na kapitoly a podkapitoly.

Tím začlenil do zápisníc aj fotografie schém zo stretnutia, čo zlepšuje predstavu, čo sa na stretnutí odohrávalo.

Nie priam priaznivé je však rozhodnutie tímu začleniť posledných 6 stretnutí do jednej zápisnice.

3. Správa o testovaní posudzovaného systému

Tím singles nám predviedol výsledný produkt s problémami spustenia. Tento problém je hlavne spôsobený slabým hardvérom na ktorom bol projekt vytvorený a testovaný. Táto skutočnosť teda nemá dopad na efektivitu výstupu. Produkt obsahuje všetky nevyhnutné komponenty a bol splnený v plnom rozsahu.

Pri prezentácii sme testovali jednoduché scenáre. Riešenie dokáže spojiť 3 protokoly do jedného a tak sa vyhnúť problému s NAT prechodom. Hovory cez RTP prechádzajú viacerými tunelmi a fungujú ako *load balancing* a súčasne záloha pri spadnutí jedného tunelu. Táto časť je riešená manažovatelným komponentom, kde sa systémom zober a posuň premiestňujú hovory do iných tunelov bez narušenia komunikácie.

Pri nasadení do reálnej prevádzky budú možné horšie výsledky z hľadiska výkonnosti vzhľadom na implementáciu niektorých komponentov v programovacom jazyku Java.

K výslednému produktu nemáme ďalšie pripomienky. Projekt bol vypracovaný podľa návrhu s menšími odchýlkami.

Pri testovaní produktu na vlastnom HW bol problém vytvoriť funkčné testovacie prostredie podľa inštalačnej príručky. V príručke chýbajú nároky na použitý hardvér. Celkovo je používateľská príručka príliš stručná a samotný „nový“ používateľ sa podľa nej len ťažko (ak vôbec) dopracuje k výsledkom spomínaným v celkovej dokumentácii. Podobne je na tom i systémová (inštalačná) príručka.

Zhodnocujeme, že výsledok by mohol byť použiteľný aj do reálnej prevádzky v prípade ďalšieho vývoja a rieši hneď niekoľko problémov a nedostatkov VoiP.

4. Záver

Zhodnotili sme dokumenty a produkt tímu č. 3. Mnoho zo spomenutých vyjadrení, ktoré opisujú nedostatky v projektovej dokumentácií, sme uviedli už pri prvom vydanom posudku.

Projektová dokumentácia tímu č. 3 má svoje muchy. Z formálnej stránky sú to najmä nesystematické delenia kapitol a celková náročnosť čítania textu (chýbajú komplexné vety vyjasňujúce danú problematiku). Z obsahovej stránky je to absencia mnohých zaujímavých informácií, najmä v častiach návrhu a implementácie.

Celkovo dokumentácii chýba záverečná kontrola a prehodenie istých kapitol do príloh.

Riadiaca dokumentácia je obsahovo kompletná a formálne prispôsobená. Zápisnice v tejto časti dokumentu sú stručné, ale jasné.

Výstupný softvér projektu sme mali možnosť vidieť prezentovaný jeho tvorcami. V praxi sme videli funkčnosť tohto systému. Avšak podľa príručiek dokumentácie by sme sa k takýmto výsledkom nedopracovali.

9 Protokoly

Tímový projekt - Preberací protokol

Odovzdávajúci tím: Singles
Prijímajúci tím: TIPSix

Zástupca tímu TIPSix Bc. Lukáš Danielovič svojim podpisom potvrdzuje korektné prevzatie dokumentov: Projektová dokumentácia a Riadiaca dokumentácia podľa požiadaviek stanovených garantom predmetu.

V Bratislava dňa 15.11.2013

Bc. Lukáš Danielovič

Tímový projekt - Preberací protokol

Odovzdávajúci tím: Singles
Prijímajúci tím: TIPSix

Zástupca tímu TIPSix Bc. Lukáš Danielovič svojim podpisom potvrdzuje korektné prevzatie dokumentu: Posudok dokumentu tímu TIPsix podľa požiadaviek stanovených garantom predmetu.

V Bratislava dňa 22.11.2013

Bc. Lukáš Danielovič

Tímový projekt - Preberací protokol

Odovzdávajúci tím: Singles
Prijímajúci tím: TIPSix

Zástupca tímu č. 6 TIPSix Bc. Lukáš Danielovič svojim podpisom potvrdzuje korektné prevzatie dokumentu: Projektová dokumentácia tímu č. 3 SIngles podľa požiadaviek stanovených garantom predmetu.

V Bratislava dňa 22.5.2014

Bc. Lukáš Danielovič