Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: POČÍTAČOVÉ A KOMUNIKAČNÉ SYSTÉMY A SIETE

Tímový projekt

Prípadová štúdia pre VoIP sieť

chiNETeam (11)

Tím č.: 11
Akademický rok: 2009/2010

Filip Burda, Bc.
Peter Havrila, Bc.
Marián Kněžek, Bc.
Klaudia Konôpková, Bc.
Juraj Nemeček, Bc.
Ján Murányi, Bc.

Obsah

Zadanie

IP Multimedia Subsystem je v súčasnosti najperspektívnejšia technológia pre implementáciu v prostredí telekomunikačných operátorov.

Cieľom projektu je vytvoriť prípadovú štúdiu zameranú na konkrétny aspekt tejto siete (zabezpečenie kvality služby, bezpečnosť a pod.)

V rámci tímového projektu bude nutné vybrať cieľovú platformu (v prípade IMS napr. OpenIMS core), navrhnúť zapojenie celej siete, pripraviť scenáre a metriky a nakoniec dosiahnuté výsledky vyhodnotiť.

Prirodzenou súčasťou práce je administrácia platformy a vytvorenie nielen dokumentácie k projektu, ale aj celej prípadovej štúdie vo vhodnom formáte.

Vedúci tímu Ing. I. Kotuliak, PhD.

1 Úvod

Milý čitateľ,

dostáva sa Vám do rúk dokument, ktorý vznikol spoločnou prácou skupiny mladých študentov na predmete Tímový Projekt na Fakulte Informatiky a Informačných Technológií Slovenskej Technickej Univerzity. Témou tejto práce je prípadová štúdia pre VoIP sieť, ktorú naša skupina analyzovala, navrhla a vytvorila z pohľadu zameranom na zabezpečovanie kvality služieb pri udalostiach s nepriaznivým vplyvom na sieť. Tento dokument popisuje celkovú problematiku z oblasti počítačových sietí a VoIP, ktorá je spracovaná jednotlivými členmi tímu. Popisujeme postupy, technológie a hlavne potreby, ktoré vyžadujú moderné multimediálne relácie vystavané na báze IP Multimedia Subsystem. Ďalej spracovávame možnosti zabezpečiť požiadavky na transportnú sieť s maximálnym dôrazom na robustnosť siete pri zabezpečovaní riadenia kvality služieb. Tu sa tím zameral na aktuálne najprogresívnejšiu technológiu používanú poskytovateľmi služieb a to *Multi-Protocol Label Switching* (MPLS) a možnosti riadenia kvality služieb pomocou pridruženej technológie MPLS *Traffic Engineering* (MPLS TE). Pre potreby testovania sme si vytvorili našu vlastnú IMS sieť.

Veríme, že tento dokument splní očakávania čitateľa a tiež poslúži ako cenné zhrnutie uvedených technológií.

Príjemné čítanie praje

chiNETeam

1.1 Motivácia

Ako napovedá zadanie projektu, IP Multimedia Subsystem je v dnešnej dobe najatraktívnejším systémom na prevádzku VoIP a multimediálnych služieb pre komerčných poskytovateľov služieb. Vychádzajúc z profesionálneho zázemia členov tímu, ktoré pokrýva VoIP implementácie, ale aj problematiku dátových transportných sietí, bolo veľmi apelujúce uskutočniť prípadovú štúdiu, v ktorej spracujeme práve spoluprácu týchto dvoch neoddeliteľných vrstiev pri implementácii celkovej infraštruktúry služieb. Pre všetkých členov tímu umožní tento projekt, okrem adekvátneho profesionálneho rastu, zároveň aj pohľad na budovanie komplexných IP sietí a VoIP služieb.

Tému si vybral celý tím jednohlasne, nakoľko sa všetci pohybujeme vo svete počítačových sietí. Jednotliví členovia majú pracovné skúsenosti z oblasti počítačových sietí, ich nasadzovania, správy a dokonca aj výučby.

Tím má záujem rozširovať si svoje vedomosti v oblasti konvergovaných sietí, NGN a IMS, nakoľko je toto oblasť pre nás atraktívna. Vytvorenie prípadovej štúdie je pre nás vynikajúcou príležitosťou ako naplniť naše osobné ciele a to rozšíriť si obzory v tejto oblasti a zároveň vytvoriť zaujímavý produkt. Je to dobrá príležitosť ako sa zamerať aj na iné technológie a nielen technológie a produkty od firmy Cisco, s ktorými sme sa doteraz stretávali. Zároveň máme k dispozícii ten najväčší kapitál a to sú naše vedomosti a chuť učiť sa nové. Myslíme si, že pre danú tému máme výborné predpoklady, nakoľko všetci sme aktívnou súčasťou Cisco akadémie a vieme využiť naše

bakalárske práce, vedecké publikácie a školské projekty na vytvorenie výsledného produktu. Naše doterajšie práce pri spojení vyvolávajú silné synergické efekty.

1.2 Prehľad dokumentu

Táto práca je analýzou k prípadovej štúdii VoIP siete. Dokument pozostáva zo 17 kapitol. Prvá kapitola je úvodom k tomuto dokumentu. V druhej kapitole si predstavíme protokol SIP a protokol SDP. V kapitole 3 rozoberáme IP Multimedia Subsystem, jeho architektúru a súčasti. Úvod do optimalizácie siete sa nachádza v kapitole 4. Piata kapitola samostatne popisuje smerovanie v TCP/IP sieťach. V kapitole 6 rozoberáme konvergenciu v sieti a časové parametre smerovacích protokolov. Kapitola 7 sa venuje problematike kvality služieb – QoS. Ôsma kapitola sa venuje redundancii na sieťovej vrstve. Kapitola 9 opisuje technológiu MPLS a nasledujúca kapitola sa venuje jednej jej súčasti – MPLS TE. V kapitole 11 diskutujeme o možnostiach protokolu IPv6 v medziach nášho projektu. Kapitola 12 je zhodnotením analýzy. Požiadavky špecifikujeme v kapitole 13. Návrh riešenia sme umiestnili do kapitoly 14. Prototyp riešenia s počiatočnou implementáciou je v nasledujúcej kapitole 15. Implementáciu je možné nájsť v kapitole 16 spolu s verifikáciou. V kapitole 17 je zhodnotenie práce. Práca pokračuje zoznamom skratiek použitých v dokumente a použitou literatúrou. Nasledujú prílohy a to konkrétne príloha A, kde sa nachádza používateľská príručka pre transportnú sieť. Príloha B je používateľskou príručkou pre inštaláciu IMS.

Táto práca vyžaduje od čitateľa aspoň základné znalosti z oblasti počítačových sietí. Práca vychádza zo spojenia viacerých našich bakalárskych prác a vedeckých článkov, odkiaľ sú prevzaté celé state. Práca sa orientuje najmä na zbernicovú topológiu, konkrétne *ethernet*, keďže je to jedna z najpoužívanejších prístupových metód. Práca sa nebude bližšie venovať bezdrôtovým riešeniam. Jadro problematiky rozoberáme najmä na sieťovej vrstve a na úrovní aplikácií.

V texte sa bude nachádzať niekoľko anglických pojmov, ktoré nebudeme prekladať do ich slovenského ekvivalentu z dôvodu nejednoznačnosti a sťaženej prehľadnosti. Pôjde najmä o pojmy z oblasti IMS.

Anglické pojmy sú označené *kurzívou*. Podstatné informácie sú vyznačené **tučným** písmom. K preloženým pojmom uvádzame aj ich anglický ekvivalent. Všetky skratky obsiahnuté v texte je možné vyhľadať v slovníku pojmov na strane 93.

2 Protokol SIP a protokol SDP

Protokol SIP je v dnešnej dobe najpoužívanejším protokolom v oblasti IP telefónie. Prvýkrát bol predložený Markom Handley a Evou Scholler ako IETF draft vo februári 1996 s názvom *Session Invitation Protocol*. Súčasne s protokolom SIPv1 bol uvedený protokol SCIP (*Simple Conference Invitation Protocol*) taktiež ako draft. Neskôr tieto dva protokoly vytvorili SIPv2. Zmenený bol názov na *Session Initiation Protocol.* Protokol bol postavený na protokole HTTP (*Hyper Text Transport Protocol*). Až vo februári 1999 dosiahol požadovanú úroveň a mohol byť uverejnený ako [RFC2543]. V júni 2002 pod vedením Jonathana Rosenberga bol označený ako prekonaný a bol nahradený [RFC3261, RFC2396].

2.1 Úlohy protokolu SIP

Protokol SIP je signalizačným protokolom. Neumožňuje prenos multimediálnych dát. Pri vývoji tohto protokolu sa dbalo, aby výsledný štandard bol veľmi jednoduchý a ľahko implementovateľný. Z tohto dôvodu v ňom boli zadefinované základné požiadavky, ktoré má umožňovať tento protokol:

- **Určenie polohy** – zistenie základných technických informácií ako IP adresa pre zabezpečenie komunikácie, ako medzi používateľskými stanicami, tak aj medzi serverom a klientom
- **Zabezpečenie dostupnosti** – zaručenie, že koncové zariadenie bude dostupné a schopné komunikovať
- **Vlastnosti koncových zariadení** – zistenie možných vlastností koncového zariadenia na prenos informácií
- **Zriadenie relácie** – dohodnutie parametrov oboch volajúcich strán, signalizácia a nadviazanie hovoru
- **Spravovanie relácie** – prenos, ukončenie a udržiavanie relácie (zmena prípadných vlastností prenosu) [1]

2.2 Protokol SIP a model RM OSI

Protokol SIP je definovaný aj nad transportným protokolom TCP aj nad protokolom UDP. V praxi sa častejšie používa protokol UDP, hlavne pre jeho vlastnosti, ktoré umožňujú malé oneskorenie. Napriek tomu RFC 3261 odporúča použitie protokolu TCP ako spoľahlivého protokolu. Predchádzajúce RFC 2543 odporúčalo protokol UDP. Táto zmena bola prijatá hlavne z dôvodu prenosu väčších správ. Hlavná nevýhoda UDP prenosu je problematická fragmentácia správ pri ich väčšom rozsahu. Preto, aj keď stanice komunikujú hlavne prostredníctvom UDP protokolu, je dôležité, aby boli schopné využívať aj protokol TCP. V prípade použitia protokolu UDP je možné v jednom datagrame poslať iba jednu SIP správu. Pri použití TCP spojenia rozdelenie správ určuje *Content-Lenght* metóda v SIP správe.

Štandardný TCP alebo UDP port, ak sa priamo nežiada inak, na ktorom aplikácie počúvajú prichádzajúce SIP správy, je 5060. V prípade použitia enkapsulácie TLS (*Transport Layer Security*) protokolu je to port 5061.

2.3 Základný hovor

Hovor, alebo dialóg, sa skladá z transakcií a tie sa skladajú z požiadaviek. Klienti teda komunikujú prostredníctvom transakcií. Každá transakcia je začatá počiatočnou požiadavkou a ukončená finálnou odpoveďou. V prípade štandardného hovoru, keď Anna (volajúca) volá Borisa (volaný), vyšle na známu adresu požiadavku *INVITE*. Boris môže odpovedať niekoľkými nepovinnými odpoveďami (*TRYING, RINGING*) a následne potvrdiť prijatie hovoru finálnou odpoveďou *OK*. Touto odpoveďou sa končí prvá transakcia. Nasleduje druhá transakcia, keď Anna potvrdí prijatie hovoru. V tomto momente sa nadviaže relácia protokolu RTP (*Real-Time Protocol*), ktorý prenáša multimediálne informácie. Táto časť nie je transakciou, bola ňou iba dohodnutá. Ak sa Anna alebo Boris rozhodnú hovor ukončiť, pošlú druhej strane požiadavku *BYE* a tá ukončenie potvrdí finálnou odpoveďou *OK*.

Dialóg je definovaný rovnakými hodnotami polí *From*, *To, Call-ID* a rovnakou kombináciou polí *Via*. Po nadviazaní hovoru všetky správy musia obsahovať dohodnuté hodnoty týchto polí. Hodnota *CSeq* identifikuje transakciu. Jej hodnota je rovnaká pre všetky požiadavky a odpovede v rámci danej transakcie. Jej číslo sa nemení iba v prípade, že ide o transakciu *ACK*, ktorá prišla ako odpoveď na 2xx odpoveď, alebo je to transakcia *CANCEL*. V oboch prípadoch sa ale mení menová hodnota poľa. Číslo zostáva rovnaké, aby bolo možné jednoducho vyhľadať, na ktorú transakciu prišlo dané *ACK* alebo *CANCEL*.

2.3.1 Identifikácia používateľa

Vo svete protokolu SIP sa jednotlivé entity identifikujú prostredníctvom SIP adresy známej ako URL (*Uniform Resource Locator*) alebo URI (*Uniform Resource Identifier*).

sip:alice@192.168.1.2

Táto adresa vychádza z RFC 2396 (*Uniform Resource Identifiers (URI): Generic Syntax*). Prvá časť určuje prekladaču URI, ako ju má správne interpretovať. V našom prípade vie, že má použiť sip schému. Ak by táto časť bola *mailto*, prekladač vie, že budú nasledovať *meno používateľa @ samotná adresa* [2].

SIP URI je definovaná ako

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

Nie všetky časti sú však potrebné. Jediná povinná zložka je časť *host*, ktorá môže byť v tvare IP adresy alebo ako plne spôsobilé doménové meno. Náš prípad využíva zložky *user* a *host*.

Táto klasifikácia je výhodná z hľadiska implementácie. Aj keď staršie zariadenia nebudú rozumieť novo vytvorenej odpovedi, ktorá je správne klasifikovaná, stále budú môcť správne reagovať [3].

2.4 Všeobecné hlavičky

Niektoré polia protokolu SIP sa nachádzajú vo všetkých správach dialógu. Tieto polia nazývame všeobecné [4].

2.4.1 Pole Call-ID

Parameter Call-ID slúži na definovanie hovoru. Pomáha pri identifikácii hovoru. Takisto pri správach *INVITE* a *REGISTER*, ak správy prechádzajú cez proxy, ktoré duplikuje správy, alebo pri identifikácii hovoru počas konferencie. Tento parameter musí byť rovnaký pre všetky požiadavky a odpovede v rámci dialógu a mal by byť rovnaký pri každej registrácii používateľa. Tento parameter musí byť generovaný náhodne z dôvodu možného odcudzenia relácie treťou stranou.

2.4.2 Pole Cseq

Toto pole musí byť obsiahnuté vo všetkých požiadavkách a odpovediach. Týmto spôsobom vieme rozpoznať jednotlivé transakcie. Ako sme už spomínali, skladá sa z dvoch častí. Kladného čísla, ktoré sa na začiatku dialógu náhodne vyberie a mena metódy, ktorá začala transakciu požiadavkou. Každá nová transakcia SIP dialógu musí toto číslo zväčšiť o jedno. Jedinou výnimkou sa stávajú transakcie s metódami *ACK* a *CANCEL*. Tieto metódy číslo nemenia, kvôli ľahšej identifikácii, na ktorú transakciu boli odpoveďou. Druhá časť poľa *Cseq* sa mení vždy. V prípade retransmisie požiadavky sa číslo nemení. Ak sa požiadavka pri zaslaní protokolom UDP stratí a po určenom čase sa opätovne pošle, jej číslo zostáva rovnaké ako pri prvom pokuse.

2.4.3 Pole From

Toto pole je jedným z polí, ktoré identifikujú dialóg. Obsahuje voliteľné „zobrazené meno“, kontaktný údaj v tvare URI a položku *tag*. Hodnota celého poľa je vždy v rámci transakcie skopírovaná z požiadavky.

Volajúci môže svoju identitu skryť. V tomto prípade je odporúčané podľa RFC 3261, aby táto hodnota bola „*Anonymous*“. Druhá časť poľa je naďalej povinná, ale môže obsahovať ľubovoľné hodnoty, pričom však musí byť syntakticky správna.

Hodnota *tag* musí obsahovať ľubovoľný náhodný reťazec obsahujúci minimálne 32 bitov. Táto hodnota musí byť globálne unikátna, pretože sa stáva jednou z identifikátorov hovoru. V prípade rozdelenia hovoru proxy serverom, táto hodnota zostáva rovnaká pre všetky rozposlané požiadavky. Spôsob generovania tohto reťazca nie je definovaný a je ponechaný na každú implementáciu zvlášť.

2.4.4 Pole To

Pole tvorí identifikátor dialógu. Jednotlivé atribúty poľa sú podobné poľu *From*. Taktiež musí obsahovať URI a pole *Tag*. Tieto atribúty sa vytvárajú rovnako. Prvá časť sa preberie z prvej požiadavky v transakcii. Ak sa nachádza v prvej požiadavke, tak ju musí vygenerovať klient. Zapíše do nej cieľ svojej požiadavky. Atribút *Tag* je vygenerovaný rovnako ako v poli *From*, náhodne.

2.4.5 Pole Via

Toto pole môže byť v jednej požiadavke alebo odpovedi viackrát. Musí ho pridať každý server, ktorý preposiela SIP správy k cieľu (proxy). Vždy nové pole *Via* sa pridá nad všetky existujúce. Pomocou poľa *Via* je umožnené používateľom využívať protokol SIP, aj keď využívajú funkciu smerovača NAT (*Network Address Translation*). Toto pole môže obsahovať niekoľko atribútov. Keď klient vytvára požiadavku, musí toto pole obsiahnuť v správe. Musí obsahovať meno protokolu a jeho verziu, teda SIP a 2.0. Pole *Via* musí ďalej obsahovať parameter *branch*, ktorý v čase musí byť pre jedného klienta unikátny. Tento atribút spoľahlivo určuje požiadavku. Jeho hodnota je náhodná a ľubovoľná, musí však obsahovať začiatočný reťazec z9hG4bK, ktorým vieme určiť verziu protokolu. Staršie verzie protokolu SIP nemohli vygenerovať takýto reťazec. Ďalej obsahuje meno protokolu 4. vrstvy RM OSI modelu, IP adresu a port zariadenia, ktoré vytvorilo pole a parameter *rport* a *received*. Parameter *received* slúži na zaznamenanie IP adresy, z ktorej prišla správa, ak sa nezhoduje s hodnotou IP adresy v prijatom poli *Via*. Rovnako parameter *rport*, ktorý zaznamená port z ktorého správa prišla, v prípade, ak sa líši od hodnoty, ktorá bola zistená z poľa *Via*. Na základe týchto hodnôt vie server prekonať NAT a kontaktovať klienta.

Predstavme si situáciu, keď je klient za smerovačom s funkciou NAT *overload* (NAT-PT - *Network Address Translation - Protocol Translation*) a kontaktuje server správou *INVITE*, ktorú server prepošle volanému používateľovi. Klient musí vložiť do požiadavky pole *Via*. Toto bude obsahovať hodnoty

SIP/2.0/UDP 192.168.1.2:5067;branch=z9hG4bK6b1a9856;rport

Tieto hodnoty určujú IP adresu klienta a zdrojový port. Smerovač s funkciou NAT preloží aj ako IP adresu, tak aj zdrojový port. Proxy server správu prijme a zistí, že IP adresa a port prijatého IP paketu sa nezhodujú s hodnotami v prijatom poli *Via*. Do existujúceho poľa *Via* pridá atribút *received*, kam vloží IP adresu, z ktorej prijal správu, a atribút *rport*, kam vloží hodnotu portu, z ktorého bol paket prijatý. Následne vytvorí nové pole *Via*, ktoré zapíše nad už existujúce a vloží doň svoju IP adresu a port. Toto pole už ale musí obsahovať novú hodnotu *branch*, ktorá je rôzna od iných hodnôt *branch* v požiadavke. Takúto správu môže poslať požadovanému klientovi. Klient odpovie napríklad odpoveďou 200 OK, ktorou oznámi že prijíma hovor. Polia *Via* prepíše z požiadavky a celú správu pošle na adresu, z ktorej požiadavku prijal, teda na adresu servera. Ten správu prijme, skontroluje prvé pole *Via*, zistí, že ho vložil on a odstráni ho. Následne prečíta nasledujúce pole *Via*, a správu odošle na adresu a port, ktoré predtým vložil do atribútu *received* a *rport*. Takýto IP paket prijme smerovač volajúceho a podľa portu ho pošle na internú IP adresu, z ktorej bola požiadavka *INVITE* odoslaná.

2.4.6 Pole Content-Type

Označuje typ obsahu tela správy. Pre nás je zaujímavý typ *application/SDP*, ktorý hovorí, že v tele správy sa budú nachádzať informácie protokolu SDP, ktorý je popísaný nižšie. Toto pole je povinné, ak je obsah tela správy nenulový. Ďalším príkladom hodnoty poľa môže byť text/html.

2.4.7 Pole Content-Length

Určuje dĺžku tela správy v desiatkovej sústave v oktetoch bitov poslaných odosielateľom. V prípade nulového obsahu musí toto pole obsahovať číslo 0. Toto pole je povinné, ak sa používa spoľahlivý protokol ako TCP [1].

2.5 Odpovede protokolu SIP

Odpovede môžeme rozdeliť na dva typy: finálne a dočasné. Dočasné odpovede, narozdiel od finálnych, neukončujú transakcie a iba spresňujú a informujú stranu klienta o prebiehajúcich udalostiach. Odpovede sa delia do šiestich skupín. Každá skupina je reprezentovaná číslom (1xx, 2xx, 3xx, 4xx, 5xx, 6xx). Jednotlivé čísla reprezentujú konkrétnu odpoveď.

Konkrétne zaradenie odpovedí do skupín spolu s najdôležitejšími prípadmi sú:

- **1xx – informatívne**, *100 TRYING, 180 RINGING*
- 2xx – oznamujúce úspešné vykonanie, *200 OK*
- **3xx – presmerovanie**, *301 MOVED PERMANENTLY* (trvalo presmerované), *302 MOVED TEMPORARILY* (dočasne presmerované)
- **4xx – chyby na strane klienta**, *400 BAD REQUEST* (chybná požiadavka), *401 UNAUTHORIZED* (neautentifikované, zakázaná požiadavka), *404 NOT FOUND* (neidentifikovaný používateľ), *407 PROXY AUTHENTICATION REQUIRED* (potrebná autentifikácia na proxy serveri), *486 BUSY HERE* (používateľ nedosiahnuteľný)
- 5xx – chyby na strane servera
- 6xx – globálne zlyhania [1, 3]

2.6 SIP a nespoľahlivé protokoly

Pôvodné RFC 2543 žiadalo od implementácií použitie protokolu UDP. Dôvod na výber tohto protokolu je nižšia odozva a flexibilnejšia kontrola preposielania dát. Dnešné linky sú už menej náchylné na odozvu a tak sa väčším problémom stáva fragmentácia správ. Protokol SIP nemá definované žiadne mechanizmy na kontrolu fragmentácie, ani na vyrovnanie sa so stratou časti správy. Preto nové RFC 3261 vyžaduje implementáciu ako protokolu UDP, tak aj protokolu TCP a na komunikáciu odporúča protokol TCP. Všetky implementácie musia byť schopné prenášať najväčšie možné správy. V prípade UDP je to 65535 bajtov (vrátane IP a UDP hlavičiek) [RFC3261].

Transakcie v zmysle nespoľahlivých protokolov môžeme rozdeliť na *INVITE* a nie *INVITE* transakcie. V oboch prípadoch sa spoliehame na retransmisiu, keďže UDP protokol nemá nástroje na detekciu straty informácie.

2.6.1 Nie INVITE transakcie

Tieto správy sú charakteristické dvojcestnou výmenou a teda jedným potvrdením. V prípade, že zo strany servera (rozumie sa ako volaná strana) nedôjde k odpovedi na požiadavku do 500 ms (hodnoty časovačov v tejto kapitole sú pre zjednodušenie prevedené na ich štandardné hodnoty definované v RFC 3261) klient požiadavku prepošle. Ak naďalej neregistruje odpoveď, toto oneskorenie zdvojnásobí. Nasledujúca požiadavka sa prepošle po jednej sekunde. Takto pokračuje, až oneskorenie retransmisie dosiahne 4 sekundy. Klient sa bude takýmto spôsobom pokúšať poslať požiadavku po dobu 32 sekúnd. Ak je počas tejto doby prijatá nepovinná odpoveď, interval retransmisie sa ihneď posunie na 4 sekundy. Po dobe 32 sekúnd je relácia považovaná za stratenú. V prípade oznámenia od nižších vrstiev o strate IP konektivity (*ICMP unreachable paket*) je relácia ukončená okamžite.

Ak je stratená odpoveď zo strany servera, klient opätovne prepošle požiadavku, na ktorú server následne odpovie. Server odpovede nepreposiela.

Nepovinným odpovediam protokol SIP doručenie negarantuje. Keďže niektoré nepovinné odpovede sa stali v určitých implementáciách potrebné, do dokumentu RFC 3261 bola pridaná požiadavka *PRACK* (ak je potrebné aby počas čakania na odpoveď nebol audio kanál prázdny, *RINGING* odpoveď zabezpečí, aby opačná strana počas doby čakania vyzváňala).

2.6.2 INVITE transakcie

INVITE transakcia je trojcestnou výmenou. Je teda nutné potvrdiť potvrdenie. Ak chce používateľ nadviazať hovor, je prirodzené, že volaná strana neprijme hovor okamžite. Preto požiadavka *INVITE* narozdiel od iných požiadaviek, nedostane žiadnu informáciu o tom, že správa bola prijatá druhou stranou. Klient *INVITE* požiadavku prepošle po pol sekunde a neskôr túto dobu zdvojnásobuje. Až do doby, kedy prijme finálnu alebo nepovinnú odpoveď. Čakanie na odpoveď môže trvať až 180 sekúnd. Ak klient prijme nepovinnú odpoveď, požiadavku *INVITE* už nepreposiela. Ak klient zo strany servera, aj napriek zaslanej nepovinnej odpovede, pošle opäť požiadavku *INVITE*, server vie, že odpoveď sa stratila a nepovinnú odpoveď prepošle.

Prijatím hovoru alebo presmerovaním, volaná strana vyšle finálnu odpoveď 2xx alebo 3xx. V prípade nepotvrdenia požiadavkou *ACK* druhou stranou, server tieto odpovede prepošle za pol sekundy a následne túto dobu zdvojnásobuje. Túto retransmisiu ukončí iba požiadavka *BYE* a v prípade 3xx, 4xx, 5xx odpovedí to je požiadavka *CANCEL*. Tretí krok výmeny je požiadavka *ACK*. Ako sme už spomínali, jej hlavné polia sú totožné s *INVITE* požiadavkou. Pole *Cseq* má číselnú hodnotu taktiež rovnakú, ale meno metódy je *ACK*. Retransmisia požiadavky *ACK* je podobná nie *INVITE* požiadavkám. Ak server neprijme *ACK* požiadavku, znovu prepošle finálnu odpoveď. Takto sa klient dozvie, že jej požiadavka bola stratená.

Odpoveď *200OK* je jediná, ktorá musí byť potvrdená globálne, až ku klientovi. Ak je medzi volanými stranami proxy server, odpovede 3xx, 4xx, 5xx môžu byť potvrdené iba proxy serverom a klient sa o nich nemusí dozvedieť. Napríklad, keď volaná strana má viacero zariadení, ktoré zvonia postupne, teda nie všetky naraz, a prvé zariadenie odmietne hovor, toto odmietnutie potvrdí len proxy server. V prípade finálnej odpovede *200OK,* táto informácia musí byť prešírená až ku klientovi. Takýmto spôsobom sa o prijatí hovoru dozvedia všetky proxy servery medzi klientom a serverom.

V prípade spoľahlivých protokolov ako je TCP, sa správy nepreposielajú. Jedinou výnimkou sú finálne odpovede 2xx, ktoré musia byť preposlané aj protokolom ako je TCP. Ak by niektorý proxy server zmenil protokol na UDP, tak aj v takomto prípade môžeme zaručiť doručenie odpovede [1, 3, RFC2543].

2.7 Protokol SDP

Protokol SIP bol od začiatku navrhovaný ako jednoduchý a prispôsobivý protokol. Umožňuje teda nadviazať hovor s rôznou škálou parametrov a vlastností. Aby boli obe strany o týchto vlastnostiach informované, musia byť informovaní prostredníctvom transakcie. V našom príklade to bolo v rámci prvej transakcie. SIP protokol slúži len na nadviazanie relácie a nemá metódy na oboznámenie náprotivka o chcenom spôsobe komunikácie. Preto sa v transakcii spolu so

SIP správami odosielajú aj vnorené správy protokolu SDP (*Session Description Protocol*). Tento obsahuje všetky informácie na vhodné nastavenie RTP prenosu.

Protokol SDP je definovaný v RFC 2327. Protokol definuje nastavenia protokolu RTP ako port nad protokolom UDP, kódovanie obrazu a zvuku, základné informácie o relácii ako aj kontaktné informácie. Informácie v protokole nie sú špeciálne kódované a prenášajú sa vo forme typ = hodnota. Najdôležitejšie hodnoty sú

- **v** – verzia, hodnota je vždy 0
- **o** – vlastník, informácie o užívateľskom mene, ID relácie, verzia, typ siete, typ adresy, adresa
- **s** – názov relácie
- **c** – spojenie, typ siete, typ adresy, adresa
- **t** – čas, počas ktorého je relácia aktívna. Hodnoty sú dve čísla, medzi ktorými je relácia aktívna. Čas je reprezentovaný v sekundách vo formáte NTP
- **m** – opis média. Hodnoty v poradí znamenajú typ média, port, typ transportného protokolu, čísla určujúce typ kódovania v poradí v akom klient preferuje ich použitie. Číslo portu je vždy párne číslo. Nasledujúce nepárne číslo je číslo portu pre protokol RTCP (*Real-Time Control Protocol*).
- **a** – atribúty média

Príkladom informácie prenášanej SDP protokolom je napríklad:

```
v=0
o=Nokia-SIPUA 63390712757083500 63390712757083500 IN IP4
85.248.125.41
s=-
c=IN IP4 213.192.59.91
t=0 0
m=audio 57854 RTP/AVP 96 0 8 97 18 98 13
a=sendrecv
a=ptime:20
a=maxptime:200
a=fmtp:96 mode-change-neighbor=1
a=fmtp:18 annexb=no
a=fmtp:98 0-15
a=rtpmap:96 AMR/8000/1
a=rtpmap:0 PCMU/8000/1
a=rtpmap:8 PCMA/8000/1
a=rtpmap:97 iLBC/8000/1
a=rtpmap:18 G729/8000/1
a=rtpmap:98 telephone-event/8000/1
a=rtpmap:13 CN/8000/1
```

3 IP Multimedia Subsystem

V posledných dvadsiatich rokoch mobilné a pevné komunikačné siete prešli veľkými zmenami. Vo svete mobilných sietí to boli najprv siete prvej generácie (1G), potom siete druhej generácie (2G) a dnes sú to už siete tretej generácie (3G), či (3.5G).

V pevných sieťach dominovali na prenos hlasového signálu *Public Switched Telephone Network* (PSTN) a *Integrated Services Digital Network* (ISDN) siete. V posledných rokoch sa záujem o Internet rapídne zvýšil a tým viac a viac používateľov začalo využívať výhody pripojenia technológie *Asymmetric Digital Subscriber Line* (ADSL).

Tento typ pripojenia naštartoval real-time komunikáciu, ako napr.: chatovacie aplikácie, online hry, či *Voice over IP* (VoIP). Aplikácie, ktoré ponúkajú služby zdieľaného prehliadania (*shared browsing*), sú vlastne peer-to-peer entitami. Schopnosť vytvoriť peer-to-peer spojenie v IP protokole je jednou z kľúčových súčastí týchto aplikácii. Táto forma v komunikácii presiahla možnosti *Plain Old Telephone Service* (POTS) sietí.

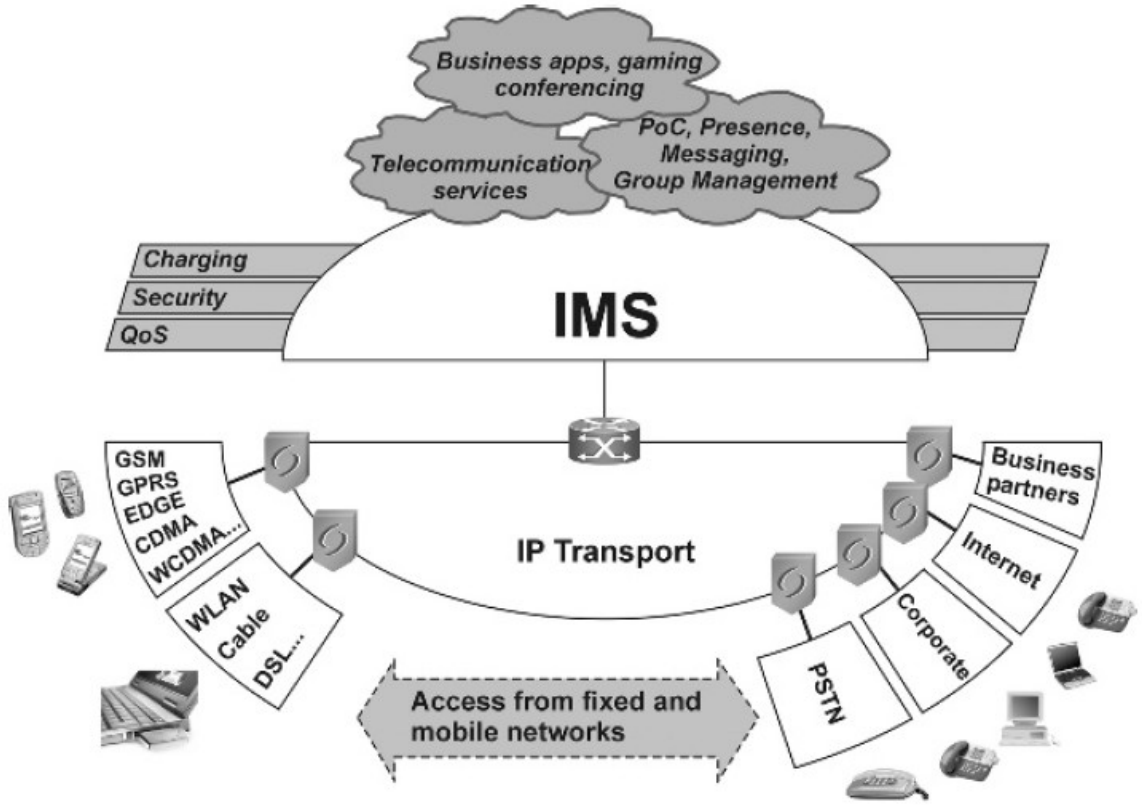
Aby aplikácie založené na protokole IP mohli komunikovať, museli mať vytvorený tento mechanizmus na spojenie s daným korešpondentom. V súčasnosti telefónne siete zaisťujú tento mechanizmus na vytvorenie spojenia. Pri vytáčaní *peer*_a, je vytvorené „ad-hoc“ spojenie medzi dvoma terminálmi prostredníctvom IP siete. Túto možnosť však ponúkajú len zariadenia poskytovateľov v Internete. A teda pri komunikácii s uzavretými systémami bolo potrebné vytvoriť globálny systém - IP Multimedia Subsystem (IMS). IMS dovolilo aplikáciám v IP sieťach vytvoriť *peer-to-peer* a *peer-to-content* spojenie ľahko a bezpečne.

Definícia podľa [5]:

„**IMS je globálna, prístupovo nezávislá (*access-indipendent*) a postavená na štandarde IP spojenia architektúra kontrolných služieb, ktorá umožňuje používateľom používať rôzne typy multimediálnych služieb vďaka bežným IP protokolom**.“

Vhodná integrácia hlasových a dátových služieb zvyšuje produktivitu a celkovú efektívnosť pri vývoji inovatívnych aplikácii integrujúcich hlas, dáta a multimédia, napr.: multimediálny chat, *push to talk* alebo konferenčný hovor.

Na Obr. 3.1 je znázornená konvergovaná komunikačná sieť pre pevné a mobilné prostredia. Je to práve IMS, ktoré zaviedlo *multimedia session control* v paketovo prepínaných (*packet-switched*) doménach a v tom istom čase prinieslo funkcionalitu prepínania okruhov (*circuit-switched*) v paketovo prepínaných sieťach.



Obr. 3.1: IP Multimedia Subsystem [5]

3.1 Architektúra IMS

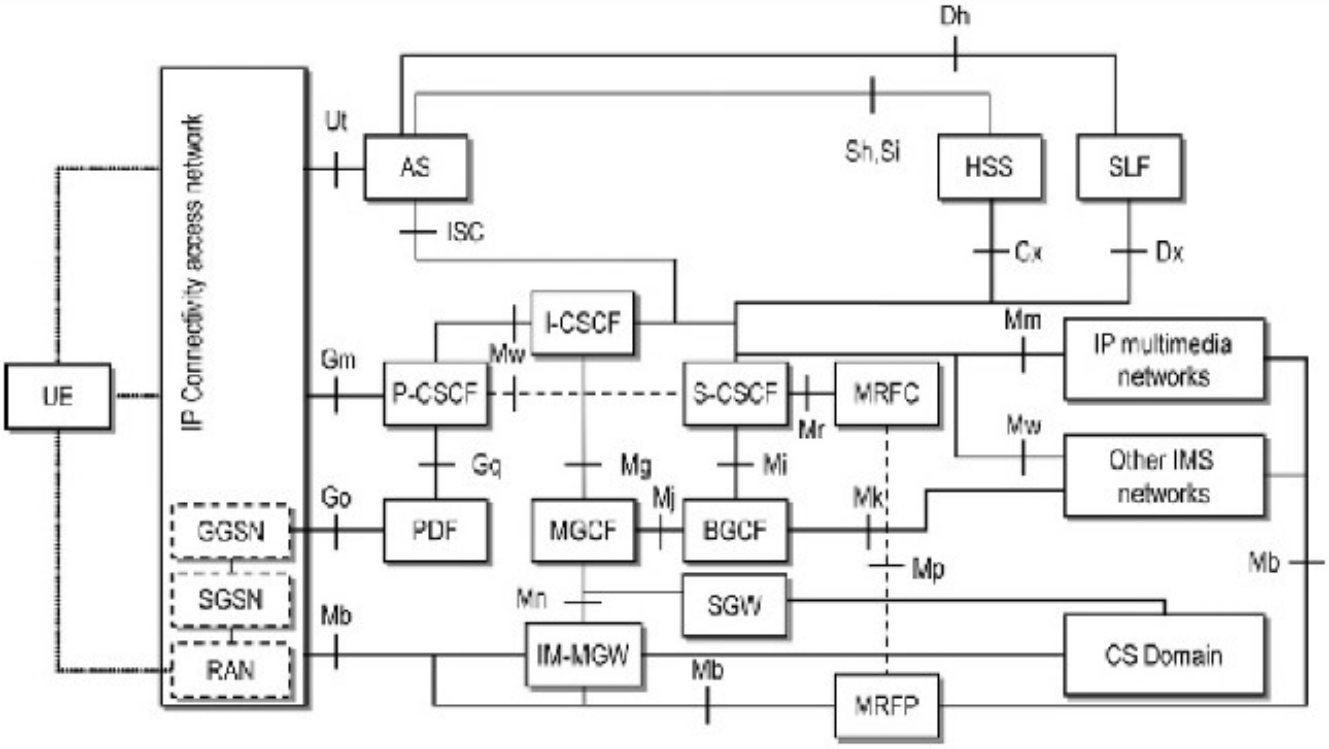
V tejto kapitole budú popísané jednotlivé entity a funkcie IMS. Treba poznamenať, že ohniskom IMS je kontrola relácií (*session control*). Keď sa pozrieme na 3GPP štandard pre IMS, neuvidíme žiadne prepínače, či iné sieťové uzly ako súčasti IMS.

3.1.1 IMS entity a funkcie

IMS entity môžeme rozdeliť do približne šiestich kategórii [5]:

- *Call Session Control Function* (CSCFs)
- Databázy (HSS, SLF)
- Služby (AS, MRFC, MRFP)
- Vnútorné funkcie (BGCF, MGCF, IMS-MGW, SGW)
- Podporné funkcie (PDF, SEG, THIG)

Jednotlivé prepojenie týchto entít je znázornené na Obr. 3.2.



Obr. 3.2: Architektúra IMS

3.1.2 Call Session Control Function (CSCF)

Call session control je primárna funkcia jadra siete. Táto funkcionalita je distribuovaná v celej sieti, aby zabezpečovala efektivitu a škálovateľnosť. Existujú tri typy *Call Session Control* funkcií:

- *Proxy-CSCF* (P-CSCF),
- *Serving-CSCF* (S-CSCF)
- *Interrogating-CSCF* (I-CSCF)

Každá funkcia CSCF má vlastnú úlohu, ale spoločne pracujú pri registrácii a zriadení relácie a pri SIP smerovaní. Navyše, všetky funkcie sú schopné posielať údaje *offline*. Rozdiel medzi týmito funkciami je znázornený na Obr. 3.3 spočívajúci v ich rozdielnom účele a procedúrach, ktoré vykonávajú [6].

Obr. 3.3: Funkcie IMS

Proxy CSCF (P-CSCF)

Proxy Call Session Control funkcia (P-CSCF) je prvým prístupovým bodom k SIP doméne z perspektívy kontroly relácie. Cez P-CSCF prechádzajú SIP správy. Prvá komunikácia je registrácia

IP adresy zariadenia. Ak zariadenie chce komunikovať s inými zariadeniami, musí najprv s daným zariadením zahájiť reláciu. Táto relácia je najprv vytvorená cez P-CSCF. Skôr ako je daná správa poslaná ďalej CSCF, P-CSCF pridá do hlavičky vlastné parametre.

Keďže SIP protokol je textovo orientovaný, 3GPP kvôli zrýchleniu inicializácie relácie poverilo P-CSCF kompresiou protokolu SIP. Ďalšie úlohy pre P-CSCF sú napríklad: IPSec bezpečnosť, interakcia s *Policy Decision Function* (PDF) a detegovanie mimoriadnych relácií (*Emergency Session Detection*).

P-CSCF je vstupným bodom do IMS, zohráva z hľadiska bezpečnosti dôležitú úlohu.

V P-CSCF je prítomné *Policy Decision Function* (PDF), ktoré rozhoduje ako sa správať v špecifických scenároch. PDF dovoľuje operátorom vytvoriť pravidlá pre prístup do siete. PDF kontroluje *Policy Enforcement Function* (PEF) na nosnej sieti. To povoľuje operátorom kontrolovať tok paketov na nosnej sieti v súlade s cieľovou a zdrojovou adresou a právami.

P-CSCF takisto preveruje smerovanie. Kontroluje, že či prijaté smerovanie v SIP žiadosti alebo odpovedi je to isté smerovanie, ktoré bolo identifikované, keď sa dané zariadenie zaregistrovalo do siete. Ak smerovacie hlavičky neobsahujú adresy zhodné s uloženými adresami počas registrácie v P-CSCF, znamená to, že boli zmenené.

P-CSCF si uchováva tieto informácie: adresu zariadenia (IP adresu) a verejné a privátne informácie používateľov.

V P-CSCF sú vedené informácie o stave relácie počas celého spojenia. V prípade, že zariadenie stratí IP konektivitu, P-CSCF o tom notifikuje IMS prostredníctvom správy *CANCEL* všetkým účastníkom danej relácie.

Keď to všetko zhrnieme, môžeme povedať, že P-CSCF je akási brána do IMS. Je zodpovedné za to, že zariadenie, ktoré pristupuje do siete, je registrované a má povolený prístup do IMS, avšak nezabezpečuje autentifikáciu a autorizáciu [6].

Interrogating CSCF (I-CSCF)

Zatiaľ čo P-CSCF je vstupným bodom do IMS, I-CSCF slúži ako brána do každej individuálnej IMS siete. I-CSCF pomáha chrániť S-CSCF a HSS pred neautorizovaným prístupom z iných sietí. Keď S-CSCF prepošle požiadavku alebo odpoveď do inej siete, správa je ďalej preposlaná do I-CSCF, ktoré danú správu prepošle ďalej do cieľovej siete.

Dôležitou funkciou I-CSCF je priradenie S-CSCF. S-CSCF je priradené v závislosti od schopnosti a samozrejme politiky poskytovateľa služieb.

Jednotlivé úlohy I-CSCF by sme mohli zhrnúť nasledovne [5]:

- Získanie mena nasledujúceho uzla (S-CSCF alebo aplikačný server) z *Home Subscriber Server* (HSS)
- Priradenie S-CSCF na základe prijatých schopností z HSS
- Priradenie prichádzajúcich smerovacích požiadaviek k S-CSCF alebo aplikačnému serveru
- Poskytovanie *Topology Hiding Inter-network Gateway* (THIG) funkcionality

Serving CSCF (S-CSCF)

S-CSCF je jadrom IMS. Kontroluje všetky aspekty odberateľských služieb, pričom eviduje stav každej relácie, ktorá bola inicializovaná. S-CSCF kontroluje správy a doručenie obsahu.

Takisto sprostredkúva informácie o stave registrácie iným aplikáciám a udržiava kontrolu počas celej doby, ako je dané zariadenie registrované.

Zo SIP perspektívy je S-CSCF registrátor, ktorý je schopný autentifikácie odberateľov, ktorí sa pokúšajú registrovať.

S-CSCF udržiava nasledovné informácie o zaregistrovanom zariadení [5]:

- HSS adresu (*HSS address*)
- Používateľský profil (*User profile*)
- P-CSCF adresu (*P-CSCF address*)
- P-CSCF doménu (*P-CSCF domain*)
- Verejnú používateľovu identitu (*Public user identity*)
- Privátnu používateľovu identitu (*Private user identity*)
- IP adresu zariadenia (*Device IP address*)

S-CSCF sprístupňuje služby poskytovaním prístupu k rôznym aplikačným serverom (*Application Servers* - ASs) v sieti. To znamená, že S-CSCF potrebuje poznať, že ku ktorej službe má odberateľ povolený prístup a samozrejme adresu servera, ktorý danú službu poskytuje. S-CSCF pristupuje k HSS, aby získalo používateľov profil, ktorý obsahuje aj profil služieb.

Takisto aj konvertovanie adresy je rola S-CSCF. Odkedy SIP smeruje na základe SIP URI, každé TEL URI musí by preložené do SIP URI. Podobne je to aj so smerovaním z IMS do PSTN. S-CSCF pristupuje do ENUM/DNS pre konvertovanie adries do SIP URI na preposielanie správ adresátovi.

ENUM/DNS aplikácie sú umiestnené na tom istom serveri alebo môžu byť ako samostatné ENUM funkcie.

S-CSCF pracuje ako statické proxy a musí si udržiavať stav všetkých registrácií a relácií, ktoré má pod kontrolou.

Jednoducho povedané S-CSCF je jadro IMS. Je to podstata alebo bod siete, ktorý poskytuje operátorom kontrolu doručenia služieb a všetkých relácií.

3.1.3 Databázy

IMS pracuje s dvoma databázami: *Home Subscriber Server* (HSS) a *Subscription Locator Function* (SLF) [6].

Home Subscriber Server (HSS)

HSS obsahuje hlavné údaje o všetkých používateľoch a údaje súvisiace so službami IMS. Sú to údaje ako: identita používateľa (verejná a privátna), informácie o registrácii, prístupové parametre a informácie o spúšťaní služieb [3GPP TS 23.002].

Hlavnou funkcionalitou HSS je šifrovanie a autorizovanie každej správy. Keď sa odberateľské zariadenie zaregistruje do siete, priradené S-CSCF vyzve zariadenie na zadanie správnych údajov, ktoré sú uložené v HSS [6].

Subscription Locator Function (SLF)

Táto funkcionalita môže byť vstavaná do inej databázy alebo môže fungovať ako samostatný server. Hlavným účelom tejto funkcie je lokalizovanie HSS a S-CSCF, ktoré je priradené danému odberateľovi. Je to indexovanie a mapovanie používateľskej identity k S-CSCF/HSS v závislosti od registrácie [6].

3.1.4 Služby

Jednotlivé služby môžeme rozdeliť na tri kategórie: *Multimedia Resource Function Controller* (MRFC), *Multimedia Resource Function Processor* (MRFP) a Aplikačné servery (AS) [5].

Služby MRFC a MRFP

MRFC a MRFP spolu obsluhujú nosné služby ako konferenciu, správy k používateľovi alebo nosné kódovanie v IMS architektúre.

MRFC má za úlohu udržiavať SIP komunikáciu od S-CSCF a k S-CSCF a kontroluje MRFP. MRFP zabezpečuje nasledovné funkcie [5]:

- Spracovanie SIP relácií prijatých z IMS
- Schopnosť zahájenia SIP žiadostí
- Schopnosť posielania informácií o konte

Aplikačné servery

Aplikačný server - AS má viac úloh v IMS, hoci nie je časťou IMS jadra. Existuje veľa možností využitia AS ako napr. aplikácie, doručovacie služby, doručenie správ, prezentačný server či video konferencia. Jednotlivé služby, ktoré poskytuje AS, sú identifikované identifikátorom (adresou).

AS vie vytvoriť SIP dialóg, čo znamená, že sa vie tváriť ako používateľský klient pre SIP [6].

3.1.5 Vnútorné funkcie

V literatúre sú spomínané štyri vnútorné funkcie, ktoré sa starajú o signalizáciu medzi IMS a *Circuit Switched Core Network* (CS CN).

Breakout Gateway Control Function (BGCF) je brána k sieťam iných poskytovateľov.

MGCF (*Media Gateway Control Function*) je pripojená do PSTN domény a zabezpečuje funkcie brány medzi SS7 (*Signaling System 7*) a IP/SIP stranou.

Keď SIP požiadavka dorazila do MGCF, v MGCF musela byť vykonaná konverzia medzi SIP protokolom a *ISDN User Part* (ISUP) alebo *Bearer Independent Call Control* (BICC). Táto správa bola poslaná a konvertovaná *Signaling Gateway* (SGW) do CS CN [5].

3.1.6 Podporné funkcie

Policy Decision Function (PDF) je zodpovedné za politiku rozhodovania sa na základe informácií získaných z relácie a multimediálne orientovaného obsahu prichádzajúce z P-CSCF.

Security Gateway (SEG) má za úlohu chrániť prevádzku medzi bezpečnostnými doménami. V IMS je prevádzka smerovaná cez SEG, špeciálne vtedy, keď zdrojová a cieľová doména sú v rôznych bezpečnostných doménach [5].

Topology Hiding Inter-network Gateway (THIG) je používaná na ukrytie konfigurácie, kapacity a topológie siete od siete, ktorá sa nachádza na druhej strane operátora. Keď chce operátor využiť ukrytie týchto funkcií, musíme umiestniť THIG na ceste smerovania správ pri prijímaní alebo odosielaní z IMS sietí.

4 Optimalizácia siete

Návrh siete so všetkými prostriedkami v štandardnej a nezmenenej konfigurácii funguje a funguje dobre. Avšak len do chvíle, keď sa objaví problém na sieti, alebo zmena v topológii siete. Čas, do ktorého sa sieť „spamätá“, je v implicitnej konfigurácii dlhý. Našim cieľom je optimalizovať sieť predovšetkým na sieťovej vrstve tak, aby sme dosiahli zlepšenie parametrov, skrátenie času konvergencie, kedy je sieť v nestabilnom stave a zároveň minimalizovali negatívne dôsledky optimalizácie.

Pri optimalizácii sa stretávame s pojmom konvergencia. Konvergencia je schopnosť a rýchlosť spolupracujúcich zariadení v sieti aktualizovať svoje údaje na nové hodnoty po zmene pomerov v sieti. Konvergencia je proces. Stav, kedy je konvergencia ukončená, budeme nazývať skonvergovaným stavom. Konvergencia je kľúčový aspekt v moderných sieťach. Pri konvergencii sledujeme najmä čas jej trvania. Čas konvergencie je trvanie konvergencie od vzniku zmeny, jej detekcie, až po záverečnú dohodu a aktualizáciu na zariadeniach. Je to čas, ktorého trvanie vyjadruje porušený skonvergovaný stav. Rýchlejšia konvergencia znamená kratšiu dobu nedostupnosti siete. Je to práve čas, ktorý sa stal kritickým faktorom dobre fungujúcej siete. Neexistuje okamžitá konvergencia, pretože tento proces vždy trvá nejaký čas.

4.1 Topológie

Sieť je nutné začať optimalizovať už od fyzickej vrstvy, teda od návrhu siete. Ak sieť nie je možné na fyzickej vrstve nijakým spôsobom zmeniť, alebo len čiastočne, nepredstavuje to prekážku pre nadväzujúcu optimalizáciu na vyšších vrstvách.

Ako Filip Burda vo svojej bakalárskej práci [7] zistil, je veľmi ťažké rozhodnúť, ktorý typ topológie je optimálny pre rýchlu konvergenciu, pretože každá má svoje nedostatky aj z pohľadu konvergencie. Všeobecne, zbernica je dobrá na rozšírenie informácie, no vďaka kolíziám sa môže stať absolútne nevhodnou voľbou. Pri topológiach založených na kruhu sa spoliehame, že nedôjde k výpadku v rámci kruhu. Výpadok automaticky môže znamenať vyradenie časti siete z prevádzky. Pri hviezde sa stáva kritickým uzlom už len centrálny rozbočovač. Navyše tým, že uzly sú vzájomne napojené na jeden centrálny uzol, konvergencia je veľmi rýchla. Z tohto pohľadu nám najlepšie vychádza práve hviezda, ktorá je dobrým kompromisom stability a konvergencie.

Ideálnejšie riešenia však ponúkajú hybridné topológie. Veľmi zaujímavou voľbou je čiastočný *mesh*, ktorý je odolný aj voči výpadkom, aj poskytuje redundanciu a aj rýchlu konvergenciu. Táto voľba kombinovaná so stromovou topológiou je v praxi dobre realizovateľná a stále z pohľadu konvergencie zaujímavá. Čiastočný *mesh* znamená prepojenie zariadení nie každého s každým, ale každého s niekoľkými. Stromová topológia je v podstate hierarchický návrh siete s postupným rozvetvovaním sa po istú úroveň.

Vhodný výber topológie je však len prvým krokom k naplneniu cieľa rýchlo konvergujúcej siete. V nasledujúcej časti sa pozrieme na rôzne typy zariadení, ich výber a príslušenstvo [8].

4.2 Zariadenia

Ak by sme chceli prepojiť každý koncový uzol so všetkými ostatnými koncovými uzlami po svete, asi by z našej planéty bolo len klbko prepletených káblov. Na vzájomnú komunikáciu zariadení sa používajú rôzne zariadenia, ktoré slúžia ako medziuzly v komunikácii. Podľa toho, ako dokážu k dátovému toku pristupovať a narábať s ním, rozdeľujeme zariadenia na opakovače, rozbočovače, mosty, prepínače a smerovače. Samozrejme zariadení je na výber omnoho omnoho viac, no tieto sú základom siete. Z pohľadu poskytovateľa služieb sú zaujímavé smerovače a prepínače na tretej vrstve, ktoré popíšeme.

Smerovače pracujú na sieťovej vrstve a dáta smerujú. Dáta sú posielané väčšinou deterministicky. Vzájomné informovanie sa smerovačov zabezpečuje dosiahnuteľnosť všetkých častí celosvetovej siete.

Smerovače sú z vyššie spomenutých zariadení najviac inteligentné, avšak spôsobujú aj najväčšie oneskorenie, čo predlžuje čas konvergencie. Problém s prepínačmi je ten, že môže existovať len práve jedna cesta medzi dvomi akýmikoľvek prepínačmi. Toto obmedzenie predchádza vzniku topologických slučiek, ktoré by vyradili sieť z prevádzky. Čas oneskorenia je o niečo menší, ale práve toto obmedzenie naopak spomaľuje rozšírenie správy o zmene do siete, čo spomaľuje konvergenciu. Pokiaľ to je možné, mali by byť vyberané predovšetkým prepínače na tretej vrstve. Tieto zariadenia sú aj prepínačmi aj smerovačmi zároveň. Výhodou je veľký počet portov, typický pre prepínače a schopnosť nielen prepínania, ale aj smerovania v jednom zariadení. Avšak smerovanie narozdiel od smerovačov je realizované hardvérovo.

Pri výbere zariadenia zohľadňujeme platformu, vhodnosť pre našu sieť a jeho využiteľnosť. Neoddeliteľnou súčasťou sú aj schopnosti jeho operačného systému. Možnosti, ktoré sú nám ponúkané, sú pre nás dôležité. Postupná evolúcia v sieťach a vylepšenia v implementácii protokolov sa zavádzajú aj do operačných systémov zariadení. Odporúčame preto pred začiatkom optimalizácie nainštalovať najnovší operačný systém. Operačný systém (OS) sa napríklad na Cisco zariadeniach nazýva IOS (*Internetwork Operating System*). Na zariadeniach od firmy Juniper to je JUNOS. V práci sa budeme orientovať na Cisco zariadenia s IOS verziami 12.3 a 12.4 pre smerovače. Staršie verzie nemusia podporovať všetky techniky rýchlej konvergencie, ktoré popíšeme a nemusí byť možná optimalizácia časov na dosiahnutie konvergencie pod jednu sekundu pre daný smerovací protokol.

V ďalších kapitolách sa zameriame len na zbernicovú topológiu a hviezdu. Základným prenosovým „médiom“ bude *ethernet*, keďže je v súčasnosti najpoužívanejší a najrozšírenejší. Určite by bolo zaujímavé sa pozrieť aj na iné média, ale berieme taktiež ohľad na laboratórne podmienky.

Na linkovej vrstve máme viacero prostriedkov na zlepšenie konvergencie. Keďže sa v tomto projekte neuvažuje o linkovej vrstve vo vnútri siete u poskytovateľa služieb, nebudeme sa ani my venovať tejto optimalizácii. Podrobnejšie o tejto optimalizácii je možné nájsť v bakalárskej práci Filipa Burdu [7] a v jeho vedeckej publikácii [9].

Sieťová vrstva tvorí podstatnú zložku siete a budeme sa jej patrične venovať na nasledujúcich stranách.

5 Smerovanie v TCP/IP

Hlavnou úlohou sieťovej vrstvy je zabezpečiť najlepšie možné doručenie paketu od odosielateľa až k príjemcovi.

Zariadenia, ktoré implementujú túto funkcionalitu siete sú smerovače. Tieto zariadenia musia učiniť rozhodnutia, cez ktoré rozhranie majú odoslať pakety tak, aby sa optimálne dostali k svojmu cieľu.

Pri tomto procese je nutné zabezpečiť, aby zariadenia, ktoré poskytujú takúto funkcionalitu, mohli so sebou komunikovať a aby sa mohli správať ako jeden konzistentný celok, ktorý si navzájom dobre rozumie. Týmto by bolo možné dosiahnuť, aby si vzájomne aktualizovali a spravovali svoje tabuľky. K tomuto účelu je možné využiť smerovacie protokoly. Tieto protokoly poskytujú algoritmy na ukladanie smerovacích informácií.

5.1 Klasifikácia smerovania

Smerovanie je proces výberu ciest v sieti medzi uzlami siete. Rozdeľujeme ho na dve základné skupiny: statické smerovanie a dynamické smerovanie.

5.1.1 Statické smerovanie

Statické smerovanie (*Static Routing*) je smerovanie, ktoré sa konfiguruje na každom smerovači individuálne, podľa návrhu topológie a rozloženia smerovačov v sieti. Pri akejkoľvek zmene topológie nastáva potreba manuálne prekonfigurovať zariadenia.

Tým, že si smerovače nevymieňajú informácie o smerovaní, neprichádza k výmene žiadnych obslužných paketov, ktoré následne nespotrebovávajú vyžadovanú šírku pásma.

Využitie statického smerovania sa v súčasnosti ukazuje výhodné v ich konfigurácii ciest v prípade predvolenej brány (*default gateway*). Je to v prípadoch, kedy je vhodné, aby administrátor po zvážení predvolil cestu, ktorou bude smerovač posielať pakety pokiaľ nenájde záznam v smerovacej tabuľke.

Ďalšie využitie statických ciest je v listových oblastiach (*Stub Networks*). Pri týchto oblastiach nedochádza k distribúcii paketov medzi viacerými oblasťami alebo autonómnymi systémami. Ak existuje len jeden východ z oblasti, je vhodné predvoliť cestu, ktorá je výstupnou z oblasti. Tieto oblasti budú popisované podrobnejšie pri protokole OSPF.

Medzi najväčšiu nevýhodu statického smerovania patrí nízka flexibilita. Čím viac sa topológia rozrastá, tým je väčšia náročnosť na správu takejto siete. To je dôvod, prečo sa statické smerovanie vo všeobecnosti považuje za nie veľmi vhodné. Obzvlášť nie je vhodné pre tzv. „živé siete“, v ktorých často dochádza k zmenám topológie.

5.1.2 Dynamické smerovanie

Pri statickom smerovaní akékoľvek rozšírenie siete znamená jej väčšiu zložitosť a náročnosť na údržbu. V dynamickom smerovaní tento problém do značnej miery odpadá.

Dynamické smerovanie, tiež známe ako adaptívne dynamické smerovanie (*Adaptive Routing*), popisuje schopnosť smerovačov dynamicky určovať najlepšie cesty v závislosti od zmeny topológie.

Za pomoci adaptívneho smerovania je možné odkloniť premávku kvôli výpadku uzla, ktorým práve prechádza optimálna trasa pri dostatočnej redundancii. V prípade výpadku uzla je potrebné vykonať nový prepočet zahrňujúci zmenu topológie.

Použitie adaptívneho smerovania nesie so sebou niekoľko negatívnych faktov. V prvom rade vyžaduje istú šírku pásma (*bandwidth*) a spotrebu procesorového času. Po získaní vhodných aktualizácií protokol príslušným algoritmom vypočíta, ktoré cesty sú najlepšie a budú použité na smerovanie, teda budú zahrnuté v smerovacej tabuľke (*routing table*).

Možnosť použiť adaptívne smerovanie v sieťach je významnejšie, ako spotreba časti šírky pásma na réžiu, alebo náročnosti na procesor. Riešenie v podobe adaptívneho smerovania je oveľa flexibilnejšie.

5.2 Klasifikácia smerovacích algoritmov a protokolov

Smerovací protokol je protokol, ktorý špecifikuje spôsob komunikácie smerovačov medzi sebou. Smerovací algoritmus je súbor inštrukcií prináležiaci výpočtu najlepšej cesty smerovacieho protokolu. Samotné smerovacie protokoly môžeme rozdeliť podľa niekoľkých aspektov.

5.2.1 Aspekt rozsahu použitia

Aspekt tkvie v rozsahu použitia daného protokolu. Protokoly, ktoré sa využívajú na smerovanie prevádzky medzi viacerými autonómnymi systémami patria do skupiny ERP (*Exterior Routing Protocol*). Dnešné protokoly, ktoré sa používajú na smerovanie v rámci viacerých autonómnych systémov sú EGP (*Exterior Gateway Protocol*), BGP (*Border Gateway Protocol*) a CSPF (*Constrained Shortest Path First*).

Protokol EGP patrí do skupiny DV (*Distance Vector*) protokolov a je to predchodca protokolu BGP. Na svoje použitie vyžaduje stromovú topológiu. Dnes sa už nepoužíva.

Pre sieť Internet bol postupne okolo roku 1994 nahradený protokolom BGP. Súčasná verzia 4 je špecifikovaná v [RFC4271] z roku 2006, kde bolo pridané najmä CIDR (*Classless Inter-Domain Routing*) a sumarizácia, ktorá je dôležitá kvôli zníženiu veľkosti smerovacích tabuliek na smerovačoch. Podrobnejšie sa protokolu BGP budeme venovať v samostatnej časti.

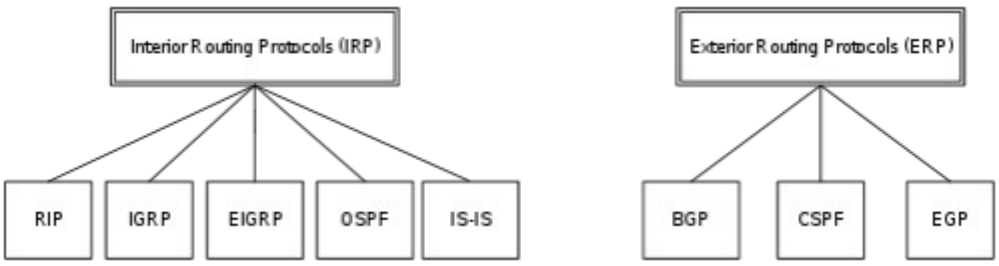
Protokol CSPF je rozšírený v MPLS (*Multiprotocol Label Switching*) sieťach. Je súčasťou CBR (*Constraint Based Routing*). Najlepšia cesta je počítaná rovnakým algoritmom ako v OSPF a IS-IS.

Iná situácia nastáva pri potrebe použitia protokolu navrhnutého na smerovanie v rámci jedného autonómneho systému. Tieto protokoly patria do skupiny IRP (*Interior Routing Protocol*). Hlavný aspekt delenia tejto skupiny protokolov tkvie vo výpočte optimálnej cesty.

Niekedy si želáme rýchlu konfiguráciu siete s jednoduchou konfiguráciou zariadení a ich údržby aj za cenu menšej efektivity. V prípade sietí, ktoré podliehajú veľkej záťaži (vzhľadom na ich kapacitu) nám zvyčajne záleží na efektivite prenášaných údajov, teda optimálnosti výsledkov výpočtov optimálnych ciest. Túto skupinu smerovacích protokolov tvoria RIP verzie 1 (*Routing Information Protocol*), RIP verzie 2, IGRP (*Interior Gateway Routing Protocol*), EIGRP (*Enhanced*

Interior Gateway Routing Protocol), OSPF (*Open Shortest Path First*), IS-IS (*Intermediate System to Intermediate System*) a Integrated IS-IS. Tieto protokoly sú detailnejšie popísané v nasledujúcich častiach.

Obr. 5.1 ukazuje rozdelenie hlavných protokolov do skupín podľa príslušnosti v skupine IRP alebo ERP.

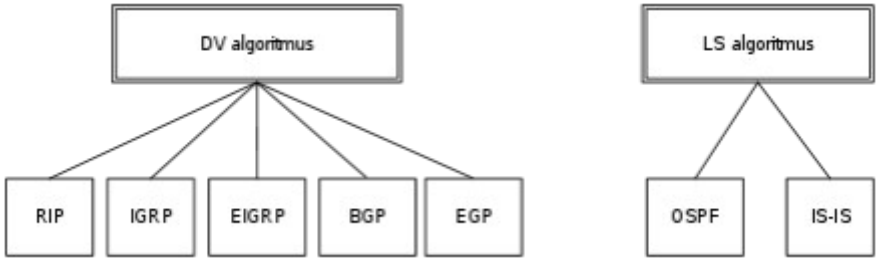


Obr. 5.1: Klasifikácia protokolov na základe rozsahu použitia

5.2.2 Aspekt smerovacieho algoritmu

Tu sa stretávame s dvoma možnosťami a s prípadnými modifikáciami. Prvú skupinu tvoria protokoly založené na algoritme *Distance-Vector* (ďalej DV). Druhú skupinu tvoria protokoly založené na algoritme *Link-State* (ďalej LS), ktorý poskytuje efektívnejšie možnosti správy výmeny smerovacích informácií za cenu relatívne vyšších nárokov oproti DV algoritmu. Tieto algoritmy budú popisované dôkladnejšie ďalej v tejto práci.

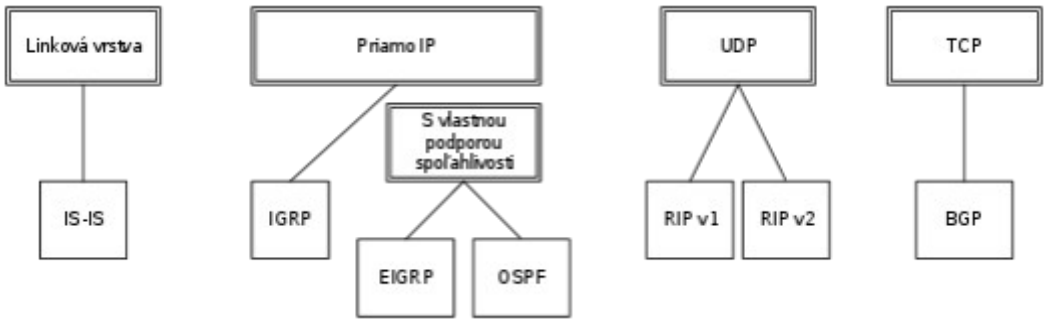
Obr. 5.2 rozdeľuje protokoly na základe ich príslušnosti k danému smerovaciemu algoritmu.



Obr. 5.2: Rozdelenie smerovacích protokolov podľa smerovacieho algoritmu

5.2.3 Aspekt transportného mechanizmu

Aspekt delenia smerovacích protokolov je založený na základe transportného mechanizmu. Od typu transportného mechanizmu záleží, akú spoľahlivosť pri prenose paketov sieťou dosiahneme. Z Obr. 5.3 je vidieť, že protokol IS-IS používa len linkovú vrstvu, IGRP používa priamo IP protokol, ktorý sám o sebe nie je spoľahlivý bez patričnej nadstavby (napr. TCP). OSPF a EIGRP si sami zaisťujú spoľahlivosť nad protokolom IP, ktorý používajú [10]. RIP verzie 1 aj 2 používa IP s nadstavbou UDP. Pozoruhodné je riešenie protokolu BGP, ktorý používa priamo TCP [RFC793] nad protokolom IP. Obr. 5.3 rozdeľuje protokoly na základe ich transportného mechanizmu.



Obr. 5.3: Rozdelenie smerovacích protokolov na základe ich transportného mechanizmu

5.3 Protokoly s DV algoritmom

Protokoly typu DV používajú ako základ Bellman-Fordov algoritmus. Tento algoritmus sa používa na nájdenie najkratšej cesty medzi dvoma uzlami siete. Algoritmus funguje aj pre negatívne hrany a vytvára kompletnú smerovaciu tabuľku.

Smerovače smerujúce cez DV protokoly si nevytvárajú komplexnú topológiu siete, nevytvárajú si nad ňou nadhľad. Protokolom DV sa nebudeme bližšie venovať, keďže nepodporujú nosnú technológiu tejto práce a to MPLS TE. Bližšie si popíšeme len externý smerovací protokol BGP.

5.3.1 Protokol BGP

Protokol BGP (*Border Gateway Protocol*) bol navrhnutý pre smerovanie medzi viacerými autonómnymi systémami (*autonomous systems*). Napriek svojej robustnosti a veľkej prispôsobivosti používa algoritmus DV. Na prenos informácií sa používa protokol TCP. Aktualizácie sú prírastkové (*incremental*) a odistené (*triggered*) bez priamej potreby ich opakovať vo vopred stanovenom intervale. Výber vstupov metriky pri BGP možno ladiť širokou paletou prostriedkov, ktoré budeme opisovať nižšie.

Z hľadiska rýchlosti konvergencie, je BGP bezpochyby najpomalší smerovací protokol v oblasti aktualizácie stavu topológie. Protokol BGP nebol navrhnutý na rýchlu konvergenciu, ale na stabilitu a správu veľkého množstva sietí s veľmi veľkým počtom ciest a smerovacích tabuliek.

V rozsiahlych sieťach WAN je bežné, že prakticky každú sekundu môže nastať zmena v sieti, teda zmena topológie, ktorú je treba oznamovať všetkým BGP uzlom v sieti. Pokiaľ by BGP konvergoval tak rýchlo ako protokoly primárne požívané v sieťach LAN, nastalo by značné zahltenie celej siete WAN oznamovacími paketmi, čo by výrazne spomalilo celú sieť.

BGP rozpoznáva dva typy smerovania. Smerovanie v rovnakom autonómnom systéme sa označuje ako IBGP (*Internal Border Gateway Protocol*) a smerovanie medzi viacerými autonómnymi systémami sa nazýva EBGP (*External Border Gateway Protocol*). Toto rozčlenenie spôsobuje problémy so spätnou synchronizáciou pre zistenie optimálnych ciest, ktoré sa zapíšu do smerovacej tabuľky.

Smerovanie v BGP má svoje pevné pravidlá. Jedno z týchto pravidiel hovorí, že autonómny systém v ktorom BGP smeruje, nebude nikdy nútiť iný autonómny systém smerovať určeným

spôsobom. Teda môžeme určiť, akým spôsobom sa bude smerovať prevádzka v našom autonómnom systéme, ale nebudeme určovať dianie za jeho hranicou, aj keď to je možné.

Atribúty BGP

Keby sme do správania BGP nezasahovali v oblasti výberu najlepšej cesty, správal by sa podobne, ako protokol RIPv1. Spočítal by, koľko smerovačov je medzi ním a cieľovou sieťou a podľa toho by si vyberal cesty.

Na zmenu tohto implicitného správania BGP sa používajú atribúty (*attributes*).

Atribúty delíme z viacerých prístupov. Prvý prístup je delenie z hľadiska podpory výrobcami na dobre známe (*well-known*) a voliteľné (*optional*). Ďalej ich môžeme deliť podľa toho, či majú byť zahrnuté v aktualizáciách na povinné (*mandatory*) a popisné (*discretionary*). Niektoré aktualizácie môžu byť prenášané od smerovača na smerovač, podľa ktorých atribúty delíme na tranzitívne (*transitive*) a na netranzitívne (*non-transitive*). Pri tranzitívnom režime, aj pokiaľ aktualizáciu nerozpozná, posiela ju ďalej s predpokladom, že bola určená pre iný BGP smerovač, ktorý ju rozpozná. Netranzitívne aktualizácie sa neposielajú ďalej, ani v prípade, pokiaľ takú aktualizáciu neviem rozpoznať.

Výber najlepšej cesty

BGP si vyberá optimálnu cestu pomocou algoritmu [11], ktorý prechádza zoznam kritérií zoradených podľa vopred definovanej dôležitosti, napríklad ďalší skok (*next-hop*), počet autonómnych systémov v ceste, lokálna preferencia (*Local Preference*), atď.

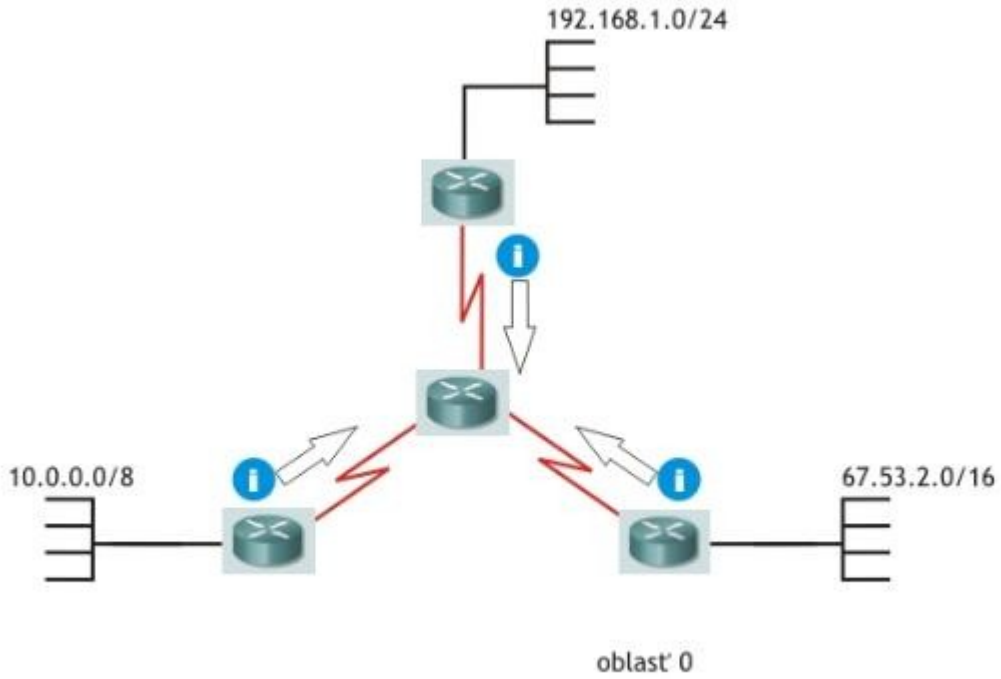
5.4 Protokoly s LS algoritmom

Základom LS protokolov je algoritmus analogický alebo príbuzný k protokolu SPF (*Shortest Path First*), ktorého autorom je Dijkstra. Základ tohto algoritmu spočíva v hľadaní najkratších ciest medzi hranami v orientovanom grafe. Špecifikácia vyžaduje, aby hrany mali len pozitívne hodnoty. Algoritmus nie je stavaný na vytvorenie celého stromu, ale na nájdenie najkratšej cesty medzi dvoma uzlami grafu, ktorý reprezentuje našu sieť.

Tento algoritmus je spravidla náročnejší na výpočet ako používajú protokoly s DV algoritmom. Tento prístup tiež vyžaduje, aby LS smerovače mali kompletný prehľad o topológii, v ktorej smerujú. Na základe toho sú schopné vnímať topológiu „z nadhľadu“ a výber najlepších ciest z topologickej tabuľky prispôsobiť tak, ako by to bolo najvhodnejšie vzhľadom na polohu konkrétneho smerovača, ktorý najlepšie cesty práve kalkuluje.

Týmto štýlom spracovania sa líšia oproti DV protokolom, ktoré sa riadia technikou smerovania pomocou zvestí (*Routing by rummor*), ktoré nie sú schopné vidieť celú topológiu zo svojho pohľadu. Dalo by sa zjednodušene povedať, že DV protokoly majú len tie informácie, ktoré im povedia ich najbližší susedia.

Aj tá najmenšia zmena topológie si vyžiada prepočet všetkých najlepších ciest v rámci celej topológie. Je to snaha, aby smerovanie bolo optimálne v rámci celej topológie. Avšak v prípade, že uvažujeme rozsiahlu topológiu s rôznymi zmenami stavov liniek, mohlo by dôjsť k nechcenému preťaženiu CPU smerovačov. Preto sa celá topológia môže rozdeliť na menšie celky, tzv. oblasti (*areas*), v ktorých sa vykonáva smerovací proces, ako to znázorňuje Obr. 5.4.



Obr. 5.4: Znalosť celej oblasti pri LS protokoloch

Existuje jedna stratégia návrhu topológie siete, vďaka ktorej sa môžeme úspešne vyhnúť zložitého výpočtu SPF algoritmu. Hovorí, že oblasť (*area*) by nemala byť navrhovaná ako príliš rozsiahla, pretože SPF algoritmus sa kalkuluje vždy pre príslušnú oblasť. Nenastanú intenzívne zmeny v sieti, ktoré sú náročné pre SPF prepočet.

Odistené aktualizácie sú založené na základe zmeny siete, ktoré až pri zmene nejakej časti topológie sietí túto informáciu rozdistribuujú. Vo svete LS protokolov napriek odisteným aktualizáciám existuje akýsi mechanizmus kontroly týchto aktualizácií, tzv. Link-State obnova (*Link-State refresh*). Táto obnova distribuuje po sieti topologickú databázu vo veľmi dlhých intervaloch (štandardne 15 alebo 30 min.), ktoré nemajú efekt na zmenu využiteľnej šírky pásma liniek. Tento mechanizmus zaisťuje aj pri krátkodobom rozšírení nepresných informácií následnú správnu konvergenciu v rámci oblasti.

V Link-State architektúre je dôležité pamätať si svojich susedov, celú topológiu a najlepšie cesty, preto je založená na existencií troch tabuliek:

- **Tabuľka susedov** (*Neighbor table*): udržiava si všetkých susedov, ktorí majú nadviazanú reláciu so smerovačom
- **Topologická tabuľka** (*Topology table*): je smerovacia mapa obsahujúca každú jednu sieť v danej oblasti
- **Smerovacia tabuľka** (*Routing table*): obsahuje výber najlepších ciest z topologickej tabuľky

5.4.1 Protokol OSPF

OSPF (*Open Shortest Path First*) je protokol založený na LS algoritme. Ide o otvorený štandard, ktorý bol vytvorený IETF skupinou.

Oblasť

Aby sa však zamedzilo enormnému posielaniu LS paketov, smerovače sa zaraďujú do oblastí (*area*). Jeden smerovač môže patriť do viacerých oblastí, ale linka patrí práve do jednej oblasti. Oblasť je logické zoskupenie smerovačov a liniek. Oblasť sa správa čiastočne autonómne a mnohé správy sú vymieňané len v rámci jednej oblasti. Znižuje sa tak počet SPF (*Shortest Path First*) kalkulácií vo fáze výpočtu. Detaily zmien v topológii sa prejavia len v rámci oblasti a zastavia sa na hranici oblasti (*area boundary*). Rozšírenie informácie je limitované. Menší počet smerovačov taktiež zmenšuje počet údajov v smerovacej a topologickej tabuľke. Všetky smerovače v tej istej oblasti majú rovnakú topologickú tabuľku (*topology table*) a musia sa pripojiť do oblasti 0 (*area 0*), spravidla priamo. Sú ale možnosti, ako možno pripojiť oblasť aj nepriamo, cez inú oblasť, pomocou tunelovania cez túto oblasť.

Smerovače, ktoré sprostredkovávajú smerovanie medzi jednotlivými oblasťami sa nazývajú ABR (*Area Border Routers*) smerovače [12]. ABR smerovače sú smerovače, ktoré majú rozhrania vo viacerých oblastiach.

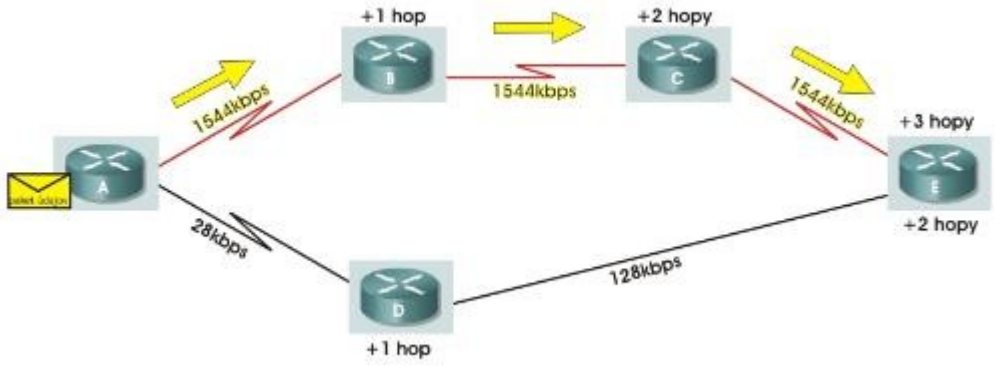
Interné OSPF smerovače sú také, ktorých rozhrania operujú len v rámci jednej oblasti, chrbticové (*backbone*) OSPF smerovače sú také, ktorých minimálne jedno rozhranie je pripojené do oblasti 0 (*area 0*).

Smerovače, ktoré sú vnímané z pohľadu topológie siete ako smerovače brány do iného autonómneho systému, sa v OSPF sieťach nazývajú ASBR *(Autonomous System Boundary Routers*).

Iba smerovače typu ABR a ASBR sú schopné sumarizovať cesty. Smerovače ABR sú výstupným uzlom zo svojej oblasti a smerovače ASBR sú výstupným uzlom pre celý autonómny systém, resp. doménu. Práve tieto smerovače by sa mali podieľať na sumarizácií.

Metrika

Výsledok výberu najlepšej cesty je v OSPF označovaný ako cena (*cost*) cesty. Ako vstupný parameter do tejto metriky bola zvolená šírka pásma. Tento parameter patrí do skupiny statických parametrov, je pevne zapísaný v konfigurácii smerovača. Obr. 5.5 ukazuje, že výber najlepšej cesty sa uskutoční na základe informácie o šírke pásma linky.



Obr. 5.5: Výber najlepšej cesty pomocou OSPF

Vzorec na výpočet ceny OSPF je zapísaný v 5.1.

$$cena = \frac{10^8}{šírka\ pásma} \quad (5.1)$$

Cena je výsledok nepriamej úrovne, teda čím menšia šírka pásma, tým väčšia cena. Metrika sa nepohybuje v desatinných číslach, z toho vyplýva, že maximálna šírka pásma, ktorá je rozoznateľná týmto vzorcom je 100 Mbps. Za touto hranicou majú všetky linky hodnotu 1. Tento nedostatok sa dá riešiť tým, že OSPF umožňuje pevne zapísať do konfigurácie novú hodnotu násobku, ktorý sa použije na prepočet ceny (zmena konštanty 10^8).

Vytvorenie relácie

OSPF relácie sa formujú iba v rámci rovnakej oblasti, nemožno formovať reláciu medzi rôznymi oblasťami systému. Najprv si smerovač určí vlastné identifikačné číslo, pod ktorým bude v sieti vystupovať. Spravidla ide o najvyššiu IP adresu pri štarte OSPF procesu na zariadení, pričom logické rozhrania sú vždy uprednostňované nad fyzickými rozhraniami. Identifikačné číslo môže byť aj pevne zapísané, čo je preferovaná voľba.

Následne sa začnú posielať *hello* správy (*Hello message*) na rozhraniach zahrnutých do OSPF procesu. Na to, aby bolo možné pokračovať, v prijatej *hello* správe je potrebné zaistiť, aby boli následovné atribúty rovnaké: *hello* a *router dead* časovače, typ siete, priame fyzické spojenie, číslo oblasti, typ hesla a heslo pre autentifikáciu, ak je zadané. Ďalej je to napríklad príznak listovej oblasti.

Smerovače si začnú vymieňať obsah svojich link-state databáz pomocou DBD (*DataBase Description packet*) paketov. Po tejto výmene sú smerovače synchronizované a spustí sa SPF algoritmus, ktorý vygeneruje príslušné cesty do smerovacej tabuľky.

Typy sietí v OSPF

OSPF rozoznáva niekoľko typov režimov. Podľa [RFC2328] sú to *point-to-multipoint* a *nonbroadcast* (NBMA). Cisco prišlo s ďalšími režimami: *point-to-multipoint nonbroadcast*, *broadcast* a *point-to-point*.

Smerovače si vytvárajú susedstvo s priamo spojenými smerovačmi. Na zdieľaných médiach, režimy *broadcast* a *nonbroadcast*, by sa pri väčšom počte muselo vytvoriť až príliš veľa susedstiev, čo by zaťažovalo zariadenia. Pri počte 10 smerovačov, by bolo vytvorených na každom smerovači 9 susedstiev a spolu by ich bolo 45 na médiu. Preto sa zaviedla voľba povereného smerovača (*Designated Router - DR*) a záložného DR (*Backup Designated Router* – BDR). Len DR a BDR smerovače požadujú od všetkých smerovačov vytvorenie susedstva. Ostatné smerovače (DROTHER) sa spoja len s DR a BDR. Väčšina typov OSPF paketov je posielaných na skupinových adresách. IP adresa 224.0.0.6 pre pakety smerujúce na DR a BDR a 224.0.0.5 pre všetky smerovače. V prípade výpadku DR rolu preberá BDR smerovač a začne sa voľba BDR. Voľba je nepreemptívna a ovplyvnená nastavenou prioritou. Smerovače s najvyššou prioritou preberajú rolu DR a BDR. Mali by to byť najviac výkonné smerovače na médiu.

LSA správy

OSPF rozlišuje niekoľko LSA (*Link-State Advertisement*) typov správ. Podľa toho, ktoré správy bude smerovač posielať a generovať sa odvíja od typu smerovača. Smerovače vo vnútri oblasti sa nazývajú interné a generujú typy 1 a 2. Sú to záplavové správy o všetkých cestách, ktoré pozná smerovač. Rozšírenie je v rámci oblasti po hranicu oblasti. Hraničné smerovače navyše generujú správy typu 3 – interná sumarizácia. Typ 4 je generovaný hraničnými smerovačmi, len keď je v sieti prítomný smerovač hraničiaci s iným autonómnym systémom a posiela externú sumarizáciu. ASBR generuje správu typu 1 a ABR ju rozposielajú do siete ako typ 4. Typ 5 reprezentuje externé cesty do iného autonómneho systému, generované ASBR a posielané každým ABR. Z ďalších typov je zaujímavý ešte typ 7, využívaný pri technike NSSA, pozri ďalej.

Typy oblastí

Dôvod, prečo je potrebné deliť OSPF sieť do oblastí je efektivita. Hneď ako toto rozdelenie urobíme, smerovače v rámci jednej oblasti budú spravovať menšiu topologickú tabuľku.

Existuje niekoľko typov oblastí, ktoré sú predurčené k špecifickým podmienkam fungovania v závislosti od topológie siete.

Chrbticová oblasť (*backbone area*), alebo *area* 0, je tranzitná oblasť, do ktorej sa pripájajú ostatné, regulárne oblasti. Ak chceme, aby si dve oblasti medzi sebou vymieňali informácie, je potrebné ich prepojiť pomocou tejto chrbticovej oblasti.

Listová oblasť (*stub area*) je oblasť, do ktorej neprúdia aktualizácie s externými cestami okrem všeobecnej brány. Cieľom týchto oblastí je filtrovať smerovacie informácie vnútri oblastí, pričom filtrovanie je vykonávané blokovaním LSA typu 5 na vstupu do oblasti. Jednotlivé smerovače dostávajú zhrnuté informácie LSA typu 3. Smerovač ABR tejto listovej oblasti má stále všetky potrebné informácie, ale do vnútra listovej oblasti sa distribuuje iba všeobecná brána. Tieto listové oblasti nemôžu byť použité na tunelovanie pre nepriame dosiahnutie chrbticovej oblasti inou oblasťou.

Celkom listová oblasť *(totally stubby area*), v ktorej sú zakázané externé cesty blokovaním všetkých ciest z ostatných oblastí LSA typu 5, sú blokované aj sumarizované cesty LSA typom 3 a 4. Cieľom tohto typu oblasti je docieliť, aby všetky smerovače poznali len naučenú všeobecnú cestu (*default route*), čo je výhodné pre úsporu šírky pásma a náročnosti smerovania na procesor smerovača. Túto požiadavku zaistí smerovač ABR vhodným filtrovaním LSA paketov. Jediná cesta ako dostať pakety z tejto oblasti je pomocou všeobecnej cesty. Opäť platí, že smerovač ABR si

ponecháva všetky informácie a LSA blokuje pre smerovače vnútri takejto oblasti. Ide o proprietárne rozšírenie.

Nie tak listová oblasť (*not so stubby area*) je oblasť, ktorá predpokladá napojenie do ešte ďalšej smerovacej oblasti [RFC3101]. Informácie z tejto ďalšej oblasti je treba oznámiť ďalším OSPF smerovačom aj mimo tejto listovej oblasti. Preto bol vymyslený LSA typ 7, ktorý pošle interný smerovač tejto listovej oblasti svojmu ABR smerovaču, ktorý tento typ LSA prijme a pošle ďalej do oblasti 0 ako LSA typ 5, ktorý býva zvyčajne označený ako externá cesta smerovaný cez ASBR. Výsledkom je, že informácie o smerovaní nadobudnuté v rámci celkom listovej oblasti sa šíria ďalej do siete. Stále pritom platí, že smerovače v takejto nie tak listovej oblasti majú odblokované LSA správy 3, 4 a 5.

Všeobecne platí, že oblasti označené ako listové a celkom listové by mali mať iba jeden ústiaci bod vo svojej oblasti. Je možné mať viacero ústiacich bodov, ale treba si uvedomiť, že potom máme viacero ABR smerovačov v týchto oblastiach, ktoré spôsobia potrebu existencie viacerých všeobecných ciest z týchto oblastí, medzi ktorými je treba vykonať vyrovnávanie záťaže. Toto vyrovnávanie záťaže je principiálne veľmi nešťastné riešenie, pretože smerovače vnútri týchto oblastí nemajú predstavu, čo sa nachádza za týmito oblasťami.

Externé cesty

V OSPF rozoznávame dva typy externých ciest, ktoré slúžia na prenos ciest z rôznych autonómnych systémov.

Cesty označované ako typ E1 (*External type 1*) sú cesty, ktorých cena (*cost*) sa zvyšuje tým, ako putujú cez autonómny systém. Je to výhodné pre systémy, ktoré majú viacero ústiacich bodov z autonómneho systému. Touto technikou je možné rozlíšiť, ktorá cesta je lepšia na toto použitie.

Cesty označované ako typ E2 (*External type 2*) sú cesty, ktorých cena (*cost*) zostáva prechodom cez autonómny systém. Je to implicitné riešenie nastavené v konfigurácii a je výhodné pre autonómne systémy s jedným výstupným bodom.

5.4.2 Protokol IS-IS

IS-IS (*Intermediate System to Intermediate System*) bol pôvodne navrhnutý ako smerovací protokol pre OSI (*Open Systems Interconnection*), čiže na svojom počiatku, keď bol vyvíjaný armádou, nebol vôbec uspôsobený na chod v TCP/IP sieťach, ale pre chod v OSI. Preto bolo nutné protokol prepracovať, jeho jadro zovšeobecniť a integrovať do neho TCP/IP model. Takéto rozsiahle prepracovanie si žiadalo nové značenie, preto sa pred IS-IS operujúci v TCP/IP sieťach pridalo slovo Integrated, teda Integrated IS-IS. IS-IS je vhodný pre veľmi rozsiahle siete, keďže v jednej oblasti môže byť až vyše tisíc smerovačov. Preto sa stal obľúbeným protokolom pre poskytovateľov služby.

IS-IS je vo všeobecnosti považovaný za robustnejší s väčšou mierou individuálnej konfigurácie čŕt (*features*). V porovnaní s protokolom OSPF, má viac možností prispôsobenia. Avšak niektoré jeho hodnoty nie sú implicitne nastavené na optimálne, ale po ich nastavení sa protokol stáva vo všeobecnosti viac efektívny ako OSPF.

Prechod na IPv6 pri tomto protokole sa odhaduje ako nenáročný. Je to spôsobené tým, že keď sa uskutočnilo prepracovanie IS-IS na prechod do TCP/IP sietí, stalo sa jadro systému natoľko univerzálne, že prechod z IPv4 na IPv6 nebude náročný. Je to rozdiel od ostatných protokolov operujúcich v TCP/IP.

Napriek tomu, že Integrated IS-IS operuje v TCP/IP, stále pracuje s OSI adresami. OSI adresy vychádzajú zo služieb CLNS (*Connectionless Network Services*) a sú alternatívnou k IP adresám.

Smerovacie domény

Formálne, protokol IS-IS rozoznáva štyri základné druhy spojení, v ktorých nadväzuje relácie:

- **Úroveň 0** (Level 0): spojenie medzi koncovým systémom ES-IS (*End System to Intermediate System*)
- **Úroveň 1** (Level 1): spojenie medzi smerovačmi v rámci oblasti
- **Úroveň 2** (Level 2): spojenie medzi smerovačmi medzi oblasťami
- **Úroveň 3** (Level 3): spojenie medzi autonómnymi systémami

V ďalšom uvažujeme len úrovne 1 a 2, keďže ide o základné úrovne používané pri smerovaní.

Úrovne smerovania

IS-IS je LS protokol, ktorý používa SPF (*Shortest Path First*) algoritmus pre výber najkratšej cesty. Na zisťovanie prítomnosti svojich susedov používa *hello* správy.

IS-IS preferuje rozdeliť autonómny systém do menších skupín – oblastí. Každá z týchto oblastí má vlastnú topológiu. Existuje preto niekoľko úrovní, v ktorých smerovače pôsobia:

- **Úroveň 1** (Level 1): Smerovače si spravujú topológiu iba vo svojej vlastnej oblasti
- **Úroveň 2** (Level 2): Smerovače pôsobiace v chrbtici siete (*backbone*), vedia iba o cestách, ktoré sú pridané do chrbtice siete
- **Úroveň 1/2** (Level 1/2): Spravujú databázy oboch úrovní

Databázy úrovne 1 a úrovne 2 sú kompletne separované. Smerovače úrovne 1 nadväzujú relácie vo svojej oblasti a medzi oblasťami sa nadväzujú vzťahy úrovne 2. Možno povedať, že smerovače úrovne 2 sú niečo ako preddefinovaná brána pre smerovače typu úrovne 1.

Metrika

Podľa RFC 1142 IS-IS rozoznáva štyri podtypy výpočtu najlepšej cesty [RFC1142]. Každý IS (*Intermediate System*) má povolené počítať metriku minimálne jedným podtypom. Pritom každý IS by mal byť schopný počítať metriku cez predvolenú metriku. Avšak rôzne IS môžu počítať najlepšiu cestu rôzne.

- **Predvolená metrika** (*default metric*): každej linke by mala byť priradená celistvá hodnota predstavujúca ľubovoľnú veličinu, napríklad šírku pásma
- **Metrika oneskorenia** (*delay metric*): je to voliteľná časť metriky predstavujúca oneskorenie medzi linkami
- **Metrika nákladov** (*expense metric*): pracujúca na základe vrstvy monetárnych cien. Je to voliteľná časť výpočtu najlepšej cesty. Podľa RFC konvencie je hodnota 1 používaná na indikáciu voľného spojenia. Vyššia hodnota indikuje väčšiu monetárnu cenu.
- **Chybová metrika** (*error metrics*): pri každej linke sú detegované chyby. Je to nepovinná časť výpočtu najlepšej cesty.

Proces kombinácií týchto metrík je cez jednoduché sčítanie hodnôt jednotlivých podtypov. Preto je dôležité tieto hodnoty voliť primerane vzhľadom k ostatným zadaným hodnotám.

Cisco smerovače implementujú len predvolenú metriku. Na informáciu o výsledku výpočtu metriky je vyhradený 1 bajt. Ôsmy bit je rezervovaný. Siedmy bit rozlišuje medzi internou a externou klasifikáciou. Interná metrika je metrika v rámci IS-IS domény, externá metrika je mimo IS-IS domény alebo naučená od iného sieťového protokolu. Na hodnotu výpočtu metriky ostáva 6 bitov. Preto na hodnotu IS-IS metriky zostáva iba interval <0; 63>. Celková suma jednotlivých metrík na linkách nemôže presiahnuť hodnotu 1023. Takáto metrika sa nazýva úzka (*narrow*). Rozšírená metrika, nazývaná ako široká, je 24 bitová pre linku a 32 bitová pre súčet cien na linkách a teda podporuje lepšiu granularitu pri určovaní výslednej ceny.

Pri implementácii metriky v TCP/IP sieťach na smerovačoch firmy Cisco výpočet metriky neobsahuje informáciu o šírke pásma. Metrika obsahuje iba implicitnú hodnotu priradenú každej linke o hodnote 10, čo spôsobuje, že pri nezmenení týchto hodnôt sa tento protokol správa ako protokol RIP, teda vzdialenosti porovnáva počtom smerovačov od štartu k cieľovej sieti.

Vytváranie susedstiev

IS-IS pri vytváraní susedstiev taktiež využíva *hello* pakety. Na zdieľanom médiu existuje DIS (*Designated Intermediate System*), čo je to isté ako DR pri OSPF. Záložný DIS sa nevolí a voľba je preemptívna. Navyše susedstvá si vytvorí vždy každý s každým. *Hello* pakety sú posielané na DIS smerovače každých 3,3 sekundy, ostatným každých 10 sekúnd. Na nie zdieľanom médiu je to tiež každých 10 sekúnd. Interval r*outer dead* je nastavený na trojnásobok *hello* času. Celá databáza sa odosiela raz za 15 minút na kontrolu konzistentnosti.

Stavba smerovacej tabuľky

V mnohých implementáciách smerovače hľadajú najlepšiu cestu pre IS-IS založenú na cene (*cost*) získanej pomocou SPF algoritmu. Túto činnosť vykonávajú smerovače úrovne 1 aj úrovne 2 kompletne samostatne. Potom sú všetky najlepšie cesty vsunuté do smerovacej tabuľky OSI. Z tejto OSI smerovacej tabuľky sa za pomoci PRC (*Partial Route Calculation*) algoritmu vyberajú cesty pre smerovaciu tabuľku IP. Všetky najlepšie cesty získané PRC algoritmom sa finálne zahrnú do IP smerovacej tabuľky.

6 Konvergencia na sieťovej vrstve

Optimalizácia na sieťovej vrstve je rozsiahla a to predovšetkým u smerovacích protokolov, ktoré sú priamo stavané na dolaďovanie. Smerovacie protokoly obsahujú mnohé časové parametre, ktorých implicitné hodnoty sú už pre dnešnú dobu príliš konzervatívne a preto je všeobecne odporúčané si ich zmeniť. Popíšeme podstatné časové a nečasové parametre, ktoré je odporúčané zmeniť a pozrieme sa bližšie aj na ich negatívne dôsledky. Zameriame sa na LS protokoly, keďže ako jediné sú schopné kooperovať s MPLS TE. Popíšeme aj zásady dobrého návrhu siete na logickej úrovni, čo ešte viac zlepší časy konvergencie. Treba si uvedomiť, že po optimalizácii budeme mať rýchlejšie konvergujúcu sieť, čo zvýši dostupnosť uzlov. Vyššia dostupnosť znamená vyššiu kvalitu služieb a vyššiu spokojnosť zákazníka.

Táto kapitola je priamo prebraná a upravená z [7].

Dosiahnutie konvergencie na sieťovej vrstve je komplexná záležitosť. Na sieťovej vrstve zabezpečujeme smerovanie pomocou smerovacích protokolov. Konvergencia sa stáva dominantnou témou a reflektuje schopnosť siete, ako rýchlo sa dokáže prispôsobiť zmenám v sieti.

Konvergencia je proces, kedy sa všetky uzly dohadujú na optimálnych cestách pre smerovanie paketov a vykonajú následnú aktualizáciu ich smerovacích tabuliek. Ak nastane akákoľvek zmena na akomkoľvek smerovači, alebo aj inom zariadení, sieť už nie je v skonvergovanom stave a je nutná opätovná konvergencia. Sieť môže mať porušený tento stav z nasledujúcich príčin:

- pridanie, odobratie alebo výpadok linky
- pridanie, odobratie alebo výpadok zariadenia
- pridanie, odobratie alebo výpadok segmentu siete
- zmena protokolu alebo niektorých jeho parametrov
- administratívny zásah, vypnutie linky alebo zariadenia z dôvodu údržby, ľudský faktor

Na sieťovej vrstve a pri smerovaní sa stretávame s viacerými variantmi pojmu konvergencia. Len pripomenieme, že sieť považujeme za skonvergovanú, keď všetky uzly dosiahli konsenzus pri nových hodnotách a všetky príslušné údaje v smerovacích tabuľkách sú aktualizované [13].

Konvergencia protokolu (*protocol convergence*) – vyjadruje proces, v ktorom si uzol aktualizuje svoju smerovaciu tabuľku a príslušné zmeny oznámi svojim susedom. Od momentu, kedy sa proces ukončil, je protokol skonvergovaný.

Konvergencia cesty (*route convergence*) – vyjadruje proces, v ktorom si uzol aktualizuje svoju smerovaciu tabuľku a dáta sú presmerovávané na novú cestu. Od momentu, kedy sa proces ukončil, je cesta skonvergovaná.

Konvergencia siete (*network convergence*) – vyjadruje proces, v ktorom si všetky uzly aktualizujú svoje smerovacie tabuľky. Od momentu, kedy sa proces ukončil, je sieť skonvergovaná.

Konvergencia siete je buď [7]:

- suma konvergencií všetkých protokolov (KP): $\sum_{i=1}^{n} KP_i$
- maximum z množiny konvergencií protokolov: $max\{KP_1, KP_2 \ldots KP_n\}$
- kombinácia predošlých dvoch prípadov

Ako vidieť z definícií, celkový pojem konvergencia, sa dá na sieťovej vrstve rozdeliť na menšie fragmenty. Čas medzi porušením skonvergovaného stavu a opätovným nadobudnutím tohto stavu označujeme ako čas konvergencie. Konvergencia nikdy nie je okamžitá. Otázka je, aký dlhý čas je potrebný na dosiahnutie tohto stavu [13].

6.1 Konvergencia v sieti

Počas konvergencie je pravdepodobné, že nastane stratovosť paketov na určitú dobu. V najhoršom prípade to je 100 %-ná stratovosť počas celej periódy. Výhodou rýchlej konvergencie v sieti je krátka perióda zvýšenej stratovosti paketov.

Konvergencia v sieti prebieha v niekoľkých krokoch [14]:

- Detekcia
- Rozšírenie informácie
- Výpočet
- Aktualizácia smerovacej tabuľky

Časový rozdiel medzi zmenou a detekciou tejto zmeny v sieti je fázou detekcie. Jej trvanie môže byť veľmi krátke, desiatky milisekúnd, pokiaľ hovoríme o reštarte protokolu, vypnutí linky a podobne. V najhoršom prípade sú to desiatky sekúnd až minúty, ak ide napríklad o výpadok zariadenia a protokol to zistí až po niekoľkonásobnom neprijatí *hello* paketov (všeobecne interval *router dead*). Práve tu je obrovský priestor na optimalizáciu. Rýchlosť detekcie ovplyvňuje aj typ interfejsu. Napríklad POS (*Packet Over SONET*) interfejs dokáže zistiť výpadok za niekoľko milisekúnd a nie je ani nutné optimalizovať interval *router dead*. Optimalizácia pozitívne vplýva na reakciu pri sledovaní objektov vo VRRP protokole, popísané nižšie. Rýchla detekcia znamená kratší čas nedostupnosti koncových zariadení a aj rýchlejšiu konvergenciu.

Rozšírenie informácie o zmene je závislé od väčšieho množstva parametrov. Najmarkantnejším parametrom je počet smerovačov, ktoré túto správu musia prijať. Správny výber topológie a predovšetkým hierarchický návrh siete so sumarizáciou smerovacích ciest na smerovačoch zabezpečí nutnosť rozšírenia správy len po určitú oblasť. Pokiaľ by sa nevyužila sumarizácia, správa by musela byť rozšírená možno až do celej siete. Pri niektorých protokoloch musia túto správu prijať všetky smerovače v rámci celej oblasti. Rozšírenie informácie ovplyvňuje rýchlosť liniek a zahltenie. Predchádzaniu zahltenia sa podrobnejšie venujeme v kapitole 7 o QoS. Ďalším parametrom je použitý protokol a jeho optimalizácia. Celkový čas rozšírenia informácie zjednodušene môžeme vyjadriť vzťahom:

```
(priemerné oneskorenie na každom uzle) * (diameter siete) (6.1)
```

Rýchlosť vo fáze výpočtu závisí od konkrétneho protokolu a implementovaného algoritmu a od aktuálnej záťaže na smerovači. Problém nastáva, ak má byť vykonaných viac výpočtov v krátkej dobe. Napríklad protokol OSPF obmedzuje implicitne výpočet na max. jedenkrát za 5 sekúnd, pričom aj prvý výpočet je pozdržaný. Parameter sa dá optimalizovať. Takéto obmedzenia v protokoloch mali v minulosti zabrániť zbytočnej záťaži pre smerovače pri viacnásobných prepočtoch, avšak v dnešnej dobe je odporúčané takéto hodnoty znížiť alebo anulovať.

Posledným krokom k dosiahnutiu skonvergovaného stavu je aktualizácia smerovacej tabuľky, prípadne všetkých smerovacích tabuliek, ak ich smerovač využíva viac. Čas je závislý od rýchlosti a vyťaženosti zariadenia. Takisto od veľkosti smerovacej tabuľky a od počtu ciest, ktoré je nutné aktualizovať. Veľkosť smerovacej tabuľky vieme ovplyvniť využívaním techník ako

sumarizácia, listová oblasť, alebo statické smerovanie. Snažíme sa udržiavať čo najmenej ciest v smerovacej tabuľke.

Konvergencia na sieťovej vrstve prebieha v niekoľkých krokoch, pričom v každej fáze je možná optimalizácia, aby sme dosiahli kratší čas celkovej konvergencie. Výsledný čas je však silne závislý od nižších vrstiev. Na sieťovej vrstve máme k dispozícií viac nástrojov na optimalizáciu. Výber vhodného smerovacieho protokolu, jeho časová optimalizácia, používanie rozšírených možností techník protokolu, sumarizácia IP adries a iné prostriedky na zmenšenie smerovacej tabuľky, QoS a mnohé iné pozitívne vplývajú na čas konvergencie [15].

6.2 Časové parametre smerovacích protokolov

V tejto kapitole sa bližšie pozrieme na časové parametre definované v smerovacích protokoloch, ktoré je možné a vhodné optimalizovať. Zmena časových parametrov nasleduje až po vyčerpaní všetkých ostatných techník definovaných smerovacím protokolom. Presný návod, aké hodnoty nastaviť neexistuje, ale sú všeobecné odporúčania. Najväčšie množstvo časových parametrov je práve v smerovacích protokoloch.

Implicitné časové hodnoty definované v protokoloch boli v dobe svojho vzniku udávané ako optimálne. Keďže protokoly „starnú“ a rýchlosti, či už prenosového pásma, alebo CPU, sa zvyšujú, môžeme jednotlivé časové parametre optimalizovať a nastaviť nižšie hodnoty. Dokonca niektoré parametre je možné anulovať, keďže svoj význam postupne strácajú. V minulosti veľké východzie hodnoty časov slúžili ako ochrana proti nadmernej záťaži zariadení. Vzhľadom na výkon dnešných zariadení sú tieto ochranné parametre takmer zbytočné. Chybovosť a stratovosť na linkách je omnoho menšia, čiže aj spoľahlivosť siete sa zvýšila. Vyššia spoľahlivosť znamená menej takzvaných falošných pozitívnych správ. Pri oscilácií linky (*flapping link*) [16], kedy linka rýchlo mení svoj stav z aktívneho na neaktívny, by mohla byť vyvolaná opätovná konvergencia. Avšak oscilácia linky väčšinou znamená hardvérovú poruchu, ale môže to byť aj problém s nastavením taktovacieho kmitočtu (*clock rate*), prípadne duplicitná IP adresa na linke. Takéto oscilácie môžu byť najväčším strašiakom po optimalizácii, keďže ich detekcia a následné spracovanie bude veľmi rýchle. Obranou je sumarizácia IP adries a tlmenie interfejsu (*dampening*).

Zmena časových parametrov je asi najjednoduchšou technikou optimalizácie a aj veľmi účinnou. Každý časový parameter môže byť menený. Rozdielne je už len rozpätie. Skracovanie časov v dnešnej dobe umožnili niekoľkonásobne rýchlejšie linky a zariadenia. Skracovanie časov síce umožní rýchlejšiu detekciu, rýchlejšie rozšírenie správy a výpočet, ale zvýši záťaž na linke a smerovači.

Smerovacie protokoly používajú viacero parametrov, ktoré sú nastaviteľné. Tieto parametre chránia samotný protokol a zariadenie pred zahltením v časoch nestability siete. Dá sa povedať, že sú istou prevenciou. Väčšina parametrov je vo svojej východzej konfigurácii nastavená značne konzervatívne. Je to preto, lebo v minulosti mala prednosť stabilná sieť pred rýchlou konvergenciou. Pri vzniku protokolov boli iné podmienky, aké sú dnes. Zariadenia a linky sú rýchlejšie, média v sieťach LAN operujú väčšinou v plne duplexnom režime a média sú väčšinou nesúperiace. Chybovosť pri prenose je zanedbateľná a celkovo stabilita siete sa zvýšila. Navyše novšie implementácie majú priamo v nastavovaní časových parametrov zabudovanú obranu proti nestabilite vo forme tlmenia. To všetko umožňuje zmenu časových parametrov na omnoho agresívnejšie hodnoty. Naším cieľom bude dosiahnutie času konvergencie pokiaľ možno pod jednu až dve sekundy na sieťovej vrstve v smerovacích protokoloch. Podmienkou je možnosť v IOS daného zariadenia nastaviť takúto hodnotu. Pokiaľ zariadenie nemá možnosť nastavenia času pod

jednu sekundu, je nevyhnutné nainštalovať novšiu verziu. Je možné, že všetky parametre, ktoré popíšeme sa nenachádzajú v danej verzii IOS_u, alebo nie sú prístupné pre danú platformu. V takom prípade sa daný parameter nebude optimalizovať. Syntax, alebo dokonca názov parametra sa môže líšiť medzi jednotlivými verziami a platformami.

Názvy techník budú vychádzať z IOS c3660-jk9o3s-mz.124-12.bin. Platforma je určená pre stredne veľké siete, verzia je 12.4 a ponúka mnohé rozšírenia a nadštandardné možnosti. Vybrali sme si tento IOS práve preto, aby sme na jednej verzii demonštrovali všetky možnosti optimalizácie. Budeme popisovať iba LS protokoly OSPF a IS-IS, keďže iba tie vedia pracovať s MPLS TE, čo je popísané nižšie.

6.2.1 Protokol OSPF

V prvom rade sa pozrieme na fázu výpočtu. Vo fáze výpočtu sa využíva SPF algoritmus. Problémom je, že fáza výpočtu sa začne až po 5 sekundách od prijatia požiadavky na prepočet! Takéto zamedzenie malo predchádzať viacerým prepočtom v krátkej dobe v prípade nestability na sieti. Neskoršie implementácie však umožňujú dynamické prispôsobovanie čakania na prepočet. Konfiguruje sa pozdržanie prvého prepočtu, pozdržanie medzi prvým a druhým prepočtom a maximálny čas pozdržania. Pozdržanie medzi druhým a tretím prepočtom a každým ďalším prepočtom v krátkej dobe, je pozdržaný o dvojnásobok predošlej hodnoty, až po maximálny čas pozdržania. Parameter je veľmi výhodné optimalizovať.

Ďalším vylepšením je ISPF (*Incremental SPF*). Po aktivácií tejto možnosti algoritmus neprepočítava celý strom, ale len zmeny v strome. Ak by bol nutný prepočet celého stromu, použije sa klasické SPF. Časové parametre pre SPF sa vzťahujú aj na ISPF. Aktivácia zabezpečí rýchlejší prepočet a rýchlejšiu konvergenciu, pričom nemá žiaden negatívny vplyv. Pokiaľ je táto možnosť k dispozícií, rozhodne by mala byť aktivovaná [17].

Na podobnom princípe pracuje aj škrtenie LSA (*LSA throttling*). Podstatou je obmedziť posielanie LSA správ susedom. Prvá správa sa vždy posiela okamžite. Na totožnom princípe ako v predošlom prípade funguje princíp pozdržania odosielania LSA správ.

Príchod LSA správ (*LSA arrival*) zakazuje prijatie tej istej LSA správy na nejaký čas. Tá istá LSA správa má rovnaké LSA ID, LSA typ a je prijatá z rovnakého smerovača.

Implicitná hodnota posielania *hello* paketov je 30 sekúnd na WAN a na ostatných typoch liniek 10 sekúnd. Tento parameter sa zdá byť nie až taký dôležitý, ale priamo s ním súvisí interval *router dead*. Ten je nastavený na štvornásobok *hello*. Znížením času *hello* sa skráti čas vo fáze detekcie. Rýchlejšia detekcia akou je jedna sekunda nie je možná, keďže interval *router dead* je minimálne jedna sekunda. Zmeniť sa dá len periodicita posielania *hello* paketov – násobič. Čím väčší násobič, tým viac *hello* paketov je poslaných za jednu sekundu. Paradoxne, čím menší násobič, tým je štatisticky rýchlejšia konvergencia, keďže prijatie *hello* paketu nastaví časovač na 1000 ms. Pri násobiči 20 by sa časovač pohyboval v rozpätí 950 – 1000 ms. Pri násobiči 4 je časovač v rozpätí 750 – 1000 ms. Zmena tohto parametra je najviac nebezpečná, keďže permanentne vyťažuje zariadenie a linku. Príliš agresívna hodnota môže spôsobiť v časoch nestability na sieti, alebo v čase vysokej záťaže, falošné správy, ktoré by zbytočne vyvolali opätovnú konvergenciu. Po zmene tohto parametra je nutné sledovať správanie sa zariadení na sieti, či nové hodnoty akceptujú. V súbehu s ostatnými technikami, ktoré popisujeme, by sa výraznejšie problémy nemali objaviť.

Po nastavení hodnôt pod jednu sekundu je odporúčané nastaviť ešte tlmenie na danom interfejsi. Tlmenie v časoch nestability na danom interfejsi exponenciálne pozdržuje čas zmeny

stavu linky. Každá zmena stavu linky vyvoláva opätovnú konvergenciu. Tlmenie je teda ochranou interfejsu a aj ochranou siete. Jej zapnutie na interfejsi vysoko odporúčame na všetkých interfejsoch, pretože obráni sieť pred zbytočnou konvergenciou a teda udržuje úroveň stability s pred optimalizácie.

OSPF ponúka kategóriu základných parametrov (*pacing*) na optimalizáciu. Všetky tieto parametre by mali byť ponechané na implicitných hodnotách a radšej využiť techniky ako sumarizácia, listová oblasť a podobne. Preto sa týmito parametrami nezaoberáme. Navyše neexistuje návod, ako tieto hodnoty zmeniť, keďže sú priamo závislé od návrhu siete.

Pri OSPF odporúčame jednoznačne zmeniť parametre pri SPF a zapnúť ISPF. Výsledkom sú lepšie časy konvergencie vo fáze výpočtu. Zmena *hello* a intervalu *router dead* zlepší časy detekcie. Aby sa neznížila stabilita interfejsu, zariadenia a siete, treba nutne zapnúť tlmenie. Škrtenie LSA a príchod LSA správ urýchlia časy rozšírenia správ.

6.2.2 Protokol IS-IS

IS-IS bol vo svojom vzniku navrhnutý tak, aby ho bolo možné optimalizovať a prispôsobovať.

Tak ako OSPF aj IS-IS využíva SPF algoritmus, ktorý je však pozdržaný až 5,5 sekúnd. Princíp je zhodný s OSPF.

IS-IS taktiež umožňuje aktivovať ISPF. Narozdiel od OSPF je možné pozdržať aktiváciu tejto techniky od jej nakonfigurovania [18].

IS-IS dokáže pomocou použitia techniky PRC prepočítať zmenu v strome bez použitia SPF. Táto možnosť je asi 10x rýchlejšia ako klasické SPF a využíva sa vtedy, keď mechanizmus smerovania sa nezmenil, ale smerovač predsa len nejakú zmenu zaznamenal. Napríklad zmenu prefixu.

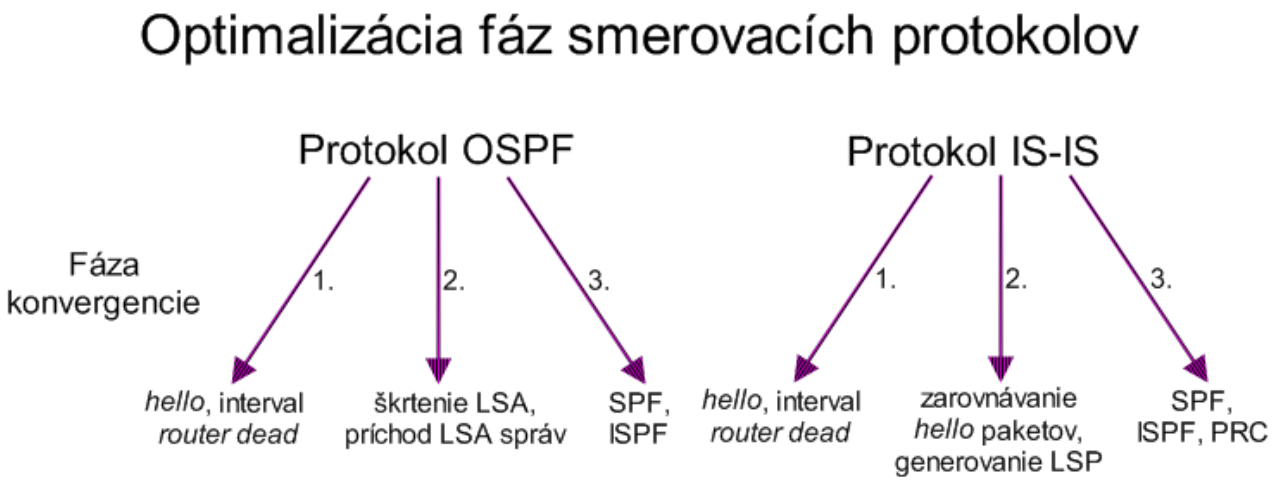
Protokol vie pozdržať generovanie LSP (*Link State Packet*) správ podobne ako pri OSPF.

Hello pakety sa odosielajú každých 10 sekúnd a na DIS každých 3,3 sekundy. Opäť platí všetko to, čo bolo popísané pri OSPF. Rozdielna je syntax a taktiež možnosť posielať *hello* správy až každú milisekundu. Interval *router dead* je trojnásobok *hello* [19].

Možnosti zmeny SPF, PRC a zapnutie ISPF zlepšia fázu výpočtu v konvergencii. Zmena *hello* a *dead* intervalu pod jednu sekundu zabezpečí lepšie časy detekcie, no je nutné zapnúť na interfejsoch tlmenie, aby sme stabilizovali sieť v prípade problémov. Generovanie LSP správ ovplyvňuje fázu rozšírenia správy do siete.

Protokol má v niektorých implementáciách implicitne zapnuté zarovnávanie *hello* paketov na dĺžku MTU (*Maximum Transmission Unit*) - *hello padding*. Túto možnosť je vhodné vypnúť, keďže nie je potrebná a zbytočne konzumuje príliš veľkú časť šírky pásma [20].

Na Obr. 6.1 sme zhrnuli všetky optimalizačné nástroje oboch smerovacích protokolov a priradili sme ich ku konkrétnej fáze konvergencie. Na základe tejto schémy budeme v návrhu optimalizovať jednotlivé parametre a študovať posun v konvergencii.



Obr. 6.1: Optimalizácia fáz smerovacích protokolov

6.2.3 Protokol RSVP

Protokol RSVP (*Resource Reservation Protocol*) slúži na signalizáciu požiadaviek medzi uzlami v sieti. Požiadavky sú smerované cestami, ktoré boli vybrané smerovacími protokolmi, preto je vhodné začať s jeho optimalizáciou až po skončení optimalizácie smerovacích protokolov. Tento protokol môže byť taktiež optimalizovaný. Protokol RSVP sa používa v modeli *IntServ* v kvalite služieb a v MPLS TE, čo je popísané v nasledujúcich kapitolách.

Optimalizované môže byť množstvo odosielaných správ, ich veľkosť a počet vo fronte. Takisto sa môže kontrolovať signalizácia a navyše sa môže optimalizovať úvodné pozdržanie signalizácie v prípade opakovaného vysielania, čakanie počas ktorého sa akumulujú potvrdenia (občerstvenie) a iné. Pomocou aktivovanej predikcie kompresie je možné redukovať réžiu UDP alebo RTP transportného protokolu.

Takisto je možné optimalizovať *hello* protokol a *keapalive* správy v protokole RSVP. Ich podporu sme v našom sledovanom IOS_e však nenašli. Je to preto, lebo náš IOS nepodporuje funkcie rýchleho presmerovania. Interval vypršania sa určuje ako počet neprijatých *hello* paketov. Ďalšie možnosti optimalizácie sa buď nevzťahujú na IP protokol, alebo nie sú momentálne pre nás podstatné [21].

6.2.4 Protokol LDP

Protokol LDP (*Label Distribution Protocol*) signalizuje správy a stará sa o výmenu návesti v prostredí MPLS. Bližšie sa čitateľ o tomto protokole dozvie v kapitole o MPLS.

Protokol LDP umožňuje optimalizovať svoj *hello* mechanizmus a časovač *holddown*. Ide o podobný mechanizmus ako v smerovacích protokoloch. Ďalej je možné upraviť časovač pre udržovanie relácie po výpadku. Veľmi dôležitá funkcia je IGP (*Interior Gateway Protocol*) synchronizácia, ktorá je však podporovaná len v LS smerovacích protokoloch. Táto funkcia aktivuje synchronizáciu medzi IGP (označované tiež ako IRP) a LDP a predíde tak strate paketov, ktoré by mohli byť smerované na základe IGP protokolu, ale nevedia byť prepnuté na základe návestia. Existujú prípady, kedy IGP má vytvorené susedstvo, ale LDP susedstvo sa zrušilo a komunikácia sa stráca. Takisto táto funkcia predchádza chybám v implemetáciach alebo v konfiguráciach [22].

Keďže funkcia pozdržuje konvergenciu IGP a eventuálne pozdrží aj konvergenciu LDP, tento príkaz sa zdá byť pre nás menej vhodný. Jeho vhodnosť prakticky verifikujeme.

Veľmi zaujímavou možnosťou je ochrana relácie v LDP protokole. Ak vytvorené susedstvo medzi dvomi smerovačmi by malo byť zrušené napríklad výpadkom linky, avšak susedia sú navzájom dosiahnuteľní alternatívnou cestou, ktorá je známa, tak sa relácia nezruší. Susedia sú dosiahnuteľní cez cielené odosielanie *hello* paketov a navzájom o svojej prítomnosti vedia. Po obnovení linky nie je nutné nanovo vytvárať susedstvo a vymieňať si stavové informácie. Tento proces totižto spomaľuje konvergenciu, preto túto funkciu odporúčame aktivovať. Zároveň šetríme prostriedky, keďže smerovače nemusia nanovo vytvárať návestia a oznamovať ich susedovi a eventuálne do siete [23].

V tejto časti sme popísali základné a podstatné časové parametre v protokoloch, ktoré budeme využívať pri implementácii, ich princíp a fungovanie. Parametrov na optimalizáciu je omnoho viac, no nie sú relevantné alebo podstatné. Táto kapitola je nosná pri optimalizácii siete s cieľom zrýchliť konvergenciu a zvýšiť dostupnosť uzlov.

6.3 Stabilita po časovej optimalizácii

Ako sme si všimli, aj samotné nastavenia protokolov prechádzajú postupnou evolúciou. V minulosti sa mnohé parametre konfigurovali len zmenou jednej hodnoty. Avšak dnes, ako napríklad SPF a LSA príkazy, majú už komplexnejšiu syntax. Zmena pozdržania SPF kalkulácii mala len jeden parameter, ktorý sa vzťahoval na všetky SPF kalkulácie. Jeho zmena na príliš nízke hodnoty v čase nestability siete by mohla mať výrazný negatívny vplyv na zariadenie a aj sieť ako takú. Avšak dnešné implementácie majú až tri parametre pre jeden príkaz. Všeobecne to je pozdržanie prvej kalkulácie, rozdiel medzi prvou a druhou kalkuláciou a maximálne pozdržanie. Pričom každá nová požiadavka na prepočet je pozdržaná o dvojnásobok času hodnoty predošlej kalkulácie počnúc časom medzi prvou a druhou kalkuláciou. Takéto správanie sa nápadne podobá na BGP tlmenie, odkiaľ mnohé smerovacie protokoly túto vlastnosť prebrali. Práve takáto úprava nám zabezpečí nastavenie nulovej hodnoty pri pozdržaní prvého prepočtu a pomerne nízku hodnotu medzi prvým a druhým prepočtom. Už sa nemusíme obávať, že by nestabilita mala devastujúci účinok, protokol sa vysporiada s nestabilitou sám.

Ďalším dôkazom evolúcie sú aj možnosti konfigurácie posielania *hello* paketov pod jednu sekundu. Keďže takéto generovanie a spracovanie paketov je veľmi rýchle, preto by určite mala byť zapnutá možnosť tlmenia na interfejsi. Opäť to isté správanie ako pri BGP tlmení. Nestabilita interfejsu v podobe oscilujúcej linky nám nezahltí sieť falošnými správami na opätovnú konvergenciu.

7 Kvalita služieb

V tejto kapitole budeme diskutovať o potrebe kvality služieb, spôsoby, algoritmy a technológie na jej dosiahnutie.

Ako aplikácie postupne umocňujú expanziu a vývoj komunikačných sietí, stávajú sa požiadavky na klasifikáciu a kategorizáciu aplikačných tokov podľa potreby danej služby stále viac viditeľnými. Aplikácie s rôznymi potrebami a požiadavkami na sieť vyžadujú istú mieru inteligencie v sieťovej technológii, ktorá by vedela zaručiť splnenie takýchto požiadaviek s konečnými kapacitnými zdrojmi siete. Sieť musí zvládať zaručiť predvídateľnú kvalitu služieb pre takzvané business-critical aplikácie, aby naplnila svoju existenčnú podstatu. Zároveň umožniť aplikáciu služieb s vysokou pridanou hodnotou ako napr. VoIP, ktoré vyžadujú garanciu časového oneskorenia pri prenose. Tieto požiadavky nás privádzajú k pojmu kvalita služieb alebo *Quality of Service* (QoS). Kvalita služieb pre komunikačné siete predstavuje súbor nástrojov, ktorých aplikovaním je možné dosiahnuť determinizmus správania sa siete a zabezpečenie požadovanej úrovne kvality služieb v prípade vyťaženia siete.

V rámci nasledujúcich častí tohto dokumentu budú predstavené základné architektúry prístupu ku kvalite služieb v TCP/IP sieťach (*Best Effort*, *Integrated Services* - *IntServ* a *Differentiated Services* - *DiffServ*) a niektoré najdôležitejšie nástroje na ich implementáciu.

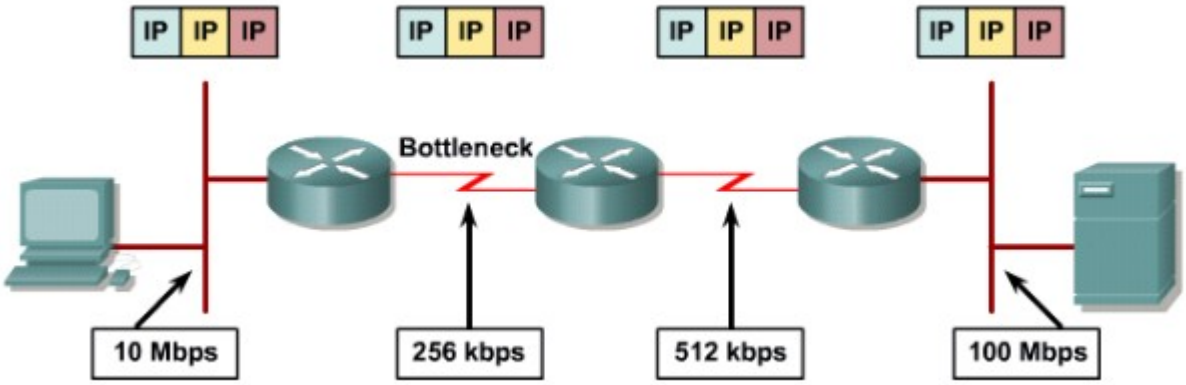
7.1 Potreba kvality služieb

Siete typu TCP/IP pri svojej prevádzke sprostredkovávajú prenosovú kapacitu pre služby, ktoré sa v sieti vyskytujú. Avšak pri prenose dátových jednotiek, rámcov, môžu nastať komplikácie vyplývajúce z fundamentálneho princípu fungovania počítačových sietí. Môže ísť o nasledovné problémy:

- Nedostatok transportnej kapacity
- Oneskorenie prenosu medzi koncovými bodmi prenosu
- Variabilné oneskorenie, takzvaný *Jitter*
- Stratu dátových jednotiek komunikácie

7.2 Nedostatok transportnej kapacity

Každá TCP/IP sieť má konečnú prenosovú kapacitu. Taktiež prenosová kapacita jednotlivých jej úsekov sa môže líšiť. Pre dátové toky prechádzajúce sieťou je však celková kapacita rovná najmenšej priepustnosti v rámci toku. Príklad možno vidieť na nasledovnom Obr. 7.1.

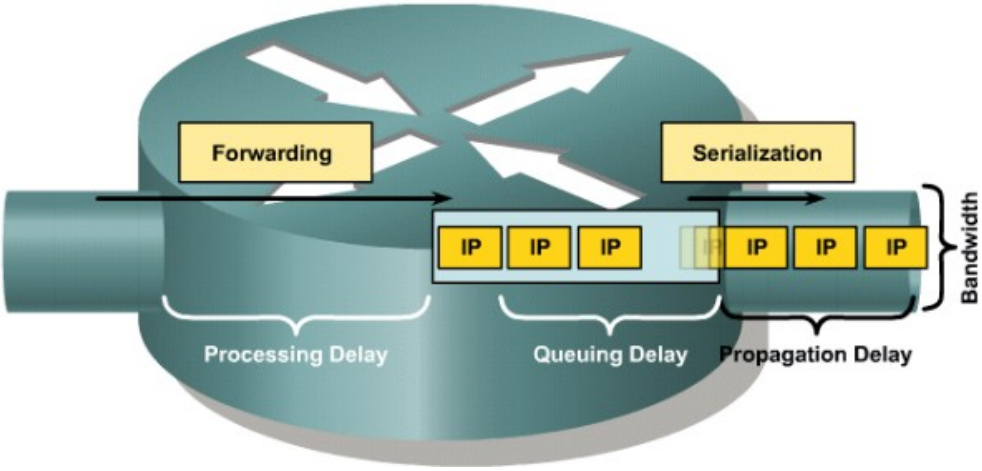


Obr. 7.1: Problém „úzkeho hrdla“ alebo Bottleneck [24]

Z obrázku je zrejmé, že dátové toky medzi stanicou a serverom budú musieť zdieľať prenosovú kapacitu najmenej výkonnej prenosovej kapacity, takzvaného „úzkeho hrdla“ siete alebo tiež *Bottleneck*. V prípade, ak sieť nerozpoznáva rôzne priority pre dátové toky, prenosová kapacita je jednoducho podiel prenosovej kapacity vydelený počtom tokov. Bez aplikácie nástrojov kvality služieb je jediným riešením zväčšenie kapacity siete, čo je však finančne náročné, ale v neposlednom rade tiež nemôže zaručiť kvalitu služieb v prípade neprirodzenej prevádzky v sieti.

7.3 Oneskorenie

Existujú služby v TCP/IP sieťach, pre ktoré je oneskorenie kľúčovým parametrom pri prevádzke. Znamená to, že pri prekročení istej hodnoty oneskorenia sa stáva ich fungovanie menej kvalitným, až neakceptovateľným. Opäť, príkladom môže byť VoIP, alebo aj videohovory alebo hry pre viacerých hráčov. Oneskorenie vzniká na viacerých miestach v prevádzke sieťových komponentov, Obr. 7.2.



Obr. 7.2: Rôzne miesta oneskorenia v sieťových komponentoch [24]

Základná kategorizácia zdrojov oneskorenia je teda nasledovná:

- **Oneskorenie spracovania** (*Processing Delay*) je oneskorenie spracovania dátového toku alebo rámca logikou sieťových zariadení
- **Oneskorenie fronty** (*Queuing Delay*) je oneskorenie, kedy rámec dátového toku je už spracovaný logikou sieťového zariadenia a bol zaradený do frontu, v ktorom čaká na príležitosť, aby bol vyslaný po dátovej linke
- **Oneskorenie serializácie** (*Serialization delay*) je oneskorenie prenosu logických dát rámca na samotné fyzické médium dátovej linky
- **Oneskorenie propagácie** (*Propagation delay*) je oneskorenie vznikajúce na základe času potrebného na prešírenie prenosového signálu na transportnom médiu

7.3.1 Variabilné oneskorenia

Oneskorenie v sieti sme si už rozobrali, na druhej strane je potrebné si uvedomiť, že aj keď sú služby schopné prispôsobiť sa istej miere konštantného oneskorenia, variácie v tomto oneskorení môžu predstavovať prílišnú prekážku v ich správnom fungovaní.

7.4 Straty v toku dát

Straty sú fundamentálne najhorším nepriateľom pre kvalitu tej danej služby. V prípade strát je možné, že aplikácie bude žiadať preposlanie stratených dát, čo znamená viacnásobne zaťaženie kapacity siete tými istými dátami. Taktiež aplikácie ako VoIP potrebujú svoje dáta v predpokladanom čase oneskorenia a pokiaľ ich v danom čase nedostanú, preposielanie stráca na význame a hlasová komunikácia trpí radikálne zhoršenou kvalitou.

7.5 Modely prístupu ku kvalite služieb

Kvalita služieb je rozsiahla tematika s pomerne početnou skupinou nástrojov na dosiahnutie vytýčených cieľov pre kvalitu služieb. Z pohľadu nasadzovania týchto nástrojov do siete je potrebné zvoliť adekvátny architektonický prístup. V súčastnosti sa abstraktné prístupy k architektúre nasadzovania kvality služieb delia na tri rôzne druhy, známe ako *Best Effort*, *Integrated Services* a *Differentiated Services*.

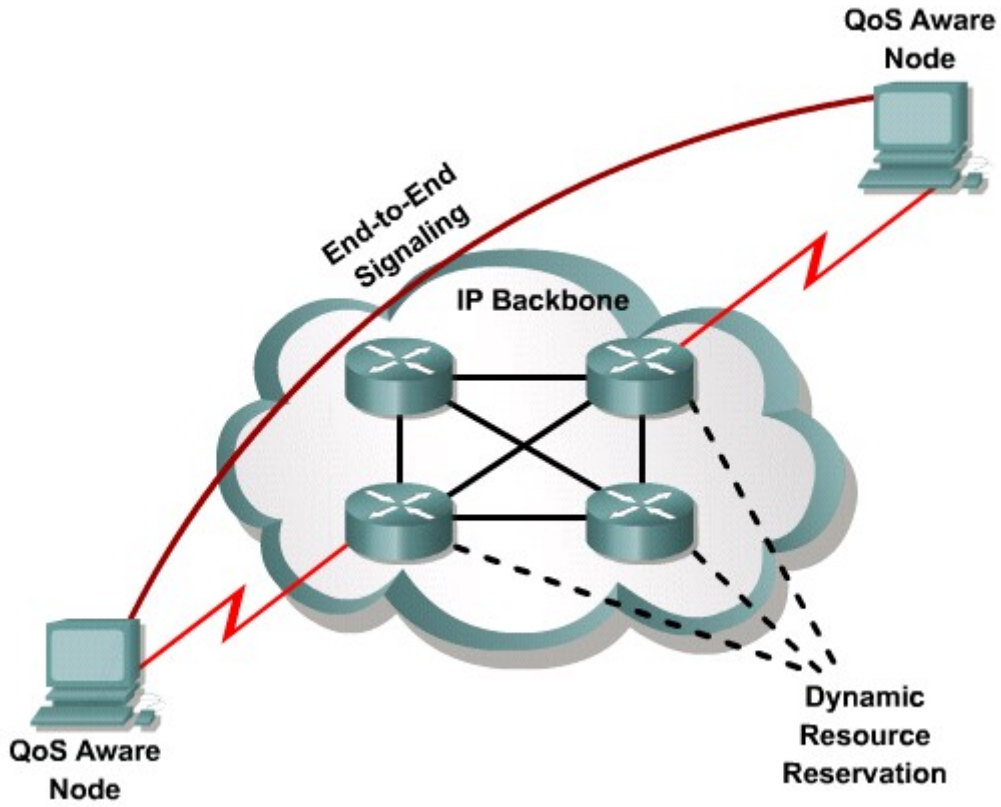
7.5.1 Model *Best Effort*

Model *Best Effort*, čo v preklade znamená „Najlepšia snaha“ predstavuje referenčnú predstavu o sieti, v ktorej sa nenachádzajú komplexné nástroje na riadenie kvality služieb. Znamená to, že v takejto sieti nie sú dátové toky jednotlivých aplikácií diferencované z pohľadu siete a všetky dosahujú rovnakú dostupnosť k sieťovým prostriedkom. Siete s týmto prístupom sú postačujúce pre služby špecifickej potreby na prenosové kapacity. Príkladom takýchto služieb sú napríklad klasické dátové toky ako sú toky aplikácii pre email, web a textovú komunikáciu. Spoločným znakom týchto služieb je, že v prípade nedostatkov prenosových kapacít v sieti, sú tieto služby schopné, aj keď sú limitované, istým spôsobom splniť svoju funkciu. Na druhej strane siete typu *Best Effort* nie sú vhodné pre služby citlivé na prenosové kapacity ako je napríklad VoIP. V prípade zahltenia siete,

napr. služba VoIP, vie deterministicky splniť svoju funkciu, nakoľko pokiaľ nie sú pre jej dátový tok splnené minimálne prenosové požiadavky, stráca svoj účel a jej používatelia nie sú schopní sa dorozumieť.

7.5.2 Model *IntServ*

Táto architektúra sa zakladá na možnosti dynamickej indikácie požiadaviek pre prenosové kapacity koncových staníc. Znamená to, že koncová stanica nahlási najbližšiemu sieťovému zariadeniu svoje požiadavky. Toto sieťové zariadenie požiadavku akceptuje (ak môže) a nahlási tieto požiadavky ďalšiemu sieťovému zariadeniu na ceste do cieľa toku dát. Každé zariadenie na ceste musí požiadavku akceptovať, aby bolo možné zabezpečiť požadovanú kvalitu. Až keď všetky zariadenia súhlasia s požiadavkami, vtedy sa pre daný tok vyčlení prenosová kapacita a zdrojová stanica môže začať komunikovať. Rezerváciu zdrojov zabezpečujú špecifické protokoly (ako RSVP) a daný protokol musí podporovať každé zariadenie v sieti vrátane koncových staníc. Na nasledujúcom obrázku je ukážka tohto princípu.

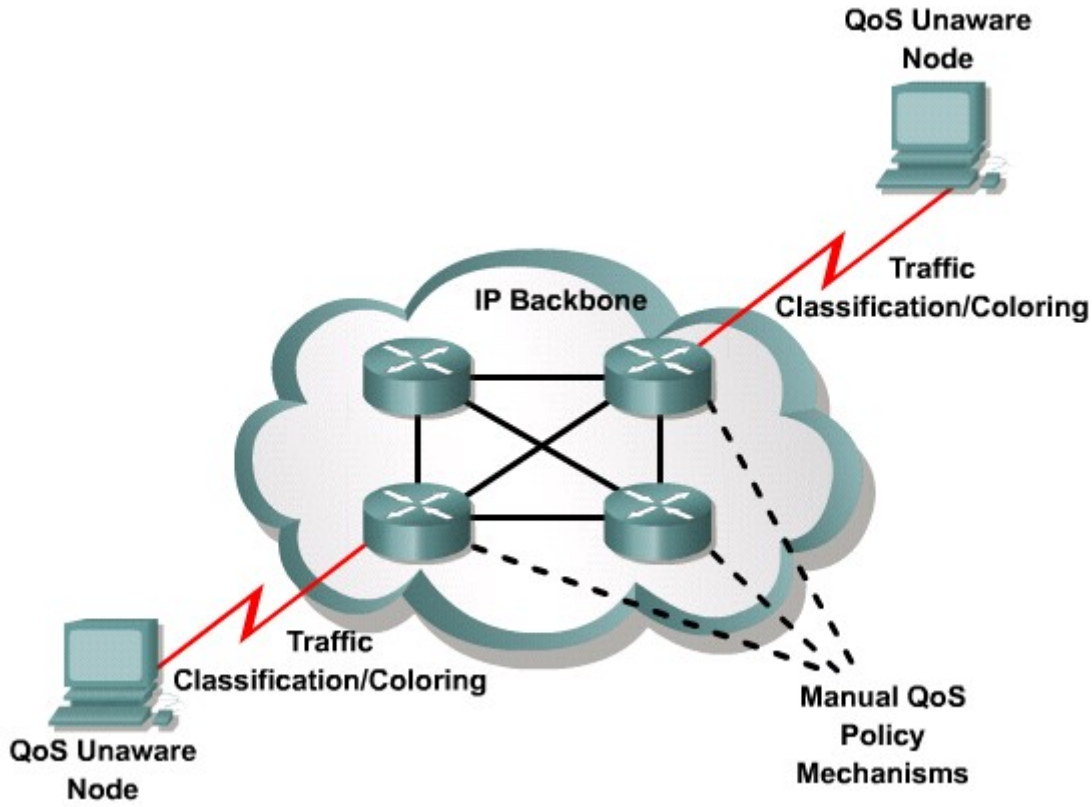


Obr. 7.3: Model architektúry IntServ [24]

7.5.3 Model *DiffServ*

Narozdiel od *Integrated services*, v tomto modeli nie sú požiadavky vôbec hlásené medzi zariadeniami. V *Differentiated Services* sú nástroje zabezpečenia kvality služieb integrované do každého sieťového zariadenia samostatne. Koncové stanice môžu byť do tohto procesu tiež zaradené, ale ide o sporadické riešenie. Výhodou tohto riešenia je jeho veľká škálovateľnosť,

flexibilita a kontrola nad kvalitou služieb. Tá je riadená iba sieťovými zariadeniami a nie koncovými stanicami, ktoré sú často pokladané za nedôveryhodné. Samotné dátové toky sú klasifikované do kategórií s priradenou kvalitou služieb podľa potreby danej služby v danej kategórii. Taktiež sa v tejto architektúre intenzívne používa značkovanie samotných dátových jednotiek (IPv4/IPv6 ToS, 802.1Q CoS, MPLS EXP), ktoré umožňuje, aby klasifikácia prebiehala podľa komplexných pravidiel iba na okrajových zariadeniach siete a jadro siete nestrácalo čas analýzou tokov. Na nasledujúcom obrázku je zobrazený princíp tejto architektúry.



Obr. 7.4: Model architektúry DiffServ [24]

7.6 Nástroje *Differentiated Services*

V rámci implementácie modelu *Differentiated Services* v tejto prípadovej štúdie je potrebné poznať nástroje, ktorých celková aplikácia vytvorí tento model. Tieto nástroje budú na nasledujúcich stránkach analyzované z pohľadu na umožnenie optimalizácie konvergencie a optimalizácie siete s podporou VoIP a iných multimediálnych dátových tokov v sieti.

7.6.1 Značenie paketov

Aby sa nemuselo na každom smerovači pre prichádzajúci paket zisťovať, do ktorého radu má byť zaradený, existuje technika značkovania paketov. Značkovanie prebehne len na jednom smerovači a ostatné smerovače v sieti budú akceptovať a veriť označeniu. Týmto ušetrime čas pri šírení informácie. V IP protokole na to slúži položka IP precedense, respektíve TOS (*Type Of*

Service). Ide o osembitové pole, ktoré označuje prioritu. V iných protokoloch existujú iné podobné polia s podobnou funkciou. Všeobecne platí, čím vyššia hodnota tohto poľa, tým je paket dôležitejší. Jednotlivé modely QoS využívajú toto pole rôzne a rôzne označujú pakety. Definovaných je len 8 základných tried. Trieda 7 a 6, teda dve najvyššie triedy, sú rezervované. Ostatné triedy je možné použiť na označovanie bežnej dátovej komunikácie. Trieda 7 je rezervovaná pre signalizáciu a trieda 6 pre smerovacie protokoly. Čiže pri výbere akéhokoľvek značenia, pakety smerovacích protokolov označujeme triedou 6.

7.6.2 Manažment radov

QoS je možné implementovať s rôznymi algoritmami na manažment radov. Významné algoritmy popíšeme z pohľadu vhodnosti pre smerovacie protokoly.

Najtriviálnejším algoritmom je algoritmus **FIFO** (*First In First Out*), ktorý prichádzajúce pakety v rovnakom poradí posiela do výstupných vyrovnávacích pamäti. Takýto prístup je nevhodný, nakoľko v skutočnosti žiaden manažment nepredstavuje.

PQ algoritmus (*Priority Queuing*) definuje štyri rady pre pakety. Každý z prichádzajúcich paketov je označený a zaradený do jedného z týchto radov. Až keď je rad s najvyššou prioritou vyprázdnený, obslúži sa nasledujúci rad. Algoritmus je vyhovujúci, pokiaľ sa v najviac prioritnom rade nachádzajú len pakety smerovacích protokolov. Celkovo však algoritmus môže spôsobiť starváciu pre ostatné dáta.

Nasledovníkom PQ algoritmu sa stal **CQ** algoritmus (*Custom Queuing*). Definuje 16 radov a každá trieda paketov má pridelené isté množstvo priestoru v rade. Rady sú obsluhované cyklicky, čím sa vytratila schopnosť striktnej priority a tak je tento algoritmus pre konvergenciu menej výhodný.

Algoritmus **WFQ** (*Weighted Fair Queuing*) zabezpečuje spravodlivý podiel šírky pásma pre všetky toky. Prioritnejšie toky dostavajú väčší podiel zo šírky pásma. Problém, ktorý vznikol pri CQ algoritme však naďalej pretrváva a ani tento algoritmus z pohľadu konvergencie nie je vhodný, nakoľko neumožňuje definovanie vlastných priorít pre rôzne triedy dátových tokov.

CBWFQ (*Class-Based WFQ*) algoritmus zaisťuje spravodlivosť a garantuje prenosové pásmo pre jednotlivé rady. Tento algoritmus umožňuje definovať triedy dátových tokov a prioritizovať medzi nimi, avšak ani tento algoritmus nevie pracovať s paketmi citlivými na oneskorenie a preto ani tento algoritmus neodporúčame v rámci tejto prípadovej štúdie.

Pridaná podpora uprednostňovania paketov do CBWFQ vytvorila algoritmus **LLQ** (*Low-Latency Queuing*), taktiež známy ako PQ-CBWFQ. Ide o kombináciu PQ algoritmu, ktorý zabezpečí uprednostnenie vybraných dátových tokov a CBWFQ algoritmus, vhodný pre ostatné dátové toky. Podobne ako pri PQ je nutné si dávať pozor, koľko a ktoré dátové toky majú byť striktne uprednostnené. Voľba tohto algoritmu je výhodná nielen z pohľadu konvergencie v sieti, ale aj pre ostatné dátové toky a v rámci tejto prípadovej štúdie bude tento algoritmus implementovaný v sieti pre podporu multimediálnych tokov pre ich potrebu minimalizovať oneskorenie.

7.6.3 Techniky predchádzania zahltenia

Na jednotlivých zariadeniach môže dochádzať k zahlteniu. O zahltení hovoríme, ak tok prijímaných dát je rýchlejší ako tok odosielaných dát. Dôsledkom je zapĺňanie vyrovnávacej pamäte a jej následné pretečenie, čo znamená stratovosť paketov. QoS obsahuje viacero techník ako včasne predchádzať zahlteniu na zariadení. Pre dáta, zabezpečujúce konvergenciu, je vhodné, aby neboli

vyhadzované a teda aby nedochádzalo k zahlteniu v ich rade. Tomu sa dá predísť rezerváciou dostatočnej šírky pásma pre daný rad. I napriek tomu je nutné uvážiť správny nástroj na predchádzanie zahltenia, keďže pri úplnom zahltení vstupnej vyrovnávacej pamäte môže dôjsť k pretečeniu a teda neprijímaniu paketov dôležitých pre rýchlu konvergenciu alebo vyvolanie falošných výpadkov v multimediálnom toku.

Vyhadzovanie paketov, až po úplnom zahltení, sa nazýva *tail drop*. Najlepšie nástroje sú algoritmy, ktoré predchádzajú zahlteniu, čiže ešte predtým, než sa vyrovnávacia pamäť úplné naplní, preventívne túto pamäť vyprázdňujú. Najlepšími algoritmami sú algoritmy včasnej náhodnej detekcie (*Random Early Detection* – RED). Algoritmus RED náhodne vyhadzuje pakety z radu od určitej nakonfigurovanej hodnoty zaplnenia vyrovnávacej pamäte. Tu je dôležité, aby pakety smerovacích protokolov neboli vyhadzované. Toto technika RED nedokáže zabezpečiť, respektíve iba v najnovšej implementácii, avšak rozlišuje len dva typy radov. Vylepšením je technika WRED (*Weighted RED*). Táto technika uvažuje už aj o precedencii a teda pri vyšších triedach môžeme znížiť agresivitu vyhadzovania. Výber modelu QoS nie je pre rýchlu konvergenciu pri smerovacích protokoloch podstatný, podstatná je garancia prenosového pásma a urýchlené preposlanie vybraných paketov. Pri použití DiffServ modelu sa treba sústrediť na správny algoritmus manažmentu radov – LLQ a aplikovať WRED na včasné predchádzanie zahltenia. Smerovacie protokoly musia byť správne označkované a smerovače musia s týmto označením správne narábať. QoS redukuje oneskorenie vo vyrovnávacej pamäti, ale pridáva oneskorenie pri spracovaní paketu v uzle, keďže je potrebné pakety analyzovať a značiť, čiže prepisovať. I napriek tomu je priam nevyhnutné aplikovať QoS.

8 Redundancia na sieťovej vrstve

Redundancia na tretej vrstve RM OSI zaisťuje dostupnosť IP adries aj po výpadku fyzického uzla siete. Je to spôsobené virtualizáciou IP adresy. Pokiaľ uvažujeme o virtualizácii IP adresy, v dôsledku enkapsulácie paketu sa virtualizuje aj MAC adresa.

Existuje viacero protokolov, ktoré túto redundanciu podporujú. Možno využiť komerčné riešenie od firmy Cisco nazvané HSRP (*Hot Standby Router Protocol*) alebo GLBP (*Gateway Load Balancing Protocol*). Alternatívne k tomu môžeme využiť riešenie VRRP (*Virtual Router Redundancy Protocol*). K výhodám použitia VRRP nepochybne patrí to, že je to otvorený štandard. Pomocou VRRP môžeme virtualizovať aj polia UNIXových serverov tak dobre, ako predvolenú bránu.

8.1 Protokol VRRP

Pôvodne bolo VRRP navrhnuté na eliminovanie chyby jedného bodu (*single point of failture*). Často sa používa pre potreby eliminácie výpadku predvolenej brány. VRRP špecifikuje protokol, ktorý dynamicky priradí zodpovednosť na jeden z VRRP smerovačov na jednom sieťovom segmente [RFC2338].

8.1.1 Terminológia v protokole VRRP

Virtuálny Smerovač (*Virtual Router*) je zariadenie vytvorené na základe operácie jedného alebo viacerých zariadení s VRRP implementáciou [25]. Je to virtuálne zariadenie, ktoré reprezentuje funkcionalitu celého klastra sieťových zariadení a je zodpovedné za sieťovú prevádzku, ktorá prináleží každému zariadeniu na serverovom klastri.

Jednoznačný identifikátor virtuálneho smerovača (*Virtual Router ID*) je jedinečne priradené číslo na rozlíšenie fyzických VRRP zariadení. VRID musí byť jedinečné pre každý sieťový segment.

Virutálna MAC adresa (*Virtual MAC address*) je automaticky priradená MAC adresa na VRRP rozhranie založená na klasickom MAC prefixe pre VRRP pakety a VRIP čísla. Prvých 5 oktetov je 00:00:5E:00:01 a posledný oktet je konfigurovaný ako VRID [26].

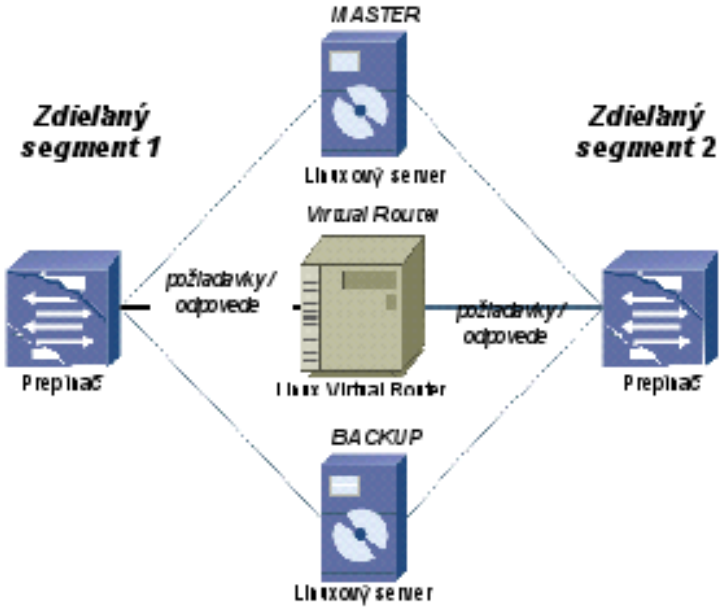
IP adresa virtuálneho smerovača (*Virtual Router IP*) tzv. VIP je IP adresa priradená k virtuálnemu smerovaču.

Master rola je zariadenie, ktoré práve reprezentuje virtuálny smerovač a vykonáva funkcionalitu virtuálneho smerovača [25]. Tiež prijíma a vysiela VRRP pakety na zistenie štatútov zariadení nachádzajúcich sa v role *Backup*.

Backup rola je zariadenie, ktoré čaká na vykonávanie funkcionality virtuálneho smerovača v prípade, že zariadenie v role *Master* zlyhá [25]. Zariadenie posiela a prijíma VRRP pakety na zistenie, či je zariadenie v role *Mater* aktívne.

Priorita je číslo z intervalu 1 až 254, ktoré môže pomôcť k voľbe zariadenia v role *Master*. Vyššie číslo priority udáva preferenciu pri výbere *Master* zariadenia.

8.1.2 Príklad použitia



Obr. 8.1: Modelový prípad pre redundanciu

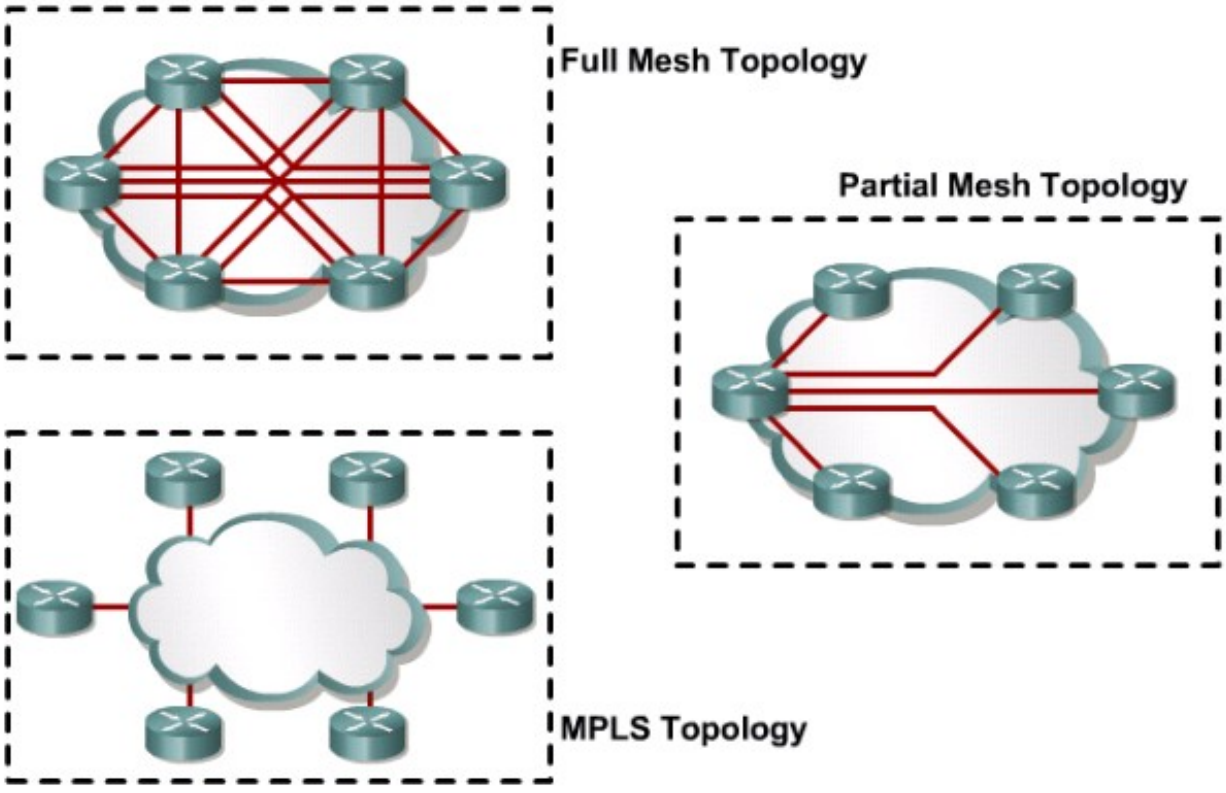
Obrázok popisuje hore uvedený princíp fungovania VRRP. Pre každý zdieľaný segment topológie sa zvolí zariadenie v *Master* role reprezentujúce virtuálny smerovač. V prípade výpadku zariadenia v role *Master* sa použije záložné zariadenie v role *Backup*. Tento výpadok je zdetegovaný neprijatím viacerých *hello* paketov na zdieľanom segmente topológie. V prípade obnovenia zariadenia v role *Master*, toto zariadenie znovu prevezme prevádzku, lebo má väčšiu prioritu.

9 Technológia MPLS

Multiprotocol Label Switching (MPLS) je technológia prepínania na základe návestí. Spája inteligenciu smerovania na tretej vrstve RM OSI modelu s rýchlosťou prepínania na druhej vrstve RM OSI modelu. Je to teda ako keby prepínanie na tretej vrstve. Tento prístup umožňuje skombinovať viacero výhod v rámci architektúry sietí poskytovateľov sieťovej konektivity. Technológia MPLS podporuje dve implementácie a to režim prepínania rámcov (*Frame Mode MPLS*) a režim prepínania buniek (*Cell Mode MPLS*). Nakoľko režim prepínania buniek je špecifické pre ATM siete, o ktorých v tomto dokumente neuvažujeme, zameriame sa len na režim prepínania rámcov. Z pohľadu zadania tímového projektu je technológia MPLS dôležitá z toho dôvodu, že poskytuje možnosti QoS medzi koncovými bodmi komunikácie. Taktiež v tomto dokumente budú spomenuté základy implementácií MPLS sietí na podpore škálovateľných VPN sietí [RFC3031].

9.1 Úvod do MPLS

Ako je známe, na fundamentálnu schopnosť smerovať dáta optimálne, je potrebné mať konektivitu z každého bodu siete do každého ďalšieho. Keď si však predstavíme momentálne existujúce spôsoby stavania sietí typu WAN, veľa implementácií toto nespĺňa. Tento fundamentálny prínos sa snaží poskytnúť práve MPLS technológia. Pohľad na tento rozdiel medzi úplnou *mesh* (*full-mesh*) topológiou, čiastočnou *mesh* (*partial-mesh*) topológiou a MPLS topológiou je znázornený na Obr. 9.1.



Obr. 9.1: MPLS maskujúci skutočnú fyzickú topológiu [27]

Ako na predošlom obrázku vidno, topológie, akou je čiastočný *mesh*, poskytujú neoptimálne smerovanie len medzi niektorými bodmi. Optimálne smerovanie poskytujú úplný *mesh* a MPLS topológie. Je potrebné si uvedomiť, že MPLS túto výhodu poskytuje tak, že zákazník si od svojho poskytovateľa nekupuje konektivitu medzi dvoma bodmi, ale kupuje si prístup k prístupovým bodom do spoločnej MPLS siete. To znamená, že zákazník sa nestará o samotné spojenia medzi svojimi bodmi, kedy mu čiastočná *mesh* topológia vychádzala lacnejšie na počet liniek. Ak si od poskytovateľa zákazník kúpi iba konektivitu pre svoje prístupové body a tiež mu poskytovateľ zabezpečí vymieňanie si informácií smerovacích protokolov medzi zákazníkovými hraničnými smerovačmi, zákazník získa z jeho pohľadu úplné *mesh* spojenie svojich bodov, avšak za cenu čiastočných *mesh* spojení. Takéto riešenie predstavuje obrovské výhody pre zákazníka pri smerovaní medzi svojimi vzdialenými sieťami a tiež pre poskytovateľa, nakoľko odpadá potreba manuálne definovať smerovanie cez prenosovú MPLS sieť.

9.2 MPLS návestie

Tento termín označuje MPLS 32-bitovú hlavičku v technológii MPLS. Táto hlavička sa pridáva za poslednú hlavičku druhej vrstvy RM OSI modelu a pred prvú hlavičku tretej vrstvy RM OSI modelu, Obr. 9.2.

Obr. 9.2: Hlavička MPLS návestia [27]

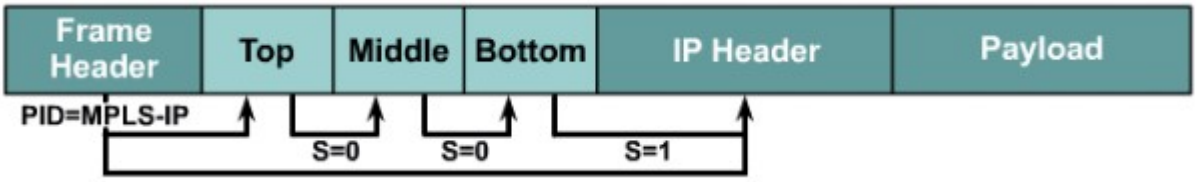
Ako z Obr. 9.2 vidno, hlavička MPLS sa skladá z viacero polí [29]. Ich funkcia je nasledovná:

- **Label** – je 20-bitová číselná hodnota používaná na smerovanie v MPLS sieti
- **EXP** – je trojbitové pole, ktoré zatiaľ nie je definované štandardom a je rezervované pre budúce rozšírenia. V rámci implementácie na zariadeniach spoločnosti Cisco Systems a iných, je toto pole využívané ako CoS (*Class of Service*) pole pre hodnoty označenia priority v rámci zabezpečovania kvality služieb
- **S** – je jednobitová informácia indikujúca posledné MPLS návestie v rámci zásobníka návestí. Bližšie vysvetlenie tohto dôležitého poľa bude poskytnuté v dokumente nižšie
- **TTL** – toto 8-bitové pole zabezpečuje pre MPLS sieť rovnakú funkcionalitu ako TTL pole v IP hlavičke, čiže určuje živosť. TTL je v MPLS návestí potrebné, nakoľko smerovanie sa uskutočňuje na základe MPLS návestia a IP hlavička (a jej TTL) je ignorovaná. Každý smerovač dekrementuje túto hodnotu o jedničku, pokiaľ nie je nastavené ignorovanie. Ak sa dosiahne hodnota menšia ako jeden, celý rámec je zahodený

9.3 Vnorené MPLS návestia

MPLS technológia umožňuje veľmi elegantný systém, ako pridať dodatočne význam pre návestie v MPLS hlavičke, nakoľko umožňuje do jedného rámca pridať viacero MPLS návestí.

Smerovače potom používajú len prvú MPLS hlavičku a ďalšie MPLS hlavičky môžu predstavovať inú ako smerovaciu informáciu. Príkladom môže byť informácia o príslušností dát do určitého logického tunela alebo kategórie pre kvalitu služieb. Tento spôsob umožňuje implementáciu ďalších veľmi atraktívnych riešení ako sú privátne siete v MPLS - MPLS VPN alebo MPLS TE (*Traffic Engineering*). Ukážku ako MPLS umožňuje existenciu viacerých hlavičiek vidno na Obr. 9.3.



Obr. 9.3: Reťazenie viacerých MPLS hlavičiek [27]

Ako vidieť na Obr. 9.3, medzi rámec a paket boli vložené tri MPLS hlavičky. Informáciu, že rámec obsahuje MPLS hlavičku nesie identifikátor v poliach hlavičiek na druhej vrstve RM OSI modelu. Zároveň tento identifikátor tiež indikuje, že za MPLS hlavičkou (alebo hlavičkami) nasleduje IP hlavička. Systém teraz môže začať spracovávať MPLS hlavičku. Pri jej spracovaní narazí na pole "S", kde nájde bit nastavený na 0. Ten indikuje, že za aktuálne spracovávanou hlavičkou je ďalšia vnorená MPLS hlavička. Toto vnorenie môže byť samozrejme viacnásobne a pokračuje až do poslednej MPLS hlavičky, ktorej pole S má hodnotu 1. V rámci tejto práce budú spracované možnosti MPLS TE. V rámci MPLS TE sa využívajú vnorené hlavičky MPLS, ktorých dodatočná informácia sa však už nepoužíva na smerovanie. Techniky MPLS TE budú v tomto dokumente podrobnejšie rozobrané v ďalších kapitolách. Ďalším možným využitím, ktoré sa v MPLS často využíva je MPLS VPN (*Virtual Private Network*). V MPLS VPN indikuje druhá vnorená hlavička lokálne jedinečné číslo tunela medzi dvoma destináciami, ktorý tvorí privátne spojenie medzi dvomi sieťami [28].

Výmenu a šírenie návestí zabezpečuje protokol LDP (*Label Distribution Protocol*). Ide o otvorený štandard. Známe sú aj iné proprietárne protokoly, ktoré sa už však v dnešnej dobe nepoužívajú.

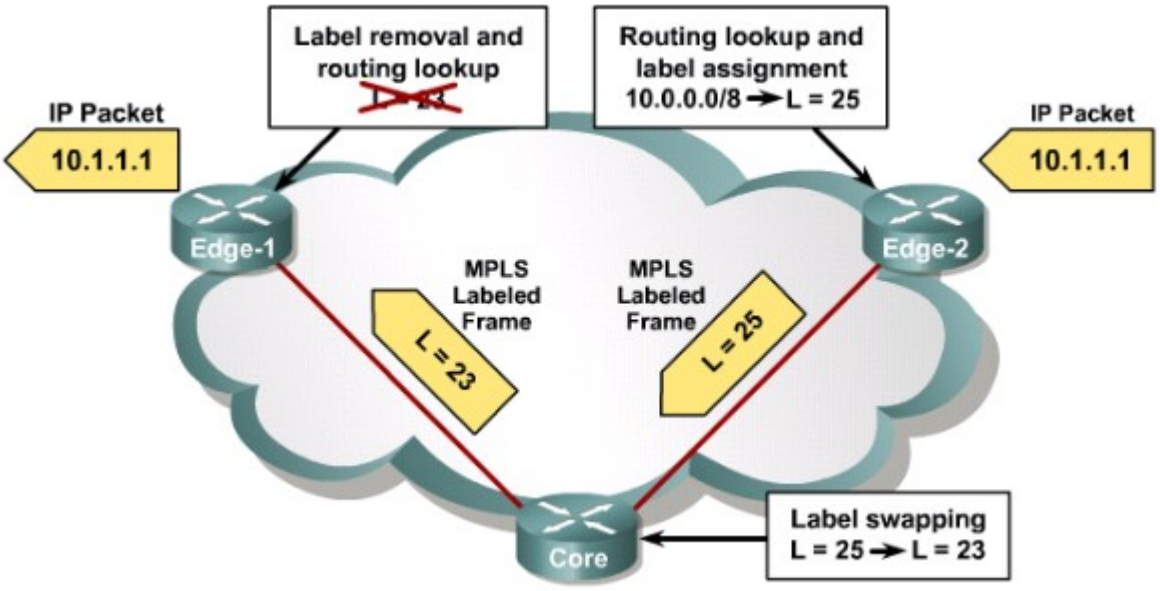
9.4 MPLS smerovanie

Technológia MPLS, ako už bolo spomenuté, vychádza zo základu, že prepínanie rámcov je rýchlejšie ako smerovanie, nakoľko je možné ho implementovať hardvérovo. Znamená to, že prepínanie, ktoré sa uskutočňuje na základe statickej MAC adresy, je efektívnejšie ako smerovanie na základe IP adresy. Preto technológia MPLS skĺbila rýchlosť a jednoduchosť prepínania z druhej vrstvy RM OSI modelu s možnosťou logickej adresácie ciest z tretej vrstvy RM OSI modelu.

Všetky tieto vlastnosti predstavuje MPLS návestie (*Label*), ktoré je často kategorizované ako hlavička dva a pól-tej vrstvy RM OSI. MPLS sieť prepína výhradne na základe tejto MPLS hlavičky [29]. Znovu je potrebné pripomenúť, že MPLS je prenosová sieť poskytovateľa a smerovanie je založené na MPLS hlavičke pridávanej do každého rámca. Avšak ide o sieť poskytovateľa služieb a platnosť MPLS hlavičiek končí, keď rámec príde na okraj MPLS siete. Tu je MPLS hlavička odstránená a smerovanie do siete zákazníka pokračuje pôvodnou technológiou. Preto MPLS doména nedosahuje koncové stanice, ani sieť zákazníka a nie sú potrebné z jeho strany žiadne technologické úpravy ohľadom MPLS.

MPLS sieť rozdeľuje svoje smerovače na dve hlavné kategórie a to hraničný (*Edge*) a vnútorný (*Core*) smerovač. Vnútorný smerovač je smerovač, ktorý smeruje na základe MPLS

návestia. Hraničný smerovač, okrem smerovania na základe MPLS návestia, musí ešte prekladať z pôvodnej technológie z nie MPLS sietí na MPLS návestia, ktoré chce ďalej smerovať do MPLS siete poskytovateľa. Ukážku základného konceptu topológie MPLS siete vidno na Obr. 9.4.



Obr. 9.4: Architektúra MPLS siete s vnútornými a hraničnými smerovačmi [27]

Ako na Obr. 9.4 vidieť, hraničné smerovače pridávajú do rámca dodatočne MPLS návestie. Na základe tohto MPLS návestia vnútorné smerovače smerujú. Výstup z MPLS siete opäť zabezpečuje hraničný smerovač, ktorý MPLS návestie z rámca odstráni a umožní tak komunikovať s nie MPLS sieťou.

10 MPLS *Traffic Engineering*

MPLS *Traffic Engineering* (MPLS TE) je jeden z najmodernejších a v súčasnosti najžiadanejších prostriedkov v sieťach u poskytovateľov služieb. Použitie MPLS v sieťach sa zvyšuje predovšetkým vďaka schopnostiam MPLS TE. MPLS využíva dostupné informácie poskytované smerovacími protokolmi na sieťovej vrstve a operuje ako ATM sieť na linkovej vrstve. S MPLS sú TE vlastnosti integrované do sieťovej vrstvy, ktoré sú využívané na lepšie riadenie využitia prenosového pásma medzi smerovačmi v SP sieti. Ide o manipuláciu dátových tokov tak, aby vhodne zapadli do povahy siete. V kombinácii s MPLS QoS môžeme efektívnejšie riadiť využiteľnosť pásma a zároveň zlepšiť alebo garantovať kvalitu služieb. Riadenie toku dát zlepšuje kvalitu v sieti a zabezpečuje predikatívne podmienky pre dáta. Využitie riadenia toku dát je jeden z prístupov k dátam a zlepšuje manažment využitia dostupných prostriedkov. Jej jednoduchosť, rýchlosť a použiteľnosť je vhodná aj pri nečakaných udalostiach, ktoré nárazovo menia bežné predpokladané podmienky v sieti, ako napríklad významná športová udalosť, škandál celebrity, nový film, aktualizácie operačných systémov, či nečakaný výpadok veľkej časti siete, poškodenie infraštruktúry a podobne. Do niekoľkých minút je možné revitalizovať sieť a aspoň čiastočne obnoviť jej funkčnosť vhodným presmerovaním dátových tokov.

Zlepšenie využitia prenosového pásma dosiahneme vytváraním TE tunelov medzi smerovačmi. Tunel je virtuálna logická cesta od počiatočného bodu ku koncovému. Tunel je jednosmerný a teda koncový uzol tunela nemôže byť zároveň počiatočným bodom toho istého tunela. Tunel môže mať predpísanú cestu, alebo nepredpísanú. Možná je aj kombinácia oboch prístupov. Predpísaná cesta znamená, že dáta – pakety, musia nutne daným uzlom prejsť. Ak nie je definovaná cesta, smerovač sa rozhoduje na základe svojej smerovacej tabuľky. Takéto správanie čiastočne pripomína prepínanie okruhov alebo prepínanie buniek v ATM sieti. Ide teda o akúsi kombináciu zasadenú do konceptu MPLS. Vytváraním TE tunelov môžeme rozkladať záťaž a tak lepšie využiť prenosové pásmo. Jednotlivým tunelom môžeme dodatočne zadefinovať ich vlastnosti a parametre, čo enormne zlepšuje flexibilitu a prispôsobivosť. Je možné riadiť napr. cenu tunela, jeho prioritu, cestu, dáta vstupujúce do tunela a ich kvalitu a podobne. TE je stavané tak, aby v prípade výpadku v sieti – konvergencii, bol čas výpadku čo najkratší. Slúžia nato dodatočné techniky ochrany uzla, linky a cesty. Čas konvergencie a stratovosť paketov sú omnoho nižšie ako pri použití akejkoľvek inej techniky, či optimalizácie.

Pred vznikom MPLS TE sa v IP sieťach riadil tok dát zmenami parametrov liniek a teda sa zmenil výpočet optimálnej cesty v smerovacích protokolov. V tomto spôsobe absentuje škálovateľnosť a ovplyvnené sú všetky dátové toky. Ďalším spôsobom bolo riadenie politikami (*Policy Based Routing* – PBR). Politika sa definuje na každom zariadení zvlášť. Možnosti a schopnosti sú širšie, no stále nám chýba jednoduchosť riešenia a väčšia škálovateľnosť. V ATM sieťach sa problém rieši úplnou topológiou (*full mesh*), čo však je náročné na manažment a aj pre zariadenia. Výpadok alebo obnovenie znamenajú enormnú záplavu riadiacich informácií, čo vo väčších sieťach spôsobuje vážne problémy s konektivitou. Záťaž je na úrovni $O(N^2)$ až $O(N^3)$ [30]. MPLS TE sa zaraďuje do kategórie CBR (*Constraint Based Routing*), čiže smerovanie na základe požiadaviek.

10.1 Využitie MPLS TE

MPLS TE sa využíva predovšetkým na [30]:

- optimalizáciu využitia prenosového pásma riadenia toku dát
- riešenie nečakaného zahltenia
- riešenie výpadkov uzlov a liniek

Aby sme mohli vôbec využívať funkcie TE, musíme ako smerovací protokol použiť jeden z LS protokolov. V súčastnosti sú len dva LS protokoly s podporou MPLS TE a to OSPF a IS-IS. IS-IS je preferovanejším protokolom, keďže protokol OSPF nepodporuje všetky funkcie, ktoré podporuje protokol IS-IS. IS-IS má už vo svojej podstate a architektúre lepšie predpoklady na implementáciu do IP siete. Je to predovšetkým robustnosť, škálovateľnosť, hierarchický návrh, možnosti optimalizácie, prispôsobivosť a určovanie hraníc oblastí na linkách a nie na smerovačoch ako v prípade OSPF. MPLS TE pridáva aspoň jedno návestie navyše na jeden tunel do zásobníka návestí.

MPLS TE používa protokol RSVP pre signalizáciu požiadaviek na jednotlivé tunely. Tento protokol využíva model *IntServ* v kvalite služieb. Nutnou podmienkou pre MPLS TE je aktivovaná technológia MPLS. V prostredí MPLS sa výmena a šírenie návestí deje najmä prostredníctvom protokolu LDP.

V nasledujúcich častiach popíšeme zaujímavé možnosti MPLS TE.

10.1.1 Atribútové bity

MPLS TE si dokáže vybrať cestu na základe ich vlastností. Máme k dispozícii atribútové bity (*Attribute Flags*), ktoré identifikujú nami definované parametre a vlastnosti linky. Ide o 32-bitovú bitmapu. Pri výbere uzlov a liniek, ktorými bude tunel prechádzať, sa môže tunel rozhodovať na základe týchto vlastností, alebo ich môže aj ignorovať. Kombinovaný prístup je možný, teda časť bitovej mapy je požadovaná a časť je pre tunel nezaujímavá. To, ktoré bity majú byť ako nastavené a ktoré sú sledované, zadávame pri špecifikácii tunela. Nastavujeme takzvanú afinitu (požadované bity) a masku (sledované atribútové bity). Takto je možné riadiť napr. cenu tunela, predpokladané oneskorenie a podobne. Napríklad hlasovým dátam vytvoríme tunel, ktorý nesmie prechádzať cez satelitnú linku, pričom zavedieme atribútový bit pre satelitné linky a nastavíme afinitu a masku tak, aby bol bit v atribútoch liniek sledovaný. Tunel automaticky vyberie optimálnu cestu bez použitia satelitnej linky. Samozrejme, je nutné administratívne každej linke zadefinovať atribúty a zjednotiť ich význam.

10.1.2 Administratívna váha

Zaujímavá je funkcia administratívnej váhy (*Administrative Weight*). Ide v podstate o metriku v MPLS TE, ktorá je implicitne zhodná s metrikou IGP protokolu. Jej zmenou získavame ďalší parameter linky. Protokol OSPF vypočíta metriku cesty len na základe rýchlostí liniek, narozdiel od protokolu EIGRP, ktorý počíta aj s oneskoreniami liniek. Protokol IS-IS v mnohých implementáciách rozoznáva hodnotu len na základe jedného parametra, najčastejšie počet skokov. Preto môžeme administratívnu váhu využiť ako ďalší parameter cesty – oneskorenie. Následne zadefinujeme, či sa má tunel rozhodovať na základe najmenšej ceny IGP protokolu – štandardne rýchlosť liniek, počet skokov, alebo na základe administratívnej váhy, čo bude najmenší súčet

oneskorení na linkách. Táto funkcia nám umožňuje diferenciáciu dátových tokov a zlepšuje proces rozhodovania sa pri výbere optimálnej cesty [30]. Ak máme dátové toky, ktoré požadujú nízke oneskorenie a v zásade prenosové pásmo nie je problém, je možné sa rozhodnúť podľa administratívnej váhy a nie IGP metriky. To môžeme použiť napríklad pri VoIP.

10.1.3 Manažment šírenia stavových informácií

Keďže sa predpokladá, že vytvorených tunelov bude násobne viac ako liniek v skutočnosti, tak sa zaviedol manažment šírenia stavových informácií. V prípade výpadku alebo obnovenia linky a uzla sa informácia šíri samozrejme okamžite. Základné pravidlo znie, že konvergencia musí byť čo najrýchlejšia a preto sa tieto informácie nepozdržujú. Inak je to s fázou výpočtu v konvergencii. Zmena na jednej linke môže vyvolať zmeny v desiatkách tunelov vo viacerých zariadeniach naraz. Každý tunel by si musel autonómne vypočítať nové hodnoty a tieto informácie prešíriť do siete, čo je veľmi vysoká záťaž. Preto sa zaviedlo rozdelenie na významné a nevýznamné zmeny. Významná zmena je každá veľká zmena a je vypočítaná bez pozdržania. Najčastejšie ide o relatívne veľký skok v rezervácii prenosového pásma (asociované s protokolom RSVP) a už spomenuté výpadky. Nevýznamné správy sú napríklad relatívne malé zmeny v rezervácii prenosového pásma. Tie sú posielané v periodických intervaloch. Sú zadefinované isté medze, ktoré keď sa presiahnu, tak i nevýznamné zmeny sa stanú významnými. Ešte existuje jeden typ správ a to chybové správy, ktoré sú považované za významné. Zmeny medzí sú konfigurovateľné a taktiež definovanie periodického posielania nevýznamných správ [30].

10.1.4 Opätovná optimalizácia tunelov

MPLS TE ponúka aj možnosť opätovnej optimalizácie tunelov. Ak je tunel aktívny a nastala zmena niekde v sieti, tunel túto zmenu môže zachytiť, ale nereflektuje ju ihneď. Ide častokrát o drobné konfiguračné zmeny ako zmena ceny linky. Prepočítavanie je periodické a možné optimalizácie. Takisto je možné nadiktovať ignorovať časovač a vykonať prepočítanie ihneď. A samozrejme je možné nakonfigurovať udalosti, na základe ktorých sa tento časovač ignoruje. Nechýba ani možnosť zablokovania opätovnej optimalizácie v akomkoľvek prípade, pokiaľ je tunel aktívny. Táto funkcia je dôležitá, lebo zmena po opätovnej optimalizácii môže viesť k zmene cesty, čo spôsobí zmenu oneskorenia v už aktívnej komunikácii medzi koncovými zariadeniami [30]. Takáto zmena je citeľná najmä pri VoIP.

10.1.5 Rozkladanie záťaže

Rovnomerné rozkladanie záťaže a taktiež nerovnomerné rozkladanie záťaže je podporované v MPLS TE. Ide o jednu z kľúčových vlastností tejto technológie. Je možné vytvoriť viacero rôznych tunelov medzi zdrojovým uzlom a destináciou s rôznymi metrikami, pričom je možné zadefinovať, aby boli všetky tieto tunely využívané. To, že aká pomerná časť dátového toku bude tiecť tým ktorým tunelom sa definuje na jednotlivých tuneloch administratívne. Aby sme využili túto funkciu naplno, treba mať TE tunel destinácie pozdĺž všetkých ciest a je treba mať zadefinovanú nenulovú hodnotu šírky prenosového pásma. Vzniká otázka, či rozkladanie záťaže neovplyvní negatívne oneskorenie v dátovom toku. Odpoveď znie: nemusí. Závislosť je od implementovaného algoritmu rozkladania záťaže a na základe akých parametrov sa záťaž rozkladá [30]. V Cisco zariadeniach je implementovaný CEF (*Cisco Express Forwarding*), ktorý rozkladá záťaž na základe zadefinovaných parametrov. Avšak pre pakety z jedného dátového toku je

výsledok vždy rovnaký a teda dáta prechádzajú vždy tým istým tunelom. Tu nám nevznikne problém s variabilným oneskorením medzi paketmi, avšak za cenu menej efektívneho rozkladania záťaže. Vhodne zvoliť kritéria pre CEF je nutné vykonať na základe analýzy dátových tokov v sieti.

10.1.6 Posunutie susedstva

Hlavným problémom TE je množstvo vytváraných tunelov. Ak chceme spojiť každý smerovač s každým, hoc len v okrajových bodoch siete, ich množstvo je veľké. Ak uvážime, že občas je nutné vytvoriť pre tú istú cestu viac rôznych tunelov, výsledné číslo je zrazu enormné. V prípade 100 smerovačov máme až 9900 tunelov pri spojení každého s každým, pretože tunel je jednosmerný. A to uvažujeme len jeden tunel pre jednu cestu. Zníženie počtu tunelov je možné pomocou funkcie posunutia susedstva - MPLS *Forwarding Adjacency*. Táto funkcia posúva TE tunel v hierarchii o jednu úroveň bližšie k jadru a teda klesá počet potrebných tunelov pri spojení každého s každým, alebo jedného s viacerými v rámci okrajového bodu siete. Samozrejme, toto posunutie v hierarchii je treba oznámiť smerovačom v nižšej úrovni. Táto funkcia redukuje počet potrebných tunelov. Treba si však dávať pozor, aby ceny tunelov a liniek v IGP pri posunutí boli konzistentné a aby sa rozkladala záťaž, ak je to požadované.

10.1.7 Automatické určenie šírky pásma

Je nutné na každej linke určiť, aká je jej skutočná šírka pásma a v TE tuneloch treba určiť, akú časť tohto pásma môže tunel využiť v prípade potreby. Funkcia automatického určovania šírky pásma (*autobandwidth*) periodicky kontroluje využitie poskytnutého pásma a signalizované požiadavky na pásmo. Smerovač sa môže rozhodnúť, že zmení aktuálne hodnoty prideleného pásma na základe reálnych požiadaviek alebo aktuálnych potrieb pre dátové toky. Tunely tak automaticky zreorganizuje a zmení im hodnoty pridelenej šírky pásma. Druhou možnosťou riadenia je monitorovací nástroj, ktorý vyhodnotí situáciu v sieti a navrhne zmeny, prípadne ich aj aplikuje [30].

10.1.8 Ochrana

V prípade zmeny pomerov v sieti nastáva konvergencia a tá trvá istý čas. V klasickej IP sieti s použitím LS smerovacieho protokolu môže byť čas na sieťovej vrstve až do 10 sekúnd, ak uvažujeme okamžitú detekciu a veľkú sieť. Časy sú konfigurovateľné, no ak sú časy nastavené príliš agresívne, môže dochádzať k falošným správam o zmene a zbytočným konvergenciám. V MPLS TE je každá konvergencia násobne väčšou záťažou. Ak konvergencia nastane na smerovači, ktorý zároveň je východzím bodom pre viacero tunelov, tak po skončení výpočtu v smerovacom protokole nastáva konvergencia v tuneloch. Časy konvergencie tunelov sú teda dlhšie, keďže sa musí čakať na informácie od smerovacieho protokolu. Zmena časovačov v smerovacích protokoloch nemôže byť nastavená až tak agresívne, ako v prípade nepoužitia MPLS TE. Druhou možnosťou ako zrýchliť konvergenciu tunelov je použitie ochrany.

Ochrana nie je konfigurácia viacerých alternatívnych tunelov, ale konfigurácia tunelov, ktoré sú priamo označené ako záložné. Túto funkciu nazývame aj ako MPLS FRR (*Fast Reroute*) – rýchle presmerovanie. Záložný tunel sleduje primárny tunel a spoločne s ním signalizuje požiadavky. Ak nastane výpadok, záložný tunel môže okamžite po detekcii prebrať úlohu primárneho tunela a tok dát ostáva neporušený. Porušenie je len v čase od výpadku po detekciu, respektíve, výmenu rolí. Je to preto, lebo záložný tunel má už vopred signalizované požiadavky

pozdĺž náhradnej cesty. Alternatívne tunely by po detekcii museli začať signalizovať požiadavky a to samozrejme predlžuje čas konvergencie, preto to nepovažujeme za ochranu [RFC4090].

Chrániť môžeme cestu, linku alebo uzol. Podpora ochrany cesty nie je dostatočne implementovaná v dnešných zariadeniach. Ochrana linky znamená, že linku, ktorú chceme chrániť, konfiguráciou záložného tunela obídeme cez iné linky a uzly. Takúto ochranu nazývame taktiež NHop ochrana (*Next-hop Router*). Uzol, na ktorom konfigurujeme záložnú cestu označujeme ako PLR (*Point of Local Repair*). Uzol, na ktorom sa spája záložný tunel s primárnym označujeme ako MP (*Merge Point*). Ochrana uzla znamená vytvorenie záložného tunela, ktorý daný uzol obchádza. Ochrana sa označuje tiež ako NNHop (*Next-next-hop router*).

Ak je nakonfigurovaných viacero záložných TE tunelov na PLR uzle, môže dôjsť k problému výberu optimálnej cesty. Preto sa zaviedla funkcia povýšenia. V periodických intervaloch sa kontrolujú záložné tunely. V prípade, že iný záložný tunel je aktuálne lepší ako momentálny záložný – pripravený prebrať úlohu primárneho, dôjde k výmene rolí a to nazývame povýšením.

Ochrany výrazne skracujú čas konvergencie, no otázna je ich podpora na zariadeniach. Ako sme si pokusne skúšali, mnohé operačné systémy smerovačov tieto funkcie majú podporované len vo vyšších radách. Ešte existujú ďalšie mechanizmy na fyzickej vrstve, ako napríklad v POS interfejsoch funkcia rýchlej konvergencie. Síce sa čas konvergencie udáva do 50 ms, alternatívny tunel musí tak či tak signalizovať svoje požiadavky a čas konvergencie v TE tuneloch sa nám zlepší len o niečo. Ani ostatné špecifické riešenia ako SONET APS, SDH MSP [31] neriešia hlavný problém tak, ako to rieši ochrana. Je však deklarované, že ochrana – MPLS FRR má lepšie časy konvergencie ako IP smerovacie protokoly [31, RFC4203, RFC4205]. Toto tvrdenie si prakticky overíme.

10.1.9 Automatická cesta

Automatická cesta (*autoroute*) naviguje IGP smerovací protokol, aby použil MPLS TE tunel ako jeho nasledujúci skok na dosiahnutie koncového uzla a destinácií za týmto uzlom. Výhodou je, že smerovací protokol si nepotrebuje vytvoriť susedstvo cez tunel. Dochádza tak k zmene hodnôt vo fáze výpočtu a teda môže dôjsť k zmene smerovacej tabuľky.

MPLS TE má problém so skupinovou komunikáciou v prípade použitia funkcie automatická cesta. Keďže skupinová komunikácia je smerovaná na základe RPF (*Reverse Path Forwarding*), to znamená, že na základe zdrojovej IP adresy, spomenutá funkcia môže spôsobiť zahadzovanie paketov, keďže v smerovacej tabuľke chýba záznam o takejto ceste. Riešením je nepoužitie danej funkcie, alebo dodatočná aktivácia funkcie, ktorá vytvorí záznam v smerovacej tabuľke tak, ako keby TE tunely ani neexistovali. Podpora je v oboch smerovacích protokoloch OSPF a IS-IS [30].

10.2 Kvalita služieb v MPLS TE

Na riadenie kvality služieb v MPLS TE môžeme použiť všetky známe modely a to *IntServ*, *DiffServ* a *Best Effort*. MPLS TE veľmi dobre spolupracuje s protokolom RSVP, ktorý slúži na signalizáciu požiadaviek dátových tokov v modeli *IntServ*. Tento model má nevýhodu v škálovateľnosti, avšak je možné použiť aj model *DiffServ*. Model *IntServ* nebudeme ďalej analyzovať, keďže sme ho už raz spomínali a v podstate nejde o žiadne zmeny pri použití v MPLS TE sieťach. Jediným rozdielom je, že signalizácia vyberá vhodný TE tunel na splnenie požiadaviek pre dátové toky, CBR.

DiffServ-Aware Traffic Engineering (DS-TE) je spojenie modelu *DiffServ* pre kvalitu služieb a TE [32]. Táto metóda taktiež požaduje protokol RSVP, keďže je súčasťou TE. DS-TE pridáva možnosť diferencovať služby tak, ako to je známe v modeli *DiffServ*. DS-TE pozostáva z piatich súčastí a to [30]:

- Dostupnosť podmnožiny šírky pásma na linke – riadené protokolom RSVP
- Plánovanie na linke – značenie paketov a manažment radov
- Požiadavky na podmnožinu šírky prenosového pásma z východzieho uzla – nároky na TE tunel
- Prístup kontroly tunela z východzieho uzla – administrácia, ktoré a koľko dát vstúpi do TE tunela
- Preempcia tunela – možnosť zostaviť TE tunel pre prioritnejší tok dát a „menej dôležitý“ TE tunel eventuálne aj zrušiť, ak je nedostatok voľnej nerezervovanej šírky pásma

10.3 MPLS TE s MPLS VPN a MPLS QoS

Spojenie MPLS TE s MPLS VPN nepredstavuje žiaden problém. Dokonca je možné pre každého zákazníka vytvoriť hneď niekoľko TE tunelov a diferencovať tak jeho služby, čo však nedáva veľký praktický zmysel. Avšak na okrajových smerovačoch je vhodné zadefinovať niekoľko TE tunelov, ktoré budú prenášať od viacerých zákazníkov ich dáta. Každý tunel bude určený pre iný typ služieb a tak môžeme zadefinovať napríklad osobitný tunel pre VoIP dáta od zákazníkov a osobitné tunely pre ostatné dáta. Spolu s využitím MPLS QoS dosiahneme požadovanú kvalitu služieb a garanciu. A samozrejme, nezáleží na fyzickej infraštruktúre a použitom linkovom protokole, keďže MPLS podporuje mnoho rôznych linkových protokolov, čo sa nazýva AToM (*Any Transport over MPLS*) [30].

11 Protokol IPv6

Doteraz sme v popise uvažovali len o smerovanom protokole IP verzie 4 - IPv4. Všetky doteraz popísané techniky, prostriedky a ich analýza bola stavaná len pre tento sieťový protokol. Nástupcom protokolu IPv4 je protokol IP verzie 6 - IPv6, ktorý prináša nové možnosti a má viacero výhod. Medzi tie najhlavnejšie by sme zaradili:

- absencia broadcast_u
- využívanie skupinového vysielania
- väčší rozsah IP adries
- globálna dosiahnuteľnosť
- efektívnejšia práca s IP adresami v zmysle sumarizácie na prefixy
- autokonfigurácia
- menšia záťaž na smerovačoch a lepšie podporovaná implementácia pre reťazenie hlavičiek
- roaming a mobilita
- viacdomovci (*multihoming*)
- a mnohé iné

Častokrát sa stretneme v literatúre aj s tým, že protokol IPv6 zlepšuje bezpečnosť [33], čo však nie je pravda, keďže neexistuje samostatné pole v hlavičke alebo povinnosť používať protokol zabezpečenia. Avšak protokol je implemetačne stavaný tak, že zlepšuje podporu a efektivitu pri použití zabezpečenia, napr. IPSec.

Protokol IPv6 má však aj mnohé negatíva a to najmä

- väčšia réžia (hlavička je dvojnásobne väčšia)
- neustále zmeny v štandardoch
- ešte stále nedostatočná podpora na strane koncových zariadení, ale aj prepájacích zariadeniach
- absencia aplikácií podporujúcich IPv6

I napriek týmto negatívám však nie je problém už dnes fungovať nad IPv6, avšak je nutné počítať s istými obmedzeniami a nedostatočnou podporou v aplikáciach.

V našom prostredí má však protokol IPv6 obrovské nedostatky. Síce existuje dobrá podpora pre IMS, keďže pôvodný zámer bol postaviť architektúru IMS len nad protokolom IPv6. Avšak absentujú aplikácie, ktoré by nám pomohli verifikovať zvýšenie dostupnosti a kvality služieb. V oblasti smerovania síce existujú smerovacie protokoly pre protokol IPv6 (RIPng, OSPFv3, IS-IS...), avšak ich implementácie sú na úrovni z pred niekoľkých rokov. To znamená, že nie je možná všetka tá optimalizácia, ktorú potrebujeme na zrýchlenie konvergencie. Nie je možné nastavovať časové parametre s takou granularitou, ako by sme si želali a ako si zaželali pred niekoľkými rokmi veľkí poskytovatelia služieb. Redundancia cez VRRP protokol, alebo príbuzné protokoly, je tiež nedostatočná, alebo nie je v súčasnosti implementovaná. Úplne najhoršie je, že nie je možné mať MPLS sieť nad protokolom IPv6, keďže nie je implementovaný protokol LDP pre IPv6 a neexistuje ani žiadna alternatíva. Čiže nie je možné využiť MPLS v spojení IPv6 a ani ostatné súčasti MPLS ako MPLS TE [34].

Existuje však jedno riešenie. Ak chce zákazník mať IPv6 sieť, tak poskytovateľ mu to môže umožniť a môže jeho jednotlivé pobočky prepojiť cez svoju IPv4 sieť. Poskytovateľ služieb musí nutne mať IPv4 sieť. Cez protokol MP-BGP (*MultiProtocol BGP*) sa budú vymieňať MPLS návestia. Pre zákazníka je proces transparentný. Tým, že poskytovateľ má IPv4 sieť, je možná optimalizácia, redundancia a MPLS TE v plnom rozsahu, ako bolo analyzované [35].

Po odbornej diskusii sme sa rozhodli, že sa pokúsime našu implemetáciu overiť aj nad protokolom IPv6 práve s týmto riešením, keďže je asi jediné relevantné. Ostatné riešenia by si žiadali dodatočnú formu tunelovania, čo ešte viac degraduje naše možnosti a úplne sa odvracia od pôvodného zámeru zvýšenia dostupnosti a použitia MPLS TE.

Otázku IPv6 ponechávame nateraz otvorenú, keďže je problém nájsť materiály k tejto problematike. Ako sme sa dočítali z dokumentácii, novšie verzie implementácií v smerovačoch by už mali podporovať všetku optimalizáciu časových parametrov, ale konkrétnu implementáciu sme zatiaľ nenašli. Našli sme však návrh pre protokol IS-IS, ktorý by mal v budúcnosti podporovať MPLS TE nad IPv6 [36]. Keďže ide len o návrh, ktorý je momentálne diskutovaný, toto riešenie je pre nás momentálne nepoužiteľné. IPv6 postupne doriešime v najbližších týždňoch a aktualizujeme analýzu.

12 Zhodnotenie analýzy

Analýza preukázala, že problematika zvýšenia dostupnosti uzlov a zlepšenia kvality služieb nie je jednoduchou záležitosťou. Postupne sme rozoberali protokol SIP a SDP, architektúru IMS, jej súčasti a vzájomné prepojenia. Do hĺbky sme sa venovali optimalizácii siete najmä na sieťovej vrstve. Popísali sme hlavné smerovacie protokoly a ich techniky, časové a nečasové parametre vhodné na optimalizáciu. Venovali sme sa ich schopnosti dosiahnutia rýchlej konvergencie. Spomenuli sme všetky ostatné významné techniky pozitívne i negatívne vplývajúce na konvergenciu. Popísali sme možnosti kvality služieb, spôsoby a prístupy a venovali sme sa technológii MPLS a jej súčasti MPLS TE. Neopomenuli sme ani ani protokol IPv6.

Analýza jasne preukazuje opodstatnenosť práce a dáva tušiť dobré výsledky pri návrhu, v ktorom si kladieme za cieľ optimalizovať siete na rýchlo konvergujúce bez zníženia ich stability a garantovanie adekvátnej kvality služieb pre aplikácie. Myslíme si, že naše riešenie bude schopné dôstojne konkurovať ostatným prístupom k tejto problematike. Je možné, že aplikovaním všetkých podstatných metód dosiahneme práve my lepšie výsledky. Očakávame totižto synergické efekty po aplikácií relevantných techník.

V nasledujúcej fáze sa zameriame na praktické nasadenie nami popísaných technológií, ich vplyv na sieť a služby a navrhneme množstvo testov, na ktorých budeme študovať zmeny. Pripravíme niekoľko modelových topológií, na ktorých názorne ukážeme, ako postupovať pri nasadzovaní jednotlivých technológií. Výsledky našej práce podrobne zdokumentujeme.

13 Špecifikácia požiadaviek

V tejto sekcii špecifikujeme hlavné aspekty nášho produktu.

Našou úlohou je zvýšiť dostupnosť jednotlivých uzlov v sieti a najmä kritických uzlov. Za kritický uzol považujeme každý uzol, ktorého nedostupnosť môže spôsobiť finančné straty a nespokojnosť zákazníka.

- Celá študovaná sieť bude pozostávať z dvoch častí: zákazník (zákazníci) a poskytovateľ služieb
- U poskytovateľa služby optimalizujeme počítačovú sieť na dosiahnutie:
  - zvýšenia dostupnosti uzlov a služieb
  - redundancie
  - rýchlej konvergencie pri zmene pomerov v sieti
  - rozloženia záťaže
  - garancie kvality služieb
- Navrhneme optimalizáciu a skvalitnenie služieb pre koncového zákazníka na strane zákazníka
- Navrhneme verifikačný postup na overenie skvalitnenia poskytovaných služieb
- Budeme sa snažiť minimalizovať negatívne dôsledky optimalizácie
- Podrobne zdokumentujeme náš postup zabezpečenia skvalitnenia služieb, odporúčania, použité a implementované nástroje a proces verifikácie

Primárne sa budeme snažiť zabezpečiť zvýšenie dostupnosti poskytovaných služieb a zabezpečiť im požadovanú kvalitu. Ako prostriedok plánujeme využiť moderné techniky optimalizácie počítačovej siete, aj na úrovni služieb a aplikácii.

14 Návrh

V tejto časti bližšie popíšeme ako chceme dosiahnuť vytýčené ciele v predošlej kapitole. Testovacie siete zostavíme na reálnych zariadeniach v našich sieťových laboratóriách, kde budeme simulovať reálnu prevádzku na sieti.

Celý projekt sme rozdelili na dva samostatné celky, ktoré vzájomne prepojíme. Ide o smerovanie a aplikačné služby.

Sieť postavíme na protokole IP a technológii MPLS. Ako médium použijeme *ethernet*. Pri návrhu siete budeme brať do úvahy možnosti v laboratóriách a zároveň použijeme zásady dobrého návrhu ako bolo analyzované. Po konzultácii s odborníkmi z viacerých spoločností poskytujúcich služby si vyberáme smerovací protokol IS-IS. Okrem toho v analýze vyšiel ako najvhodnejší kandidát pre prostredie poskytovateľa služieb.

Na kontrolu a monitorovanie siete využijeme program postavený na protokole SNMP. Zvýšenie dostupnosti uzlov a služieb zabezpečíme pomocou redundancie, čiže využitím VRRP protokolu. Aplikujeme optimalizáciu časových a ostatných parametrov v konfiguračných súboroch smerovačov. V prípade zmeny pomerov v sieti vieme garantovať maximálne dvoj-sekundovú nedostupnosť. Spolu s redundanciou vieme minimalizovať počet kritických výpadkov, kedy sa uzol stane absolútne nedostupným.

Pri MPLS si môžeme vybrať z dvoch základných typov navrhovania MPLS sietí. Ide o taktický a strategický prístup. Pri taktickom návrhu vytvárame TE tunely podľa potreby a pri strategickom návrhu vytvárame v niektorých bodoch siete prepojenie každého s každým. Rozhodli sme sa pre taktický návrh. Vytvoríme niekoľko základných TE tunelov v sieti a budeme púšťať dáta do siete. Budeme postupne zvyšovať nároky na prenosové pásmo, až vznikne potreba nového tunelu alebo presmerovania časti komunikácie. Interaktívne zasiahneme do siete a vytvoríme nový TE tunel. Budeme merať rýchlosť a náročnosť tohto procesu, aby sme vedeli kvantifikovať schopnosť poskytovateľa reagovať na zmeny pomerov v sieti.

Kvalitu služieb nastavíme na strane zákazníka, na strane poskytovateľa služieb, na úrovni prepínania a smerovania a na úrovni aplikácii, čiže naprieč celým spektrom vrstiev RM OSI modelu v celej sieti.

Cieľom implementácie projektu IMS je poskytnúť odberateľom, zákazníkom moderné a technicky vyspelé služby. V dnešnej dobe je prenos hlasu neustále priorita číslo 1 pri zavádzaní multimediálnych služieb. Neustále však narastajú požiadavky na zložitejšie služby. Pri zavedení systému chceme zákazníkom garantovať prenos v najvyššej možnej kvalite bez citeľných obmedzení kvalitou linky a pásma.

Jadro nášho projektu na základe analýzy bude stáť na implementácii OpenIMS Core. Samotný systém IMS bude implementovaný na fyzických výpočtových strojoch. Jednotlivé počítače budú obsluhovať nasledujúce služby CSCF, HSS a AS. Medzi aplikačné servery zahrnieme systémy podporujúce služby ako VoIP a podobne.

V rámci systému chceme poskytovať niekoľko úrovní kvality našich služieb. Rôzne platiacim zákazníkom budeme garantovať rôznu kvalitu služby. V prípade zlatého zákazníka, zákazník ktorý potrebuje mať garantovanú dostupnosť a kvalitu služieb navzdory zvýšeným

nákladom, budú nami poskytované služby mať lepšiu priechodnosť našou transportnou sieťou. Túto službu môžeme garantovať mechanizmom zabezpečenia kvality služieb QoS.

Túto formu služieb chceme zabezpečiť rôznymi média bránami v našom systéme. Na ich základe hraničné smerovače budú rôzne označovať pakety RTP prúdu. Zdroj prúdu bude jednotný, ale na základe klasifikácie klienta bude využívať iné média brány.

V prípade úspešného aplikovania smerovacích mechanizmov a mechanizmov garantovania služieb budeme môcť pozorovať rôznu kvalitu vysielaného obrazu u zákazníkov, ktorý požadujú zvýšenú dostupnosť a kvalitu služieb a iných. K tomuto bude môcť dôjsť iba v prípade zahltenia transportnej siete. Ak táto podmienka nebude naplnená, klienti nemôžu pociťovať výkyvy v kvalite ponúkaných služieb.

Ako verifikačné nástroje použijeme BASH skripty, ktoré jeden z členov tímu používa na kontrolu rýchlosti konvergencie času, používateľské aplikácie na VoIP hovory a konferenčné hovory. Na generovanie dátových tokov si pomôžeme aplikáciami ako napr. PackETH, alebo spustíme FTP prenos a podobne.

Výsledkom bude zvýšenie dostupnosti uzlov a kvality služieb v sieti. Celý progres vytvárania produktu budeme priebežne podrobne dokumentovať. Na záver vytvoríme okrem hlavnej dokumentácie aj technickú dokumentáciu a používateľskú príručku. K projektu priložíme všetky relevantné materiály použité počas tvorby projektu.

V nasledujúcej časti bližšie popíšeme sledované parametre a princípy merania. Podkapitola je vo veľkej miere prevzatá z [7], keďže základná metodika je zhodná.

14.1 Návrh sledovaných parametrov a vyhodnocovanie

Našou úlohou je optimalizovať sieť tak, aby sme dosiahli pokiaľ možno čo najrýchlejšiu konvergenciu s čo najmenšími vedľajšími negatívnymi efektmi. Keďže sa orientujeme na koncového používateľa, primárne budeme merať a vyhodnocovať parametre, ktoré sú zaujímavé z pohľadu koncových zariadení.

Rozhodli sme sa merať konvergenciu siete a stratovosť paketov pri danej periodicite vysielania paketov. Ako periodicita je stanovená hodnota 30 ms, keďže touto rýchlosťou sú vysielané pakety z nášho IMS klienta počas hovoru.

Konvergencia siete je čas od porušenia skonvergovaného stavu až po znovu nadobudnutie tohto stavu. Práve v tomto časovom úseku dochádza k stratovosti paketov, keďže sieť práve vykonáva konvergenciu. Našou úlohou je minimalizovať počet kritických uzlov v sieti, kedy konvergencia ovplyvňuje koncové zariadenia takým spôsobom, že dochádza k strate paketov v ich komunikáciách. Pokúsime sa minimalizovať dĺžku konvergencie času s čo najmenšími negatívnymi efektmi. Hľadáme kompromis medzi rýchlosťou konvergencie a úrovňou zvýšenej záťaže. Uvažujeme primárne o výpadku jednej linky, respektíve jedného zariadenia v meranom časovom okamihu, keďže aj v praxi je nízka pravdepodobnosť viacerých výpadkov v rovnakom čase. Pre zaujímavosť vykonáme aj merania s viacerými simultánnymi výpadkami. Ďalej uvažujeme o zahltení uzlov a liniek na ceste.

Merania budú vykonané simuláciou výpadku fyzickým odpojením kábla z daného zariadenia. Meranie budeme niekoľkokrát opakovať. Presný počet bude závislý od typu merania,

typu výpadkov, od očakávanej variability hodnôt a od skutočnej variability. Merania s väčšou variabilitou a merania s neočakávanými hodnotami budeme opakovať viackrát. Hodnoty porovnáme so známymi hodnotami, predovšetkým uvedených v [7].

Zmeriame hodnoty konvergencie v neoptimalizovanej topológii a v optimalizovanej topológii. Optimalizácia bude vykonaná v jednom kroku, naraz pre všetky technológie. Výsledkom budú tabuľky vytvorené z meraní. Výsledná konfigurácia sa stane naším odporúčaním pre navrhnutú sieť. Cieľom bude minimalizovať počet konvergencií s teoretickou nekonečnou dobou trvania, skrátiť čas konvergencie a dosiahnuť prakticky nulovú hodnotu konvergencie všade tam, kde to bude možné. Týmto zvýšime dostupnosť a kvalitu služieb pre koncových používateľov.

Pri implementácii zmeriame časy konvergencie siete a stratovosť paketov pre vybrané výpadky liniek a zariadení. Implementáciu upravíme na fyzickej, linkovej a sieťovej vrstve podľa zistení z analýzy a návrhu. Vyberieme najvhodnejšie prostriedky a optimalizujeme ich. Časy opätovne premeriame a porovnáme hodnoty s hodnotami pred optimalizáciou. Tie výpadky, ktoré koncové zariadenia nepostihnú, budú vyhodnotené ako nulová konvergencia i keď v skutočnosti konvergencia v sieti trvá a dátový prenos ovplyvňuje. Nulová konvergencia znamená, že daný dátový prenos medzi dvomi koncovými zariadeniami nebol ovplyvnený konvergenciou, ale iná dvojica koncových zariadení byť môže.

Testy budeme vykonávať z testovacieho zariadenia. Ide o notebook Asus F5RL. Použitý je operačný systém Linux Ubuntu 8.10. Notebook bude slúžiť ako koncové zariadenie. Budeme testovať dosiahnuteľnosť iného koncového zariadenia v testovacom laboratóriu, predovšetkým IMS klienta. Z notebooku vysielame ping. Pre ping sme sa rozhodli z dôvodu jeho jednoduchosti, avšak zároveň poskytuje úplne všetky pre nás dôležité parametre. Primárne testujeme dosiahnuteľnosť koncových zariadení, teda ak je ping neúspešný, je problém v sieti. Ping vysiela komunikáciu v pároch, požiadavka a odpoveď. Požiadavku generuje vysielač a odpoveď prijímateľ. Simultánne bude prebiehať aspoň jedna dátová komunikácia medzi IMS klientmi. Použité hardvérové a softvérové prostriedky pre IMS klientov budeme striedať (počítače v laboratóriu, naše notebooky, telefóny, atď.)

Naše merania na počet stratených paketov a konvergenciu siete bude nasledovný. Budeme generovať kontinuálne komunikáciu prostredníctvom ping_ov (ďalej písané ako jedno slovo). Čas od prvého neúspešného pingu (neprijatá odpoveď) až po posledný neúspešne prijatý je čas konvergencie siete. Konvergenciu siete vyhodnotíme za ukončenú, až keď bude niekoľko pingov po sebe úspešných. Presný počet je závislý od topológie na fyzickej a logickej úrovni, predovšetkým od počtu uzlov rozkladajúcich záťaž a alternatívnych ciest do destinácie. Môže totižto nastať situácia, že časť siete je skonvergovaná, respektíve nastáva rozloženie záťaže a stratovosť paketov nie je 100 %-ná. Nami napísaný pomocný program vyhodnocuje konvergenciu siete ako čas medzi prvým neúspešne poslaným pingom až po posledne neúspešným.

Z notebooku odosielame ping v nasledovnom formáte:

```
ping -i 0.03 -T tsonly 84.47.53.2 | tee -i ping_test_01_01
```

Ping je názov programu. Prepínač *-i 0.01* je interval medzi dvomi vyslanými paketmi udávaný v sekundách. Tento zápis vyjadruje vysielanie paketov každých 30 ms. Prepínač *-T tsonly* určuje typ záznamu časovej známky. My zaznamenávame len čas vysielania. Nasleduje IP adresa destinácie.

Za lomenou čiarou nasleduje názov programu *tee*, ktorý rozdvojí vstup na výstup na obrazovku a zároveň bude zapisovať všetok vstup aj do súboru. Prepínač *-i* vyjadruje nereakciu na prerušenia. Keďže ping vypíname klávesovou skratkou Ctrl+C, čo je prerušenie, tak bez tohto

prepínača by sme nezaznamenali záverečnú štatistiku do súboru. Nasleduje názov súboru do ktorého zapisujeme.

Ping sa stane našou testovacou aplikáciou, ktorú budeme využívať vo všetkých testoch a označovať ju týmto pomenovaním. Počet stratených paketov je závislý od periodicity odosielania. V prípade stratovosti, aplikácia automaticky pozdržiava vyslanie nasledujúceho paketu, preto počet stratených paketov prenásobený periodicitou odosielania paketov sa nerovná konvergencii siete. Stratených paketov je vždy menej. Ide o vlastnosť tejto aplikácie. Podobná vlastnosť sa prejavuje vo viacerých protokoloch vyšších vrstiev a aplikáciách, akými sú aj IMS klienti.

Budeme testovať niekoľko vybraných topológií. Meriame konvergenciu siete (KS) a stratovosť paketov (SP). Vo fáze implementácie vyhodnocujeme konvergenciu siete z pohľadu koncových zariadení.

V nasledujúcej časti si predstavíme platformu Ericsson SDS na vývoj aplikácií v IMS. Ide o spojenie analýzy možností použitia tohto riešenia s praktickou implementáciou.

14.2 Platforma ESDS na vývoj aplikácii v IMS

Ericsson Service Development, skrátene Ericsson SDS (ESDS), je platforma určená na vývoj aplikácií architektúry IMS v prostredí upraveného nástroja *Eclipse*. Platforma je primárne navrhnutá na fungovanie pod operačným systémom Microsoft Windows. Vďaka zahrnutiu všetkých kľúčových prvkov architektúry IMS Ericsson SDS umožňuje a uľahčuje návrh, vývoj, a testovanie klientských aj serverových aplikácií určených pre nasadenie v systéme IMS [37].

Ericsson SDS v sebe zahŕňa nasledovné prvky siete IMS [37]:

- simulovaný Aplikačný server (*Application server*) – označovaný aj ako tzv. SIP *container*, ktorý umožňuje do IMS siete pridávať služby
- DNS server – umožňuje preklad domén na IP adresy, telefónne čísla či transportné protokoly
- domovský server účastníka (*Home Subscriver Server*) – podporuje poskytovanie údajov, ako sú používateľská identita (*user identity*), prístupové parametre používateľa *(user access parameters*), identity verejných služieb (*Public Service Identities*), a informácie týkajúce sa spúšťania konkrétnych služieb siete IMS (*service-triggering information*)
- *Call session control* funkcia (*Call Session Control Function*) – v prostredí Ericsson SDS sú P-CSCF, I-CSCF a S-CSCF spojené do jedného CSCF prvku
- *Brakeout Gateway Control* funkcia (*Brakeout Gateway Control Function*) – umožňuje simulovať smerovanie hovorov do a z klasických telefónnych sietí

Keďže Ericsson SDS je primárne vývojové prostredie, jeho súčasťou je aj niekoľko knižníc a nástrojov zameraných na vývoj klientskych aplikácií, ako aj aplikácií aplikačného servera. Pre vývoj IMS klientov sa v Ericsson SDS nachádza služba zvaná *Ericsson IMS Platform* (ICP - *IMS Client Platform*) [37]. Prostredníctvom jej knižnice a API volaní sa z tvorby IMS klienta odstraňuje potreba vytvárania objektov na posielanie, prijímanie a spracovávanie komunikácie s IMS sieťou poskytovateľa. ICP je teda agentom, ktorý sprostredkúva komunikáciu medzi samotným IMS klientom, a sieťou. Pre vývoj programov aplikačného servera je v Ericsson SDS vytvorená podpora pomocou sprievodcov, ktorí uľahčujú vytvorenie skeletu nových aplikácií [38].

Okrem základných prvkov siete IMS sú ešte súčasťou Ericsson SDS aj iné nástroje používané v procese vývoja softvéru na platforme [37]:

- nástroj na vizualizáciu IMS siete (*Visual Network*)
- *IMS Java Client utility* – API, pomocou ktorého je možné vyvíjať IMS aplikácie na mobilné zariadenia s podporou mobilnej verzie Javy
- emulátor prenosných zariadení (*Handheld emulator*) – v ktorom je možné simulovať aplikácie vyvíjané pre zariadenia s *OS Symbian*, napríklad na *Sony Ericsson* P1
- simulátor výskytu manažmentu skupín (*Presence and Group List Management Simulator*)
- simulátor *Push-to-talk* funkcionality v mobilnej sieti (*Push-To-Talk over Cellular Enabler Simulator*)
- simulátor systému okamžitých správ (*IMS Messaging Enabler Simulator*)
- simulátor televízneho vysielania protokolom IP (*IPTV Simulator*)

Platforma Ericsson SDS pre vývojárov predstavuje veľmi užitočný a komplexný nástroj pre vývoj aplikácií pre IMS siete. Pre vývojárov je k dispozícii kompletná dokumentácia k poskytovaným API v podobe *JavaDoc ALEX* knižnice, nachádzajúcej sa priamo v programe [37]. Bohužiaľ, pri testovaní základnej funkcionality systému sme narazili na niekoľko problémov, ktoré sa nám nepodarilo odstrániť. Systém sme testovali na virtuálnom počítači v programe *VirtualBox*. Hostovaným operačným systémom bolo *Windows XP Profession SP3* legálne získané v rámci MSDNAA dostupnej na FIIT. Testovanou platformou bola Ericsson SDS vo verzii 4.1 FD1 nainštalovaná podľa [39].

Prvým, aj keď menej vážnym nedostatkom, bola absencia balíčka *Microsoft Visual C++ 2005 Redistributable Package* v inštalácií Ericsson SDS, ako aj vo virtuálnom počítači. Dôsledkom absencie týchto knižníc bol fakt, že DNS server dodávaný ako súčasť celej platformy nebolo možné z prostredia programu spustiť. Pridaním balíčka sa však problém odstránil.

Druhým problémom, na ktorý sme počas testovania narazili, sa týkal PGM servera. Na základe postupu popísanom v [40] sme sa snažili spustiť ukážkovú IMS aplikáciu dodávanú spolu s platformou. Po jej úspešnej inštalácií malo nasledovať jej spustenie, avšak chyba v konfigurácií PGM servera nám znemožnila tento krok dokončiť. Chybu sa nepodarilo odstrániť ani po kompletnom reštarte operačného systému a všetkých súvisiacich služieb.

Tretí problém, ktorý sa vyskytol pri testovaní platformy sa týkal prípravy jednoduchého programu aplikačného servera, pričom postup vývoja aplikácie sme získali zo [38]. Tento mal na prijatú SIP správu s metódou *MESSAGE* odosielateľovi odpovedať najskôr správou 200 OK a potom *MESSAGE SIP* správou obsahujúcou text „*Hello World*!“. Po vložení zdrojových kódov do vygenerovaného skeletu, vygenerovaní nového používateľa a niekoľkých ďalších konfiguračných krokoch sa však nepodarilo program na aplikačnom serveri Sailfin v1 spustiť. Pre vylúčenie chyby v konfiguračnom postupe sme platformu Ericsson SDS odinštalovali, počítač reštartovali, a po nasledovnej inštalácií platformy postup znovu zopakovali. Chybu sa nám však odstrániť nepodarilo.

Jediný pozitívny výsledok, ku ktorému sme v rámci testovania platformy Ericsson SDS dospeli, bolo úspešné uskutočnenie hovoru medzi dvoma účastníkmi prihlásenými k tomu istému serveru. Jeden z klientov bol spustený z prostredia virtuálneho počítača a išlo o zdarma dostupnú verziu IMS klienta s názvom Mercuro. Tento mal nastavenú so systémom štandardne dodávanú identitu alice@ericsson.com, kde doménu ericsson.com hosťovala bežiaca platforma Ericsson SDS. V prípade druhého klienta išlo o program Monster a bol spustený na inom počítači nachádzajúcom sa v lokálnej sieti. Tento program bol do systému prihlásený s taktiež štandardne dodávanou

identitou bob@ericsson.com. Úspešný hovor bol vykonaný v smere Alica volá Bobovi a došlo k úspešnému prenosu zvuku aj textových správ. Vykonať hovor opačným smerom sa však nepodarilo, ani po výmene klientov na oboch stranách.

15 Prototyp riešenia

V súčasnej fáze projektu sme vytvorili prototyp riešenia, ktorý odráža hrubý návrh nášho riešenia a spĺňa špecifikované požiadavky. Na nasledujúcich stranách popíšeme implementáciu IMS a transportnej siete. Merania špecifikovaných parametrov zatiaľ nevykonáme. Sústredíme sa na úspešnú implementáciu podľa stanovených kritérií a konfiguráciu komponentov v IMS tak, aby sme v ďalšej fáze projektu mohli pohodlne merať a pokračovať v implementácii podľa potrieb a požiadaviek zákazníka.

V tejto časti postupne rozoberáme jednotlivé komponenty našej IMS siete a uvádzame základný postup inštalácie a konfigurácie. Následne popisujeme navrhnutú transportnú sieť s doteraz aplikovanými technológiami a postup základnej konfigurácie technológií. Optimalizácia transportnej siete bude predmetom neskoršej fázy implementácie, keď budeme vyhodnocovať a porovnávať namerané hodnoty.

Cieľom prototypu riešenia je demonštrovať naše schopnosti praktickej aplikácie analyzovaných oblastí a schopnosť prepojiť svet IMS a transportnú sieť. Návrh zapojenia je stavaný tak, aby bolo možné v druhej fáze projektu pokračovať v implementácií na rovnakej topológii a vyhodnocovať merania.

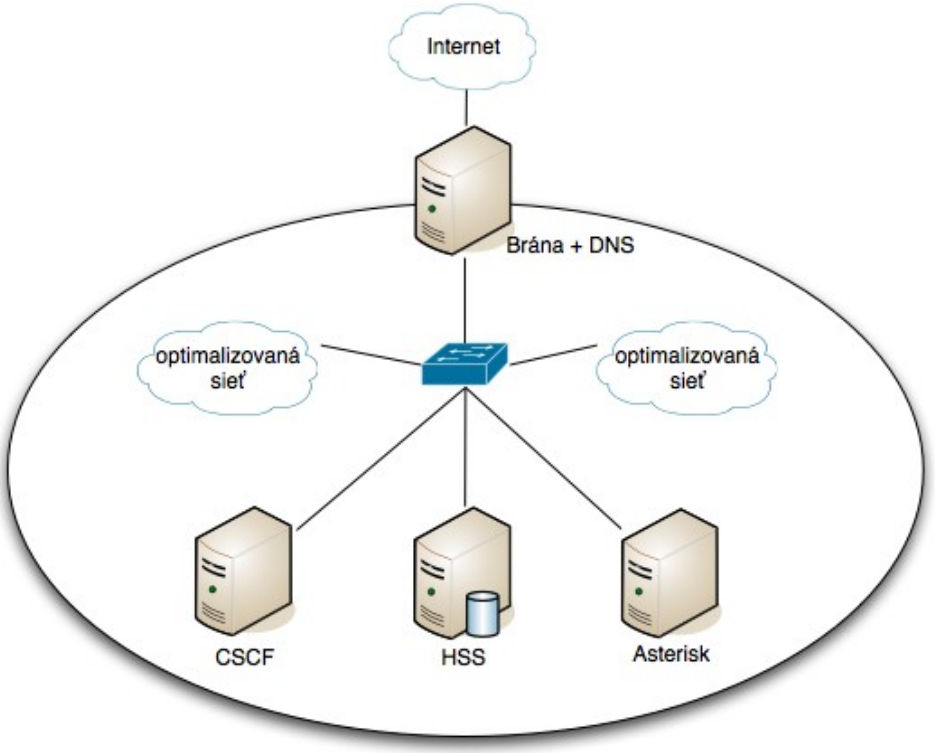
15.1 Topológia v IMS podsieti

Pre nasledovné rozloženie jednotlivých úloh funkcionality IMS sme sa rozhodli na základe analýzy. Z tej vyplynula potreba diverzifikácie výpočtových zdrojov, modularity a modifikovateľnosti komponentov IMS domény. Z dôvodu jednoduchšej údržby a členitosti celkového systému sme jednotlivé funkcie atomizovali do požadovanej miery. Funkcionalitu HSS sme separovali na jediné zariadenie. Takto bude hlavná databáza v prípade potreby aktualizácie, alebo implementovania nových súčastí domény, neustále dostupná. Takisto sa vyhneme problémom v prípade výpadku niektorej súčasti domény. Takto zvyšujeme pravdepodobnosť zachovania dostupnosti služieb.

Na základe výsledkov analýzy navrhujeme topológiu, ktorá sa primárne skladá z nasledujúcich komponentov:

- CSCF – riadiaca jednotka, ktorá umožňuje prístup k doméne a jej službám z hľadiska používateľa.
- HSS – zariadenie s databázou, ktoré je zodpovedné za uchovávanie identity a používateľských informácií jednotlivých používateľov.
- GATEWAY – výpočtové zariadenie, ktoré nám umožňuje vzdialený prístup prostredníctvom protokolu SSH. Administrátorom umožňuje konfiguráciu jednotlivých komponentov architektúry IMS. Tento počítač taktiež poskytuje funkčnosť DNS serveru.
- ASTERISK – systém umožňujúci priame záťažové testovanie základných funkcií VoIP. Systém je použitý iba vo fáze počiatočného testovania. V budúcnosti môže byť použitý ako aplikačný server domény.

Na Obr. 15.1 je znázornená navrhovaná topológia IMS domény začlenená do celkovej architektúry navrhovanej siete.



Obr. 15.1: Topológia navrhovanej IMS domény

V nasledujúcich častiach bližšie popíšeme jednotlivé použité komponenty.

15.1.1 CSCF

CSCF funguje v rámci IMS domény ako riadiaca jednotka zodpovedná za zostavovanie relácií, smerovanie SIP správ a registráciu. Ako sme už uviedli v analýze, poznáme tri typy CSCF:

- *Proxy-CSCF* (P-CSCF)
- *Serving-CSCF* (S-CSCF)
- *Interrogating-CSCF* (I-CSCF)

V architektúre *OpenIMS Core* sú reprezentované ako inštancie programu SER (SIP Express Routes) s rôznymi konfiguračnými súbormi. V našej topológii sú jednotlivé programy spustené na jednom výpočtovom stroji.

Pre inštaláciu danej funkcie sme si zvolili stroj v školskom laboratóriu D-105. Na výpočtový stroj bola nainštalovaná čistá distribúcia operačného systému Linux - Debian Lenny. Nakonfigurované boli najprv sieťové rozhrania. Tak ako všetky súčasti našej domácej domény, výpočtový stoj používa ethernetové rozhranie eth0. Bola mu staticky pridelená IP adresa 5.200.0.2/28 a nastavená predvolená brána pre odchádzajúce dáta a DNS server. Obe hodnoty ukazujú na rovnakú IP adresu 5.200.0.1. Následne sme overili konektivitu komunikáciou s verejnou sieťou.

Podľa postupu zverejnenom vývojármi projektu *OpenIMS Core*, sme zaviedli funkcionalitu CSCF. Postup inštalácie je nasledovný:

1. Inštalácia operačného systému Debian Lenny so zvolením inštalácie základných balíčkov
2. Príprava zdrojových kódov *OpenIMS Core* systému. Tie sme uložili v adresári */opt/OpenIMSCore/ser_ims* programom *svn* zo systému *Subversion*

   svn checkout http://svn.berlios.de/svnroot/repos/openimscore/ser_ims/trunk ser_ims
3. Kompilácia získaných zdrojových kódov programom make

   make install-libs all
4. Zmena prednastavenej IP adresy. Skriptom *configurator.sh* sme zmenili v konfiguračných súboroch výskyt prednastavenej IP adresy 127.0.0.1 na IP adresu sieťového rozhrania 5.200.0.2.
5. Zavedenie konfiguračných súborov na požadované miesto v adresári */opt/OpenIMSCore*
6. Spustenie inštancie programu SER pomocou skriptov *pcscf.sh*, *icscf.sh* a *scscf.sh*.
7. Po náležitom nastavení funkcionality HSS sme overili funkčnosť konfigurácie domény IMS.

15.1.2 HSS

Táto služba je zodpovedná za uchovávanie a poskytovanie používateľských informácií v rámci IMS siete. Okrem toho je to hlavný prvok zodpovedný za autentifikáciu a autorizáciu používateľov IMS siete. Ako sme už spomenuli, táto funkcia je implementovaná na samostatnom počítači, ktorý je zapojený do spoločnej počítačovej siete.

V rámci architektúry systému OpenIMS, HSS sa skladá z dvoch hlavných častí:

- MySQL databáza
- FHoSS servera

Nakoľko MySQL databáza je z hľadiska IMS siete len úložným priestorom, s ktorým je možné komunikovať iba prostredníctvom MySQL protokolu, je funkcionalita HSS doplnená o FHoSS obslužný server, ktorý zastáva úlohu agenta spracúvajúceho jednotlivé požiadavky.

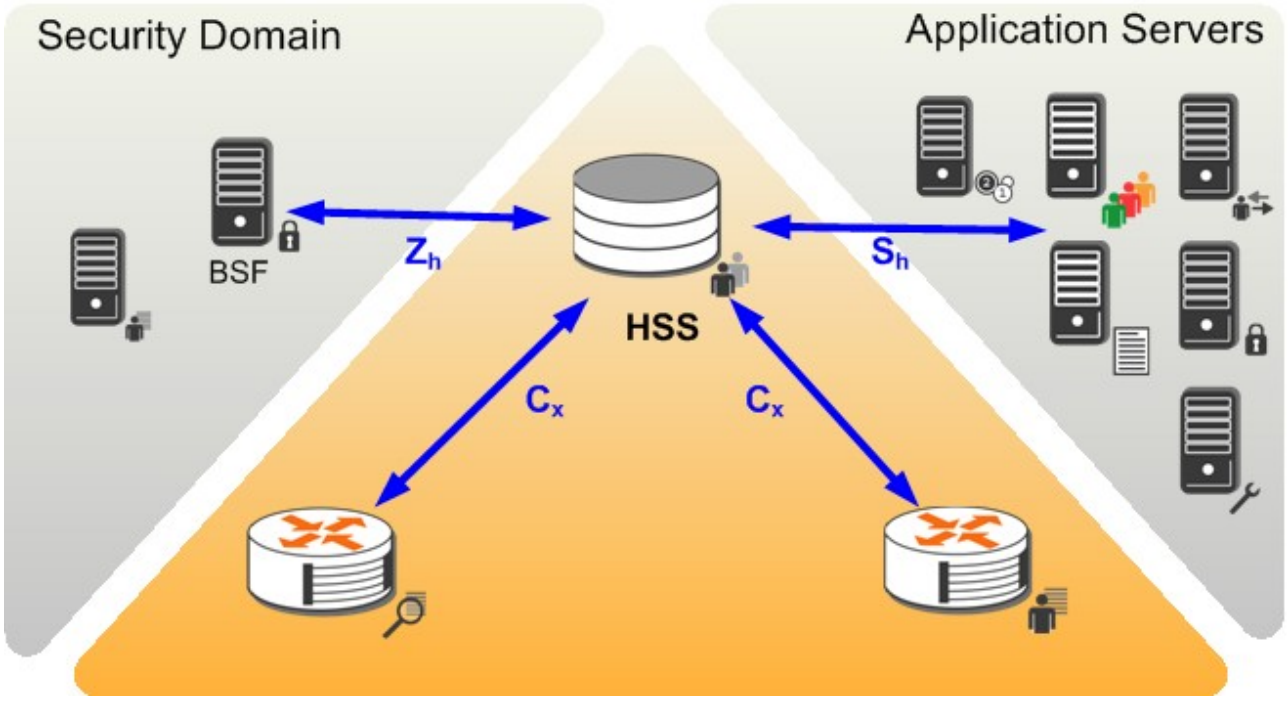
FHoSS server je aplikáciou napísanou v programovacom jazyku Java a z hľadiska architektúry sa skladá z troch vrstiev:

- vrstva sieťového rozhrania (*interface layer*)
- vrstva dátového prístupu (*data access layer*)
- vrstva používateľského rozhrania (*graphical user interface layer*)

Vrstva sieťového rozhrania je zodpovedná za prijímanie požiadaviek a odosielanie odpovedí na jednom z troch dostupných rozhraní, ktoré znázorňuje Obr. 15.2.

Vrstva dátového prístupu je postavená na systéme *Hibernate*, teda na implementácii architektúry relačného databázového systému poskytujúceho perzistentné triedy v Jave. Prostredníctvom tejto triedy FHoSS získava údaje z pridelenej MySQL databázy a poskytuje ich zvyšným dvom vrstvám.

Vrstva používateľského rozhrania používa systém *Apache Struts*, teda HTTP server, v ktorom je možné funkcionalitu web stránok programovať v jazyku Java. Toto rozhranie umožňuje správcom IMS siete spravovať používateľské údaje, definovať nové účty a iné rôzne operácie spojené s funkcionalitou HSS uzla.



Obr. 15.2: Rozhrania FHoSS servera [41]

Z technického hľadiska je HSS v rámci nášho projektu implementovaný na bežnom počítači. Nakoľko OpenIMS vyžaduje prostredie operačného systému Linux, bola na tomto uzle použitá minimálna inštalácia distribúcie Debian v najnovšej stabilnej verzii zvanej *Lenny*.

Postup inštalácie HSS uzla bol nasledovný:

1. Inštalácia operačného systému Debian Lenny so zvolením inštalácie základných balíčkov
2. Inštalácia balíčkov potrebných na prevádzku systému FHoSS – *make*, *open-jdk 1.5*, *ant*, *bison*, *flex*, *libxm2* a *openssl*
3. Inštalácia MySQL servera
4. Získanie najnovšej stabilnej verzie FHoSS servera prostredníctvom systému *Subversion* – repozitár sa nachádza na stránke

   http://svn.berlios.de/svnroot/repos/openimscore/FHoSS/trunk

   V čase písania tejto správy tu bola k dispozícii revízia 784.
5. Kompilácia zdrojových kódov FHoSS servera – prostredníctvom programu *ant*.
6. Import predpripravených nastavení systému a používateľov Alice a Bob do MySQL databázy – v našej revízii sa tieto nachádzali v adresári *scripts* a išlo o súbory *hss_db.sql*, *userdata.sql* a *icscf.sql*
7. Spustenie servera, overenie funkcionality

15.1.3 Internetová brána a DNS server

Ďalšie zariadenie v našej sieti slúži ako internetová brána a DNS server zároveň, no budeme ho zjednodušene označovať ako bránu. Funguje na linuxovej distribúcií Debian Lenny a plní v našej sieti tri úlohy:

- dostupnosť vzdialenej správy všetkých počítačov prostredníctvom protokolu SSH

- smerovanie riadiacej prevádzky ostatných uzlov v IMS sieti do siete Internet a preklad adries
- funkcionalita DNS servera v IMS sieti

Nakoľko je testovanie a implementácia IMS siete a optimalizovanej prístupovej siete možné až do fázy merania oddeliť, naša IMS sieť potrebovala počítač, ktorý by bol prostredníctvom vzdialenej správy dostupný z Internetu, a umožnil tak členom tímového projektu s IMS sieťou pracovať aj bez nutnosti fyzickej prítomnosti v laboratóriu. Keďže Internetové pripojenie dostupné v laboratóriu, v ktorom sa tento počítač nachádza, je chránené bránou firewall a s výnimkou tohto stroja sú všetky uzly v IMS sieti pripojené do samostatnej lokálnej siete bez prístupu na Internet, vzdialená správa ostatných prvkov v IMS sieti je možná iba vytváraním SSH spojení z tohto počítača po lokálnej sieti. Tento stroj je zároveň zodpovedný za smerovanie a preklad adries a pre všetky ostatné uzly v IMS sieti slúži zároveň ako predvolená brána pre prístup do siete Internet.

Pre efektívnejšie využitie zdrojov v sieti, ako aj diverzifikáciu služieb s cieľom umožniť ich modularitu bol na tento počítač umiestnený taktiež DNS server, ktorý umožňuje každému prvku v IMS sieti zistiť IP adresy ostatných služieb. Na tento účel sme použili najznámejší DNS server *Bind9*, získaný v najnovšej stabilnej verzii dostupnej z repozitárov distribúcie Debian. Zoznam DNS zón pre tento program sme získali úpravou predpripraveného zoznamu dodávaného so systémom OpenIMS.

Postup inštalácie všetkých služieb na tomto počítači bol nasledovný:

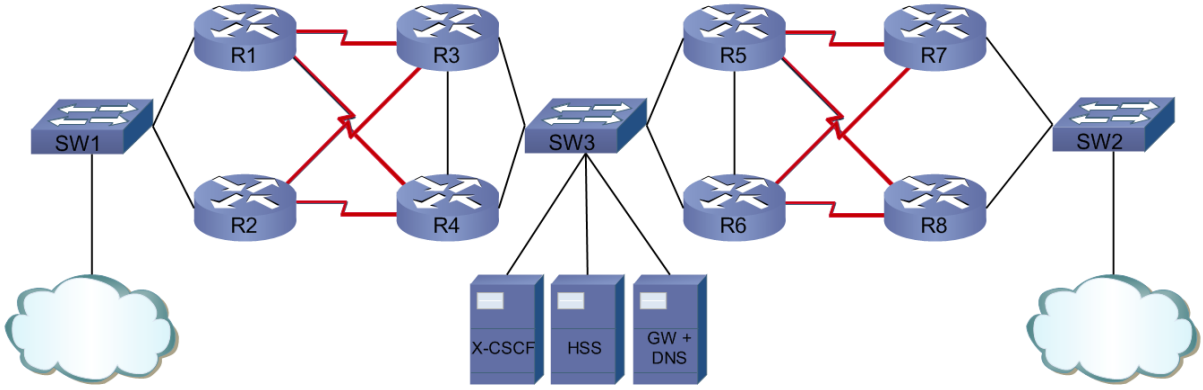
1. Inštalácia operačného systému Debian Lenny so zvolením inštalácie základných balíčkov
2. Inštalácia a konfigurácia brány firewall pre ochranu IMS siete pred nepovoleným prístupom zo siete Internet
3. Konfigurácia smerovania a prekladu adries
4. Inštalácia DNS servera *Bind9* z repozitárov distribúcie Debian Lenny vo verzii 9.5.1
5. Úprava DNS záznamov dodávaných spoločne so systémom OpenIMS a ich nasadenie v DNS serveri
6. Spustenie servera a overenie funkcionality

15.1.4 Asterisk

Za účelom zrýchlenia dostupnosti aspoň minimálnej sady služieb nad protokolom SIP sme vo fáze návrhu na jeden z dostupných počítačov nainštalovali distribúciu Trixbox, ktorá je špeciálne zameraná na poskytnutie služieb softvérovej ústredne Asterisk. Trixbox je postavený na linuxovej distribúcií Centos a okrem samotného programu Asterisk v sebe od inštalácie zahŕňa aj nástroj pre zjednodušenie jeho konfigurácie zvaný FreePBX, ako aj niekoľko ďalších nástrojov, ako správcu odkazovej schránky a podobne. Asterisk sme použili ako dočasnú náhradu za IMS sieť pri testovaní navrhnutého prototypu, nakoľko niektoré súčasti IMS siete ešte v čase testovania neboli v prevádzke. Vo fáze implementácie zadania môže tento server byť použitý ako aplikačný server, prípadne ako iný podporný prostriedok v IMS sieti.

15.2 Transportná sieť

Rozhodli sme sa pre implementáciu nasledovnej siete, ktorá sa nachádza na Obr. 15.3.



Obr. 15.3: Návrh transportnej siete

Cieľom siete je zaistiť dostatočnú redundanciu a rýchlu konvergenciu v prípade výpadku uzla siete alebo liniek medzi uzlami. Smerovače R1 až R4 by bolo možné nahradiť dvoma smerovačmi (napr. R1 a R3). Avšak v prípade výpadku jedného z týchto dvoch uzlov by sa úsek siete stal nedostupným. Redundanciou z obrázka zaistíme, že aj v prípade výpadku viacerých uzlov bude sieť schopná smerovania prevádzky.

Rozhodli sme sa pre dve brány do Internetu, resp. k poskytovateľom služieb – SP (*Service Provider*). Je vylúčené, aby operátor mal iba jeden fyzický prístup do siete WAN. Na obrázku túto časť siete máme znázornenú od SW1 až po „obláčik“. V skutočnosti ide o kritickú časť, kde by pri neželanom výpadku mohlo nastať prerušenie všetkých hovorov operátora.

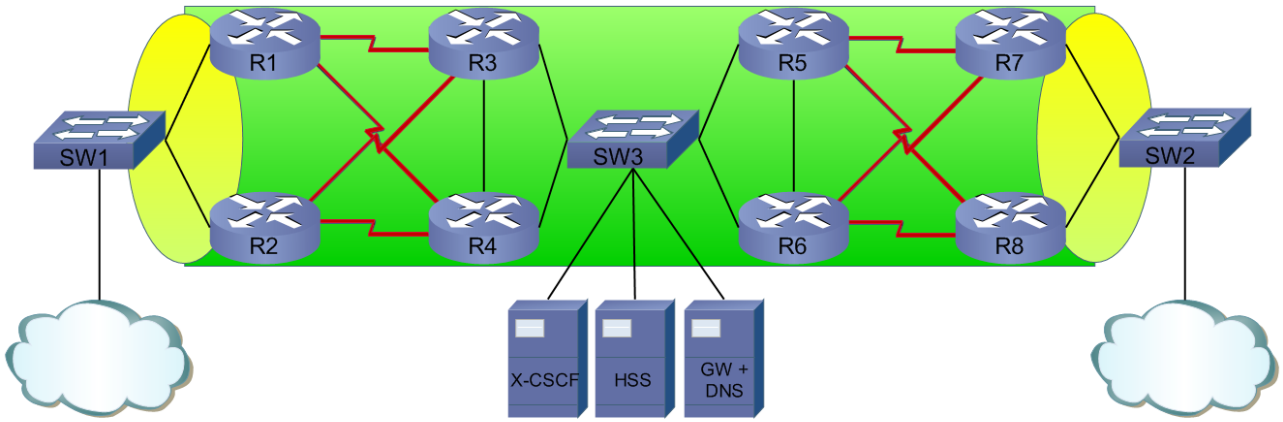
Druhé spojenie do siete WAN je realizované od {R5, R6} až po SW2. Vyťaženie oboch strán bude upresnené pri záťažových testoch. Je možné, že niektoré služby (prípadne používateľov) budeme preferovať jednou cestou a zvyšnú prevádzku budeme smerovať cez druhý prístup k WAN. Každopádne nám redundancia prístupu do WAN rieši problémy s poruchou na strane SP.

V strede topológie siete je jadro prevádzky, ktoré obsahuje uzly so službami. Topológia je navrhnutá univerzálne, teda v jadre môžeme mať ľubovoľný počet rôznych typov služieb.

V prípade implementácie siete operátorom by sme nahradili všetky sériové linky ethernetovými linkami s vysokou šírkou pásma (1 Gbps). V súčastnosti nám laboratórne podmienky zatiaľ neumožnili takúto sieť odsimulovať.

Nazdávame sa, že v našej skúšobnej prevádzke nám pomalšie linky môžu byť prínosom, keďže sme nimi schopní jednoduchšie simulovať správanie sa s preťaženými linkami a tak vyskúšať rôzne techniky QoS.

Na Obr. 15.4 je detailnejšie vidieť VRRP zóny, na ktorých sa uvedené zariadenia budú správať ako jeden logický uzol. Ide o žlté elipsy. Zelená zóna určuje našu sieť a naše zariadenia, ktoré budeme konfigurovať. Do tejto zóny sme zahrnuli len zariadenia, ktoré sú súčasťou transportnej siete.

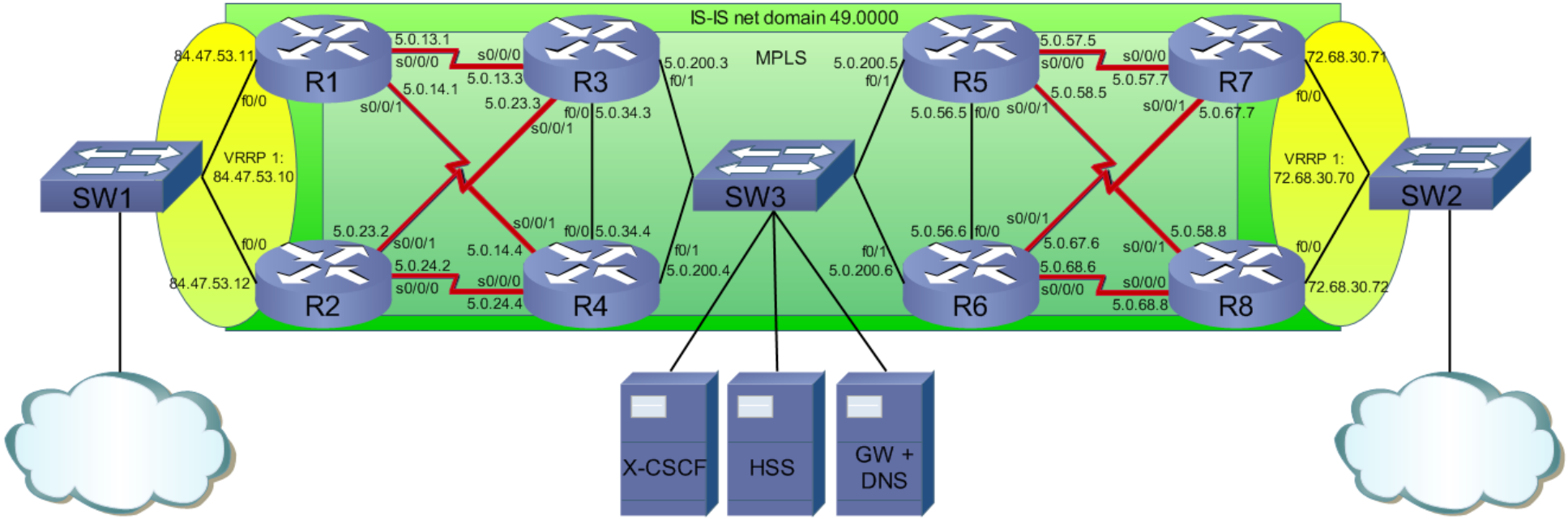


Obr. 15.4: Návrh transportnej siete so zvýraznenými VRRP zónami

Na Obr. 15.5 je vyhotovená celá schéma aj s IP adresami a fyzickým označením interfejsov. Interným sieťovým protokolom bude IS-IS kvôli jeho vysokým možnostiam optimalizácie. Narozdiel od protokolu OSPF ponúka lepšie možnosti optimalizácie, lepšie časy konvergencie a širšiu paletu možností optimalizácie pri MPLS TE.

V ďalších podkapitolách rozoberieme jednotlivé technológie a obohatíme ich o postup konfigurácie. Postup a konfigurácia sú zamerané na zariadenia od firmy Cisco, keďže sú ako jediné dostupné v laboratóriu. Na prípadné odchýlky u iných výrobcov priebežne upozorníme.

Ukážková konfigurácia smerovačov je prezentovaná tak, ako boli príkazy zadávané do smerovača. To znamená aj príkazy, ktoré sú implicitné a nezobrazujú sa v konfigurácií operujúceho zariadenia.



Obr. 15.5: Detailný návrh transportnej siete

15.2.1 Protokol VRRP

Protokol VRRP ako protokol pre zvýšenie dostupnosti prvého skoku pre koncové zariadenia sme zatiaľ použili na okrajoch siete – smerom k SP. Jeho konfigurácia je v Tabuľke 15.1.

Tabuľka 15.1: Konfigurácia protokolu VRRP

Konfiguračný príkaz	Popis príkazu
Router(config)# **interface** *type number*	Nastavenie VRRP na rozhraní smerovača
Router(config-if)# **vrrp** *group* **ip** *ipaddress*	Nastavenie IP adresy virtuálneho smerovača
Router(config-if)# **vrrp** *group* **priority** *level*	Zmena priority – úroveň 1-254
Router(config-if)# **vrrp** *group* **preempt** [**delay** *seconds*]	Preemptívne vymieňanie si rolí v prípade zmeny priority
Router(config-if)# **vrrp** *group* **timers advertise** [**msec**] *interval*	Konfigurácia časovačov

15.2.2 Protokol IS-IS

Prototyp, ktorý bol implementovaný v rámci prvotného návrhu siete, využíva smerovací protokol *Integrated* IS-IS. Tento protokol sme implementovali pre jeho excelentné optimalizačné vlastnosti. Tieto vlastnosti máme vedecky zdokumentované a podložené v [7].

V prototype bol tento protokol zavedený ako hlavný interný smerovací protokol. Implementovali sme ho s nastavenou sieťovou doménou OSI modelu 49.0000. Číslo 49 označuje privátnu doménu. Zároveň boli v prototype všetky väzby medzi susednými smerovačmi v rámci IS-IS protokolu ponechané s implicitným nastavením komunikácie. To znamená, že všetky smerovače komunikujú v oboch úrovniach architektúry IS-IS siete.

15.2.3 MPLS

V rámci aktuálneho prototypu bola implementovaná podpora MPLS smerovania. Všetky smerovače (R1 až R8) mali túto podporu aktivovanú. Aktivácia MPLS v prototype siete vyžaduje len niekoľko splnených požiadaviek. Prvou je podpora CEF (*Cisco Express Forwarding*) na zariadeniach spoločnosti Cisco Systems. Na zariadeniach od iných výrobcov existuje podobná funkcionalita. Druhou je implementovaná podpora pre protokol LDP (*Label Distribution Protocol*) v smerovačoch. Protokol LDP sa aktivuje implicitne po aktivovaní MPLS podpory na danom rozhraní a je potrebný na vybudovanie MPLS smerovania v rámci skonvergovanej topológie. Protokol LDP sme vybrali preto, lebo ide o otvorený štandard. A nezabudli sme zmeniť hodnotu MTU na MPLS aktívnych interfejsoch. Jej hodnota má byť rovnaká alebo väčšia ako typických 1500 bajtov pre ethernet v súčte so všetkými možnými MPLS hlavičkami prechádzajúcimi daným interfejsom.

Konfigurácia protokolu IS-IS a MPLS sa nachádza súhrnne v Tabuľke 15.2.

Tabuľka 15.2: Konfigurácia protokolu IS-IS a MPLS

Konfiguračný príkaz	Popis príkazu
Router(config)# **mpls ip**	Príkaz aktivuje MPLS smerovanie v rámci IOS smerovača
Router(config)# **mpls traffic-engeneering tunnels**	Príkaz aktivuje pre MPLS smerovania podporu MPLS TE
Router(config)# **router isis**	Aktivuje IS-IS protokol na danom smerovači a sprístupní jeho konfiguráciu
Router(config-router)# **net** *net_address*	Konfigurácia sieťovej adresy v rámci OSI adresácie
Router(config-router)# **metric-style wide**	Aktivuje rozšírenú metriku v IS-IS
Router(config-router)# **mpls traffic-eng router-id** *interface*	Nastavenie identifikátora smerovača v rámci MPLS TE v IS-IS
Router(config-router)# **mpls traffic-eng** *level*	Nastavenie prikazuje posielať informácie z MPLS TE pomocou spojení danej úrovne v IS-IS
Router(config-if)# **ip router isis**	Aktivuje smerovanie v IS-IS na danom rozhraní
Router(config-if)# **mpls ip**	Aktivuje podporu MPLS pre dané rozhranie
Router(config-if)# **mpls mtu** *bytes*	Zmení hodnotu MTU pre interfejs
Router(config-if)# **mpls traffic-eng tunnels**	Aktivuje podporu MPLS TE pre dané rozhranie
Router(config-if)# **ip rsvp bandwidth** *bandwidth* **sub-bandwidth** *sub-bandwidth*	Aktivuje RSVP na danom rozhraní a alokuje prenosové pásmo na danom rozhraní

15.3 Zhodnotenie

Podstatou hrubého návrhu a prototypu riešenia bolo vytvorenie modelovej topológie pre IMS sieť a transportnú sieť a vzájomne previazať obe súčasti. V súčasnej fáze implementácie existuje zapojená a funkčná transportná sieť so základnými konfiguračnými príkazmi. Sieť IMS je nakonfigurovaná a napojená do siete tak, aby mohla byť verifikovateľná dostupnosť. Prototyp riešenia svoj cieľ splnil.

V ďalšej fáze projektu budeme optimalizovať transportnú sieť tak, aby sme dosiahli lepšie časy konvergencie, zvýšili redundanciu a zvýšili dostupnosť uzlov. Vykonáme merania a porovnáme hodnoty s neoptimalizovanou sieťou. Do IMS zavedieme nové servery a nové služby, ako napríklad IPTV a budeme vyhodnocovať ich dostupnosť a kvalitu. V rámci celej siete aplikujeme kvalitu služieb. Celý proces podrobne zdokumentujeme.

16 Implementácia

Vzhľadom na to, že sme pred samotnou implementáciou pre potreby tímového projektu implementovali vybrané riešenia už o niečo skôr za účelom tvorby vedeckých publikácií, rozhodli sme sa, že dané výsledky tu uvedieme. Budeme vychádzať z článku „Optimization of Network Redundant Technologies Collaboration in IMS Carrier Topology “, ktorý bol zaslaný a prijatý na workshop ONIT 2010, ktorý sa bude konať v máji v Berlíne. Preklad tohto článku uvádzame ako našu prvú implementáciu v rámci tohto projektu. Článok bol primárne zameraný na sieťovú problematiku a riešili sme v ňom optimalizáciu len čiastočne.

Ďalej v tejto kapitole nasleduje ďalší prípad implementácie aj s výsledkami, ktoré boli naším cieľom pre tento projekt. V druhej implementácii sme sa viac zamerali na IMS a dôkladnú optimalizáciu všetkých aspektov tak, ako bolo uvedené v pláne projektu. Čitateľovi odporúčame spolupracovať s prílohami A a B, kde je bližšie uvedená inštalácia, konfigurácia a sieťová optimalizácia.

V nasledujúcej časti uvádzame prepis článku „Optimization of Network Redundant Technologies Collaboration in IMS Carrier Topology “ s miernymi úpravami pre potreby tohto projektu. Ponechali sme originálne obrázky a grafy.

16.1 Optimalizácia spolupráce redundantných sieťových technológií založených na architektúre IMS

16.1.1 Úvod k implementácii

Technológia TE (*Traffic Engineering*) použitá v IMS sieťach je v súčasnosti aktuálna téma hlavne pre sieťových operátorov, ktorí vyžadujú rozšírené riadenie kvality služieb. Kombináciou IP SLA so sledovaním objektov (*Object Tracking*) a MPLS TE môžeme vytvoriť automatické riešenie pre aplikovanie nových pravidiel pre nosnú topológiu poskytovateľov služieb.

IP SLA poskytuje príležitosť na sledovanie špecifických parametrov liniek a zariadení. V tejto časti sa zameriame na optimalizáciu konvergencie a distribúcie záťaže medzi existujúcimi linkami v sieti, ktoré je vykonávané automaticky. V dnešnej dobe pracujú podobné riešenia hlavne manuálne. Popísali sme inovatívne riešenie, ktoré hľadá optimálne riešenie automaticky - bez potreby zásahu administrátorov.

16.1.2 Motivácia k implementácii

Architektúra IMS vytvára hybnú silu vo výskume telekomunikačných technológií a dátových sietí. Rovnako, ako sa dva oddelené svety spájajú do jedného spoločného prostredia, tak existujú viac ako postačujúce náležitosti, pre ktoré by operátori chceli zaistiť hladké začlenenie architektúry IMS do jadier svojich sietí. V našom výskume sme sa zamerali na základné operácie zaisťujúce chod údajov v smerovacích procesoch v oblasti jadra architektúry IMS.

V poslednej generácii telekomunikačných sietí sa nachádzajú niektoré, v súčasnosti bežné črty, ako napríklad kvalita služieb a vyrovnávanie záťaže. Tieto črty by mohli byť natívne pre celú

sieť. Tieto črty sú vysoko prirodzené pre dátové siete a čiastočne aj pre IP siete, ktoré sú smerované najkratšou cestou. Tento prístup vytvára obmedzenia na dostupnosť týchto sietí, na využitie celej dostupnej šírky pásma a využitie iných ciest, ako tie, ktoré sú vybrané za najlepšie (z pohľadu najkratšej cesty k cieľu).

Na tieto obmedzenia sa v súčasnosti zameriavajú výskumníci po celom svete. Začínajú nadväzovať na koncept TE, ktorý bol prezentovaný už začiatkom 90-tych rokov. Táto relatívne stará technológia sa začala používať na nové prostredia, čiastočne aj s pomocou MPLS. Je možné tieto techniky spojiť dokopy. V prípade takéhoto spojenia môžeme konštatovať, že TE je forma manipulácie prevádzky, ktorá sa hodí pre typy sieťových topológií, na ktoré sa v našej práci zameriavame.

Táto časť sa zameriava na TE spolu s MPLS v prostredí architektúry IMS. Spojenie TE s MPLS býva často označované ako tzv. MPLS TE (*Multi-Protocol Label Switching Traffic Engineering*). V našom prístupe kombinujeme riešenia Cisco IP SLA so sledovaním prevádzky a MPLS TE. Týmto spojením vytvoríme jedinečné automatické riešenie na aplikáciu nových pravidiel na zmenu priority (napríklad v prípadoch výpadku liniek alebo vyťaženia liniek). IP SLA poskytuje možnosti sledovania špecifických parametrov liniek a zariadení. Následne, tieto výsledky môžu byť aplikované na sledovanie objektov pre vytváranie vstupov, ktoré budú aplikované do smerovacej tabuľky na základe špecifickej udalosti.

Náš hlavný cieľ je optimalizovať konvergenciu a vytvoriť vhodné vyrovnávanie záťaže medzi existujúcimi linkami v sieti. Náš prístup zvyšuje dostupnosť služieb, celkovú kvalitu služieb a snaží sa pritom uspokojiť požiadavky SLA (*Service Level Agreement*) medzi zákazníkom a poskytovateľom služieb.

16.1.3 Prehľad existujúcich riešení implementácie

TE sme použili na vyriešenie základných problémov zobrazených na Obr. 16.1. V tomto konkrétom príklade, všetky linky sú typu OC-3 s približnou šírkou pásma 150 Mbit/s. Ďalej uvažujme, že vieme, že smerovač R1 posiela údaje rýchlosťou 90 Mbit/s do smerovača R6, a zároveň smerovač R7 posiela údaje zvyšnou rýchlosťou 80 Mbit/s na smerovač R6.

Pokiaľ by sme uvažovali použitie techniky klasického výberu najkratšej cesty, tak R2 má linku do R5 cez najbližší uzol R6. Výsledkom takéhoto nastavenia bude zahltenie na linkách medzi smerovačmi R2 a R5. Samozrejme, alternatívne linky cez cestu R2-R3-R4-R6 ostávajú pod plnou záťažou. Je tu možnosť použitia TE pomocou manipulácie s cenami. To by malo za následok vyvažovanie všetkých alternatívnych ciest a potom vyrovnávanie záťaže medzi týmito cestami.

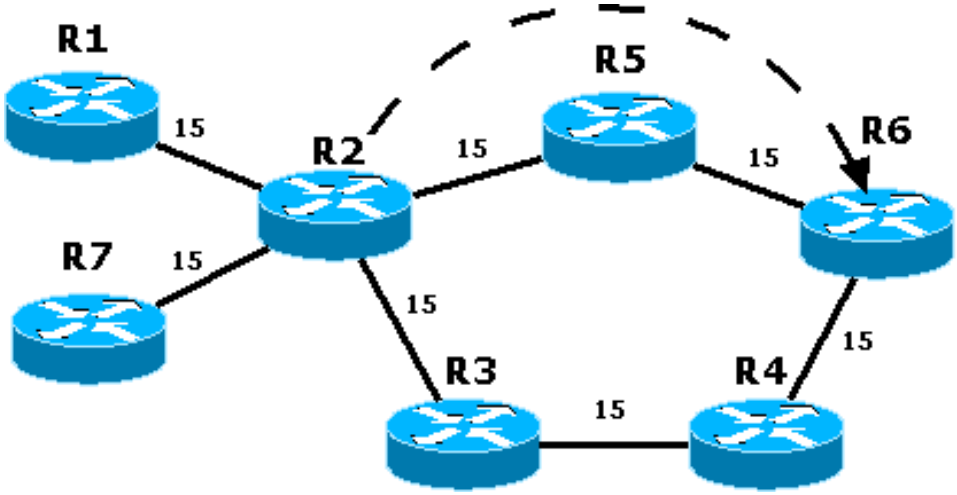
Toto riešenie je použiteľné pre malé siete, ale v prípade nasadení vo veľkých sieťových topológiách sa môže stať problematickým. Lepší prístup je pomocou techniky vyrovnávania záťaže (*Load Sharing*), ktoré lepšie reflektuje rôzne linkové informácie nachádzajúce sa medzi cestami (napr. šírka pásma).

Ďalšia použiteľná alternatíva je aj technológia ATM (*Asychronous Transfer Mode*), kde sa vytvárajú permanentné virtuálne okruhy tzv. PVC (*Permanent Virtual Circuits*) medzi koncovými bodmi. Zaťaženie potom môže byť zdieľané medzi permanentnými virtuálnymi okruhmi. Aby bola táto technológia použiteľná, je vyžadované, aby nebolo nevhodne manipulované s cenami liniek.

IP SLA je špecifická funkcia vyvíjaná firmou Cisco na podporu monitorovania špecifických parametrov. IP SLA dokáže monitorovať rozdielne obmedzenia uzla, linky alebo cesty zo smerovačov pomocou využitia vhodných akcií a tiež informovaním administrátorov pomocou

protokolu SNMP (*Simple Network Manegement Protocol*). IP SLA je nástroj na uspokojenie ustanovených obmedzení v SLA [42].

Ďalšia špecifická črta firmy Cisco, ktorú môžeme použiť, sa nazýva sledovanie objektov. Pomocou sledovania objektov môžeme monitorovať také objekty, ako napr. IP SLA, stav rozhrania, stav IP adresy (pod tým sa myslí, či je dostupná alebo nedostupná v smerovacej tabuľke), výskyt cieľovej siete v smerovacej tabuľke, výpočet metriky v ceste a pod. Môžu byť vytvorené aj kompozitné objekty tam, kde ostatné objekty sú spojené cez boolovskú logiku alebo cez hraničný systém [43].



Obr. 16.1: Príklad IP siete s potenciálom pre využitie TE

Kombinácia vlastností manažmentu ATM cez PVC a rozšíriteľnosť infraštruktúry IP vyústila v MPLS sieťach do MPLS TE. MPLS umožňuje zmeniť PDU (*Protocol Data Unit*) a tak umožní mať pre nie najspodnejšie návestia v zásobníku návestí iné ako smerovacie účely. Najväčšie využitia pre tieto návestia v jednom PDU sú pre VPN siete a pre identifikáciu tunelov TE.

Avšak tunely sú stále vytvorené prevažne manuálne ako súčasť sieťového dizajnu. Pridanie nových tunelov TE do siete môže byť vykonané pomocou strategického prístupu vytváraním sietí, ktorých logická topológia je úplný graf TE tunelov. Druhá možnosť je taktický prístup cez monitorovanie záťaže liniek s pridaním TE tunelov, ak by to bolo potrebné [44, 45].

Existuje tiež prístup založený na triedach prémiových služieb v diferencovaných službách (*DiffServ*) nad MPLS sieťou. Výhoda tohto riešenia je aj v implementácii celého riešenia ako *frameworku*. Avšak je vyžadované nutne použiť diferencované služby a merané sú iba vybrané parametre. My by sme radi sledovali desiatky rozdielnych parametrov, čo tento prístup neumožňuje. Obmedzuje sa len na diferencované služby a MPLS. Závislosť na diferencovaných službách nie je vhodná, ak nechceme byť striktne limitovaní na architektúru kvality služieb QoS.

Je tu taktiež práca založená na doručovaní kvality služieb v NGN sieťach. Tento článok prezentuje využitie kvality služieb v NGN sieťach pre aplikácie koncových používateľov. Zaoberá sa viacerými konceptmi pre umožnenie kontroly úrovní QoS. Chceli by sme prezentovať automatický prístup, neobmedzovať sa na pevnú klasifikáciu a značkovanie kvality služieb, čo nám táto práca neumožňuje.

Existuje práca, ktorej základ tvorí modelovanie a simulácia agregácie sieťovej prevádzky cez MPLS siete. Táto práca sa bližšie zameriava na ustanovenie hovoru na SIP a operácie so SIP_om. My sa nesústredíme iba na signalizáciu SIP_u, ale aj na pôsob prepravy RTP obsahu a taktiež iného mediálneho obsahu.

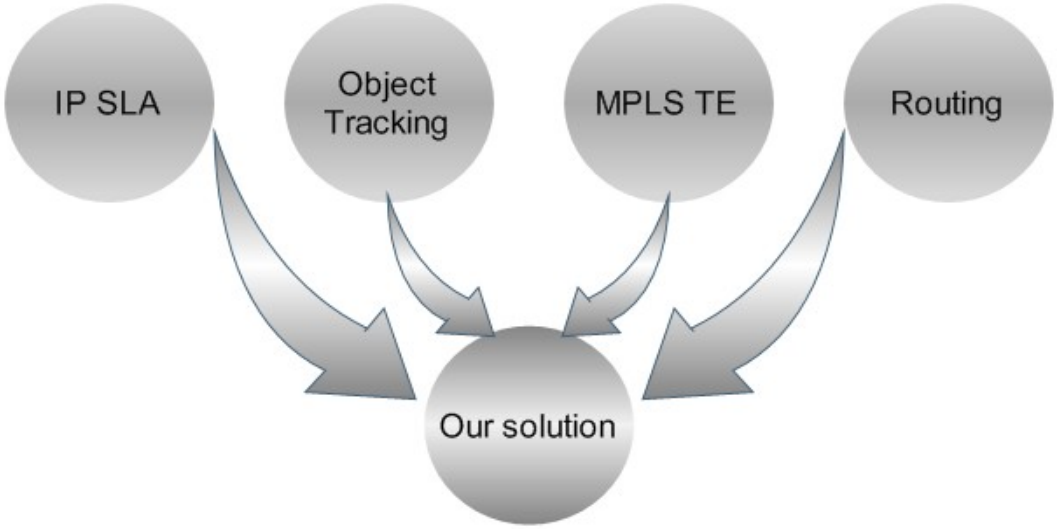
16.1.4 Navrhované riešenie

Naše monitorovanie liniek a zariadení je vykonané pomocou špecializovaných aplikácií, z najväčšej časti pomocou protokolu SNMP. Môžeme využiť IP SLA na sledovanie parametrov liniek, ktoré bude informovať administrátorov cez protokol SNMP. Máme k dispozícii veľké množstvo parametrov v rôznych vzájomných kombináciách použitia, ktoré môžu byť aktívne monitorované zo smerovačov. Je veľa možností, kedy a akým spôsobom informovať administrátora. Nastáva tu ale problém s tým, že čas medzi poslaním zachytávanej správy, jej prijatím a čítaním je príliš dlhý. Pokiaľ nemáme vytvorený žiadny záložný plán, môžu ubehnúť minúty. Naším ďalším úsilím bude automatizácia tohto procesu založenom na protokole SNMP.

Naše navrhované riešenie je popísané aj pomocou Obr. 16.2. Kombinujeme IP SLA so sledovaním objektov a zároveň aj sledovanie objektov so statickými cestami, ktoré sú definované v smerovacej tabuľke. Zameriavame sa na tunely MPLS TE. Položili sme si iba jednu nevyhnutnú podmienku - TE tunely sú definované staticky v smerovacej tabuľke. Monitorované parametre spolu s IP SLA sú mapované na sledovanie objektov pomerom 1:1.

V našom riešení vytvárame kompozitné sledované objekty, ktoré menia svoj stav podľa viacerých podmienok, ktoré sa stretli v rôznych sledovaných objektoch. Takýto kompozitný objekt je mapovaný so statickou cestou. Pokiaľ sú zistené niektoré kritické hodnoty, tak sledovaný objekt zmení svoj stav automaticky. Takže potom, čo kompozitné objekty zmenia svoj stav, takisto aj statická cesta zmení svoj stav. Pokiaľ vypadne statická cesta, nahradia ju ostatné statické cesty s horšou prioritou, alebo prípadne dynamické cesty.

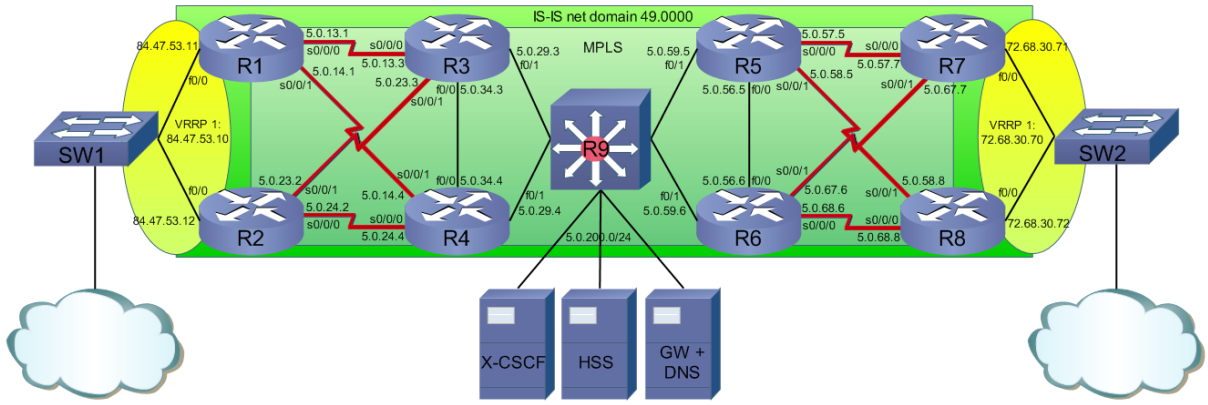
Navrhovaný prístup predstavuje jedinečné automatické riešenie. Záložný plán sa použije automaticky v prípade nutnosti. Problémy môžu nastať so zvyšovaním počtu objektov IP SLA. Tieto objekty zvyšujú šírku pásma a zaťaženie CPU. Dokonca aj záložný plán musí byť pripravený ako plnohodnotná súčasť sieťového návrhu. Stále, v prípade potreby môžu byť administrátorom zasielané aj SNMP správy a taktiež administrátori môžu manuálne zmeniť politiku siete, ak to považujú za vhodné.



Obr. 16.2: Navrhnuté riešenie: kombinácia IP SLA, sledovania objektov, MPLS TE a smerovania

16.1.5 Navrhnutá testovacia architektúra

Na Obr. 16.3 prezentujeme našu architektúru jadra IMS zapojenú do redundantnej siete operátora. Sieť operátora sa skladá z ôsmych smerovačov s viacerými redundantnými cestami. Máme dva výstupné body sieťovej topológie hlavne kvôli simulácii prechodu cez sieť operátora. Jeden výstupný bod je na kraji vľavo, druhý je na kraji vpravo. V centre sieťovej topológie sa nachádza jadro IMS siete.



Obr. 16.3: Naša testovacia schéma

Máme dva MPLS TE tunely nastavené na smerovačoch R1, R2, R7 a R8. Tieto tunely sú vytvorené naprieč našou testovacou sieťou, z jedného koncového bodu do druhého. Jeden tunel je umiestnený na smerovači R1 a vedie cez R3, R5 a R7. Druhý tunel je umiestnený na smerovači R1 a vedie cez R4, R6 do R8.

Na ostatných smerovačoch sú umiestnené tunely podobným spôsobom. Prvý tunel označme ako primárny a druhý ako sekundárny tunel. Primárny tunel je umiestnený medzi vrchnými smerovačmi (R1, R3, R5, R7). Vzhľadom k tomu, že ide o sieť operátora, tak predpokladáme, že medzi výstupnými bodmi bude nakonfigurovaný protokol eBGP. Primárny tunel je umiestnený v smerovacej tabuľke staticky s danou preferenciou. Primárny aj sekundárny tunel sa učia informácie o cestách pomocou dynamického smerovacieho protokolu, v našom prípade bol použitý protokol IS-IS s nízkou preferenciou.

Statický primárny tunel je sledovaný pomocou objektu. Všetka prevádzka je určená do sietí, ktoré sa nachádzajú za koncovými bodmi siete, pomocou statických tunelov. Pokiaľ sa stretlo niekoľko podmienok, tak vypadne sledovaný objekt, čo vedie k vymazaniu statickej cesty zo smerovacej tabuľky. Po odstránení tejto statickej cesty zo smerovacej tabuľky, bude umiestnený dynamicky naučený primárny aj sekundárny tunel do smerovacej tabuľky. Potom bude fungovať sieťová prevádzka na princípe vyrovnávania záťaže.

Pre účely nášho testovania bude sieťová prevádzka vstupovať iba do smerovača R1 a vystupovať bude cez koncový bod za smerovačmi R7 a R8 (viď Obr. 16.3). Predpokladáme, že zákazníci sú už registrovaní a že sa začal iniciovať hlasový alebo video hovor. Jeden zo zákazníkov sa nachádza za ľavým koncovým bodom a jeden za pravým koncovým bodom siete. Využitie šírky pásma sa zvyšuje po inicializácii hovorov, čo vedie k zvýšeniu hodnoty RTT (*Round Trip Time*). Kvôli fyzicky zapojeným linkám medzi smerovačmi so šírkou pásma iba 128 kbit/s, môže byť uskutočnený len jeden hovor bez posunutia kvality za hranu únosnosti. Po inicializácii druhého hovoru sa zvýši hodnota RTT, nastane *jitter* (variabilné oneskorenie) a strata paketov. Medzi dvoma hovormi dochádza k rovnakému postihnutiu kvality.

Pre účely nášho druhého merania sme konfigurovali objekty IP SLA. Máme vybraté hodnoty RTT a piemerný *jitter* pre sledovanie IP SLA objektov. V prvom IP SLA objekte konfigurujeme typ paketu *icmp-echo* s hodnotou TOS na 184, čo je decimálna reprezentácia triedy EF. Pre naše testovacie účely je hranica (*threshold*) nastavená na 20 ms. Frekvencia posielania týchto paketov a kontrolovanie kvality linky je jedna sekunda. Druhý objekt IP SLA je konfigurovaný rovnakým spôsobom ako bol konfigurovaný prvý objekt. Hraničná hodnota je nastavená na 4 ms. Táto hodnota je stanovená pre podmienky našich meraní. Reakcia je konfigurovaná na priemerný *jitter* s hornou hranicou 4 ms a spodnou hranicou 3 ms. Ak sú prekročené hraničné limity, tak okamžite je vykonaná príslušná akcia.

Každý IP SLA objekt je mapovaný na svoj vlastný jedinečný objekt v technológii sledovania objektov (1 a 2). Jeden kompozitný objekt, vytvorený s logickou booleovskou hodnotou, je navrhnutý ako objekt 3. Keď je ľubovoľný objekt mimo prevádzky, tak je celý kompozitný objekt mimo prevádzky. Tento kompozitný objekt je použitý na konfiguráciu statickej cesty. Sledované objekty 1 a 2 sú oneskorené. Ak nie sú oneskorené a ak jeden z IP SLA objektov zlyhá v teste, tak je okamžite vykonaná príslušná akcia. Oneskorujeme stavy „zapnutý“ a „mimo prevádzky“ trojnásobkom frekvencie objektov IP SLA. Ak zlyhá test trikrát v rade, tak je sledovaný objekt považovaný za vypnutý. Rovnaké pravidlo sa aplikuje pre stav „zapnutý“.

16.1.6 Výsledky meraní

Behom meraní sme uskutočnili prvý hovor. Šírka pásma tunela bola dostatočná pre presne jeden hovor s akceptovateľnou hodnotou RTT a *jitter*_ovou charakteristikou pre A/S rozhrania o šírke pásma 128 kbit/s tak, ako je ukázané v Tabuľke 16.1.

Potom sme uskutočnili druhý hovor. Keď bol ustanovený druhý hovor, tak namerané hodnoty RTT konštantne rástli (viď Obr. 16.4). Po 4 sekundách oboch hovorov začína rapídne klesať kvalita hovoru až za akceptovateľnú hranicu (viď Tabuľka 16.1).

Po zavedení optimalizácie je sieť schopná detegovať klesanie kvality hovoru a dynamicky prepnúť prevádzku v TE kvôli rastúcim požiadavkám pre priepustnosť siete. V prípade optimalizovaného prostredia, po ustanovení druhého hovoru a zhoršení hovorovej charakteristiky, sa vykoná príslušné meranie IP SLA. Objekt 1 zlyhá okamžite v ďalšom testovanom intervale. Pre nastávajúce tri sekundy bude objekt 1 prinútený k tomu, aby bol oneskorený pred výsledným zmenením jeho stavu. Tento postup slúži ako ochrana proti tomu, aby nenastala predčasne zmena na zálohu tunelu.

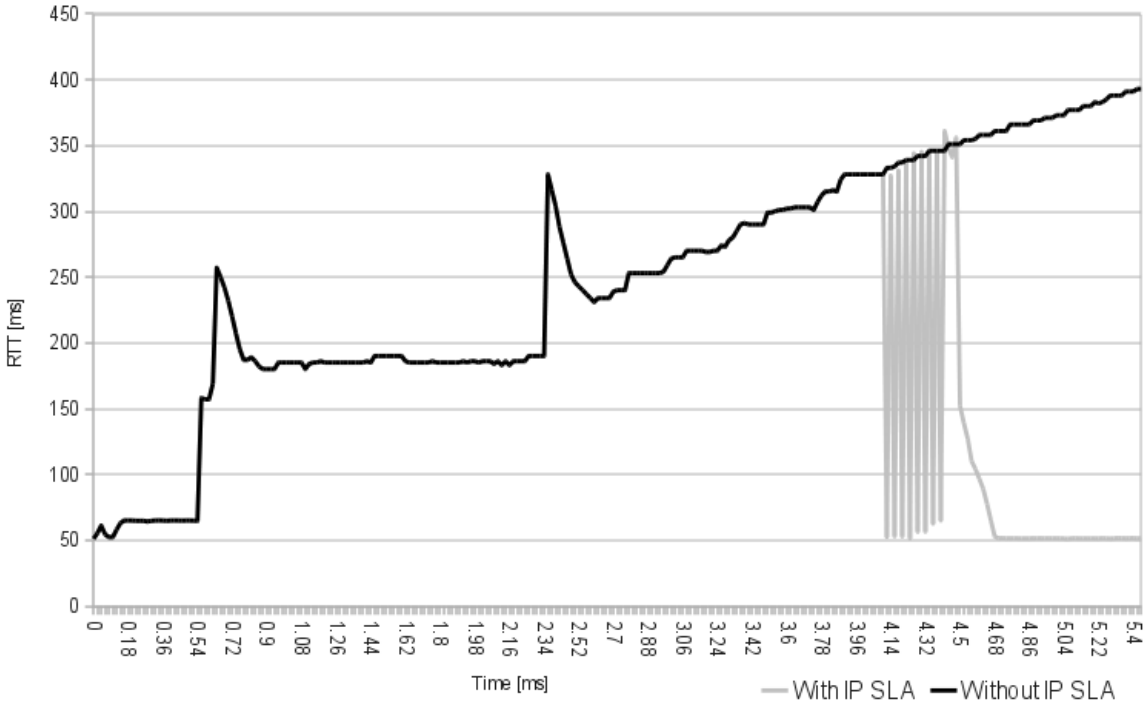
Potom, ako expiroval tento časovač, tak sa objekt 1 zmenil na stav „vypnutý“, a zároveň sa objekt 3 zmenil na stav „mimo prevádzky“. Toto viedlo k zmazaniu príslušných statických ciest v smerovacej tabuľke. Dynamické cesty sa okamžite aktivovali, vyúsťujúc do vytvorenia rovnakého primárneho tunela a sekundárneho tunela. V takomto nastavení si smerovače môžu medzi týmito dvoma tunelmi začať vyrovnávať záťaž. V nasledujúcom čase 0.5 sekundy, daným prostredím pre vyrovnávanie záťaže medzi dvoma udržiavanými hovormi, sa vrátila kvalita oboch hovorov na prípustnú úroveň. Objekt IP SLA stále drží oba tunely aktívne, pretože s riešením vyrovnávania záťaže ostáva *jitter*_ová charakteristika v uzle nad prípustnou úrovňou. Z toho dôvodu IP SLA samo zistilo, že sa je možné vrátiť na riešenie jedného tunela. Počas našich experimentov sa nevyskytli žiadne negatívne efekty.

Tabuľka 16.1: Správanie sa RTP s a bez proaktívnej zálohy pri zaťažení tunela

	RTT [ms]		Jitter [ms]
	Bez IP SLA	*S IP SLA*	*Bez IP SLA*
1 hovor	51,4	51,4	1
2 hovory (0-4 s)	~305	~305	1
2 hovory (po 4 s)	>2000	51,4	4-6

Na Obr. 16.4 je graf meraných hodnôt RTT. Zaznamenávanie hodnôt začalo v čase, keď bol uskutočnený druhý hovor. Náš optimalizovaný systém potreboval čas okolo 4 sekúnd, na detekciu, propagovanie, počítanie a aktualizáciu preposielacej informačnej bázy, tzv. FIB (*Forwarding Information Base*), hlavne na zvýšenie požiadaviek priepustnosti behom vyrovnávania záťaže tunelov.

Po približne 4 sekundách bol náš systém schopný dosiahnuť akceptovateľnú kvalitu hovoru. Poznamenajme, že v čase 0.5 sekundy po 4. sekunde mali hodnoty RTT najväčší rozptyl po aplikovaní našej konfigurácie. V tomto čase prichádzajú oneskorené odpovede a retransmisie zo sekvencie paketov a zapríčiňujú tieto „skákajúce“ hodnoty idúce z hornej do normálnej úrovne hodnôt RTT. Tieto pakety boli zväčša zničené VoIP klientmi. Po 4.5. sekunde sú klienti schopní komunikovať s očakávanou kvalitou. Bez IP SLA sa môžu hodnoty RTT konštantne zvýšiť až na 2000 ms (viď Tabuľka 16.1).



Obr. 16.4: Porovnanie hodnôt RTT v čase pre systém s a bez IP SLA

16.2 Implementácia druhej testovacej topológie

V druhej testovacej topológii sme vychádzali primárne z nášho prototypu. Mierne sme upravili schému a tú uvádzame na Obr. 16.5. Hlavnou zmenou je, že sme do stredu siete pridali

prepínač na tretej vrstve. Ten slúži ako prepínač pre IMS sieť a ako smerovač pre ostatné susedné zariadenia. Serial linky sú nakonfigurované ako 128 kbit/s.

Cieľom testov bola štúdia konvergencii siete, počty stratených paketov a kvalita hovoru pri rôznych konšteláciách výpadkov v sieti. Testovali sme čas konvergencie siete (KS) pri výpadku linky, merali sme počet stratených paketov (SP) a subjektívnu kvalitu hovoru. Kvalitu hovoru vyhodnocovali kolegovia vzájomnou komunikáciou. Následne počítali, koľko sekúnd nebolo druhú stranu počuť a aký dlhý čas bol hovor nekvalitný, až priam nezrozumiteľný.

Meranie sme vykonali na tejto topológii v dvoch sadách. Prvá sada je určená pre neoptimalizovanú sieť a druhá sada pre optimalizovanú sieť. Vychádzame z návrhu a teda klasifikujeme zákazníkov a dátové toky na Gold, Silver a ostatné dáta. V neoptimalizovanej sieti neuvažujeme o rozdelení zákazníkov do tried, v optimalizovanej časti už áno.

V prvej sade sme navrhli dovedna šesť testov. V druhej sade sme vykonali až 10 testov. Pri testoch sme zaťažili sieť klasickým SIP hovorom prostredníctvom IMS klientov. Hovor prebiehal z ľavej okrajovej strany na pravú okrajovú stranu našej siete, čiže naprieč celou našou transportnou sieťou. Registrácia prebiehala smerovaním paketov do centra našej siete, do IMS. Pri niektorých testoch sme si vypomohli programom SIPp, ktorý slúži na generovanie SIP dátovej komunikácie a tým sme simulovali viacero hovorov. SIPp je bližšie popísaný v prílohe B. Počas všetkých testoch sme boli pripojení s testovacím notebookom na pravý okraj siete a používali sme ping v špeciálnom formáte tak, ako sme to uviedli v návrhu. Ping bol smerovaný na IMS klienta na ľavom okraji siete. Ping nám slúži na zaznamenávanie a vyhodnocovanie sledovaných parametrov KS a SP pre sieť z pohľadu koncových zákazníkov. Ping bol odosielaný v rozostupoch 30 ms, pretože aj IMS klient vysiela pakety v týchto rozostupoch. Paralelne sme zachytávali komunikáciu programom Wireshark na oboch IMS klientoch, ktorý si zavolali.

V prvej sade sme vykonali šesť testov. Testy sme zhrnuli do tabuľky 16.2. Všetky testy obsahovali práve jeden SIP hovor a náš ping. Menili sa len fyzické výpadky fyzickým odpojením kábla a počty SIPp hovorov.

- Test 1 predstavuje jeden SIP hovor, 0 SIPp hovorov a ping bez výpadkov.
- Test 2 prestavoval to isté, len nastal výpadok R2 rozhrania f0/0.
- V treťom teste nastal výpadok medzi prepínačom a smerovačom R3.
- V štvrtom teste nastalo rozpojenie medzi smerovačmi R3 a R4.
- V piatom teste pribudol jeden SIPp hovor a vypojili sme rozhranie f0/0 na R1.
- V šiestom teste sme taktiež použili jeden SIPp hovor a nastal výpadok na R2 rozhrania f0/0.

Výpadok SIP hovoru a oneskorenie SIP sú udávané ako subjektívne, čas udáva posun oproti vyslovenému a počutému v klientovi. Hovor bol po obnovení výpadku rušený, nekvalitný po dobu 1-2 sekúnd. Následne sa ustálil do pôvodnej kvality. Hodnoty RTT pingu subjektívny pocit potvrdzuje.

Po prepočítaní hodnôt RTT na počet paketov sa zistilo, že štandardné oneskorenie bolo 0.1 sekundy. Pri zahltení (SIP hovor, SIPp hovor a ping) to bolo 0.2 sekúnd, ale po skonvergovaní to skákalo od 30 ms. do 1500 ms. po dobu niekoľkých sekúnd po skonvergovaní a následne sa to ustálilo. Subjektívne hodnoty boli o niečo vyššie ako skutočne namerané, čo neprekvapuje, keďže konvergencia siete sa končí na sieťovej vrstve, pričom klienti konvergujú až na aplikačnej úrovni. Takisto treba započítať schopnosti klienta a najmä použitého kodeku pri hovoroch, ktoré tiež robia posun v čase. SIPp aj použitý Mercuro IMS klient používajú kodek G.711 PCMU.

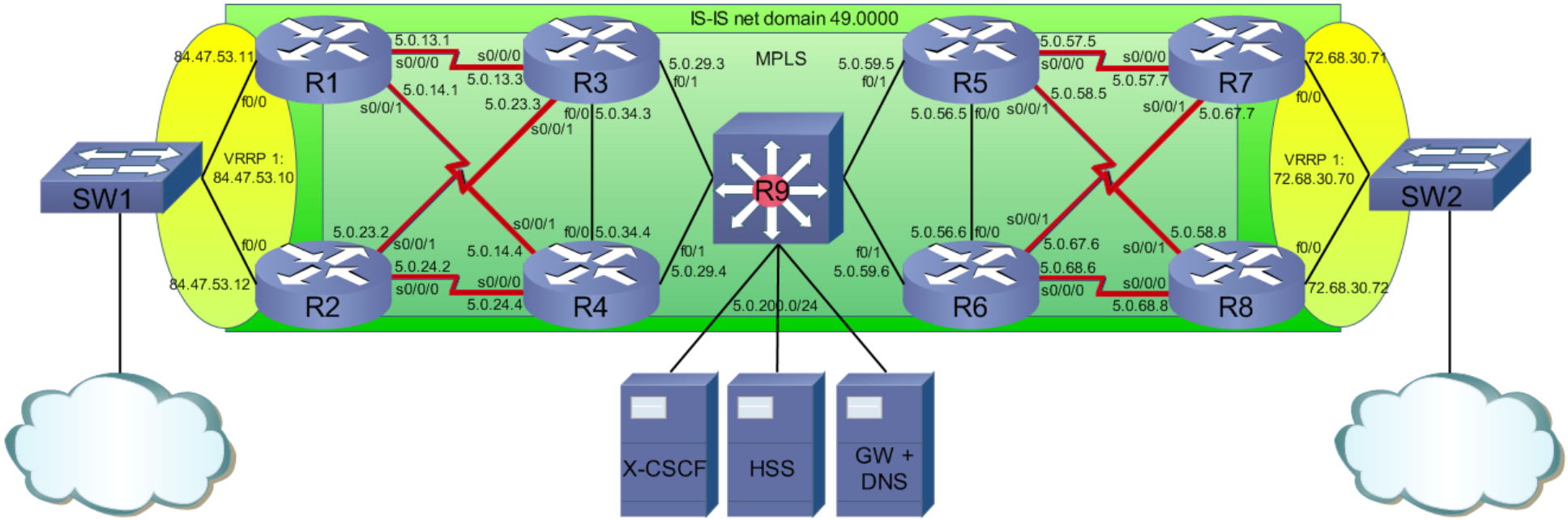
V Tabuľke 16.3 popisujeme druhú sadu testov, ktorá predstavuje optimalizovanú sieť. Optimalizácia prebehla predovšetkým na transportnej sieti. Konfigurácia sa nachádza na elektronickom nosiči a detailný popis príkazov sa nachádza v prílohe A. Na strane IMS sme vytvorili aplikačné servery pre dve triedy zákazníkov Gold a Silver. Všetky potrebné nastavenia sú v prílohe B.

V druhej sade sme vykonali až desať testov. Prvých šesť testov je zhodných so sadou 1 pre lepšie porovnanie. Všetky testy obsahovali práve jeden Silver SIP hovor a náš ping. Menili sa len fyzické výpadky fyzickým odpojením kábla a počty Gold SIPp hovorov.

- Test 1 predstavuje jeden Silver SIP hovor, 0 Gold SIPp hovorov a ping bez výpadkov.
- Test 2 prestavoval to isté, len nastal výpadok R2 rozhrania f0/0.
- V treťom teste nastal výpadok medzi prepínačom a smerovačom R3.
- V štvrtom teste nastalo rozpojenie medzi smerovačmi R3 a R4.
- V piatom teste pribudlo veľa Gold SIPp hovorov a vypojili sme rozhranie f0/0 na R1.
- V šiestom teste sme taktiež použili veľa Gold SIPp hovorov a nastal výpadok na R2 rozhrania f0/0.
- V teste 7 sme mali 1 Silver SIP hovor, 0 SIPp hovorov a ping, bol vykonaný výpadok na R8, fyzické odpojenie oboch serial káblov z R8.
- V teste 8 sme mali 1 Silver SIP hovor, 0 SIPp hovorov a ping; výpadok medzi R8 a R7, fyzické odpojenie kábla z R8.
- V teste 9 sme mali 1 Silver SIP hovor, 0 SIPp hovorov a ping; výpadok medzi R8 a R5, fyzické odpojenie kábla z R8.
- V teste 10 sme mali 1 Silver SIP silver hovor, veľa Gold SIPp hovorov a ping.

Pod pojmom „veľa“ SIPp hovorov si treba predstaviť desať hovorov, čo rozhodne prekračuje kapacitu našich liniek v sieti. Testovali sme aj viac simultánnych hovorov, ale hodnota 10 bola asi maximum pre našu sieť.

V teste 2 meranie 3 došlo k zacykleniu v sieti a dočasnej topologickej slučke. Optimalizácia bola príliš agresívna. V teste 5 prichádzali pakety v nesprávnom poradí, vzniklo veľmi veľké oneskorenie, od 50 ms do 2000 ms. Test 6 bol z pohľadu pingu nemerateľný, keďže linka bola extrémne zahltená. Až po konvergencii a čiastočnom ukončení SIPp hovorov sa obnovil ping (okolo 15 sekúnd).



Obr. 16.5: Detailný návrh upravenej transportnej siete

Tabuľka 16.2: Sada testov 1

Sada 1	Test1		Test2		Test3		Test4		Test5		Test6	
test č.	KS	SP	KS	SP	KS	SP	KS	SP	KS	SP	KS	SP
1	0	0	3.67	111	6.59	5	0	0	0.03	1	3.18	86
2			3.49	109	6.52	5			0.03	1	2.84	88
priemer	0.000	0.00	3.576	110.00	6.553	5.00	0.000	0.00	0.030	1.00	3.013	87.00
min.	0.000	0	3.486	109	6.519	5	0.000	0	0.030	1	2.842	86
max.	0.000	0	3.666	111	6.587	5	0.000	0	0.030	1	3.184	88
ping frekvencia [s]	0.030		0.030		0.030		0.030		0.030		0.030	
teória / prax min.	0.000		2.684		6.512		0.000		0.030		2.684	
teória / prax max.	0.000		3.759		6.688		0.000		0.030		3.759	
je v danom rozsahu?	áno		áno		áno		áno		áno		áno	
výpadok SIP hovoru	0.000		4.000		7.000		0.000		0.000		7.000	

(SP = počet stratených paketov, KS = čas konvergencie siete)

Tabuľka 16.3: Sada testov 2

Sada 2	Test1		Test2		Test3		Test4		Test5		Test6		Test7		Test8		Test9		Test10	
test č.	KS	SP	KS	SP	KS	SP	KS	SP	KS	SP	KS	SP	KS	SP	KS	SP	KS	SP	KS	SP
1	0.030	1	1.410	31	2.190	36	0.036	1	0.046	1	-	-	0.992	30	0.007	0	1.010	31	0.047	1
2	0.063	2	1.046	30	1.224	33	0.000	0			-	-	1.265	32	0.102	3	1.194	17		
3			4.927	123	1.192	31	0.069	2			-	-	1.253	39	0.094	3	0.937	17		
priemer	0.047	1.50	2.461	61.33	1.535	33.33	0.035	1.00	0.046	1.00	-	-	1.170	33.67	0.068	2.00	1.047	21.67	0.047	1.00
min.	0.030	1	1.046	30	1.192	31	0.000	0	0.046	1	-	-	0.992	30	0.007	0	0.937	17	0.047	1
max.	0.063	2	4.927	123	2.190	36	0.069	2	0.046	1	-	-	1.265	39	0.102	3	1.194	31	0.047	1
ping frekvencia [s]	0.030		0.030		0.030		0.030		0.030		0.030		0.030		0.030		0.030		0.030	
teória / prax min.	0.000		0.747		0.634		0.000		0.000		0.747		0.747		0.000		0.747		0.000	
teória / prax max.	0.000		1.439		2.032		0.070		0.070		1.439		2.032		0.070		1.439		0.000	
je v danom rozsahu?	áno		áno okrem č. 3		áno		áno		áno		áno		áno		áno		áno		áno	
výpadok SIP hovoru	0.000		1.500		1.500		0.000		0.000		3.000-9.000		1.000-2.000		zašumenie		1.000		0.000	

(SP = počet stratených paketov, KS = čas konvergencie siete)

16.3 Zhodnotenie implementácie

V dnešnej dobe existuje niekoľko prístupov na optimalizáciu kooperácie sietí v jadre IMS. Tieto prístupy pracujú manuálne, nie je dostupné žiadne automatické riešenie. Navrhli sme inovatívne riešenie, ktoré automaticky nájde neoptimálne využitie šírky pásma, bez vyžadovania zásahu sieťových administrátorov. Avšak v tomto stave sú vyžadované záložné plány pre okamžitú reakciu.

Ukázali sme, že IP SLA so sledovaním objektov môže byť efektívne kombinované s ostatnými technológiami ako MPLS TE. Po zlyhaní linky alebo jej preťažení, môže byť presmerovaná sieťová prevádzka na alternatívne cesty, a to plne automatickým spôsobom. Tento prístup eliminuje potrebu zásahu sieťových administrátorov, ktorý je relatívne pomalý. Máme otvorené problémy pre ďalší vývoj, ako napr. optimalizácia vytvorenia TE tunela a stratégia jeho nasadenia.

Druhý prístup, ktorý bol cieľom tohto projektu dopadol úspešne. Sieť sme optimalizovali tak, aby dosiahla lepšie časy konvergencie. To sekundárne spôsobilo vyššiu dostupnosť uzlov a služieb a teda aj vyššiu kvalitu hovorov. Delením zákazníkov do tried sme zabezpečili garanciu kvality služieb. Implementácia bola úspešná, časy a kvalita sa rapídne zlepšili, čo viedlo k vyššej spokojnosti zákazníkov.

16.4 Verifikácia

Verifikácia prebiehala priebežne počas implementácie. Namerané hodnoty a sledované parametre sme porovnávali s teóriou a s publikáciami venujúcimi sa tejto problematike. Sledované parametre konvergencia siete a stratovosť paketov a teda dostupnosť uzlov bola kontrolovaná a porovnávaná najmä s bakalárskou prácou Filipa Burdu [7]. Keďže sa v tejto problematike už dlhšiu dobru orientuje a používa vlastné nástroje na zaznamenávanie zmien tokov v sieti, vie spoľahlivo vyhodnotiť merania. Namerané údaje sme samozrejme skontrolovali aj voči relevantným zdrojom. Väčšinou išlo o webové stránky, kde administrátori a správcovia siete si vymieňajú svoje výsledky v optimalizácii, čiže ide o hodnoty z praxe. Takisto máme k dispozícii mnohé prezentácie o optimalizácii. Publikačná činnosť je v tejto oblasti slabšia, keďže hodnoty sú skôr stavané na konkrétnu sieť.

Sieť IMS je jednoduché verifikovať, keďže nenastala veľká optimalizácia v tejto sekcii. Dôležité si bolo overiť funkčnosť IMS siete a správnosť jej chovania. Bližší popis je možné nájsť aj v prílohe B, kde uvádzame verifikačné prístupy. Keďže členovia tímu Juraj Nemeček a Ján Murányi majú bohaté pracovné skúsenosti s IMS sieťami, považujeme ich za dostatočne kompetentných na určenie správnej konfigurácie. Sieť sme verifikovali aj voči oficiálnej dokumentácii. Najsilnejším testom pre nás bolo, že klienti nemali problém s nadviazaním hovoru a hovor samotný prebiehal bezproblémovo. Problémy nastali len v prípade, že sme vygenerovali príliš veľa testovacích dátových tokov pomocou programu SIPp (viac v prílohe B) a servery HSS a P-CSCF nezvládali nápor používateľov. Za inštaláciu, konfiguráciu a správu IMS siete im aj touto cestou ďakujeme.

Hlavným nástrojom na verifikáciu konvergencie siete a stratovosti paketov je BASH skript, ktorý bol pre tento účel vytvorený. Tento vlastný pomocný program vytvára ku každému záznamu súhrn štatistík. Aplikácia je stavaná tak, aby bola v maximálnej možnej miere spoľahlivá. Pre istotu sme merania vyhodnotili aj manuálne a prípadné odchýlky upravili. Prípadné extrémne hodnoty

sme manuálne prehodnotili. Extrémne hodnoty sú spôsobené stratenými paketmi po ukončení konvergencie siete. K takejto občasnej stratovosti dochádza z dôvodu veľmi rýchleho posielania paketov. Záporné hodnoty z výstupu konvergencie siete znamenajú predčasne ukončené testovanie, respektíve neukončenú konvergenciu siete, pričom ale konvergencia protokolu už mohla nastať. Pomocný program sa nachádza na elektronickom médiu. Je určený na automatické vyhodnocovanie parametrov a slúži ako podklad pre ďalšie spracovanie výsledkov.

17 Zhodnotenie

V rámci tímového projektu bol vytvorený kompletne funkčný prototyp siete poskytovateľa služieb. Tento prototyp siete bol následne obohatený pokročilými technológiami pre optimalizáciu konvergencie a zaručenia kvality služieb, ktoré boli dostupné na špecializovaných smerovacích zariadeniach spoločnosti Cisco Systems pre IP siete. Optimalizáciou boli pokryté všetky parametre sieťovej topológie od smerovacieho protokolu IS-IS, cez model kvality služieb DiffServ až po veľmi pokročilé a vyspelé techniky dátového inžinierstva v MPLS sieti s klasifikáciou zákazníckych tried v IMS sieti.

V rámci procesu vytvárania a hlavne vyhodnocovania, tím neustále spracovával štatistické výsledky z testovacej prevádzky hlasových a konferenčných služieb v sieti. Z týchto meraní bol v tejto prípadovej štúdii zhodnotený vplyv optimalizačných vlastností rôznych technológií na kvalitu prevádzky v sieti. Sledovaná bola nie len schopnosť siete uprednostňovať služby podľa rôznych tried zákazníkov v IMS sieti, ale aj schopnosť siete vysporiadať sa s výpadkami na rôznych linkách v minimálnom čase. V rámci tejto siete bol zavedený systém OpenIMS ako jadro logiky komunikačného systému, ktorý umožnil nielen spracovanie signalizácie a spracovania hovorov, ale vďaka zavedeniu RTP proxy serverov aj plnú kontrolu nad dátovými tokmi medzi účastníkmi dátových relácií. Tento systém sme úspešne integrovali s logikou IP siete, aby sme boli schopní s minimálnou záťažou za pomoci oboch systémov klasifikovať účastníkov a udržiavať kontinuálnu kontrolu nielen nad signalizáciou hovorov, ale aj nad samotným dátovým tokom hovoru.

Všetky údaje z meraní boli následne spracované a odôvodnené na základe svojho vplyvu na optimalizáciu v tejto experimentálnej sieti. Tieto výsledky predstavujú prínos tejto prípadovej štúdie nakoľko predstavujú veľmi dôležitý údaj pre kvalitatívne a kvantitatívne porovnávanie schopnosti sietí z pohľadu optimálnosti pre multimediálne dátové toky, ktoré vyžadujú transportné služby v reálnom čase.

IMS sieť ponechávame nainštalovanú a nakonfigurovanú aj pre budúce generácie v laboratóriu D-105. Bude ju možné využiť napríklad pri spracúvaní bakalárskych a diplomových projektov. Takisto vieme zabezpečiť prístup z domu do našej IMS siete.

Slovník skratiek

ABR - Area Border Routers
ADSL - Asymmetric Digital Subscriber Line
AS - Application Servers
ASBR - Autonomous System Boundary Routers
ATM - Asychronous Transfer Mode
AToM - Any Transport over MPLS
BDR - Backup Designated Router
BGCF - Breakout Gateway Control Function
BGP - Border Gateway Protocol
BICC - Bearer Independent Call Control
CBR - Constraint Based Routing
CBWFQ - Class-Based WFQ
CEF - Cisco Express Forwarding
CIDR - Classless Inter-Domain Routing
CLNS - Connectionless Network Services
CoS - Class of Service
CQ - Custom Queuing
CSCF - Call Session Control Function
CS CN - Circuit Switched Core Network
CSPF - Constrained Shortest Path First
DBD - DataBase Description packet
DIS - Designated Intermediate System
DNS - Domain Name Server
DR - Designated Router
DS-TE - DiffServ-Aware Traffic Engineering
DV - Distance Vector
eBGP - External Border Gateway Protocol
ERP - Exterior Routing Protocol
ESDS - Ericsson Service Development
ES - End System
FIB - Forwarding Information Base
FIFO - First In First Out
FRR - Fast Reroute
GLBP - Gateway Load Balancing Protocol
HSRP - Hot Standby Router Protocol
HSS - Home Subscriber Server
HTTP - Hyper text transport protocol
IBGP - Internal Border Gateway Protocol
ICP - IMS Client Platform
I-CSCF - Interrogating-CSCF
IGP - Interior Gateway Protocol
IMS - IP Multimedia Subsystem
IOS - Internetwork Operating System
IP - Internet Protocol
IRP - Interior Routing Protocol
ISDN - Integrated Services Digital Network
IS - Intermediate System
IS-IS - Intermediate System to Intermediate System
ISPF - Incremental SPF
ISUP - ISDN User Part
KP - Konvergencia protokolu
KS - Konvergencia siete
LDP - Label Distribution Protocol
LLQ - Low-Latency Queuing
LSA - Link-State Advertisement
LS - Link-State
LSP - Link State Packet
MGCF - Media Gateway Control Function
MP-BGP - MultiProtocol BGP
MPLS - Multi-Protocol Label Switching
MPLS TE - MPLS Traffic Engineering
MP - Merge Point
MRFC - Multimedia Resource Function Controller
MRFP - Multimedia Resource Function Processor
MTU - Maximum Transmission Unit
NAT - Network Address Translation
NAT-PT - Network Address Translation - Protocol Translation
NBMA - nonbroadcast
NGN - New Generation Networks
NHop - Next-hop Router
NNHop - Next-next-hop router
OSI - Open Systems Interconnection
OSPF - Open Shortest Path First
PBR - Policy Based Routing
P-CSCF - Proxy-CSCF
PDF - Policy Decision Function
PDU - Protocol Data Unit
PEF - Policy Enforcement Function
PLR - Point of Local Repair
POS - Packet Over SONET
POTS - Plain Old Telephone Service
PQ - Priority Queuing
PRC - Partial Route Calculation
PSTN - Public Switched Telephone Network
PVC - Permanent Virtual Circuit
QoS - Quality of Service
RED - Random Early Detection
RIP - Routing Information Protocol

RPF - Reverse Path Forwarding
RSVP - Resource Reservation Protocol
RTCP - Real-Time Control Protocol
RTP - Real-Time Protocol
RTT - Round Trip Time
SCIP - Simple Conference Invitation Protocol
S-CSCF - Serving-CSCF
SDP - Session Description Protocol
SEG - Security Gateway
SER - SIP Express Routes
SGW - Signalling Gateway
SIP - Session Invitation Protocol
SLA - Service Level Agreement
SLF - Subscription Locator Function
SNMP - Simple Network Manegement Protocol
SPF - Shortest Path First
SP - Service Provider
SP - Stratovosť paketov
SS7 - Signaling System 7
TCP - Transmission Control Protocol
THIG - Topology Hiding Inter-network Gateway
TLS - Transport Layer Security
TOS - Type Of Service
UDP - User Datagram Protocol
URI - Uniform Resource Identfier
URL - Uniform Resource Locator
VIP - Virtual IP
VoIP - Voice over IP
VPN - Virtual Private Network
VRID - Virtual Router ID
VRRP - Virual Router Redundancy Protocol
WFQ - Weighted Fair Queuing
WRED - Weighted RED

Literatúra

[1] Hersent, Petit, Gurle: IP Telephony - Deploying Voice-over-IP Protocols. John Wiley & Sons Ltd, 2005. 416 s. ISBN-13: 978-0470023594.

[2] Johnston: SIP: understanding the Session Initiation Protocol. ARTECH HOUSE, 2004. 310 s. ISBN-13: 978-1580536554.

[3] Camarillo: SIP Demystified. McGraw-Hill Professional, 2001. 320 s. ISBN-13: 978-0071373401.

[4] KELLY: VoIP for dummies.Wiely Publishng, 2005. 312 s. ISBN-13: 978-0764588433.

[5] Poikselka, M., Mayer, G., Khartabil, H.: The IMS: IP Multimedia Concepts and Services. John Wiley & Sons Ltd, 2006. 439 s. ISBN-13: 978-0-470-01906-1.

[6] Russell, T.: The IP Multimedia Subsystem (IMS): Session Control and Other Network Operations. McGraw-Hill Osborne Media, 2008. 224 s. ISBN: 0071488537.

[7] Burda, F.: Konvergencia v počítačových sieťach. Bakalárska práca, FIIT STU, 2009.

[8] Reynders, D., Wright, E.: Practical TCP/IP and Ethernet Networking. IDC Technologies, August 2003. 306 s. ISBN 0-7506-5806-1.

[9] Burda, F.: Convergence in Computer Networks. In: Proceedings of the 5th Student Research Conference in Informatics and Information Technologies. Bratislava, Slovak Republic, 2009. pp. 327-334.

[10] EIGRP, IP Routing, IGRP. http://www.rhyshaden.com/eigrp.htm (28.10.2009)

[11] BGP Path Selection, http://rbcciequest.wordpress.com/2008/02/27/bgp-path-selection/ (28.10.2009)

[12] OSPF routing protocol, dijkstra algorithm, ospf stub area. http://www.rhyshaden.com/ospf.htm (28.10.2009)

[13] Poretsky, S.: Terminology for Benchmarking IGP Data Plane Route Convergence. Internet Draft. Jún 2003. http://www.ietf.org/proceedings/03jul/I-D/draft-ietf-bmwg-igp-dataplane-conv-term-00.txt (2008-09-11)

[14] Dubois, N., Fondeviole, B., Michel, N.: Fast convergence project. France Telecom R&D, 2004.

[15] Pužmanová, R.: Routing and Switching, Time of Convergence?. Addison-Wesley, 2002. ISBN 0-201-39861-3.

[16] Daugherty, B.: Optimizations for Routing Protocol Stability and Convergence. Cisco Systems, Inc., 2002.

[17] Shaikh, A., Greenberg, A.: Experience in Black-box OSPF Measurement. AT&T Research, 1998.

[18] ISISv6/BGP Fast Convergence Tuning. 6NET Project, 2003.

[19] Alaettinoglu, C., Jacobson, V., Yu, H.: Toward Millisecond IGP Convergence. Packet Design, Inc., 2000.

[20] Previdi, S., Horrocks, P.: Integrated IS-IS Design and Deployment Guide, 1998.

[21] Cisco Systems, Inc.: Cisco IOS Quality of Service Solutions Command Reference, Október 2009, s. 208-350. http://www.cisco.com/en/US/docs/ios/qos/command/reference/qos_cr.pdf (2009-10-26)

[22] Jork, M., Fang, L.: LDP IGP Synchronization. Internet Draft. December 2008. http://tools.ietf.org/html/draft-ietf-mpls-ldp-igp-sync-04 (2009-10-26)

[23] Cisco Systems, Inc.: MPLS LDP Session Protection. Máj 2007. http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/fssespro.pdf (2009-10-26)

[24] Cisco Systems Curriculum: Optimizing Converged Networks v5.0. Cisco Networking Academy, 2007.

[25] Cross V.: Linux on e-server IBM zSeries and S/390: Virtual Router Redundancy Protocol on VM Guest LANs. http://www.redbooks.ibm.com/redpapers/pdfs/redp3657.pdf (28.10.2009)

[26] Cross V.: Linux on e-server IBM zSeries and S/390: Virtual Router Redundancy Protocol on VM Guest LANs. http://www.redbooks.ibm.com/redpapers/pdfs/redp3657.pdf (28.10.2009)

[27] Cisco Systems Curriculum: Implementing Secure Converged Wide-area Networks v5.0. Cisco Networking Academy, 2007.

[28] Pepelnjak. I., Guichard. J.: MPLS and VPN architectures - volume II. Cisco Press 201 West 103rd Street Indianapolis, IN 46290 USA, 2001.

[29] Pepelnjak. I., Guichard. J.: MPLS and VPN architectures - volume I. Cisco Press 201 West 103rd Street Indianapolis, IN 46290 USA, 2001.

[30] Osborne, E., Simha, A.:Traffic Engineering with MPLS. Cisco Press, Júl 2002. 608 s. ISBN 1-58705-031-5.

[31] Álvarez, S.: MPLS Traffic Engineering Traffic Protection using Fast Re-route (FRR). Cisco Systems, Inc, August 2008.

[32] Cisco Systems, Inc.: MPLS Traffic Engineering - DiffServ Aware (DS-TE) http://www.cisco.com/en/US/docs/ios/12_2s/feature/guide/fsdserv3.pdf (2009-10-17)

[33] Satrapa, P.: Internetový protokol IPv6. CZ.NIC, z. s. p. o., 2008. ISBN 978-80-904248-0-7.

[34] 6net, An IPv6 Deployment Guide. The 6NET Consortium, September 2005.

[35] Polyrakis, A., Kalogeras, D.: 6PE: IPv6 over MPLS. N.T.U.A Network Management Center. CISCO EXPO, Január 2005.

[36] Bartlett, M., Berger, J., Harrison, J.: IPv6 Traffic Engineering in IS-IS. Internet Draft. September 2009. http://www.ietf.org/id/draft-ietf-isis-ipv6-te-07.txt (2009-11-07)

[37] Service Development Studio (SDS) 4.1 Developer’s Guide. Ericsson AB, 2009. 418s.

[38] Service Development Studio (SDS) 4.1 Tutorial. Ericsson AB, 2009. 232s.

[39] Service Development Studio (SDS) 4.1 FD1 Installation Instructions. Ericsson AB, 2009. 102s.

[40] Service Development Studio (SDS) 4.1 Windows Client Sample Application Description. Ericsson AB, 2009. 54s.

[41] Dokumentácia pre Open IMS HSS implementáciu. http://www.openimscore.org/docs/FHoSS/index.html (2009-12-06)

[42] Cisco Systems, Inc.: Cisco IOS IP SLAs Configuration Guide. Technical report, August 2008.

[43] Cisco Systems, Inc.: Configuring Enhanced Object Tracking. Technical report, March 2009.

[44] Awduche, D., O., Jabbari, B.: Internet traffic engineering using multi-protocol label switching (MPLS). Computer Networks, September 2002, vol. 40, no. 1, pp. 111-129.

[45] Szviatovszki, B., Szentesi, A., Juttner, A.: Minimizing re-routing in MPLS networks with preemption-aware constraint-based routing. Computer Communications, July 2002, vol. 25, no. 11-12, pp. 1076-1084.

Relevantné RFC dokumenty

[RFC142] Kline, Ch., Wong, J.: Time-Out Mechanism in the Host-Host Protocol. RFC142, Máj 1971.

[RFC793] Information Sciences Institute University of Southern California. Transmission Control Protocol. RFC793, September 1981.

[RFC1142] Oran, D.: OSI IS-IS Intra-domain Routing Protocol. RFC1142, Február 1990.

[RFC2327] Handley, M., Jacobson, V.: SDP: Session Description Protocol. RFC2327, Apríl 1998.

[RFC2328] Moy, J.: OSPF Version 2. RFC2328, Apríl 1998.

[RFC2338] Knight, S., Weaver, D., Whipple, D.: Virtual Router Redundancy Protocol. RFC2338, Apríl 1998.

[RFC2396] Berners-Lee, T., Fielding, R., Masinter, L.: Uniform Resource Identifiers (URI): Generic Syntax. RFC2396, August 1998.

[RFC2543] Handley, M., Schooler, E., Schulzrinne, H.: SIP: Session Initiation Protocol. RFC2543, Marec 1999.

[RFC3031] Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. RFC3031, Január 2001.

[RFC3101] Murphy, P.: The OSPF Not-So-Stubby Area (NSSA) Option. RFC3101, Január 2003.

[RFC3261] Camarillo, G., Rosenberg, J., Schulzrinne, H.: SIP: Session Initiation Protocol. RFC3261, Jún 2002.

[RFC4090] Pan, P., Swallow, G., Atlas, A.: Fast Reroute Extensions to RSVP-TE for LSP Tunnels. RFC4090, Máj 2005.

[RFC4203] Kompella, K., Rekhter, Y.: OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS). RFC4203, Október 2005.

[RFC4205] Kompella, K., Rekhter, Y.: Intermediate System to Intermediate System (IS-IS) Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS). RFC4205, Október 2005.

[RFC4271] Hares, S., Li, T., Rekhter, Y.: A Border Gateway Protocol 4 (BGP-4). RFC4271, Január 2006.

Príloha A: Používateľská príručka pre transportnú sieť

V tejto prílohe detailne popíšeme spôsob optimalizácie z pohľadu transportnej siete a význam jednotlivých konfiguračných príkazov zadaných na Cisco zariadeniach.

A-1: Základná konfigurácia

V prvom rade bude sieť „oživená“ pomocou protokolu IS-IS. Pre protokol IS-IS je potrebné najprv aktivovať proces protokolu IS-IS, nastaviť CLNS adresáciu a priradiť rozhrania do smerovacieho procesu. Druhým aspektom základnej konfigurácie sú MPLS parametre. Konfigurácia bude teraz predstavená s popisom týchto parametrov. Význam príkazov je popísaný v podobe komentáru za znakom „!“, ktorý je umiestnený pred samotným príkazom.

```
! konfigurácia názvu systému s lokálnym významom
hostname R1
!
! vstup do konfigurácie pre logické rozhranie využívané pre identifikáciu systému pre proces IS-IS
! protokolu
interface Loopback0
        ! konfigurácia IP adresy pod rozhraním Loopback0
        ip address 5.1.1.1 255.255.255.255
        ! príkaz pridáva toto rozhranie a jeho sieť do procesu IS-IS protokolu, preto sa adresa
        ! 5.1.1.1 stane v sieti dostupnou
        ip router isis
!
! vstup do konfigurácie pre rozhranie FastEthernet 0/0 (skrátene f0/0)
interface FastEthernet0/0
        ! popis rozhrania použitý pre lepšiu prehľadnosť konfigurácie
         description Rozhranie ku klientom
        ! konfigurácia IP adresy na rozhraní Lo0
         ip address 84.47.53.11 255.255.255.0
        ! implicitná konfigurácia rozhrania pre obojsmerný tok dát
         duplex auto
        ! implicitná konfigurácia pre rozhrania typu FastEthernet pre autokonfiguráciu rýchlosti
         speed auto
        ! konfigurácia VRRP protokolu pre umožnenie redundancie pre bránu klientov
         vrrp 1 ip 84.47.53.10
         vrrp 1 priority 254
        ! príkaz pridáva toto rozhranie a jeho sieť do procesu IS-IS protokolu
        ip router isis
!
! vstup do konfigurácie pre rozhranie Serial0/0/0 (skrátene s0/0/0)
interface Serial0/0/0
```

```
        ! konfigurácia IP adresy na rozhraní
        ip address 5.0.13.1 255.255.255.0
        ! príkaz pridáva toto rozhranie a jeho sieť do procesu IS-IS protokolu
        ip router isis
        ! aktivácia MPLS smerovania na rozhraní spolu s implicitným LDP protokolom
        mpls ip
!
! vstup do konfigurácie pre rozhranie Serial0/0/1 (skrátene s0/0/1), konfigurácia rozhrania je
! principiálne identická s rozhraním s0/0/0
interface Serial0/0/1
        ip address 5.0.14.1 255.255.255.0
        ip router isis
        mpls ip
!
! konfiguráca smerovacieho procesu IS-IS v jeho základnej podobe
router isis
        ! nastavenie typu metriky na jej novšou podobu pomocou TLV hodnôt, táto konfigurácia je
        ! vyžadovaná pre ďalšiu konfiguráciu MPLS TE
        metric-style wide
        ! proces IS-IS protokolu pri komunikácii so smerovačmi identifikuje sám seba pomocou IP
        ! adresy, ktorú týmto nastavením zvolíme z rozhrania Lo0
        mpls traffic-eng router-id Loopback0
        ! identifikácia domény systému IS-IS vychádza z adresácie v CLNS prostredí. Pre našu sieť
        ! sme zvolili nasledovný systém adresovania spojený z identifikácie domény „49.0000“
        ! a koncového systému vo formáte zahrňujúceho numerické meno smerovača, preto pre R1 je
        ! CLNS adresa „0000.0000.0001“, posledná časť má v CLNS sieti význam služby v systéme
        ! a hodnota „00“ je port systému. Táto hodnota je týmto IS-IS procesom.
        net 49.0000.0000.0000.0001.00
        ! rozhanie f0/0 nepotrebuje aktívne komunikovať pomocou protokolu IS-IS a preto je
        ! nastavené nasledovným príkazom ako pasívne rozhranie. Sieť takéhoto rozhrania bude
        ! propagovaná protokolom IS-IS, avšak samotné rozhranie nebude vytvárať IS-IS susedstvá
        ! s inými smerovačmi.
        passive-interface fastEthernet 0/0
```

Táto konfigurácia bola predvedená pre smerovač R1. V základnej konfigurácií sú konfigurácie smerovačov ekvivalentné z pohľadu týchto parametrov a rozdiely sú iba v hodnotách IP adries a CLNS doménového mena. Neokrajové smerovače v sieti nemajú aktivovaný protokol VRRP.

A-2: Optimalizácia

V rámci optimalizácie siete boli v sieti vybudované MPLS TE tunely pre jednotlivé triedy zákazníkov. Ďalej, smerovací protokol IS-IS bol nastavený na čo možno najlepší čas konvergencie pomocou optimalizácie časových parametrov. Nasledovná ukážka konfigurácie pochádza zo smerovača R8, ktorého konfigurácia by sa dala považovať za vzorovú konfiguráciu pre ostatné smerovače. Konfigurácia bude najprv predvedená vcelku a následne budú dôležité časti zdôvodnené a vysvetlené.

Konfigurácia smerovača R8

```
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R8
!
boot-start-marker
boot-end-marker
!
no aaa new-model
dot11 syslog
!
ip cef
ip cef load-sharing algorithm include-ports source destination
!
multilink bundle-name authenticated
mpls traffic-eng tunnels
!
class-map match-all GOLD
 match access-group name GOLD_DEST_IP
class-map match-all EF
 match dscp ef
 match mpls experimental topmost 5
class-map match-all SILVER
 match access-group name SILVER_DEST_IP
class-map match-all CS4
 match dscp cs4
 match mpls experimental topmost 4
!
!
policy-map GOLD_SILVER_MARK
 class GOLD
  set dscp ef
  set mpls experimental 5
  set mpls experimental topmost 5
 class SILVER
  set dscp cs4
  set mpls experimental 4
  set mpls experimental topmost 4
policy-map QUEUE_POLICY
 class EF
    priority percent 70
 class CS4
    bandwidth remaining percent 70
!
voice-card 0
 no dspfarm
!
archive
 log config
  hidekeys
!
controller DSL 0/1/0
 line-term cpe
!
interface Loopback0
```

```
 ip address 5.8.8.8 255.255.255.255
!
interface Tunnel1
 ip unnumbered Loopback0
 tunnel destination 5.9.9.9
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng path-option 10 explicit name GOLD
 tunnel mpls traffic-eng load-share 2
 no routing dynamic
!
interface Tunnel2
 ip unnumbered Loopback0
 tunnel destination 5.9.9.9
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng path-option 10 explicit name SILVER
 tunnel mpls traffic-eng load-share 2
 no routing dynamic
!
interface Tunnel3
 ip unnumbered Loopback0
 tunnel destination 5.9.9.9
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng path-option 10 explicit name GOLD_BACKUP
 tunnel mpls traffic-eng load-share 2
 no routing dynamic
!
interface FastEthernet0/0
 dampening
 ip address 72.68.30.72 255.255.255.0
 ip router isis
 ip policy route-map GOLD_ROUTE_MAP_ODD_EVEN
 duplex full
 speed 100
 mpls ip
 mpls mtu 1520
 mpls traffic-eng tunnels
 vrrp 1 ip 72.68.30.70
 vrrp 1 timers advertise msec 250
 vrrp 1 priority 254
 vrrp 1 track 3 decrement 2
 isis circuit-type level-1
 isis hello-multiplier 4
 isis hello-interval minimal
 no isis hello padding
 ip rsvp bandwidth
 service-policy GOLD_SILVER_MARK in
!
interface FastEthernet0/1
 no ip address
 shutdown
 duplex auto
 speed auto
!
interface FastEthernet0/2/0
!
interface FastEthernet0/2/1
!
interface FastEthernet0/2/2
!
interface FastEthernet0/2/3
!
```

```
interface Serial0/0/0
 dampening
 ip address 5.0.78.8 255.255.255.0
 ip router isis
 mpls ip
 mpls mtu 1520
 mpls traffic-eng tunnels
 no fair-queue
 isis circuit-type level-1
 isis hello-multiplier 4
 isis hello-interval minimal
 no isis hello padding
 ip rsvp bandwidth
 service-policy QUEUE_POLICY out
!
interface Serial0/0/1
 dampening
 ip address 5.0.58.8 255.255.255.0
 ip router isis
 mpls ip
 mpls mtu 1520
 mpls traffic-eng tunnels
 isis circuit-type level-1
 isis hello-multiplier 4
 isis hello-interval minimal
 no isis hello padding
 ip rsvp bandwidth
 service-policy QUEUE_POLICY out
!
interface wlan-controller1/0
 no ip address
 shutdown
!
interface Vlan1
 no ip address
!
router isis
 net 49.0000.0000.0000.0008.00
 ispf level-1-2
 metric-style wide
 spf-interval 1 1 100
 prc-interval 1 1 100
 lsp-gen-interval 1 1 100
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng level-1
 passive-interface Loopback0
!
ip forward-protocol nd
!
ip http server
no ip http secure-server
!
ip explicit-path name GOLD enable
 next-address 5.0.78.7
 next-address 5.0.59.9
!
ip explicit-path name SILVER enable
 next-address 5.0.58.5
 next-address 5.0.59.9
!
ip explicit-path name GOLD_BACKUP enable
```

```
 next-address 5.0.58.5
 next-address 5.0.59.9
!
ip access-list extended GOLD_ACL
 permit ip any host 5.200.0.10
 permit ip host 5.200.0.10 any
ip access-list extended GOLD_ACL_EVEN
 permit ip 0.0.0.0 255.255.255.254 host 5.200.0.10
ip access-list extended GOLD_ACL_ODD
 permit ip 0.0.0.1 255.255.255.254 host 5.200.0.10
ip access-list extended GOLD_DEST_IP
 permit ip any host 5.200.0.10
ip access-list extended SILVER_DEST_IP
 permit ip any host 5.200.0.20
!
route-map GOLD_ROUTE_MAP permit 10
 match ip address GOLD_ACL
 set interface Tunnel3 Tunnel1
!
route-map GOLD_ROUTE_MAP_ODD_EVEN permit 10
 match ip address GOLD_ACL_ODD
 set interface Tunnel1 Tunnel3
!
route-map GOLD_ROUTE_MAP_ODD_EVEN permit 20
 match ip address GOLD_ACL_EVEN
 set interface Tunnel3 Tunnel1
!
track 1 interface Serial0/0/0 line-protocol
!
track 2 interface Serial0/0/1 line-protocol
!
track 3 list boolean and
 object 1
 object 2
 delay up 30
!
line con 0
 exec-timeout 0 0
line aux 0
line vty 0 4
 privilege level 15
 no login
```

Rozbor ukážkovej konfigurácie a význam príkazov

Globálne nastavenia a protokol IS-IS

Prvým dôležitým optimalizačným nastavením je príkaz:

```
ip cef load-sharing algorithm include-ports source destination
```

Tento príkaz zabezpečí, aby v prípade rozkladania záťaže na viacero ciest boli pri rozhodovaní, ktorá relácia bude smerovaná ktorou cestou, brané do úvahy aj porty na transportnej vrstve. Toto nastavenie bolo nesmierne dôležité, nakoľko všetci Gold aj všetci Silver zákazníci používajú len svoje RTP proxy servery s IP adresami 5.200.0.10 pre Gold a 5.200.0.20 pre Silver. Preto s implicitným nastavením, ktoré by vyberalo cestu len na základe zdrojovej alebo cieľovej IP

adresy, by bolo možné, že alternatívne cesty by neboli vyťažené rovnomerne, čo by viedlo k suboptimálnemu zaťaženiu siete.

Druhým dôležitým optimalizačným príspevkom tohto projektu je optimalizácia IS-IS parametrov. V rámci IS-IS protokolu na smerovačoch Cisco sme využili tieto príkazy pod konfiguračným rozhraním IS-IS.

```
mpls traffic-eng tunnels
router isis
       net 49.0000.0000.0000.0008.00
       ispf level-1-2
       metric-style wide
       spf-interval 1 1 100
       prc-interval 1 1 100
       lsp-gen-interval 1 1 100
       mpls traffic-eng router-id Loopback0
       mpls traffic-eng level-1
       passive-interface Loopback0
```

● **mpls traffic-eng tunnels –** aktivuje MPLS TE pomocou tunelových rozhraní na smerovači spoločnosti Cisco Systems. Bez tohto príkazu by boli možnosti MPLS TE napriek všetkým ostatným nastaveniam v konfigurácii nečinné.

● **ispf level-1-2** – konfiguruje výnimku pre algoritmus SPF, aby proces IS-IS na smerovači v prípade zmeny topológie neprepočítaval celú topológiu, ale len ovplyvnenú časť stromovej štruktúry, ktorú protokol IS-IS vytvára.

● **metric-style wide** – implikuje smerovaču, aby používal v protokole IS-IS výhradne nový tvar smerovacích informácií v tvare TLV. Toto nastavenie je nutné pre konfiguráciu a prenos MPLS TE informácií pomocou IS-IS protokolu.

● **spf-interval 1 1 100** – konfigurácia minimálneho časového intervalu medzi dvoma SPF kalkuláciami. V tomto prípade sme nastavili počiatočnú kalkuláciu na minimálnu hodnotu, aby sme maximalizovali rýchlosť konvergencie. Parametre v tomto prípade sú:

- Interval v sekundách medzi dvoma nasledujúcimi kalkuláciami (1, 120)
- Minimálny čas čakania pred prvou kalkuláciou SPF v milisekundách (1, 120000)
- Minimálny čas čakania medzi prvou a druhou kalkuláciou SPF algoritmu v milisekundách (1, 120000)

● **prc-interval 1 1 100** – je príkaz s obdobným významom, avšak v tomto prípade nastavuje časové parametre medzi kalkuláciami čiastočnej zmeny topológie.

● **lsp-gen-interval 1 1 100** – je príkaz s obdobnými parametrami ako predošlé dva príkazy, tentokrát definujúci minimálny čas medzi dvoma vytvoreniami LSP paketu.

● **mpls traffic-eng router-id Loopback 0** – je príkaz, ktorý je súčasťou konfigurácie MPLS TE. Tento príkaz priraďuje IP adresu z logického rozhrania Lo0 ako globálny identifikátor pre tvorbu MPLS TE tunelov. Toto nastavenie je súčasťou IS-IS protokolu, nakoľko tento identifikátor je distribuovaný pomocou IS-IS protokolu.

● **mpls traffic-eng level-1 –** Nakoľko IS-IS je nosný protokol pre MPLS TE informácie (spolu s RSVP protokolom), týmto príkazom sme špecifikovali, že tieto informácie budú prenášané pomocou susedstiev v úrovni 1 (level-1).

Rozhranie F0/0

V ďalšej časti bude popísaná optimalizácia v rámci rozhrania a parametrov MPLS TE, IS-IS a VRRP protokolu na tomto ukážkovom rozhraní. Ukážkovým rozhraním je rozhranie FastEthernet 0/0 na smerovači R8.

```
interface FastEthernet0/0
       dampening
       ip address 72.68.30.72 255.255.255.0
       ip router isis
       ip policy route-map GOLD_ROUTE_MAP_ODD_EVEN
       duplex full
       speed 100
       mpls ip
       mpls mtu 1520
       mpls traffic-eng tunnels
       vrrp 1 ip 72.68.30.70
       vrrp 1 timers advertise msec 250
       vrrp 1 priority 254
       vrrp 1 track 3 decrement 2
       isis circuit-type level-1
       isis hello-multiplier 4
       isis hello-interval minimal
       no isis    hello padding
       ip rsvp bandwidth
       service-policy GOLD_SILVER_MARK in
```

● **dampening –** tento príkaz aktivuje vstavanú ochranu smerovacích protokolov proti takzvanému *flappingu*, čo je nekontrolovaná zmena stavu rozhrania. Rýchle zmeny stavu na rozhraniach môžu spôsobiť veľkú záťaž na prepočítavanie algoritmov v smerovacích protokoloch. Toto nastavenie umožní pozdržanie poskytnutia informácií o stave zariadenia týmto procesom, ak zmeny stavu prekonajú svojou nestabilitou definované parametre.

● **ip policy route-map GOLD_ROUTE_MAP_ODD_EVEN** – pre toto rozhranie priraďuje smerovanie na základe politiky, ktorej pravidlá sú definované v objekte takzvanej *route-map*. Podrobné vysvetlenie politiky skrývajúcej sa za *GOLD_ROUTE_MAP_ODD_EVEN* bude predstavené neskôr.

● **mpls ip** – pre toto rozhranie aktivuje smerovanie pomocou MPLS protokolu a zároveň aktivuje LDP protokol na distribúciu návestí.

● **mpls mtu 1520** – je podporný parameter pre smerovanie v MPLS sieti, nakoľko MPLS pridáva do každého rámca 4-bajtovú informáciu. Táto vlastnosť by mohla spôsobiť nadštandardnú veľkosť rámca na linkovej vrstve a preto toto nastavenie inkrementuje maximálnu povolenú dĺžku MTU pre toto rozhranie a pridáva 20 bajtov pre až 5 vnorených návestí MPLS.

Nasledovné príkazy konfigurujú protokol VRRP pre klientov nachádzajúcich sa za smerovačmi R8 a R7 v sieti. Protokol VRRP umožní, aby klienti mali nastavenú ako východiskovú bránu IP adresu, ktorú smerovače R8 a R7 medzi sebou virtuálne zdieľajú. VRRP konfigurácia pozostáva z týchto príkazov:

● **vrrp 1 ip 72.68.30.70** – definuje zdieľanú IP adresu východiskovej brány pre klientov

● **vrrp 1 timers advertise msec 250** – VRRP protokol oznamu aktívnosť smerovača každých 250 milisekúnd. (Implicitná hodnota je 1 sekunda)

● **vrrp 1 priority 254** – výber smerovača, ktorý bude pre zdieľanú IP adresu v danom momente poskytovať dátové služby závisí od protokolu VRRP, ktorý vyberá hlavný (*master*) smerovač pomocou priority (1-254), pričom väčšie čísla sú preferované.

● **vrrp 1 track 3 decrement 2** – VRRP proces v rámci Cisco zariadení umožňuje sledovať ľubovoľné rozhranie smerovača alebo objekt a dekrementovať prioritu v rámci VRRP procesu v prípade, ak sledované rozhranie alebo objekt zmení stav. V tomto nastavení sledujeme objekt *track 3* a v prípade zmeny stavu objektu dekrementujeme prioritu o 2.

Smerovače spoločnosti Cisco Systems umožňujú sledovať viacero parametrov podľa požiadaviek administrátora. V rámci posledného predstaveného príkazu sme využili túto schopnosť aby sme dynamicky menili parametre protokolu VRRP. Objekt 3 a kompletný použitý systém na sledovanie objektov pozostáva z týchto príkazov:

```
track 1 interface Serial0/0/0 line-protocol
!
track 2 interface Serial0/0/1 line-protocol
!
track 3 list boolean and
      object 1
      object 2
      delay up 10
```

● **track 1 interface Serial0/0/0 line-protocol** a **track 2 interface Serial0/0/1 line-protocol -** definujú dva objekty, ktoré sledujú stav linkovej vrstvy na rozhraniach Serial0/0/0 a Serial0/0/1.

● **track 3 list boolean and –** tretí objekt je definovaný ako kompozitný objekt. Tento druh objektu je spojenie viacerých objektov pomocou booleovskej logiky. V našom prípade sme definovali objekt 3 ako kompozitný objekt s booleovskou logikou operátora AND pomocou príkazu **track 3 list boolean and**. Pod týmto príkazom sme následne spriahli objekty 1 a 2. Dodatočne bolo dokonfigurované pozdržanie zmeny stavu objektu 3 na 10 sekundový interval po obnovení objektu ako ochrana proti *flapping*-u, teda rýchlim zmenám stavu linkovej vrstvy na rozhraní.

Nasledujúce parametre sú optimalizačné parametre pre protokol IS-IS na tomto rozhraní.

● **isis circuit-type level-1** – toto rozhranie komunikuje len v rámci susedstiev prvej úrovne.

● **isis hello-multiplier 4** – časovač *hold down*, ktorého vypršanie značí stratu susedstva s iným IS-IS smerovačom, nie je možné nastaviť na nižšiu hodnotu ako jedna sekunda. Tento príkaz zmení hodnotu násobiča na 4, čiže sa odosielajú 4 *hello* správy za jednu sekundu, teda každých 250 ms.

● **isis hello-interval minimal** – proces IS-IS nastaví *hold down* časovač na hodnotu 1 sekunda a čas medzi dvoma *hello* správami odvodí pomocou delenia tohto času násobičom.

● **no isis hello padding** – vypne zarovnávanie IS-IS *hello* paketov na dĺžku MTU. Týmto značne ušetríme prenosové pásmo a zrýchlime konvergenciu. Táto funkcia v dnešnej dobe nemá veľký význam a preto sa odporúča použiť túto optimalizáciu.

● **ip rsvp bandwith** – aktivuje pre rozhranie f0/0 možnosť rezervovať prenosové pásmo na tomto zariadení. Bez explicitne zadanej hodnoty sme umožnili rezervovať 75 % prenosového pásma rozhrania.

● **service-policy GOLD_SILVER_MARK in –** je konfiguračný príkaz pre aplikovanie politiky správy kvality služieb v prostredí Cisco IOS. Táto politika GOLD_SILVER_MARK je na rozhraní aplikovaná v smere dovnútra a skrýva sa pod ňou klasifikácia a značkovanie paketov pomocou DSCP a MPLS EXP bitov. Politika GOLD_SILVER_MARK bude bližšie predstavená neskôr.

Rozhranie tunela

MPLS TE tunel je v rámci Cisco IOS implementovaný ako logické rozhranie nasledovnými príkazmi.

```
interface Tunnel1
      ip unnumbered Loopback0
      tunnel destination 5.9.9.9
      tunnel mode mpls traffic-eng
      tunnel mpls traffic-eng path-option 10 explicit name GOLD
      no routing dynamic
```

V rámci konfigurácie R8 sú implementované tri tunely, GOLD, SILVER a GOLD_backup (teda záložný GOLD tunel). V rámci podrobného rozboru príkazov bude predstavená konfigurácia prvého z nich a to Tunnel 1, určeného pre Gold triedu zákazníkov.

● **interface tunnel 1 –** rozhranie tunela sa prvýkrát zadefinuje týmto príkazom. Tento príkaz vytvorí logické rozhranie tunela.

● **ip unnumbered Loopback 0 –** na rozhraní bez špecifikovanej IP adresy aktivuje IP procesy (ako smerovanie) a v prípade potreby zoberie ako identifikátor IP adresu z rozhrania Loopback 0. Nasleduje séria konfiguračných príkazov pre parametre tunela:

● **tunnel destination 5.9.9.9** – špecifikuje koncový bod tunela v sieti

● **tunnel mode mpls traffic-eng** – tunnel je typu MPLS TE

● **tunnel mpls traffic-eng path-option 10 explicit name GOLD** – pomocou zoznamu adries v *explicit-path GOLD* sú priamo definované IP body, ktorými musí tunel viesť

Ako bolo už spomenuté, pre tunel sme definovali cestu topológiou explicitne pomocou nasledovného zoznamu IP bodov v sieti, cez ktoré musí tunel viesť:

```
ip explicit-path name GOLD enable
      next-address 5.0.78.7
      next-address 5.0.59.9
```

Teraz už máme optimalizovaný smerovací protokol IS-IS, redundanciu pomocou protokolu VRRP pre koncových klientov a vytvorených dosť MPLS TE tunelov pre využívanie viacerých explicitných ciest pre klientov do IMS siete. V tejto časti siete predstavíme spôsob, akým rozdeľujeme záťaž na viacero tunelov. Logika je zaviesť politiku smerovania, ktorá rozdelí záťaž podľa príslušnosti klienta do GOLD alebo SILVER triedy a zároveň v rámci triedy na oba dostupné tunely pre GOLD. Tohto efektu sme dosiahli pomocou IMS siete tak, že všetci klienti GOLD triedy používajú RTP proxy s adresou 5.200.0.10 a všetci SILVER klienti s adresou 5.200.0.20. Nakoľko každý jeden hovor používa príslušnú adresu, mohli sme jednoducho vytvoriť IP pravidlá na klasifikáciu tokov pre tieto triedy. Avšak, aby sme dosiahli aj rozdelenie záťaže na vytvorené dva tunely, špecifikovali sme tieto pravidla ešte na párne a nepárne IP adresy klientov. Pravidlá na identifikáciu klientov sú nasledovné.

Akýkoľvek GOLD klient:

```
ip access-list extended GOLD_ACL
      permit ip any host 5.200.0.10
      permit ip host 5.200.0.10 any
```

Akýkoľvek GOLD klient s párnou IP adresou:

```
ip access-list extended GOLD_ACL_EVEN
      permit ip 0.0.0.0 255.255.255.254 host 5.200.0.10
```

Akýkoľvek GOLD klient s nepárnou IP adresou:

```
ip access-list extended GOLD_ACL_ODD
      permit ip 0.0.0.1 255.255.255.254 host 5.200.0.10
```

Nakoľko smerovač R8 je hraničný smerovač ku klientom, tak identifikuje GOLD a SILVER skupiny podľa IP adresy smerom k predpísaným RTP proxy serverom:

```
ip access-list extended GOLD_DEST_IP
      permit ip any host 5.200.0.10
ip access-list extended SILVER_DEST_IP
      permit ip any host 5.200.0.20
```

Následne už stačí definovať politiku smerovania pre GOLD zákazníkov:

```
route-map GOLD_ROUTE_MAP_ODD_EVEN permit 10
      match ip address GOLD_ACL_ODD
      set interface Tunnel1 Tunnel3
!
route-map GOLD_ROUTE_MAP_ODD_EVEN permit 20
      match ip address GOLD_ACL_EVEN
      set interface Tunnel3 Tunnel1
```

Route-map GOLD_ROUTE_MAP_ODD_EVEN je sekvenčný list logiky, ktorý pre dáta spĺňajúce pravidlá v GOLD_ACL_ODD (príkaz **match ip address GOLD_ACL_ODD**) smeruje dáta primárne tunelom 1 a v prípade nedostupnosti tunelom 3 (príkaz **set interface Tunnel1 Tunnel3**).

Druhým pravidlom v politike je pre dáta spĺňajúce pravidlo GOLD_ACL_EVEN (párne IP adresy – príkaz **match ip address GOLD_ACL_EVEN**), ktoré sú smerované primárne tunelom 3 a v prípade nedostupnosti tunela 3 tunelom 1 (príkaz **set interface Tunnel3 Tunnel1**).

Posledným aspektom konfigurácie je samotné manipulovanie s dátami v rámci DiffServ modelu pre kvalitu služieb. MPLS TE tunely umožnili rezervovať prenosové pásmo a rozdeliť záťaž na viacero tunelov. Avšak stále je potrebné v každom bode v topológii rozdeľovať dátové toky. Nasledovné nastavenia sú ukážkové nastavenia pre smerovač R8. Táto časť konfigurácie zabezpečuje, aby boli dáta GOLD a SILVER zákazníkov klasifikované a označkované v rámci IP aj MPLS hlavičky.

Prvá časť konfigurácie vytvára klasifikáciu dátových tokov na triedy (*class*). Klasifikujeme dáta podľa IP adries RTP proxy serverov, alebo podľa predošlého značkovania. Tieto triedy sa budú následne aplikovať do viacero politík pre kvalitu služieb.

Trieda definujúca toky dát smerujúce na RTP proxy GOLD zákazníkov:

```
class-map match-all GOLD
      match access-group name GOLD_DEST_IP
```

Trieda definujúca toky dát označkované triedou EF v DSCP poli IP hlavičky, alebo MPLS EXP polom s hodnotou 5:

```
class-map match-any EF
      match dscp ef
      match mpls experimental topmost 5
```

Trieda definujúca toky dát smerujúce na RTP proxy SILVER zákazníkov:

```
class-map match-all SILVER
      match access-group name SILVER_DEST_IP
```

Trieda definujúca toky dát označkované triedou CS4 v DSCP poli IP hlavičky alebo MPLS EXP polom s hodnotou 4:

```
class-map match-any CS4
      match dscp cs4
      match mpls experimental topmost 4
```

Nasledujúca skupina príkazov definuje mapu politík kvality služieb GOLD_SILVER_MARK, ktorá je určená na klasifikáciu a označenie dát pomocou IP DSCP poľa a MPLS EXP poľa. Ako vidieť z hierarchického usporiadanie príkazov, pre triedy GOLD a SILVER politika vykonáva značkovanie. Táto politika je priradená v sieti na všetky rozhrania v smere dnu, kde po prvýkrát vstupujú klientské dáta do siete:

```
policy-map GOLD_SILVER_MARK
      class GOLD
              set dscp ef
              set mpls experimental 5
              set mpls experimental topmost 5
       class SILVER
              set dscp cs4
              set mpls experimental 4
              set mpls experimental topmost 4
```

Druhá politika je politika QUEUE_POLICY, ktorá definuje prioritu dát (prioritný rad s rezervovanými 70 percentami priepustnosti rozhrania) označených pomocou značky triedy EF v DSCP poli IP hlavičky. Pre dáta so značkou CS4 v tom istom poli definuje rezerváciu priepustnosti poľa na úrovni 70 percent z doteraz nerezervovanej priepustnosti rozhrania.

```
policy-map QUEUE_POLICY
       class EF
               priority percent 70
       class CS4
               bandwidth remaining percent 70
```

Príloha B: Používateľská príručka pre inštaláciu IMS

V tejto kapitole je vysvetlený postup inštalácie prostredia a samotného jadra IMS. Vzhľadom na technický jazyk a pre zvýšenú prehľadnosť porušíme zavedenú štábnu kultúru ohľadne označovania anglických pojmov kurzívou vo vybraných prípadoch.

B-1: Príprava prostredia

Táto časť opisuje prípravu prostredia pred inštaláciou implementácie Open IMS Core. Požiadavky pre Open IMS Core sú uvedené na stránke *http://www.openimscore.org/installation_guide*, avšak uvádzame aj konkrétnu konfiguráciu nášho riešenia.

Použitý hardvér

Pre oddelenie internetovej brány, HSS, x-CSCF a aplikačných serverov bolo použitých päť osobných počítačov s parametrami:

- 733 Mhz Celeron
- 256 MB RAM
- 1x integrovaná 82557/8/9/0/1 Ethernet Pro 100 (100Mbit)
- 1x PCI Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+ 100Mbit
- 10GB WDC WD100EB-11BHF0
- CD-ROM

Použitý softvér

- Operačný systém Debian Lenny 5.0.4
    - stiahnutý netinst amd64 z:

      http://cdimage.debian.org/cdimage/squeeze_di_alpha1/amd64/iso-cd/debian-testing-amd64-netinst.iso
- GCC 4.3: apt-get install gcc
- make: apt-get install make
- JDK 1.6: apt-get install sun-java6-jdk
    - kontrola: java -version
    - výber medzi nainštalovanými verziami : update-alternatives --config java
- ant: apt-get isntall ant
- MySQL server: apt-get install mysql-server
- bison: apt-get install bison

- ant: apt-get install ant
- libxml 2.6 development: apt-get install libxml2-dev
- libmsql development: apt-get install libmysql++-dev
- openssl: apt-get install openssl
- bind: apt-get install bind9

V prípade, že nie je možné stiahnuť balíčky, editujeme súbor: */etc/apt/source.list* a aktualizujeme zoznam balíčkov: *apt-get update*. Ak ani po týchto krokoch nie je možné stiahnuť balíčky, skontrolujeme sieťové pripojenie do internetu.

Nainštalovanie všetkých použitých balíčkov pre jednotlivé servery uvádzame v nasledujúcich výpisoch. Používateľovi stačí skopírovať nasledujúce pasáže. AS1 a AS2 predstavujú aplikačné servery.

Internetová brána

apt-get install acpi-support-base acpid adduser apt apt-utils aptitude at base-files base-passwd bash bash-completion bc bind9 bind9-host bind9utils bsd-mailx bsdmainutils bsdutils busybox ca-certificates console-common console-data console-tools coreutils cpio cpp cpp-4.3 cron dbus dbus-x11 dc debconf debconf-i18n debian-archive-keyring debian-faq debianutils defoma dhcp3-client dhcp3-common dhcp3-server dictionaries-common diff dmidecode dnsutils doc-debian doc-linux-text dpkg e2fslibs e2fsprogs ed eject exim4 exim4-base exim4-config exim4-daemon-light file findutils fontconfig fontconfig-config freepats ftp gcc-4.2-base gcc-4.3-base gconf2 gconf2-common gettext-base gnupg gpgv grep groff-base grub grub-common gstreamer0.10-alsa gstreamer0.10-ffmpeg gstreamer0.10-plugins-bad gstreamer0.10-plugins-base gstreamer0.10-plugins-good gstreamer0.10-plugins-ugly gstreamer0.10-x gzip hicolor-icon-theme hostname iamerican ibritish ifupdown info initramfs-tools initscripts ation-report iproute iptables iputils-ping ispell klibc-utils laptop-detect less liba52-0.7.4 libaa1 libacl1 libasound2 libatk1.0-0 libatk1.0-data libattr1 libavc1394-0 libavcodec51 libavformat52 libavutil49 libbind9-40 libblkid1 libbz2-1.0 libc6 libc6-i686 libcaca0 libcairo2 libcap1 libcap2 libcdaudio1 libcdio7 libcdparanoia0 libcomerr2 libconsole libcucul0 libcups2 libcurl3-gnutls libcwidget3 libdatrie0 libdb4.5 libdb4.6 libdbus-1-3 libdc1394-22 libdevmapper1.02.1 libdirectfb-1.0-0 libdns45 libdrm2 libdv4 libdvdnav4 libdvdread3 libedit2 libept0 libevent1 libexempi3 libexif12 libexosip2-4 libexpat1 libfaad0 libfftw3-3 libflac8 libfontconfig1 libfontenc1 libfreebob0 libfreetype6 libgc1c2 libgcc1 libgconf2-4 libgcrypt11 libgdbm3 libgl1-mesa-glx libglib2.0-0 libglib2.0-data libglu1-mesa libgmp3c2 libgmyth0 libgnutls26 libgpg-error0 libgpm2 libgsm1 libgssglue1 libgstreamer-plugins-base0.10-0 libgstreamer0.10-0 libgstreamer0.10-0-dbg libgtk2.0-0 libgtk2.0-bin libgtk2.0-common libhal1 libid3tag0 libidl0 libidn11 libiec61883-0 libiptcdata0 libisc45 libisccc40 libisccfg40 libjack0 libjpeg62 libkeyutils1 libklibc libkrb53 libldap-2.4-2 liblocale-gettext-perl liblockfile1 liblwres40 liblzo2-2 libmad0 libmagic1 libmms0 libmpcdec3 libmpeg2-4 libmpfr1ldbl libmusicbrainz4c2a libmysqlclient15off libncurses5 libncursesw5 libneon27-gnutls libnewt0.52 libnfsidmap2 libofa0 libogg0 liboil0.3 libopenspc0 liborbit2 libosip2-2 libosip2-3deb libpam-modules libpam-runtime libpam0g libpango1.0-0 libpango1.0-common libpcap0.8 libpci3 libpcre3 libpixman-1-0 libpkcs11-helper1 libpng12-0 libpopt0 libpostproc51 libraw1394-8 libreadline5 librpcsecgss3 libsasl2-2 libselinux1 libsepol1 libshout3 libsidplay1 libsigc++-2.0-0c2a libslang2 libsndfile1 libsoundtouch1c2 libsoup2.4-1 libspeex1 libsqlite3-0 libss2 libssl0.9.8 libstdc++6 libsysfs2 libtag1c2a libtasn1-3 libtext-charwidth-perl libtext-iconv-perl libtext-wrapi18n-perl libthai-data

libthai0 libtheora0 libtiff4 libts-0.0-0 libusb-0.1-4 libuuid1 libvisual-0.4-0 libvisual-0.4-plugins libvolume-id0 libvorbis0a libvorbisenc2 libwavpack1 libwildmidi0 libwrap0 libx11-6 libx11-data libxapian15 libxau6 libxcb-render-util0 libxcb-render0 libxcb-xlib0 libxcb1 libxcomposite1 libxcursor1 libxdamage1 libxdmcp6 libxext6 libxfixes3 libxfont1 libxft2 libxi6 libxinerama1 libxml2 libxmuu1 libxrandr2 libxrender1 libxv1 libxxf86vm1 links linux-image-2.6-686 linux-image-2.6.26-2-686 locales lockfile-progs login logrotate lsb-base lsof lzma m4 makedev man-db manpages mawk mime-support mktemp mlocate module-init-tools mount mutt mysql-common nano ncurses-base ncurses-bin ncurses-term net-tools netbase netcat-traditional nfs-common nmap ntpdate openbsd-inetd openssh-blacklist openssh-blacklist-extra openssh-client openssh-server openssl openssl-blacklist openvpn openvpn-blacklist oss-compat passwd patch pciutils perl perl-base perl-modules portmap procmail procps psmisc python python-central python-minimal python2.5 python2.5-minimal readline-common reportbug rsyslog screen sed ser de sysv-rc sysvinit sysvinit-utils tar tasksel tasksel-data tcpd tcpdump telnet texinfo time traceroute ttf-dejavu ttf-dejavu-core ttf-dejavu-extra tzdata ucf uctb2bua uctiptv uctiptvas udev update-inetd usbutils util-linux vim-common vim-tiny w3m wamerican wget whiptail whois x-ttcidfont-conf x11-common xauth xfonts-encodings xfonts-utils zlib1g

HSS

apt-get install acpi-support-base acpid adduser apt apt-utils aptitude at base-files base-passwd bash bash-completion bc bind9 bind9-host bind9utils bsd-mailx bsdmainutils bsdutils busybox ca-certificates console-common console-data console-tools coreutils cpio cpp cpp-4.3 cron dbus dbus-x11 dc debconf debconf-i18n debian-archive-keyring debian-faq debianutils defoma dhcp3-client dhcp3-common dhcp3-server dictionaries-common diff dmidecode dnsutils doc-debian doc-linux-text dpkg e2fslibs e2fsprogs ed eject exim4 exim4-base exim4-config exim4-daemon-light file findutils fontconfig fontconfig-config freepats ftp gcc-4.2-base gcc-4.3-base gconf2 gconf2-common gettext-base gnupg gpgv grep groff-base grub grub-common gstreamer0.10-alsa gstreamer0.10-ffmpeg gstreamer0.10-plugins-bad gstreamer0.10-plugins-base gstreamer0.10-plugins-good gstreamer0.10-plugins-ugly gstreamer0.10-x gzip hicolor-icon-theme hostname iamerican ibritish ifupdown info initramfs-tools initscripts ation-report iproute iptables iputils-ping ispell klibc-utils laptop-detect less liba52-0.7.4 libaa1 libacl1 libasound2 libatk1.0-0 libatk1.0-data libattr1 libavc1394-0 libavcodec51 libavformat52 libavutil49 libbind9-40 libblkid1 libbz2-1.0 libc6 libc6-i686 libcaca0 libcairo2 libcap1 libcap2 libcdaudio1 libcdio7 libcdparanoia0 libcomerr2 libconsole libcucul0 libcups2 libcurl3-gnutls libcwidget3 libdatrie0 libdb4.5 libdb4.6 libdbus-1-3 libdc1394-22 libdevmapper1.02.1 libdirectfb-1.0-0 libdns45 libdrm2 libdv4 libdvdnav4 libdvdread3 libedit2 libept0 libevent1 libexempi3 libexif12 libexosip2-4 libexpat1 libfaad0 libfftw3-3 libflac8 libfontconfig1 libfontenc1 libfreebob0 libfreetype6 libgc1c2 libgcc1 libgconf2-4 libgcrypt11 libgdbm3 libgl1-mesa-glx libglib2.0-0 libglib2.0-data libglu1-mesa libgmp3c2 libgmyth0 libgnutls26 libgpg-error0 libgpm2 libgsm1 libgssglue1 libgstreamer-plugins-base0.10-0 libgstreamer0.10-0 libgstreamer0.10-0-dbg libgtk2.0-0 libgtk2.0-bin libgtk2.0-common libhal1 libid3tag0 libidl0 libidn11 libiec61883-0 libiptcdata0 libisc45 libisccc40 libisccfg40 libjack0 libjpeg62 libkeyutils1 libklibc libkrb53 libldap-2.4-2 liblocale-gettext-perl liblockfile1 liblwres40 liblzo2-2 libmad0 libmagic1 libmms0 libmpcdec3 libmpeg2-4 libmpfr1ldbl libmusicbrainz4c2a libmysqlclient15off libncurses5 libncursesw5 libneon27-gnutls libnewt0.52 libnfsidmap2 libofa0 libogg0 liboil0.3 libopenspc0 liborbit2 libosip2-2 libosip2-3deb libpam-modules libpam-runtime libpam0g libpango1.0-0 libpango1.0-common libpcap0.8 libpci3 libpcre3 libpixman-1-0 libpkcs11-helper1 libpng12-0 libpopt0 libpostproc51 libraw1394-8 libreadline5 librpcsecgss3 libsasl2-2 libselinux1 libsepol1 libshout3 libsidplay1 libsigc++-2.0-0c2a libslang2 libsndfile1

libsoundtouch1c2 libsoup2.4-1 libspeex1 libsqlite3-0 libss2 libssl0.9.8 libstdc++6 libsysfs2 libtag1c2a libtasn1-3 libtext-charwidth-perl libtext-iconv-perl libtext-wrapi18n-perl libthai-data libthai0 libtheora0 libtiff4 libts-0.0-0 libusb-0.1-4 libuuid1 libvisual-0.4-0 libvisual-0.4-plugins libvolume-id0 libvorbis0a libvorbisenc2 libwavpack1 libwildmidi0 libwrap0 libx11-6 libx11-data libxapian15 libxau6 libxcb-render-util0 libxcb-render0 libxcb-xlib0 libxcb1 libxcomposite1 libxcursor1 libxdamage1 libxdmcp6 libxext6 libxfixes3 libxfont1 libxft2 libxi6 libxinerama1 libxml2 libxmuu1 libxrandr2 libxrender1 libxv1 libxxf86vm1 links linux-image-2.6-686 linux-image-2.6.26-2-686 locales lockfile-progs login logrotate lsb-base lsof lzma m4 makedev man-db manpages mawk mime-support mktemp mlocate module-init-tools mount mutt mysql-common nano ncurses-base ncurses-bin ncurses-term net-tools netbase netcat-traditional nfs-common nmap ntpdate openbsd-inetd openssh-blacklist openssh-blacklist-extra openssh-client openssh-server openssl openssl-blacklist openvpn openvpn-blacklist oss-compat passwd patch pciutils perl perl-base perl-modules portmap procmail procps psmisc python python-central python-minimal python2.5 python2.5-minimal readline-common reportbug rsyslog screen sed ser de sysv-rc sysvinit sysvinit-utils tar tasksel tasksel-data tcpd tcpdump telnet texinfo time traceroute ttf-dejavu ttf-dejavu-core ttf-dejavu-extra tzdata ucf uctb2bua uctiptv uctiptvas udev update-inetd usbutils util-linux vim-common vim-tiny w3m wamerican wget whiptail whois x-ttcidfont-conf x11-common xauth xfonts-encodings xfonts-utils zlib1g

CSCF

acpi-support-base acpid adduser apt apt-utils aptitude at base-files base-passwd bash bash-completion bc bind9-host binutils bison bsd-mailx bsdmainutils bsdutils build-essential busybox bzip2 ca-certificates console-common console-data console-tools coreutils cpio cpp cpp-4.3 cron dc debconf debconf-i18n debian-archive-keyring debian-faq debianutils dhcp3-client dhcp3-common dictionaries-common diff dmidecode dnsutils doc-debian doc-linux-text dpkg dpkg-dev e2fslibs e2fsprogs ed eject exim4 exim4-base exim4-config exim4-daemon-light file findutils flex ftp g++ g++-4.3 gcc gcc-4.2-base gcc-4.3 gcc-4.3-base gettext-base gnupg gpgv grep groff-base grub grub-common gzip hostname iamerican ibritish ifupdown info initramfs-tools initscripts ation-report iproute iptables iputils-ping ispell klibc-utils laptop-detect less libacl1 libapr1 libaprutil1 libattr1 libbind9-40 libblkid1 libbz2-1.0 libc6 libc6-dev libc6-i686 libcap2 libcomerr2 libconsole libcwidget3 libdb4.5 libdb4.6 libdbd-mysql-perl libdbi-perl libdevmapper1.02.1 libdns45 libedit2 libept0 libevent1 libexpat1 libgc1c2 libgcc1 libgcrypt11 libgdbm3 libgmp3c2 libgnutls26 libgomp1 libgpg-error0 libgpm2 libgssglue1 libidn11 libisc45 libisccc40 libisccfg40 libkeyutils1 libklibc libkrb53 libldap-2.4-2 liblocale-gettext-perl liblockfile1 liblwres40 libmagic1 libmpfr1ldbl libmysql++-dev libmysql++3 libmysqlclient15-dev libmysqlclient15off libncurses5 libncursesw5 libneon27-gnutls libnet-daemon-perl libnewt0.52 libnfsidmap2 libpam-modules libpam-runtime libpam0g libpcap0.8 libpci3 libpcre3 libplrpc-perl libpopt0 libpq5 libreadline5 librpcsecgss3 libsasl2-2 libselinux1 libsepol1 libserf-0-0 libsigc++-2.0-0c2a libslang2 libsqlite3-0 libss2 libssl0.9.8 libstdc++6 libstdc++6-4.3-dev libsvn1 libtasn1-3 libterm-readkey-perl libtext-charwidth-perl libtext-iconv-perl libtext-wrapi18n-perl libtimedate-perl libusb-0.1-4 libuuid1 libvolume-id0 libwrap0 libx11-6 libx11-data libxapian15 libxau6 libxcb-xlib0 libxcb1 libxdmcp6 libxext6 libxml2 libxml2-dev libxmuu1 linux-image-2.6-686 linux-image-2.6.26-2-686 linux-libc-dev locales login logrotate lsb-base lsof lzma m4 make makedev man-db manpages mawk mime-support mktemp mlocate module-init-tools mount mutt mysql-client mysql-client-5.0 mysql-common nano ncurses-base ncurses-bin ncurses-term net-tools netbase netcat-traditional nfs-common openbsd-inetd openssh-blacklist openssh-blacklist-extra openssh-client openssh-server openssl passwd patch pciutils perl perl-base perl-modules portmap procmail procps python python-central python-

minimal python2.5 python2.5-minimal readline-common reportbug rsyslog screen sed subversion sysv-rc sysvinit sysvinit-utils tar tasksel tasksel-data tcpd tcpdump telnet texinfo time traceroute tzdata ucf udev update-inetd usbutils util-linux vim-common vim-tiny w3m wamerican wget whiptail whois x11-common xauth zlib1g zlib1g-dev

AS1

apt-get install acpi-support-base acpid adduser apt apt-utils aptitude at base-files base-passwd bash bash-completion bc bind9-host binutils bison bsd-mailx bsdmainutils bsdutils build-essential busybox bzip2 ca-certificates console-common console-data console-tools coreutils cpio cpp cpp-4.3 cron dc debconf debconf-i18n debian-archive-keyring debian-faq debianutils dhcp3-client dhcp3-common dictionaries-common diff dmidecode dnsutils doc-debian doc-linux-text dpkg dpkg-dev e2fslibs e2fsprogs ed eject exim4 exim4-base exim4-config exim4-daemon-light file findutils flex ftp g++ g++-4.3 gcc gcc-4.2-base gcc-4.3 gcc-4.3-base gettext-base gnupg gpgv grep groff-base grub grub-common gzip hostname iamerican ibritish ifupdown info initramfs-tools initscripts ation-report iproute iptables iputils-ping ispell klibc-utils laptop-detect less libacl1 libapr1 libaprutil1 libattr1 libbind9-40 libblkid1 libbz2-1.0 libc6 libc6-dev libc6-i686 libcap2 libcomerr2 libconsole libcwidget3 libdb4.5 libdb4.6 libdbd-mysql-perl libdbi-perl libdevmapper1.02.1 libdns45 libedit2 libept0 libevent1 libexpat1 libgc1c2 libgcc1 libgcrypt11 libgdbm3 libgmp3c2 libgnutls26 libgomp1 libgpg-error0 libgpm2 libgssglue1 libidn11 libisc45 libisccc40 libisccfg40 libkeyutils1 libklibc libkrb53 libldap-2.4-2 liblocale-gettext-perl liblockfile1 liblwres40 libmagic1 libmpfr1ldbl libmysql++-dev libmysql++3 libmysqlclient15-dev libmysqlclient15off libncurses5 libncursesw5 libneon27-gnutls libnet-daemon-perl libnewt0.52 libnfsidmap2 libpam-modules libpam-runtime libpam0g libpcap0.8 libpci3 libpcre3 libplrpc-perl libpopt0 libpq5 libreadline5 librpcsecgss3 libsasl2-2 libselinux1 libsepol1 libserf-0-0 libsigc++-2.0-0c2a libslang2 libsqlite3-0 libss2 libssl0.9.8 libstdc++6 libstdc++6-4.3-dev libsvn1 libtasn1-3 libterm-readkey-perl libtext-charwidth-perl libtext-iconv-perl libtext-wrapi18n-perl libtimedate-perl libusb-0.1-4 libuuid1 libvolume-id0 libwrap0 libx11-6 libx11-data libxapian15 libxau6 libxcb-xlib0 libxcb1 libxdmcp6 libxext6 libxml2 libxml2-dev libxmuu1 linux-image-2.6-686 linux-image-2.6.26-2-686 linux-libc-dev locales login logrotate lsb-base lsof lzma m4 make makedev man-db manpages mawk mime-support mktemp mlocate module-init-tools mount mutt mysql-client mysql-client-5.0 mysql-common nano ncurses-base ncurses-bin ncurses-term net-tools netbase netcat-traditional nfs-common openbsd-inetd openssh-blacklist openssh-blacklist-extra openssh-client openssh-server openssl passwd patch pciutils perl perl-base perl-modules portmap procmail procps python python-central python-minimal python2.5 python2.5-minimal readline-common reportbug rsyslog screen sed subversion sysv-rc sysvinit sysvinit-utils tar tasksel tasksel-data tcpd tcpdump telnet texinfo time traceroute tzdata ucf udev update-inetd usbutils util-linux vim-common vim-tiny w3m wamerican wget whiptail whois x11-common xauth zlib1g zlib1g-dev

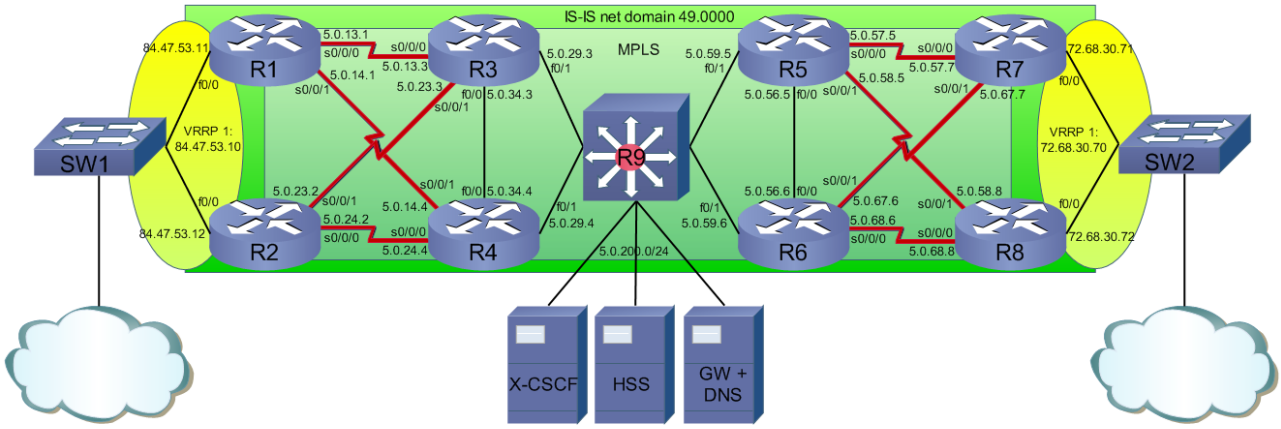
AS2

apt-get install acpi-support-base acpid adduser apt apt-utils aptitude at base-files base-passwd bash bash-completion bc bind9-host binutils bison bsd-mailx bsdmainutils bsdutils build-essential busybox bzip2 ca-certificates console-common console-data console-tools coreutils cpio cpp cpp-4.3 cron dc debconf debconf-i18n debian-archive-keyring debian-faq debianutils dhcp3-client dhcp3-common dictionaries-common diff dmidecode dnsutils doc-debian doc-linux-text dpkg dpkg-dev e2fslibs e2fsprogs ed eject exim4 exim4-base exim4-config exim4-daemon-light file findutils flex ftp g++ g++-4.3 gcc gcc-4.2-base gcc-4.3 gcc-4.3-base gettext-base gnupg gpgv grep groff-

base grub grub-common gzip hostname iamerican ibritish ifupdown info initramfs-tools initscripts ation-report iproute iptables iputils-ping ispell klibc-utils laptop-detect less libacl1 libapr1 libaprutil1 libattr1 libbind9-40 libblkid1 libbz2-1.0 libc6 libc6-dev libc6-i686 libcap2 libcomerr2 libconsole libcwidget3 libdb4.5 libdb4.6 libdbd-mysql-perl libdbi-perl libdevmapper1.02.1 libdns45 libedit2 libept0 libevent1 libexpat1 libgc1c2 libgcc1 libgcrypt11 libgdbm3 libgmp3c2 libgnutls26 libgomp1 libgpg-error0 libgpm2 libgssglue1 libidn11 libisc45 libiscc40 libisccfg40 libkeyutils1 libklibc libkrb53 libldap-2.4-2 liblocale-gettext-perl liblockfile1 liblwres40 libmagic1 libmpfr1ldbl libmysql++-dev libmysql++3 libmysqlclient15-dev libmysqlclient15off libncurses5 libncursesw5 libneon27-gnutls libnet-daemon-perl libnewt0.52 libnfsidmap2 libpam-modules libpam-runtime libpam0g libpcap0.8 libpci3 libpcre3 libplrpc-perl libpopt0 libpq5 libreadline5 librpcsecgss3 libsasl2-2 libselinux1 libsepol1 libserf-0-0 libsigc++-2.0-0c2a libslang2 libsqlite3-0 libss2 libssl0.9.8 libstdc++6 libstdc++6-4.3-dev libsvn1 libtasn1-3 libterm-readkey-perl libtext-charwidth-perl libtext-iconv-perl libtext-wrapi18n-perl libtimedate-perl libusb-0.1-4 libuuid1 libvolume-id0 libwrap0 libx11-6 libx11-data libxapian15 libxau6 libxcb-xlib0 libxcb1 libxdmcp6 libxext6 libxml2 libxml2-dev libxmuu1 linux-image-2.6-686 linux-image-2.6.26-2-686 linux-libc-dev locales login logrotate lsb-base lsof lzma m4 make makedev man-db manpages mawk mime-support mktemp mlocate module-init-tools mount mutt mysql-client mysql-client-5.0 mysql-common nano ncurses-base ncurses-bin ncurses-term net-tools netbase netcat-traditional nfs-common openbsd-inetd openssh-blacklist openssh-blacklist-extra openssh-client openssh-server openssl passwd patch pciutils perl perl-base perl-modules portmap procmail procps python python-central python-minimal python2.5 python2.5-minimal readline-common reportbug rsyslog screen sed subversion sysv-rc sysvinit sysvinit-utils tar tasksel tasksel-data tcpd tcpdump telnet texinfo time traceroute tzdata ucf udev update-inetd usbutils util-linux vim-common vim-tiny w3m wamerican wget whiptail whois x11-common xauth zlib1g zlib1g-dev

Prepojenie počítačov

Pri našej poslednej a záverečnej implementácii sme použili nasledovnú schému (Obr. B-1). Počítače v IMS sieti používajú adresný rozsah 5.200.0.0/24.



Obr. B-1: Detailný návrh transportnej siete

Nastavenia pre *Gateway* (internetová brána), označený ako GW sú nasledovné:

Nastavenie sieťových rozhraní v súbore */etc/network/interfaces*

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo eth0
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
address 5.200.0.1
netmask 255.255.255.0
```

Pre aplikovanie zmien vykonaných v tomto súbore je potrebné reštartovať sieťové rozhrania príkazom:

```
#/etc/init.d/networking restart
```

V súbore */etc/resolv.conf* uvedieme natavenie DNS servera.

```
nameserver 5.200.0.1
```

Nastavenie sieťových rozhraní je u ostatných serveroch obdobné. Pre úplnosť uvádzame nastavenie súboru */etc/network/intefaces*.

CSCF:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo eth0
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
address 5.200.0.2
netmask 255.255.255.0
gateway 5.200.0.1
```

HSS:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo eth0
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
address 5.200.0.3
netmask 255.255.255.0
gateway 5.200.0.1
```

AS1:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
```

```
# The loopback network interface
auto lo eth0
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
address 5.200.0.10
netmask 255.255.255.0
gateway 5.200.0.1
```

AS2:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo eth0
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
address 5.200.0.20
netmask 255.255.255.0
gateway 5.200.0.1
```

B-2: Získanie zdrojových súborov

Zdrojové súbory sa nachádzajú v predvolenom adresári, ktorý odporúča návod Open IMS Core na počítači CSCF s IP adresou 5.200.0.2:

```
mkdir /opt/OpenIMSCore
cd /opt/OpenIMSCore
mkdir ser_ims
svn checkout http://svn.berlios.de/svnroot/repos/openimscore/ser_ims/trunk
ser_ims
```

Na počítači HSS s IP adresou 5.200.0.3:

```
mkdir /opt/OpenIMSCore
cd /opt/OpenIMSCore
mkdir FHoSS
svn  checkout  http://svn.berlios.de/svnroot/repos/openimscore/FHoSS/trunk
FHoSS
```

Kompilácia

Zdrojové súbory sú skompilované podľa návodu Open IMS Core. V prípade, že kompilácia neprehne úspešne, je potrebné skontrolovať, že či všetky aplikácie v softvérových požiadavkách sú aktuálne a správne nainštalované.

Program ser_ims

```
cd /opt/OpenIMSCore/ser_ims
make install-libs all
cd ..
```

FHoSS

```
cd /opt/OpenIMSCore/FHoSS
ant compile
ant deploy
cd ..
```

V prípade, že sa kompilácia nevykonala úspešne, je potrebné skontrolovať verziu používanej javy: *java -version*. Nastaviť: *export JAVA_HOME=/usr*

B-3: Nastavenia

Skôr ako systém bude spustený, je potrebné vykonať nasledovné nastavenia.

Open IMS core

Zmeníme prednastavené IP adresy. Spustíme skript *configurator.sh*, ktorý nás navedie otázkami na zmenu prednastavenej IP adresy. Osobitne treba zmeniť nastavenia pre FHoSS, ktoré sa nachádzajú v *opt/OpenIMSCore/FHoSS/deploy*

Upravíme pôvodné nastavenia aplikačných serverov:

v */etc/ser* na 5.200.0.10 (Gold) a 5.200.0.20 (Silver)

Konfigurácia Gold aplikačného servera

```
debug=3           # debug level (cmd line: -dddddddddd)
fork=no           # pri starte vypisuj priamo do stderr (nebez v pozadi)
log_stderror=yes  # (cmd line: -E)

listen=5.200.0.10 # IP adresa, na ktorej bude server prijimat poziadavky
check_via=no      # (cmd. line: -v)
dns=no            # (cmd. line: -r)
rev_dns=no        # (cmd. line: -R)
fifo="/tmp/ser_fifo"

# ------------------ inicializacia modulov --------------------------
loadmodule "/usr/lib/ser/modules/mysql.so"
loadmodule "/usr/lib/ser/modules/sl.so"
loadmodule "/usr/lib/ser/modules/tm.so"
loadmodule "/usr/lib/ser/modules/rr.so"
loadmodule "/usr/lib/ser/modules/maxfwd.so"
loadmodule "/usr/lib/ser/modules/usrloc.so"
loadmodule "/usr/lib/ser/modules/textops.so"
loadmodule "/usr/lib/ser/modules/uri.so"
loadmodule "/usr/lib/ser/modules/nathelper.so"
loadmodule "/usr/lib/ser/modules/domain.so"
loadmodule "/usr/lib/ser/modules/xlog.so"

# ----------------- nastavenie modulov -----------------------------
modparam("domain", "db_mode",   0)  # non-caching rezim pre DB
modparam("domain", "db_url", "mysql://ser:ser@5.200.0.3/ser")

modparam("nathelper", "natping_interval", 0)
```

```
modparam("nathelper", "natping_interval", 30)
modparam("nathelper", "ping_nated_only", 1)

# poloha socketu rtpproxy servera
modparam("nathelper", "rtpproxy_sock", "unix:/var/run/rtpproxy/rtpproxy.sock")

# ----------------  smerovacie pravidla ----------------------------
route{

      # -----------------------------------------------------------------
      # Uvodna kontrola spravnosti spravy
      # -----------------------------------------------------------------
      # kontrolujeme ci sprava nie je preposielana
      # cez prilis vela serverov, alebo ci nie je nadmerne velka
      if (!mf_process_maxfwd_header("10")) {
            sl_send_reply("483","Too Many Hops");
            break;
      };
      if (msg:len >=  2048 ) {
            sl_send_reply("513", "Message too big");
            break;
      };
      xlog("L_ALERT","Initial checks complete %rm \n");

      # nas AS musi mat kontrolu nad celou relaciou, takze si do kazdej SIP
      # spravy vlozime pole record-route
      if (!method=="REGISTER") {
            record_route();
            xlog("L_ALERT","record_route()\n");
      }




      # -----------------------------------------------------------------
      # Ukoncenie hovoru
      # -----------------------------------------------------------------
      if (method=="BYE" || method=="CANCEL") {
            xlog("L_ALERT","call tear down section - unforce_rtp_proxy()\n");
            unforce_rtp_proxy();
      };


      # ine spravy v dialogu je potrebne spracovat podla tela SIP spravy
      if (loose_route()) {
            xlog("L_ALERT","loose_route()\n");
            append_hf("P-hint: rr-enforced\r\n");

            if (method=="INVITE") {
                  xlog("L_ALERT","loose-route force_rtp_proxy()\n");
                  # pre kazdy INVITE zmenime obsah SDP tak aby ukazoval na nas
                  force_rtp_proxy();
            };

            route(1);
            break;
      };

      if (!uri==myself) {
            append_hf("P-hint: outbound\r\n");
            route(1);
```

```
            break;
      };

      if (uri==myself) {
            xlog("L_ALERT","uri == myself\n");
            if (!uri==myself) {
                  append_hf("P-hint: outbound alias\r\n");
                  route(1);
                  break;
            };
      };
      append_hf("P-hint: usrloc applied\r\n");
      route(1);
}

# ----------------  spracovanie sprav ----------------------------
route[1]
{
      xlog("L_ALERT","route[1] %rm \n");

      # odpovede na SIP spravu spracuj v onreply_route[1]
      t_on_reply("1");

      # obsah SDP na INVITE treba tiez upravit
      if ( status =~ ".*180.*" || status =~ ".*200.*" )
      {
            xlog("L_ALERT","route[1] force_rtp_proxy() PRVE\n");


            # ak sprava obsauje SDP, uprav ho
            if (!search("^Content-Length:[ ]*0")) {
                  xlog("L_ALERT","route[1] force_rtp_proxy() DRUHE\n");
                  force_rtp_proxy();
            };
      };

      xlog("L_ALERT","route[1] fix_nated_contact()\n");
      # v pripade ze sa klient nachadza za NAT alebo ma viac IP adries,
      # uprav obsah Contact pola
      fix_nated_contact();

      # preposli upraveny INVITE na S-CSCF
      if (!t_relay_to_udp("5.200.0.2", "6060")) {
            # ak sa hovor nepodari zostavit a transakcia bola INVITE alebo ACK
            # tak ukonci relaciu

            if (method=="INVITE" || method=="ACK") {
                  xlog("L_ALERT","route[1] unforce_rtp_proxy()\n");
                  unforce_rtp_proxy();
            };

            sl_reply_error();
      };
}

# spracovanie odpovedi na INVITE tranzakciu
onreply_route[1] {

  xlog("L_ALERT","onreply_route[1]\n");
  xlog("L_ALERT","onreply_route[1] status=%rs \n");
```

```
  if ( status=~"(180)|(183)|2[0-9][0-9]" ) {
      xlog("L_ALERT","onreply_route[1] status matched\n");
      # ak sa v odpovedi nachadza SDP, uprav ho
      if (!search("^Content-Length:[ ]*0")) {
            force_rtp_proxy();
      xlog("L_ALERT","onreply_route[1] - force_rtp_proxy()\n");
    };
  };
}
```

Konfigurácia Silver aplikačného servera

Konfigurácia Silver aplikačného servera je až na IP adresu zhodná s konfiguráciou Gold servera. Pre úplnosť ju uvádzame:

```
debug=3           # debug level (cmd line: -dddddddddd)
fork=no           # pri starte vypisuj priamo do stderr (nebez v pozadi)
log_stderror=yes  # (cmd line: -E)

listen=5.200.0.20 # IP adresa, na ktorej bude server prijmat poziadavky
check_via=no      # (cmd. line: -v)
dns=no            # (cmd. line: -r)
rev_dns=no        # (cmd. line: -R)
fifo="/tmp/ser_fifo"

# ------------------ inicializacia modulov --------------------------
loadmodule "/usr/lib/ser/modules/mysql.so"
loadmodule "/usr/lib/ser/modules/sl.so"
loadmodule "/usr/lib/ser/modules/tm.so"
loadmodule "/usr/lib/ser/modules/rr.so"
loadmodule "/usr/lib/ser/modules/maxfwd.so"
loadmodule "/usr/lib/ser/modules/usrloc.so"
loadmodule "/usr/lib/ser/modules/textops.so"
loadmodule "/usr/lib/ser/modules/uri.so"
loadmodule "/usr/lib/ser/modules/nathelper.so"
loadmodule "/usr/lib/ser/modules/domain.so"
loadmodule "/usr/lib/ser/modules/xlog.so"

# ----------------- nastavenie modulov -----------------------------
modparam("domain", "db_mode",   0)  # non-caching rezim pre DB
modparam("domain", "db_url", "mysql://ser:ser@5.200.0.3/ser")

modparam("nathelper", "natping_interval", 0)
modparam("nathelper", "natping_interval", 30)
modparam("nathelper", "ping_nated_only", 1)

# poloha socketu rtpproxy servera
modparam("nathelper", "rtpproxy_sock", "unix:/var/run/rtpproxy/rtpproxy.sock")

# ----------------  smerovacie pravidla ----------------------------
route{

      # ----------------------------------------------------------------
      # Uvodna kontrola spravnosti spravy
      # ----------------------------------------------------------------
      # kontrolujeme ci sprava nie je preposielana
      # cez prilis vela serverov, alebo ci nie je nadmerne velka
      if (!mf_process_maxfwd_header("10")) {
            sl_send_reply("483","Too Many Hops");
            break;
```

```
	};
	if (msg:len >=  2048 ) {
		sl_send_reply("513", "Message too big");
		break;
	};
	xlog("L_ALERT","Initial checks complete %rm \n");

	# nas AS musi mat kontrolu nad celou relaciou, takze si do kazdej SIP
	# spravy vlozime pole record-route
	if (!method=="REGISTER") {
		record_route();
		xlog("L_ALERT","record_route()\n");
	}




	# ----------------------------------------------------------------
	# Ukoncenie hovoru
	# ----------------------------------------------------------------
	if (method=="BYE" || method=="CANCEL") {
		xlog("L_ALERT","call tear down section - unforce_rtp_proxy()\n");
		unforce_rtp_proxy();
	};


	# ine spravy v dialogu je potrebne spracovat podla tela SIP spravy
	if (loose_route()) {
		xlog("L_ALERT","loose_route()\n");
		append_hf("P-hint: rr-enforced\r\n");

		if (method=="INVITE") {
			xlog("L_ALERT","loose-route force_rtp_proxy()\n");
			# pre kazdy INVITE zmenime obsah SDP tak aby ukazoval na nas
			force_rtp_proxy();
		};

		route(1);
		break;
	};

	if (!uri==myself) {
		append_hf("P-hint: outbound\r\n");
		route(1);
		break;
	};

	if (uri==myself) {
		xlog("L_ALERT","uri == myself\n");
		if (!uri==myself) {
			append_hf("P-hint: outbound alias\r\n");
			route(1);
			break;
		};
	};
	append_hf("P-hint: usrloc applied\r\n");
	route(1);
}

# ---------------  spracovanie sprav ----------------------------
route[1]
```

```
{
      xlog("L_ALERT","route[1] %rm \n");

      # odpovede na SIP spravu spracuj v onreply_route[1]
      t_on_reply("1");

      # obsah SDP na INVITE treba tiez upravit
      if ( status =~ ".*180.*" || status =~ ".*200.*" )
      {
            xlog("L_ALERT","route[1] force_rtp_proxy() PRVE\n");


            # ak sprava obsauje SDP, uprav ho
            if (!search("^Content-Length:[ ]*0")) {
                  xlog("L_ALERT","route[1] force_rtp_proxy() DRUHE\n");
                  force_rtp_proxy();
            };
      };

      xlog("L_ALERT","route[1] fix_nated_contact()\n");
      # v pripade ze sa klient nachadza za NAT alebo ma viac IP adries,
      # uprav obsah Contact pola
      fix_nated_contact();

      # preposli upraveny INVITE na S-CSCF
      if (!t_relay_to_udp("5.200.0.2", "6060")) {
            # ak sa hovor nepodari zostavit a transakcia bola INVITE alebo ACK
            # tak ukonci relaciu

            if (method=="INVITE" || method=="ACK") {
                  xlog("L_ALERT","route[1] unforce_rtp_proxy()\n");
                  unforce_rtp_proxy();
            };

            sl_reply_error();
      };
}

# spracovanie odpovedi na INVITE tranzakciu
onreply_route[1] {

  xlog("L_ALERT","onreply_route[1]\n");
  xlog("L_ALERT","onreply_route[1] status=%rs \n");

  if ( status=~"(180)|(183)|2[0-9][0-9]" ) {
      xlog("L_ALERT","onreply_route[1] status matched\n");
      # ak sa v odpovedi nachadza SDP, uprav ho
      if (!search("^Content-Length:[ ]*0")) {
            force_rtp_proxy();
      xlog("L_ALERT","onreply_route[1] - force_rtp_proxy()\n");
    };
  };
}
```

Konfigurácia DNS servera

Konfigurácia DNS servera je nasledovná:

```
cat /etc/bind/named.conf.local

zone "open-ims.test" IN {
```

```
        type master;
        file "pri/open-ims.dnszone";
        notify no;
};
cat /etc/bind/pri/open-ims.dnszone


$ORIGIN open-ims.test.
$TTL 1W
@                       1D IN SOA       5.200.0.1. root.5.200.0.1. (
                                        2010031001      ; serial
                                        3H              ; refresh
                                        15M             ; retry
                                        1W              ; expiry
                                        1D )            ; minimum

                        1D IN NS        ns
ns                      1D IN A         5.200.0.1

pcscf                   1D IN A         5.200.0.2
_sip.pcscf              1D SRV 0 0 4060 pcscf
_sip._udp.pcscf         1D SRV 0 0 4060 pcscf
_sip._tcp.pcscf         1D SRV 0 0 4060 pcscf


icscf                   1D IN A         5.200.0.2
_sip                    1D SRV 0 0 5060 icscf
_sip._udp               1D SRV 0 0 5060 icscf
_sip._tcp               1D SRV 0 0 5060 icscf

open-ims.test.          1D IN A         5.200.0.2
open-ims.test.          1D IN NAPTR 10 50 "s" "SIP+D2U" ""      _sip._udp
open-ims.test.          1D IN NAPTR 20 50 "s" "SIP+D2T" ""      _sip._tcp


scscf                   1D IN A         5.200.0.2
_sip.scscf              1D SRV 0 0 6060 scscf
_sip._udp.scscf         1D SRV 0 0 6060 scscf
_sip._tcp.scscf         1D SRV 0 0 6060 scscf


trcf                    1D IN A         5.200.0.2
_sip.trcf               1D SRV 0 0 3060 trcf
_sip._udp.trcf          1D SRV 0 0 3060 trcf
_sip._tcp.trcf          1D SRV 0 0 3060 trcf


bgcf                    1D IN A         5.200.0.2
_sip.bgcf               1D SRV 0 0 7060 bgcf
_sip._udp.bgcf          1D SRV 0 0 7060 bgcf
_sip._tcp.bgcf          1D SRV 0 0 7060 bgcf


mgcf                    1D IN A         5.200.0.2
_sip.mgcf               1D SRV 0 0 8060 mgcf
_sip._udp.mgcf          1D SRV 0 0 8060 mgcf
_sip._tcp.mgcf          1D SRV 0 0 8060 mgcf


hss                     1D IN A         5.200.0.3
```

```
ue                      1D IN A         5.200.0.2

presence                1D IN A         5.200.0.2

goldproxy                A              5.200.0.10
silverproxy              A              5.200.0.20

iptv                    1D IN A         5.200.0.1

pcrf                    1D IN A         5.200.0.2
clf                     1D IN A         5.200.0.2
```

DNS server po konfigurácii reštartujeme

```
/etc/init.d/bind9 restart
```

Databázový server MySQL

Do databázy sme vložili vzorové údaje. Následné príkazy sú vykonané na stroji HSS.

```
mysql –u root –p –h localhost < ser_ims/cfg/icscf.sql
mysql –u root –p –h localhost < FHoSS/scripts/hss_db.sql
mysql –u root –p –h localhost < FHoSS/scripts/userdata.sql
```

B-4: Spustenie komponentov

Komponenty spúšťame nasledovne:

```
cd opt/OpenIMSCore/
S-CSCF:  ./scscf.sh
P-CSCF:  ./pcscf.sh
I-CSCF:  ./icscf.sh

cd opt/OpenIMSCore/FHoSS/deploy
./startup.sh export JAVA_HOME=/usr
```

B-5: Meranie a testovanie

V tejto časti príručky popíšeme meranie a testovanie IMS zo strany klientov.

IMS klienti

Na testovanie boli použití 3 rôzni softvéroví klienti:

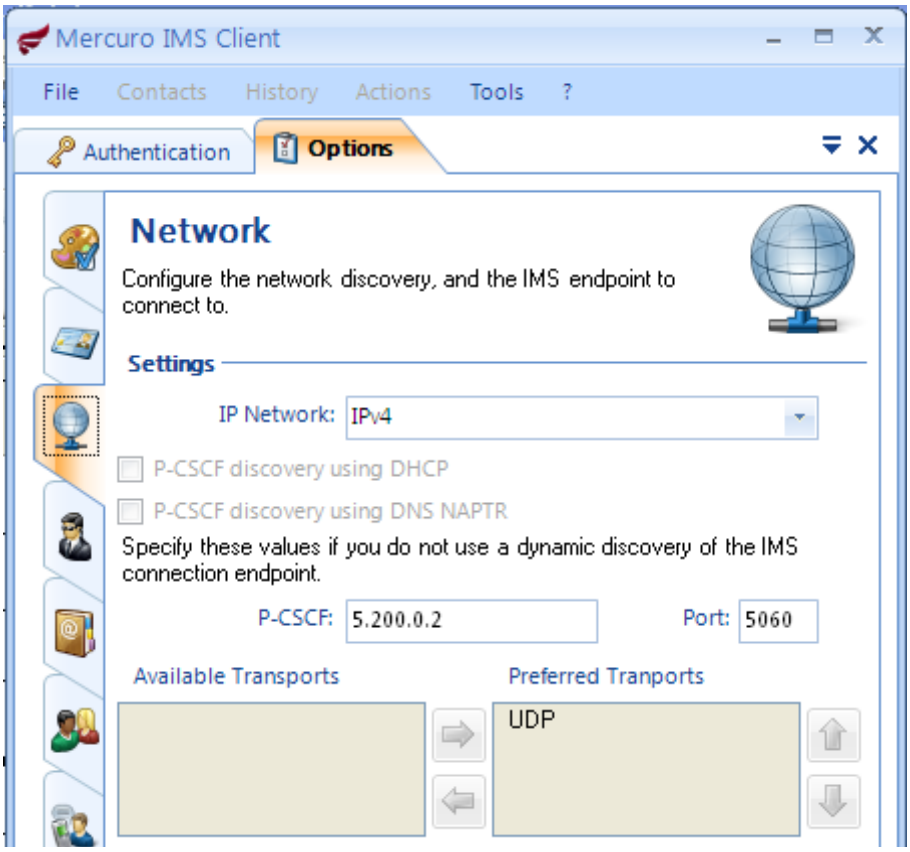
● Monster – funguje na všetkých operačných systémoch
http://www.monster-the-client.org

● Mercuro – najstabilnejšia verzia
http://www.mercuro.net

● UCT IMS Client – len linux

http://uctimsclient.berlios.de

Uvádzame nastavenia IMS klienta Mercuro na obrázkoch Obr. B-2 a B-3. V iných klientoch sú nastavenia analogické:



Obr. B-2: Návrhová obrazovka pre nastavenia IMS klienta Mercuro

Obr. B-3: Návrhová obrazovka pre konfiguráciu IMS klienta Mercuro

SIPp

SIPp je voľne dostupný testovací nástroj protokolu SIP. Umožňuje definovanie vlastných scenárov a vytváranie RTP prúdov vhodných na testovanie zátaže a výkonu komunikačnej infraštruktúry. Vlastné scenáre sa definujú v značkovacom jazyku XML.

Pre potreby projektu sme vytvorili štyri scenáre, ktoré nám umožnili vytvoriť vhodné prostredie pre meranie výkonu zaťaženej sieti.

Program funguje v režime klient-server.

Inštancia programu SIPp v režime server prijíma a odpovedá na požiadavky ako používateľ *Alice*.

Komunikáciu inicializuje inštancia v režime klient ako používateľ *Daniel*.

XML súbor inštancie v režime klient spustíme príkazom:

```
sipp -sf <meno súbor> <cielová IP adresa:cielový port> -m <počet vykonaní scenára> -i <zdrojová IP adresa> -l <maximálny počet súbežných hovorov> -r <maximálny počet hovor vytvorených za jednu sekundu>
```

Nasleduje ukážkový XML výpis:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="rozhovor Daniel a Alice, rezim: klient">

<!-- Posli INVITE, v pripade nepotvrdenia posli o 500ms opat.
-->
<send retrans="500">
<![CDATA[
INVITE sip:alice@open-ims.test SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
Max-Forwards: 20
Route: <sip:orig@scscf.open-ims.test:6060;lr>
P-Preferred-Identity: <sip:daniel@open-ims.test>
Privacy: none
P-Access-Network-Info: 3GPP-UTRAN-TDD;utran-cell-id-3gpp=C359A3913B20E
From: <sip:daniel@open-ims.test>;tag=[call_number]
To: <alice@open-ims.test>
Call-ID: [call_id]
CSeq: 10 INVITE
Supported: 100rel
Contact: <sip:daniel@[local_ip]:[local_port]>
User-Agent: Sipp v1.1-TLS, version 20061124
Allow: ACK, BYE, CANCEL, INVITE, REFER, OPTIONS, INFO, REGISTER, NOTIFY, UPDATE,
SUBSCRIBE, PRACK
Content-Type: application/sdp
Content-Length: [len]

v=0
o=user1 53655765 2353687637 IN IP4 [local_ip]
s=-
c=IN IP4 [local_ip]
t=0 0
m=audio 30000 RTP/AVP 0 8
a=rtpmap:0 PCMU/8000
a=sendrecv
]]>
</send>

<!-- Prijate spravy s nasedovnymi kodmi je dobrovolne a  nebude povazovane za
spravu mimo dialogu, co by ukoncilo vykonavanie programu. Parameterom next
definujeme skok v pripade prijatia danej spravy. Miesto skoku je definovane
znackou label.                      -->
<recv response="100" optional="true">
</recv>
```

```
<recv response="180" optional="true">
</recv>

<recv response="183" optional="true">
</recv>


<recv response="403" optional="true" next="1">
</recv>

<recv response="404" optional="true" next="1">
</recv>

<recv response="408" optional="true" next="1">
</recv>

<!-- Prijmi odpoved 200, a uchovaj hlavicku Record-Route:.
       -->
<recv response="200" rrs="true">
</recv>

<!-- Posli ACK, miesto pola [routes] vloz uchovanu hlavicku Record-Route:.
      -->
<send crlf="true">
<![CDATA[
ACK [next_url] SIP/2.0
[last_Via:]
Max-Forwards: 20
[routes]
From: <sip:daniel@open-ims.test>;tag=[call_number]
[last_To:]
Call-ID: [call_id]
CSeq: 10 ACK
Content-Length: 0
]]>
</send>

<!-- Prehraj nahrany PCAP subor (zaciatok RTP prudu).
      -->
<nop>
  <action>
    <exec play_pcap_audio="g711a.pcap"/>
  </action>
</nop>

<!-- Pozastav vykonavanie na 8 sekund, co je asi dlzka suboru g711a.pcap.
      -->
<pause milliseconds="8000" crlf="true" />

<!-- Posli BYE, v pripade nepotvrdenia posli o 500ms opat.             -->
<send retrans="500">
<![CDATA[
BYE sip:[next_url] SIP/2.0
[last_Via:]
Max-Forwards: 20
[routes]
From: <sip:daniel@open-ims.test>;tag=[call_number]
[last_To:]
Call-ID: [call_id]
CSeq: 11 BYE
Contact: <sip:daniel@[local_ip]:[local_port]>
```

```
Content-Length: 0
]]>
</send>
<!-- Ak prijmes 200 OK skoc na navestie 2 a tak ukonči rozhovor.             -->
<recv response="200" crlf="true" next="2">
</recv>

<!--Bola prijata chybova sprava, odpovedaj ACK.                          -->

<label id="1"/>

<send crlf="true">
<![CDATA[
ACK sip:alice@open-ims.test SIP/2.0
[last_Via:]
Max-Forwards: 20
From: <sip:daniel@open-ims.test>;tag=[call_number]
[last_To:]
Call-ID: [call_id]
CSeq: 10 ACK
Content-Length: 0
]]>
</send>

<label id="2"/>

<ResponseTimeRepartition value="10, 20"/>

<CallLengthRepartition value="10"/>

</scenario>
```

XML súbor inštancie v režime server spúšťame príkazom:

```
sipp -sf <meno súbor> -i <IP adresa, na ktorej server počúva>
```

Uvádzame ukážkový XML súbor:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="rozhovor Daniel a Alice, rezim: server">

<!-- Prijmi správu INVITE.
       -->
<recv request="INVITE">
</recv>

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

```
<!-- Pozastav vykonavanie na 2000ms.
-->
<pause milliseconds="2000"/>

<!-- Posli 200OK, v pripade nepotvrdenia posli o 500ms opat.
       -->
<send retrans="500">
<![CDATA[
SIP/2.0 200 OK
[last_Via:]
[last_Record-Route:]
[last_From:]
[last_To:];tag=[call_number]
[last_Call-ID:]
[last_CSeq:]
Contact: <sip:alice@[local_ip]:[local_port]>
Allow: INVITE,REGISTER,ACK,BYE,INFO,REFER,NOTIFY,SUBSCRIBE,MESSAGE,CANCEL
Content-Type: application/sdp
Content-Length: [len]

v=0
o=- 53655765 2353687637 IN IP4 [local_ip]
s=-
c=IN IP4 [media_ip]
t=0 0
m=audio 40000 RTP/AVP 8 0 18
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:18 G729/8000
]]>
</send>

<!-- Posli spravu ACK.                                                    -->
<recv request="ACK" crlf="true">
</recv>

<!-- Prehraj nahrany PCAP subor (zaciatok RTP prudu).
      -->
<nop>
      <action>
            <exec play_pcap_audio="g711a.pcap"/>
      </action>
</nop>

<!-- Prijmi spravu BYE.                                                                  -->
<recv request="BYE">
</recv>

<!-- Posli odpoved 200 OK
      -->
<send>
<![CDATA[
SIP/2.0 200 OK
[last_Via:]
[last_From:]
[last_To:]
[last_Call-ID:]
[last_CSeq:]
Content-Length: 0
]]>
</send>
```

```
</scenario>
```

Pred exekúciou horeuvedených scenárov je potrebné klientov vopred zaregistrovať v jadre IMS. Registrácia nemôže byť súčasťou horeuvedených scenárov, pretože program SIPp neumožňuje v jednom scenári použiť rôzne hodnoty polí *Call-ID*. Následné scenáre prihlásia daných používateľov.

Registráciu používateľa Daniel spúšťame príkazom

```
sipp -sf <meno súbor> <cielová IP adresa:cielový port> -m 1 -i <zdrojová IP adresa>
```

Vzorový výpis XML súboru je nasledovný:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="registracia Daniel">

<!-- Posli poziadavku REGISTER.                                          -->

<send retrans="500">
<![CDATA[
REGISTER sip:open-ims.test SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
Max-Forwards: 20
From: "daniel" <sip:daniel@open-ims.test>;tag=[call_number]
To: "daniel" <sip:daniel@open-ims.test>
P-Access-Network-Info: 3GPP-UTRAN-TDD;utran-cell-id-3gpp=C359A3913B20E
Call-ID: reg///[call_id]
CSeq: 1 REGISTER
Contact: <sip:daniel@[local_ip]:[local_port]>
Expires: 3600
Content-Length: 0
User-Agent: Sipp v1.1-TLS, version 20061124
Authorization: Digest username="daniel@open-ims.test", realm="open-ims.test"
Supported: path
]]>
</send>

<!-- Prijmi odpoved 401, parameterom auth ulozime hodnotu pola Proxy-
Authenticate a nasledne pouzijeme polom [authentication].
                  -->

<recv response="401" auth="true" rtd="true">
<action>
<ereg regexp=".*" search_in="hdr" header="Service-Route" assign_to="1" />
</action>
</recv>

<!-- Posli poziadavku REGISTER, pri autentifikacii pouzi uvedene prihlasovacie
meno a heslo.
      -->
<send retrans="500">
<![CDATA[
REGISTER sip:open-ims.test SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
Route: [$1]
Max-Forwards: 20
From: "daniel" <sip:daniel@open-ims.test>;tag=[call_number]
To: "daniel" <sip:daniel@open-ims.test>
```

```
P-Access-Network-Info: 3GPP-UTRAN-TDD;utran-cell-id-3gpp=C359A3913B20E
Call-ID: reg///[call_id]
CSeq: 2 REGISTER
Contact: <sip:daniel@[local_ip]:[local_port]>
Expires: 3600
Content-Length: 0
User-Agent: Sipp v1.1-TLS, version 20061124
[authentication username=daniel@open-ims.test password=daniel]
Supported: path
]]>
</send>

<!-- Prijmi odpoved 200OK.                                          -->
<recv response="200">
</recv>

<ResponseTimeRepartition value="10, 20"/>
<CallLengthRepartition value="10"/>

</scenario>
```

Registráciu používateľa Alice spúšťame príkazom

```
sipp -sf <meno súbor> <cielová IP adresa:cielový port> -m 1 -i <zdrojová IP adresa>
```

Vzorový výpis XML súboru je nasledovný:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="Registracia Alice">

<!-- Posli poziadavku REGISTER.                                     -->
<send retrans="500">
<![CDATA[
REGISTER sip:open-ims.test SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
Max-Forwards: 20
From: "alice" <sip:alice@open-ims.test>;tag=[call_number]
To: "alice" <sip:alice@open-ims.test>
P-Access-Network-Info: 3GPP-UTRAN-TDD;utran-cell-id-3gpp=C359A3913B20E
Call-ID: reg///[call_id]
CSeq: 1 REGISTER
Contact: <sip:alice@[local_ip]:[local_port]>
Expires: 300
Content-Length: 0
User-Agent: Sipp v1.1-TLS, version 20061124
Authorization: Digest username="alice@open-ims.test", realm="open-ims.test"
Supported: path
]]>
</send>


<!-- Prijmi odpoved 401, parameterom auth ulozime hodnotu pola Proxy-
Authenticate a nasledne pouzijeme polom [authentication].
                -->
<recv response="401" auth="true" rtd="true">
<action>
<ereg regexp=".*" search_in="hdr" header="Service-Route" assign_to="1" />
</action>
</recv>
```

```
<!-- Posli poziadavku REGISTER, pri autentifikacii pouzi uvedene prihlasovacie
meno a heslo.
      -->
<send retrans="500">
<![CDATA[
REGISTER sip:open-ims.test SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
Route: [$1]
Max-Forwards: 20
From: "alice" <sip:alice@open-ims.test>;tag=[call_number]
To: "alice" <sip:alice@open-ims.test>
P-Access-Network-Info: 3GPP-UTRAN-TDD;utran-cell-id-3gpp=C359A3913B20E
Call-ID: reg///[call_id]
CSeq: 2 REGISTER
Contact: <sip:alice@[local_ip]:[local_port]>
Expires: 300
Content-Length: 0
User-Agent: Sipp v1.1-TLS, version 20061124
[authentication username=alice@open-ims.test password=alice]
Supported: path
]]>
</send>

<!-- Prijmi odpoved 200OK.                                                  -->
<recv response="200">
</recv>

<ResponseTimeRepartition value="10, 20"/>
<CallLengthRepartition value="10"/>

</scenario>
```

Príloha C: Elektronický nosič

Štruktúra priečinkov a podpriečinkov elektronického nosiča je nasledujúca.

- \Bind – konfigurácia Bind servera
- \Cisco_configuration – naša odporúčaná konfigurácia smerovačov
- \GoldAS – konfigurácia Gold aplikačného servera
- \HSS – konfigurácia HSS databázy
- \SilverAS – konfigurácia Silver aplikačného servera
- \SIPp - konfigurácia aplikácie SIPp
- \Stranka_timu_11 – naša webová stránka

Obsah priečinkov nezobrazujeme pre veľké množstvo súborov. Bližšie informácie sú priamo na elektronickom nosiči.