Masarykova univerzita
Fakulta informatiky

RTSP server pre paketové zrkadlo

Bakalárska práca

Boris Parák

Brno, jar 2010

Prehlásenie

Prehlasujem, že táto bakalárska práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Boris Parák

Vedúci práce: RNDr. Miloš Liška

Poďakovanie

Ďakujem vedúcemu práce, RNDr. Milošovi Liškovi, za odbornú pomoc pri vypracovávaní tejto bakalárskej práce a Mgr. Pavlovi Troubilovi za to, že sa podelil o svoje skúsenosti s vývojom modulov pre paketové zrkadlo.

Zhrnutie

Cieľom tejto práce je implementácia *RTSP* serveru vo forme modulu pre modulárne paketové zrkadlo *RUM2* v jazyku *C*, ktorý užívateľom umožní kontaktovať paketové zrkadlo priamo z multimediálneho prehrávača a zjednoduší tak jeho používanie. Súčasťou práce je popis architektúry paketového zrkadla *RUM2*, popis fungovania sieťových protokolov *RAP*, *RTSP*, *SDP* a analýza existujúcich implementácií protokolu *RTSP* na strane serverov i klientov.

Kľúčové slová

paketové zrkadlo, modul, RTSP, SDP, RAP, jazyk C, skupinová komunikácia, multimédiá, počítačová sieť, multicast

Obsah

1 Úvod

Pokroky v oblasti počítačových sietí a lepšia dostupnosť širokopásmových pripojení k Internetu pre domácnosti umožnili rýchly rozvoj multimediálnej komunikácie a jej popularizáciu medzi bežnými užívateľmi. Prenos obrazu a zvuku na veľké vzdialenosti v reálnom čase už nepatrí len do výskumných laboratórií a video-konferencie či prenosy vopred zaznamenaného videa na požiadanie je možné realizovať pomocou voľne dostupných nástrojov bez hlbšej znalosti ich fungovania.

Prenos multimediálnych dát v počítačovej sieti je zvyčajne realizovaný medzi jedným zdrojom dát a skupinou klientov. Dáta cestujúce cez sieť k jednotlivým klientom sú identické, preto je rozumné využiť transportné mechanizmy, ktoré minimalizujú množstvo duplicitných dát a zvýšia tak efektívnosť prenosu. Jedným z takýchto mechanizmov je *skupinová komunikácia*[1] (ďalej len *multicast*) pomocou paketov protokolu IP[2] (ďalej len paket).

Multicast je špeciálnym typom vysielania v počítačovej sieti. Cieľová adresa v hlavičke paketu nepatrí jednému fyzickému uzlu v sieti (tzv. *unicast*), ale označuje skupinu uzlov. Zdroj dát pošle do siete len jeden paket, ten je duplikovaný až v prípade nutnosti na uzloch, ktoré sú čo najbližšie ku klientským staniciam. Zdroj teda nemusí poznať počet klientov a dáta sú čo najdlhšie prenášané len v jednej kópii.

Komplikácie nastávajú, keď je potrebné zabezpečiť multicast v sieti, ktorá ho z bezpečnostných alebo technických dôvodov nepodporuje. Typickým príkladom sietí s obmedzenou alebo žiadnou podporou multicastu sú siete využívajúce *preklad sieťových adries*[3] alebo prostredia, v ktorých sú pakety smerujúce na skupinové adresy odfiltrované smerovačmi snažiacimi sa zabrániť preťaženiu siete a potenciálnym útokom na preťažené uzly. Pri využití multicastu tiež nastávajú problémy s udržovaním kvality služby a zabezpečením prenášaných dát.

Možným riešením je predstavenie konceptu *modulárneho paketového zrkadla* simulujúceho skupinovú komunikáciu využívajúc spojenia inicializované koncovými užívateľmi [9]. Paketové zrkadlo je aktívnym uzlom v počítačovej sieti, ktorý rozosiela dáta prichádzajúce z vopred definovaného zdroja užívateľom, ktorí ho kontaktovali pomocou transportného protokolu *UDP*[4] a boli zaradení do zoznamu klientov. Od zdroja k zrkadlu teda sieťou

1. RFC 1112 - Host Extensions for IP Multicasting
2. RFC 791 - Internet Protocol
3. RFC 1631 - The IP Network Address Translator
4. RFC 768 - User Datagram Protocol

prechádza len jedna kópia dát a duplikovanie prebieha až na zrkadle.

Modulárna architektúra umožňuje rozšírenie funkcionality o smerovanie, zabezpečenie, moderovanie alebo ďalšie spracovanie dát prechádzajúcich zrkadlom. Jednotlivé moduly sú rozdelené do tried definovaných v *Reflector Administration Protocol* [8] (ďalej len *RAP*), ktorý predpisuje formát správ pre komunikáciu so zrkadlom.

Implementáciou tohto konceptu je paketové zrkadlo *RUM2* [10]. V súčasnosti je možné zaradiť nového klienta do zoznamu klientov len pomocou *UDP* paketu periodicky zasielaného na adresu zrkadla. Tento postup je nutné dodržať napríklad pri sledovaní prednášok Fakulty informatiky Masarykovej univerzity v reálnom čase s pomocou nástroja *ping* [13]. Komplikuje to ovládanie zrkadla pre menej skúsených užívateľov.

Cieľom tejto bakalárskej práce je implementácia *Real Time Streaming Protocol* [16] (ďalej len *RTSP*) servera ako modulu paketového zrkadla *RUM2*. Užívateľom to umožní kontaktovať zrkadlo priamo z multimediálneho prehrávača pomocou štandardizovaného protokolu. Nebude nutné využívať špeciálnu aplikáciu periodicky kontaktujúcu zrkadlo a spúšťať prehrávač schopný prehrávať prichádzajúce *UDP* dáta.

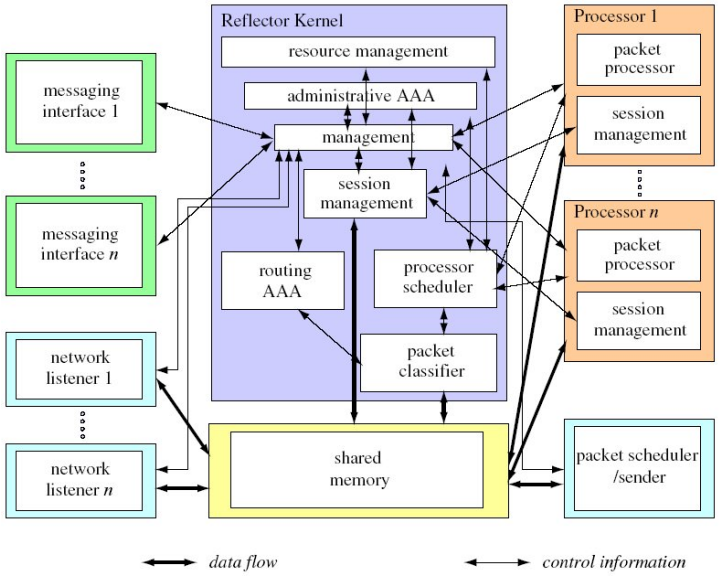
Jadro práce je rozdelené na 3 kapitoly. Prvá sa venuje základným princípom fungovania zrkadla, protokolom *RTSP* a *SDP*, rozhraniu pre komunikáciu s modulmi v *RUM2* a triedam modulov. Druhá kapitola sa venuje analýze existujúcich serverových a klientských riešení, tretia vlastnej implementácii, problémom s ňou spojenými a testovaniu modulu. Záver nahliada na možnosti budúceho rozširovania paketového zrkadla.

Výsledná implementácia *RTSP* modulu sa nachádza na priloženom médiu vo forme zdrojového kódu, vopred preloženej knižnice pripravenej na použitie a pomocného filtrovacieho modulu. K obom modulom je priložená kompletná dokumentácia popisujúca ich použitie a fungovanie.

2 Teoretický základ

2.1 Architektúra paketového zrkadla

Paketové zrkadlo *RUM2* je modulárny nástroj schopný replikácie a zpracovávania dát, ktoré ním prechádzajú. Podporuje kontrolu prístupu, autentizáciu užívateľov, smerovanie, filtrovanie obsahu paketov a dynamické zmeny v konfigurácii pomocou textového administračného protokolu. Je napísané v jazyku *C* [11] a umožňuje dynamické pripájanie modulov pomocou funkcií dostupných v knižnici *ltdl*, ktorá je súčasťou nástroja *Libtool* [3] uľahčujúceho tvorbu dynamických knižníc. Tieto vlastnosti ho predurčujú na využitie v oblasti video-konferencií a všeobecne na distribúciu videa v počítačových sieťach.



Obr. 2.1: Architektúra modulárneho paketového zrkadla [9].

Architektúra znázornená na obrázku 2.1 rozdeľuje zrkadlo *RUM2* na šesť hlavných častí. Jadro zrkadla obsahuje statické moduly zodpovedné za správu pamäte, administráciu, plánovanie, komunikáciu s externými modulmi či klasifikovanie a smerovanie prichádzajúcich paketov. Zdieľaná pamäť obsahuje dátové štruktúry potrebné pre fungovanie zrkadla a pakety pripravené na spracovanie alebo odoslanie užívateľom. Dáta sú modulom predávané pomocou referencií, nenastáva zbytočné kopírovanie a minimalizuje sa réžia a oneskorenie medzi zdrojom dát a užívateľom. Moduly *listener* načúvajúce na *UDP* portoch zabezpečujú prijímanie dát a ich ukladanie do vyrovnávacej

pamäte. Každý modul má pridelený jeden port a pridáva dáta do jednej relácie, ktorá je týmto portom jednoznačne identifikovaná. Moduly *processor* sú určené na spracovávanie prichádzajúcich dát, ich činnosť je plánovaná jadrom zrkadla. Paket môže prejsť viacerými spracovateľskými modulmi.

Komunikačné rozhranie zrkadla je realizované pomocou skupiny modulov načúvajúcich na *TCP*[5] portoch rozdelených podľa komunikačných protokolov, ktoré akceptujú. Príkladom sú protokoly *HTTP*[6], *SOAP*[7], *RPC*[8] alebo ďalej popisovaný špecializovaný protokol *RAP*. Jediným výstupným modulom *RUM2* je plánovač paketov. Je zodpovedný za odosielanie spracovaných paketov užívateľom. Kópie paketov sú odosielané všetkým užívateľom na základe ich príslušnosti k existujúcim reláciám.

2.1.1 Moduly

Každý modul zrkadla *RUM2* je jednoznačne identifikovaný pomocou triedy a mena. Táto konvencia umožňuje predávanie správ a požiadaviek jednotlivým modulom a zároveň sprehľadňuje záznamy o činnosti zrkadla.

Protokol *RAP* vo verzii 1.0 definuje nasledujúce triedy modulov paketového zrkadla [8]:

reflector
: Virtuálna trieda, ktorá neobsahuje žiadne moduly, je určená na skupinový prístup k modulom určitej triedy alebo ku všetkým pripojeným modulom naraz.

listener
: Trieda obsahujúca všetky moduly prijímajúce dáta zo siete, identifikátory jednotlivých modulov obsahujú meno protokolu a port, na ktorý sú viazané.

processor
: Trieda modulov vykonávajúcich transformácie a filtrovanie dát, meno modulu popisuje jeho účel a poradové číslo. Špeciálnym modulom je plánovač identifikovaný ako *master*.

5. RFC 793 - Transmission Control Protocol
6. RFC 2068 - Hypertext Transfer Protocol
7. RFC 3288 - Using the Simple Object Access Protocol (SOAP) in BEEP
8. RFC 1050 - RPC: Remote Procedure Call Protocol specification

sender
: Obsahuje jediný modul identifikovaný menom *master*, ide o plánovač paketov zabezpečujúci odosielanie dát užívateľom.

aaa
: Obsahuje moduly zabezpečujúce autorizáciu, autentizáciu a správu užívateľov. Moduly v tejto triede majú na starosti napríklad kontrolu prístupu pomocou *ACL* (*Access Control List*).

management
: Trieda tvoriaca jadro zrkadla, patria do nej tri základné moduly. Hlavný modul označovaný ako *master* ovládajúci spúšťanie, správu a zastavovanie ostatných modulov zrkadla. Správca pamäte označovaný ako *memory* a správca relácií nazvaný *session*.

msg-interface
: Trieda modulov komunikačného rozhrania, jednotlivé moduly sú pomenované podľa protokolu a portu, prípadne iného identifikátora komunikačného kanálu. Moduly patriace do tejto triedy umožňujú kontrolu zrkadla za chodu pomocou správ protokolu *RAP*.

Základné moduly patriace do tried *management*, *aaa* a *sender* nie je možné spúšťať a zastavovať, sú statickou súčasťou zrkadla. U zvyšných tried je možné zvoliť dynamické pripájanie modulov vo forme knižníc so špecifickým rozhraním.

2.1.2 Rozhranie pre komunikáciu s modulmi

Pripájanie dynamických modulov je realizované pomocou knižnice *ltdl*, ktorá obaľuje linkovací mechanizmus *dlopen* a uľahčuje tak jeho použitie v programe. Moduly paketového zrkadla sú vlastne dynamicky linkované knižnice, v ktorých je vstupným bodom funkcia *initialize*. Táto funkcia pripraví štruktúru obsahujúcu informácie potrebné pre spustenie. Okrem iného zaregistruje funkcie predstavujúce rozhranie pre komunikáciu s modulom.

Rozhranie poskytuje nasledujúcu funkcionalitu popísanú podrobne v hlavičkovom súbore *module.h* zrkadla *RUM2* [10]:

name
: Funkcia zostavujúca meno modulu z parametrov obsiahnutých v požiadavke na spustenie modulu. Môže ísť napríklad o port alebo IP adresu rozhrania. Nepatrí medzi povinné funkcie.

conflicts
: Funkcia vytvárajúca zoznam potenciálne konfliktných modulov. V prípade, že zrkadlo nájde medzi práve aktívnymi modulmi konfliktný modul, zastaví spúšťanie a oznámi túto skutočnosť užívateľovi. Nepatrí medzi povinné funkcie.

init
: Funkcia zabezpečujúca inicializáciu modulu, alokovanie zdrojov pre interné dátové štruktúry, otvorenie súborov, prípravu sieťových rozhraní a prípadné registrovanie modulu na čakanie v radách. Povinná funkcia, po jej úspešnom vykonaní musí byť modul schopný štartu.

main
: Funkcia obsahujúca hlavnú slučku zvyčajne zastavenú externou požiadavkou. Povinná funkcia, modul je spustený v novom vlákne.

stop
: Funkcia volaná po zastavení modulu, môže byť použitá na uvoľnenie zdrojov obsadených počas vykonávania hlavnej slučky. Nepatrí medzi povinné funkcie.

clean
: Modul je možné zastaviť alebo reštartovať. Funkcia *clean* zabezpečuje uvoľnenie všetkých zdrojov alokovaných modulom v prípade zastavenia a zachovanie zdrojov alokovaných pred vykonaním *init* v prípade reštartu. Povinná funkcia.

push_data
: Funkcia poskytujúca synchrónne odovzdanie dát modulu. Nepatrí medzi povinné funkcie.

push_message
: Funkcia poskytujúca synchrónnu komunikáciu s modulom pomocou správ protokolu RAP. Nepatrí medzi povinné funkcie.

events
: Funkcia umožňujúca modulu získavať informácie o globálnych udalostiach. Nepatrí medzi povinné funkcie.

config
: Funkcia zabezpečujúca uloženie aktuálnej konfigurácie modulu vo forme správ protokolu RAP. Umožňuje opätovné spustenie zrkadla s rovnakou konfiguráciou. Povinná funkcia.

Okrem definovaných funkcií komunikačného rozhrania môže modul obsahovať neobmedzený počet pomocných funkcií. Vhodné zvolené členenie prispieva k prehľadnosti kódu.

2.2 Reflector Administration Protocol

Reflector Administration Protocol je textový aplikačný protokol určený na ovládanie paketového zrkadla a získavanie informácií o jeho stave [8]. Poskytuje informácie o aktívnych reláciách, alokovaných zdrojoch a dostupných moduloch. Umožňuje spúšťanie, zastavovanie a kontrolu stavu modulov. Štruktúru správ demonštruje príklad 2.2.

```
START RAP/1.0
Module: listener/udp
Port: 1234

START RAP/1.0
Module: processor/filter

PROCESS RAP/1.0
From: 0.0.0.0/0
To: 0.0.0.0/0
Listener: listener/udp-0.0.0.0:1234
Processor: processor/filter-0
```

Príklad 2.2: *RAP* správy spúšťajúce moduly.

Protokol využíva šestnásť základných typov správ. Najpoužívanejšie sú správy zabezpečujúce ovládanie modulov a zisťovanie stavu zrkadla. Príkladom sú nasledujúce typy správ [8]:

AVAIL
: Informuje klienta o dostupných moduloch a ich parametroch. Táto správa je spracovaná v module *management/master* bez ohľadu na cieľ uvedený v hlavičke.

LIST
: Informuje klienta o spustených moduloch alebo o detailoch vybraného modulu, ktorý je uvedený v hlavičke správy ako cieľ.

START
Spustí modul definovaný v hlavičke správy so zvolenými parametrami (napr. adresa, port alebo veľkosť paketov).

PROCESS
Zaradí zvolený *processor* modul do radu na spracovanie paketov. Príkladom použitia je manipulácia zoznamu klientov, ktorým bude paket zaslaný, pomocou definovania bitových masiek. Bitová maska je násobená s *IP* adresami jednotlivých klientov pri výbere cieľov pre paket, ku ktorému patrí. Na základe výsledku tohto súčinu sa zrkadlo rozhoduje o zaslaní/nezaslaní paketu danému klientovi.

STAT
Informuje klienta o štatistikách modulu, ktorý je uvedený ako cieľ v hlavičke správy. Klasickým príkladom použitia je získanie údajov o počte klientov a množstve dát v jednotlivých reláciách od *management/session* alebo štatistík využitia radov správ a dát od *management/master*.

2.3 Real Time Streaming Protocol

Real Time Streaming Protocol je textový aplikačný protokol poskytujúci kontrolu nad prenosom dát v reálnom čase [16]. Umožňuje kontrolovať prenos audia a videa z živého vysielania alebo zo záznamu, poskytuje podporu pre rôzne doručovacie mechanizmy (*UDP*, *TCP* alebo *RTP*[9]) a je navrhnutý tak, aby dokázal ovládať niekoľko prenosov jednoznačne určených identifikátormi relácií. Ide o protokol založený na modele klient/server. Principiálne je podobný protokolu *HTTP*, líšia sa v niekoľkých bodoch:

- V *RTSP* prebieha komunikácia v oboch smeroch, server aj klient môžu posielať požiadavky.
- *RTSP* implementuje niekoľko metód špecializovaných na ovládanie multimediálnych prenosov.
- *RTSP* je stavový protokol, stav je udržovaný na základe identifikátorov relácií.
- Implicitným kódovaním správ v *RTSP* je *UTF-8*[10].

9. RFC 1889 - RTP: A Transport Protocol for Real-Time Applications
10. RFC 2279 - UTF-8, a transformation format of Unicode and ISO 10646

Štruktúra správ

```
PLAY rtsp://localhost:6666/ RTSP/1.0
CSeq: 4
Session: 1774427278
Range: npt=0.000-
User-Agent: VLC media player

RTSP/1.0 200 OK
CSeq: 4
Session: 1774427278
```

Príklad 2.3: *RTSP* správa a odpoveď.

RTSP požiadavka sa skladá z troch častí. Riadok identifikujúci požiadavku a hlavička správy patria medzi povinné časti, telo správy je voliteľné a zvyčajne obsahuje informácie, ktoré sú nezávislé na *RTSP*. V odpovedi prvý riadok obsahuje identifikátor protokolu a návratový kód so slovným popisom. Protokol *RTSP* preberá návratové kódy protokolu *HTTP* a dopĺňa ich o kódy popisujúce jeho špecifické stavy [16]. Štruktúru demonštruje príklad 2.3.

Prvý riadok požiadavky obsahuje identifikátor metódy, identifikátor požadovaného zdroja a verziu protokolu. Štandard definuje nasledujúce povinné metódy [16]:

DESCRIBE (odporúčané)
: Metóda akceptovaná v smere klient → server, žiada o popis prezentácie alebo média určeného v požiadavke. Zvyčajne definuje akceptované formáty popisu ako napríklad *Session Description Protocol* (ďalej len *SDP*).

OPTIONS
: Metóda akceptovaná v smere klient ↔ server, ktorá nemení stav relácie a môže byť vyvolaná kedykoľvek. V odpovedi cieľ zverejňuje zoznam podporovaných metód.

PAUSE (odporúčané)
: Metóda akceptovaná v smere klient → server, ktorá zabezpečuje dočasné zastavenie prenosu dát pomocou transportného mechanizmu.

PLAY
Metóda akceptovaná v smere klient → server. Oznamuje serveru, že môže začať posielať dáta pomocou transportného mechanizmu dohodnutého v predchádzajúcej *SETUP* požiadavke a následnej odpovedi.

SETUP
Metóda akceptovaná v smere klient → server, ktorá zabezpečuje inicializáciu transportného mechanizmu. Popisuje protokoly, porty a generuje identifikátor relácie.

TEARDOWN
Metóda akceptovaná v smere klient → server. Zastaví prenos dát a uvoľní všetky zdroje, ktoré využíval. Pre opätovné spustenie prenosu je nutné opakovať volania *SETUP* a *PLAY*.

Hlavička obsahuje poradové číslo správy, voliteľné položky a voľby špecifické pre jednotlivé metódy. Voliteľnou položkou je napríklad použité kódovanie alebo časová známka, medzi voľby metód patrí identifikátor relácie alebo informácia o použitom transportnom protokole. Telo správy sa najčastejšie vyskytuje v odpovedi na požiadavku *DESCRIBE*. Obsahuje popis vyžiadaného zdroja, zvyčajne je použitý *SDP*. Informuje klienta o formáte audia/videa, transportnom protokole a použitých komunikačných portoch. Príklad komunikácie *RTSP* modulu s klientom sa nachádza v prílohe A.

2.4 Session Description Protocol

Session Description Protocol je textový aplikačný protokol popisujúci multimediálne relácie za účelom ich oznamovania a inicializácie [6]. Klient sa podľa obsahu popisovača pripraví na prijímanie/prehrávanie špecifikovaného média a zároveň môže získať doplnkové informácie o jeho pôvode. *SDP* obsahuje nasledujúce informácie:

- Názov a účel relácie.
- Dĺžka trvania relácie.
- Médiá tvoriace reláciu.
- Popis transportného mechanizmu a formátu médií.

Príkladom využitia *SDP* je popis médií kontrolovaných *RTSP* serverom. Štruktúru popisovača demonštruje príklad 2.4.

```
v=0
o=fi.muni 2890844526 2890842807 IN IP4 127.0.0.1
s=FI Lectures
i=Live stream from D1/D2/D3
e=support@video.muni.cz
c=IN IP4 127.0.0.1
t=0 0
a=recvonly
m=video 1234 udp 33
```

Príklad 2.4: *SDP* popisovač média v tele *RTSP* správy.

Najdôležitejšími položkami *SDP* popisovača sú informácie o spojení v položke *c* a popis média v položke *m*. V našom prípade ide o audio a video dáta, ktoré má klient očakávať na lokálnom *UDP* porte číslo 1234. Zvyšné položky poskytujú len doplňujúce informácie ako napríklad verzia protokolu (*v*), slovný popis média (*s* a *i*) alebo kontakt na vlastníka/správcu (*e*).

3 Analýza dostupných implementácií RTSP

V záujme kompatibility *RTSP* modulu s najrozšírenejšími multimediálnymi prehrávačmi som sa rozhodol preskúmať dostupné zdrojové kódy a fungovanie rôznych implementácií. Pri serverových implementáciách som hodnotil aj ich vhodnosť na začlenenie do modulu.

3.1 Serverové implementácie

Z oblasti knižníc implementujúcich protokol *RTSP* som vybral knižnice *LIVE555 Streaming Media*[11] napísané v jazyku *C++*[12] a z kompletných implementácií multimediálny server *Feng* [12] napísaný v jazyku *C*.

3.1.1 LIVE555 Streaming Media

Táto kolekcia knižníc je poskytovaná pod licenciou *GNU Lesser General Public License*[13]. Poskytuje nástroje na prácu s protokolmi *RTP*, *RTSP*, *SIP*[14] a najrozšírenejšími video/audio formátmi.

Hodnotenie vzhľadom na použiteľnosť v *RTSP* module:

+ Kompatibilné s väčšinou prehrávačov.
+ Prebieha neustály vývoj a oprava chýb.
+ Obsahuje množstvo príkladov použitia.
- Napísané v jazyku *C++*, komplikovanejšie spájanie so zrkadlom.
- *RTSP* modul vyžaduje len zlomok funkcionality, zbytočne zložité.
- Ťažko prispôsobiteľné.

Knižnice *LIVE555 Streaming Media* nie sú vhodné na použitie v module, priniesli by viac komplikácií ako úžitku.

---

11. Dostupné na http://www.live555.com/liveMedia/
12. N1905 - C++ International Standard
13. Dostupná na http://www.gnu.org/copyleft/lesser.html
14. RFC 3261 - SIP: Session Initiation Protocol

3.1.2 Feng

Feng je kompletné serverové riešenie na distribúciu multimediálnych dát, je poskytovaný pod *GNU Lesser General Public Licence v2.1*[15]. Obsahuje podporu pre protokoly *RTSP* a *RTP*, najrozšírenejšie audio/video formáty a funguje na princípe modulov.

Hodnotenie vzhľadom na použiteľnosť v RTSP module:

+ Kompatibilný s väčšinou prehrávačov.
+ Prebieha neustály vývoj a oprava chýb.
+ Modulárna architektúra, viditeľnejšie rozčlenenie.
+ Napísaný v jazyku *C*, jednoduchšie spájanie so zrkadlom.
+ Kompletný server, príklad aplikačnej logiky.
- *RTSP* modul pre *RUM2* vyžaduje len zlomok funkcionality, zbytočne zložité.

Feng je vhodný na priame spojenie so zrkadlom. Veľká časť jeho funkcionality je však pre *RTSP* modul nadbytočná. Rozhodol som sa vybrať z *Feng* len syntaktické analyzátory *RTSP* správ a zvyšok použiť ako predlohu pre implementáciu *RTSP* servera ako modulu zrkadla *RUM2* v jazyku *C*.

3.2 Klientské implementácie

Pri skúmaní klientských implementácií som sa zameral na multimediálne prehrávače. Tie budú klientmi hotového modulu a práve na ich fungovaní teda záleží najviac. Trojicu najpoužívanejších prehrávačov tvoria *MPlayer* [14], *VLC Player* [17] a *Windows Media Player*[16]. Prvé dva prehrávače sú poskytované pod licenciou *GNU General Public License*[17] a podpora *RTSP* je v nich realizovaná pomocou fragmentov z rôznych knižníc.

Oba využívajú prevažne statické správy, do ktorých sú dynamicky vkladané iba nevyhnutné prvky ako napríklad identifikátor *RTSP* relácie. Informujú tiež o tom, že ich implementácie *RTSP* zatiaľ nevyhovujú štandardu. *Windows Media Player* je komerčný produkt spoločnosti *Microsoft*, zdrojové kódy nie sú k dispozícii. Jeho kompatibilita s modulom je skúmaná v rámci testovania v stati 4.4.

15. Dostupná na http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html
16. Dostupný na http://www.microsoft.com/windows/windowsmedia/default.mspx
17. Dostupná na http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

3.2.1 MPlayer

MPlayer preberá a rozširuje *librtsp* knižnicu z projektu *xine*[18]. Aplikačná logika je realizovaná pomocou *rtsp_request_∗* funkcií implementovaných pre každú povinnú metódu *RTSP* zvlášť a pomocných funkcií zabezpečujúcich čítanie prichádzajúcich správ, plánovanie odosielania a odosielanie správ serveru. Klient sa identifikuje ako *RealMedia Player Version 6.0.9.1235*, od servera požaduje metódy *DESCRIBE*, *SETUP*, *PLAY*, *TEARDOWN* a *OPTIONS* spolu s doručovaním dát pomocou protokolu *RTP*. Pri preklade je možné zvoliť rozšírenú podporu *RTSP* pomocou externých knižníc *LIVE555 Streaming Media*.

MPlayer nedokáže prehrávať audio/video doručované vo forme *UDP* paketov na vopred špecifikovaný port. Pri pokuse o prehrávanie nastane kritická chyba a prehrávač je násilne ukončený. Kompatibility modulu a *MPlayera* sa dá dosiahnuť len vytvorením doplnkového modulu pre *RUM2*, ktorý bude dáta prechádzajúce zrkadlom upravovať do formátu akceptovateľného *MPlayerom*. Toto riešenie je rozoberané ako ďalšia možnosť rozširovania paketového zrkadla v kapitole 5. Štruktúru správ zasielaných *MPlayerom* demonštruje príklad 3.1.

```
OPTIONS * RTSP/1.0
CSeq: 1
User-Agent: RealMedia Player Version 6.0.9.1235
ClientChallenge: 9e26d33f2984236010ef6253fb1887f7
PlayerStarttime: [28/03/2003:22:50:23 00:00]
CompanyID: KnKV4M4I/B2FjJ1TToLycw==
GUID: 00000000-0000-0000-0000-000000000000
RegionData: 0
ClientID: Linux_2.4_6.0.9.1235_play32_RN01_EN_586
```

Príklad 3.1: *RTSP* požiadavka *OPTIONS* od prehrávača *MPlayer*.

Prípadná podpora pre *MPlayer* bude zahrnutá až v neskorších verziách modulu.

3.2.2 VLC Player

Vo *VLC Player* je implicitným *RTSP* klientom implementácia založená na *LIVE555 Streaming Media* a prehrávač sa serveru identifikuje ako *VLC media*

18. Dostupný na http://www.xine-project.org/home

player (LIVE555 Streaming Media v2010.02.10). *VLC Player* implementuje aj klienta založeného na knižnici *librtsp*, ten je však použitý až v prípade zlyhania implicitného klienta (napríklad pri neočakávanom ukončení spojenia zo strany servera). Prehrávač má robustné sieťové rozhranie a dokáže automaticky rozpoznať viaceré video/audio formáty.

Od servera očakáva implementáciu povinných metód *DESCRIBE*, *SETUP*, *PLAY*, *TEARDOWN* a voliteľnej metódy *OPTIONS*. *VLC Player* využije na svoje nastavenie informácie obsiahnuté v *SDP* popisovači, ktorý dostane od servera v tele odpovede na požiadavku *DESCRIBE*. V súčasnosti je používaný v kombinácii s paketovým zrkadlom práve kvôli svojej schopnosti prehrávať video/audio dáta prichádzajúce na *UDP* port, na ktorom lokálne načúva. Je preto ideálnym klientom pre *RTSP* modul. Štruktúru zasielaných správ demonštruje príklad 3.2.

```
OPTIONS rtsp://localhost:6666/ RTSP/1.0
CSeq: 1
User-Agent: VLC media player (LIVE555 Streaming Media)
```

Príklad 3.2: *RTSP* požiadavka *OPTIONS* od prehrávača *VLC Player*.

VLC Player je pre *RTSP* modul referenčnou implementáciou klienta. Všetka funkcionalita bude testovaná práve vzhľadom na tento prehrávač.

4 Vlastná implementácia RTSP modulu

Implementácia *RTSP* modulu pre paketové zrkadlo *RUM2* sa skladá zo štyroch častí. Prvou časťou je návrh fyzického rozčlenenia kódu a požadovanej aplikačnej logiky modulu, druhou časťou je hľadanie nástrojov vhodných na implementáciu navrhnutej aplikačnej logiky, treťou časťou je vlastná implementácia v jazyku C a štvrtou časťou je testovanie funkčnosti a škálovateľnosti implementovaného modulu.

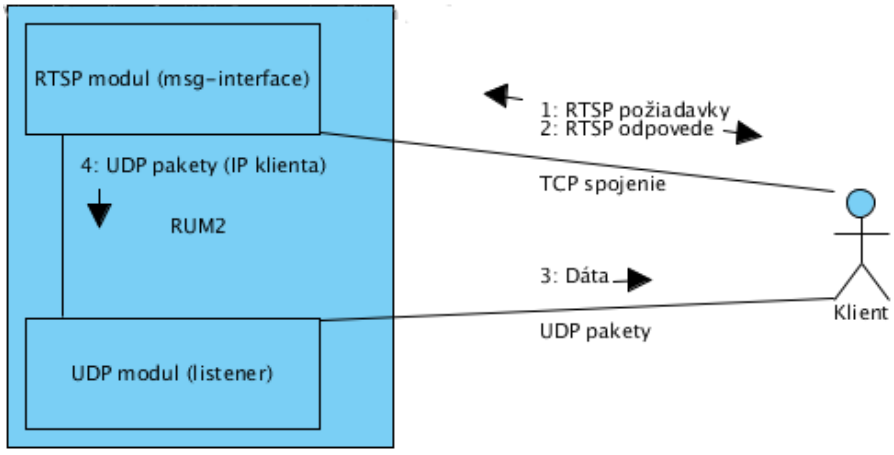
4.1 Návrh modulu

Existujúce moduly pre paketové zrkadlo *RUM2* dodržujú konvenciu jazyka *C* pre fyzické rozdeľovanie kódu do hlavičkových a zdrojových súborov. V hlavičkových súboroch sa nachádzajú prototypy použitých funkcií, definície štruktúr a výčtových typov, konštanty a makrá. Zdrojové súbory obsahujú rozhranie umožňujúce komunikáciu paketového zrkadla s modulom a aplikačnú logiku rozdelenú do funkcií definovaných prototypmi v hlavičkovom súbore. V oboch súboroch sa tiež nachádzajú nevyhnutné direktívy *include* sprístupňujúce externé hlavičkové súbory potrebné pre preklad a komentáre popisujúce použitie, parametre a fungovanie funkcií alebo význam jednotlivých dátových štruktúr a ich položiek. Vzhľadom na menší počet implementovaných funkcií bude stačiť rozdelenie na jeden hlavičkový súbor, jeden zdrojový súbor a súbory obsahujúce kód syntaktických analyzátorov pochádzajúcich z implementácie servera *Feng*. Modul nebude zasahovať do dátového toku, je určený na pridávanie a odstraňovanie klientov na základe prijatých *RTSP* správ, preto je zaradený do triedy *msg-interface*. Ďalším krokom v návrhu je definovanie požiadaviek na aplikačnú logiku.

Analýza dostupných klientských implementácií protokolu *RTSP* v stati 3.2 ukázala, že klienti od servera požadujú implementáciu metód *DESCRIBE*, *SETUP*, *PLAY*, *TEARDOWN* a očakávajú implementáciu odporúčanej metódy *OPTIONS*. Z toho vyplýva, že *RTSP* modul musí tieto metódy podporovať. Vzhľadom na to, že modul bude zabezpečovať poskytovanie dát vznikajúcich v reálnom čase, nemá zmysel implementovať napríklad metódu *PAUSE*. Ostatné metódy sú v *RFC 2326* [16], popisujúcom protokol *RTSP*, označené ako voliteľné a nie sú nutné k správnemu fungovaniu modulu. *RTSP* správy zasielané od modulu ku klientovi sú, až na sekvenčné čísla, časové známky a identifikátory relácií, prevažne rovnaké. V kóde môžu byť uložené ako konštanty, do ktorých budú vkladané len spomínané informácie. To isté platí aj o *SDP* popisovači modulom ovládaného média.

Správy prichádzajúce od klientov obsahujú štyri dôležité informácie. Prvou je identifikátor metódy, druhou verzia protokolu *RTSP*, treťou sekvenčné číslo správy *CSeq* a štvrtou identifikátor relácie *Session* (v prípade, že klient tento identifikátor už získal z odpovede na požiadavku *SETUP*). Sú nevyhnutné pre fungovanie modulu. Z prijatých správ budú získavané pomocou syntaktických analyzátorov protokolu *RTSP*. Modul ich využije hlavne na kontrolu prichádzajúcich správ a identifikáciu jednotlivých klientov. Identifikátor klientskej aplikácie *User-Agent*, potvrdenie transportných mechanizmov *Transport* a pokyn na štart prehrávania z určitého miesta v poskytovanom médiu *Range* sú ignorované. Modul totiž nedokáže robiť zmeny v transportnom mechanizme, ktorý je poskytovaný zrkadlom paketov.

Kľúčovým krokom pri tvorbe *RTSP* modulu je formulovanie princípov komunikácie medzi modulom a časťami zrkadla ovládajúcimi zaraďovanie klientov do konkrétnych dátových relácií. Táto funkcionalita totiž nie je zahrnutá v štandardnom rozhraní, ktoré zrkadlo využíva na komunikáciu s modulmi. Klient sa do relácie hlási periodickým kontaktovaním toho modulu triedy *listener*, ktorý prijíma požadované dáta od zdroja. Ak teda chce klient od zrkadla dostávať napríklad video/audio dáta, ktoré zdroj posiela na *UDP* port s číslom 1234, musí periodicky kontaktovať zrkadlo práve na porte 1234. Bude tak zaradený do rovnakej relácie ako zdroj a zrkadlo mu začne posielať kópie dát práve z tejto relácie na adresu, ktorú našlo v hlavičke prijatého paketu.



Obr. 4.1: Princíp fungovania *RTSP* modulu.

Táto funkcionalita je simulovateľná pomocou špeciálne vytvorených *UDP* paketov s podvrhnutou zdrojovou *IP* adresou. Klient kontaktuje *RTSP* mo-

dul, prejde povinným procesom *SETUP* → *PLAY* a modul za neho začne periodicky kontaktovať zrkadlo (jeho príslušný *listener* modul) paketmi, ktoré majú v hlavičke podvrhnutú adresu klienta. Z pohľadu *listener* modulu sa nič nezmenilo a *RTSP* modul sa správa len ako most medzi štandardizovanou komunikáciou protokolom *RTSP* a neštandardizovaným periodickým zasielaním *UDP* paketov. *RTSP* modul bude v periodickom kontaktovaní pokračovať, až kým klient nepožiada o ukončenie doručovania dát zaslaním *TEARDOWN* alebo kým neuzatvorí *TCP* spojenie nadviazané s modulom. Princíp ilustruje obrázok 4.1. Implementačné detaily fungovania späté s jazykom *C* sú rozoberané v stati 4.3.1.

Periodickým kontaktovaním paketového zrkadla pridávajú klienti do výstupného prúdu dát pakety, ktorými zrkadlo kontaktovali. Tieto pakety nemajú zmysluplný obsah, zvyčajne sa v nich nachádza predvolený reťazec znakov. Príkladom je slovo *ping*, ktoré do tela paketov vkladá už spomínaný nástroj *ping*. Túto konvenciu od neho prebral aj *RTSP* modul. Pakety majúce v tele slovo *ping* nepredstavujú problém, ak klientský prehrávač pracuje v režime predpokladajúcom určité straty alebo skreslenie dát. V podobnom režime pracuje prehrávač *VLC Player*, ak je užívateľom ručne nastavený na načúvanie na lokálnom *UDP* porte a prehrávanie akýchkoľvek dát, ktoré prijme a identifikuje ako vyhovujúce. Ak sa v prúde prehrávateľných audio/video dát vyskytne poškodený alebo neznámy paket, prehrávač ho zahodí a snaží sa pokračovať v prehrávaní obsahu nasledujúcich paketov. Keď však použijeme na inicializáciu prehrávača *SDP* popisovač z tela odpovede na požiadavku *DESCRIBE*, prehrávač bude očakávať len dáta vopred definovaného formátu. Keď narazí na neznámy alebo poškodený paket, ukončí prehrávanie a užívateľovi oznámi koniec prehrávaného média.

Plynulé prehrávanie u klienta umožňuje doplnkový modul s názvom *Filter*, ktorý patrí do triedy *processor*. Jeho úlohou je odstraňovať z výstupných radov pakety, ktoré obsahujú slovo *ping*. Implementačné detaily fungovania tohto modulu sú rozoberané v stati 4.3.2.

4.2 Nástroje a knižnice

Pri vývoji *RTSP* modulu som použil niekoľko nástrojov a knižníc prinášajúcich zjednodušenie alebo požadovanú funkcionalitu. V prvom rade modul vyžaduje prítomnosť hlavičkových súborov paketového zrkadla *RUM2*, ktoré poskytujú informácie o funkciách a dátových štruktúrach používaných pri komunikácii medzi zrkadlom a modulom. Tieto súbory sú z dôvodu zjednodušenia závislostí pri preklade pribalené k modulu. Okrem hlavičkových súborov zrkadla a hlavičkových súborov sprístupňujúcich funkcie štandardnej knižnice jazyka *C*

modul využíva nasledujúce dôležité knižnice:

- *GLib*, nízkoúrovňová knižnica nástrojov projektu *GNOME* [5]
- *libev*, knižnica implementujúca výkonný udalostný model [15]
- syntaktické analyzátory servera *Feng*

Knižnica *GLib* je v module použitá kvôli pokročilým dátovým štruktúram, ktoré implementuje. Na ukladanie záznamov o klientoch je použitá štruktúra *HashTable*. Sprístupňuje *IP* adresy klientov pomocou identifikátorov *RTSP* relácií, ktoré k nim patria. Funkcie patriace do tejto knižnice sú použité aj na alokovanie a uvoľňovanie pamäte, generovanie identifikátorov relácií a získavanie časových známok. Jej použitie umožňuje hlavičkový súbor *glib.h* a direktíva *-lglib-2.0* pri preklade. Paketové zrkadlo túto knižnicu nevyužíva, je linkovaná výlučne pre potreby *RTSP* modulu.

Knižnica *libev* implementuje udalostný model, ktorý *RTSP* modulu umožňuje reagovať na zmeny v stave *socketu*, na ktorom načúva. *Socket* je otvorený v neblokujúcom režime, udalosti *READ* a *WRITE* modulu oznamujú prítomnosť prichádzajúcej správy alebo možnosť zápisu odpovede. Spojenie nie je blokované jedným klientom, prijímanie a odosielanie správ je asynchrónne. Táto technika prispieva k výkonnosti a efektivite modulu. Jej použitie umožňuje hlavičkový súbor *ev.h* a direktíva *-lev* pri preklade. Paketové zrkadlo túto knižnicu nevyužíva, je linkovaná len pre potreby *RTSP* modulu.

Syntaktické analyzátory servera *Feng* nie sú knižnicou v pravom slova zmysle. Ide o stavové automaty akceptujúce správy protokolu *RTSP* zapísané v syntaxi prekladača *Ragel* [2]. *Ragel* umožňuje prekladať konečné automaty, zapísané pomocou špeciálnych regulárnych a vkladacích operátorov, do kódu jazyka *C* (okrem iného). Výsledné súbory je možné preložiť bežným prekladačom jazyka *C* a využívať na analýzu správ protokolu *RTSP*. Funkcie predstavujúce rozhranie analyzátorov sú v kóde modulu definované ako externé a spájanie sa vykonáva až pri preklade. Dynamické knižnice *GLib* a *libev* musia byť v systéme prítomné aj počas fungovania *RTSP* modulu. Prekladač *Ragel* je potrebný len vo fáze prekladu.

Modul *Filter* je principiálne aj implementačne jednoduchší, sú v ňom použité len hlavičkové súbory sprístupňujúce funkcie a štruktúry rozhrania zrkadla *RUM2* a hlavičkové súbory sprístupňujúce funkcie štandardnej knižnice jazyka *C*. Automatizácia prekladu oboch modulov je realizovaná pomocou obľúbeného nástroja *GNU Make* [4].

Pri tvorbe modulov *RTSP* a *Filter* boli použité aj pomocné nástroje *Doxygen* [7] a *Subversion* [1]. *Doxygen* je určený na tvorbu dokumentácie zo

špeciálne formátovaných komentárov nachádzajúcich sa v zdrojovom kóde. Dokumentáciu je možné vygenerovať vo formátoch LaTeX, *PDF*[19], *HTML*[20], *RTF*[21] alebo ako manuálové stránky systému *Unix*[22]. *Subversion* je systém na kontrolu verzií zdrojového kódu. Umožňuje dlhodobé sledovanie zmien v projekte, tímovú spoluprácu, rozdeľovanie kódu do vývojových vetiev a zároveň slúži ako zálohovací systém. Použitie verzovacieho systému značne uľahčuje prácu so zdrojovým kódom a poskytuje možnosť návratu k predchádzajúcim verziám.

4.3 Popis fungovania modulov RTSP a Filter

Predchádzajúce state popisovali princípy, ktoré moduly využívajú pre svoje fungovanie. Nasleduje zjednodušený pohľad na fungovanie modulov *RTSP* a *Filter* implementovaných v jazyku *C* detailnejšie rozoberajúci niektoré kľúčové časti kódu. Táto stať má za úlohu osvetliť základy fungovania oboch modulov krok za krokom. Pre hlbšie pochopenie je potrebné nahliadnuť do komentovaného zdrojového kódu nachádzajúceho sa na priloženom médiu. Príklad *RAP* správ spúšťajúcich moduly zrkadla sa nachádza v prílohe B.

4.3.1 RTSP modul

RTSP modul prechádza počas svojho fungovania niekoľkými štádiami. Pri spustení modulu paketové zrkadlo *RUM2* vyberie na základe zhody mena modulu a mena súboru príslušnú dynamickú knižnicu, ktorá obsahuje preložený modul. Na sprístupnenie symbolov modulu použije funkcie knižnice *ltdl* a spustí vlastnú inicializáciu. V prvom kroku inicializácie je modulu priradené dočasné meno, trieda a ukazateľ na štruktúru rozhrania *iface*. Zo štruktúry rozhrania zrkadlo získa mená funkcií implementujúcich jednotlivé časti rozhrania, tie dodržujú zavedené konvencie a obsahujú prefix *m_* nasledovaný menom funkcie zodpovedajúcim jej účelu. Funkcie rozhrania medzi modulom a zrkadlom boli detailnejšie popísané v stati 2.1.2.

Po inicializácii rozhrania a získaní mena nasleduje vlastná inicializácia modulu. Prebehne získanie parametrov obsiahnutých v *RAP* požiadavke na spustenie modulu, alokácia pamäte pre dátové štruktúry a príprava *socketu*, na ktorom bude modul načúvať. *IP* adresa, port *RTSP* modulu a port cieľového *UDP listener* modulu patria medzi voliteľné parametre, ak nie sú

19. ISO 32000-1:2008 - Portable document format
20. RFC 1866 - Hypertext Markup Language - 2.0
21. Špecifikácia na http://www.biblioscape.com/rtf15_spec.htm
22. Špecifikácia na http://www.unix.org/online.html

špecifikované, použijú sa implicitné hodnoty (načúvanie na 0.0.0.0 : 554 a kontaktovanie lokálneho portu 1234). V tejto fáze je pripravená štruktúra *RTSP_Server* obsahujúca dáta nutné k fungovaniu servera, ktorý je vlastne jadrom modulu. V štruktúre sa nachádza napríklad zoznam klientov, lokálna adresa servera, popisovač *socketu* alebo štruktúry knižnice *libev* umožňujúce využívanie udalostného modelu.

V závere inicializácie modulu prebieha alokovanie pamäte a príprava šablóny pre *UDP* pakety, do ktorých budú neskôr vkladané *IP* adresy klientov. Pakety budú použité na periodické kontaktovanie *UDP listener* modulu v mene klientov *RTSP* servera. Odosielanie paketov s pozmenenými hlavičkami je z bezpečnostných dôvodov možné len s prístupovými právami užívateľa *root* alebo so špeciálnym príznakom nastaveným pomocou *setcap*[23]. Príznak môže nastaviť len užívateľ *root*. Po jeho nastavení môže modul fungovať aj s prístupovými právami bežného užívateľa.

Po úspešnom dokončení inicializácie zrkadlo spustí *m_main* a tým uvedie modul do chodu. Modul musí byť schopný štartu a zotrvať v chode až kým ho zrkadlo explicitne neukončí. V tejto fáze modul zapíše svoje spustenie do záznamu činnosti zrkadla, odštartuje vlákno funkcie *watcher* a začne čakať na prichádzajúce správy. *Watcher* každých 60 sekúnd skontroluje zoznam klientov a za každého klienta v zozname kontaktuje *listener* modul *UDP* paketom s podvrhnutou *IP* adresou. K zoznamu klientov majú prístup dve vlákna, jedným je *watcher* a druhým je hlavné vlákno spracúvajúce prichádzajúce *RTSP* správy a pridávajúce nových klientov. Prístup k zoznamu klientov je teda kritickou sekciou a musí byť synchronizovaný. Synchronizáciu zabezpečuje *mutex*, ktorý nedovolí dvom vláknam vstúpiť do tej istej kritickej sekcie. Kritickú sekciu znázorňuje príklad 4.2.

```
pthread_mutex_lock(&srv->srv_mutex);
g_hash_table_foreach(srv->client_list,
                     add_new_listener,
                     module);
pthread_mutex_unlock(&srv->srv_mutex);
```

Príklad 4.2: Ukážka kritickej sekcie.

Prijatie správy spustí neblokujúce akceptovanie spojenia, pripravenie dátovej štruktúry pre klienta, prečítanie správy a jej analýzu pomocou syntaktických analyzátorov. Ako prvá prebehne predbežná analýza, ktorá zistí, či

23. Použitie na http://linux.die.net/man/8/setcap

prijatá správa patrí protokolu *RTSP*. Štruktúra *RTSP_Request* je naplnená verziou protokolu a požadovanou metódou. Ak správa vyhovuje, modul pokračuje v analýze hlavičiek a vo vykonávaní kódu požadovanej metódy. Kontrolu sekvenčných čísel v *RTSP* module predstavuje len kontrola ich existencie a nezápornosti. Toto riešenie umožňuje opakované spúšťanie a zastavovanie doručovania z prehrávača *VLC Player*, ktorý pokračuje v číslovaní správ v nových spojeniach.

Metódy *OPTIONS* a *DESCRIPTION* zasielajú klientovi vopred definované správy informujúce o schopnostiach modulu, detailoch transportu dát a formáte poskytovaného audia/videa. Spávy zasielané ako odpovede na tieto metódy sú v kóde reprezentované ako konštanty a ich obsah sa počas fungovania modulu nemení. Jedinou výnimkou je sekvenčné číslo správy vkladané až pri odosielaní. Umožňuje to opakovné zasielanie odpovedí, ktoré u klienta úspešne prejdú kontrolou náväznosti sekvenčných čísel. V odpovedi na *DESCRIPTION* sa nachádza *SDP* popisovač média. Vzhľadom na to, že je reprezentovaný konštantným reťazcom znakov, nie je nutné pre potreby *RTSP* modulu implementovať logiku pracujúcu so správami protokolu *SDP*.

Metóda *SETUP* vygeneruje identifikátor relácie na základe *IP* adresy, portu a aktuálnej časovej známky. Identifikátor zašle klientovi v odpovedi. Metódy *PLAY* a *TEARDOWN* kontrolujú prítomnosť a správnosť tohto identifikátora, kontroluje sa jeho zhoda s uloženou kópiou. *PLAY* zaradí klienta do zoznamu klientov a *TEARDOWN* klienta zo zoznamu odstráni. Každá metóda nastavuje v dátovej štruktúre klienta odpoveď, tá je klientovi odoslaná v momente dostupnosti spojenia na zápis. Dátové štruktúry sú uvoľňované po detekovaní chyby alebo ak klient prejde celým procesom a korektne zavolá *TEARDOWN*. Výmena končí uzatvorením spojenia.

Čistenie pamäte a odstraňovanie klienta zo zoznamu klientov sa z bezpečnostných dôvodov vykonáva aj po vypršaní nastaviteľného časového limitu. Odpočítavanie je pozastavené len po zavolaní metódy *PLAY*, klienti totiž počas prehrávania nezasielajú žiadne správy. Po ukončení periodického kontaktovania *listener* modulu bude zrkadlo pokračovať v zasielaní dát klientovi po dobu, ktorá je určená ako doba platnosti dátovej relácie. Tento časový interval nie je možné ovplyvniť z *RTSP* modulu, je závislý výlučne na nastaveniach zrkadla.

4.3.2 Filter modul

Uvedenie modulu triedy *processor* do chodu je rozdelené na dve časti. Prvá pozostáva zo spustenia vlastného modulu a jeho inicializácie podľa parametrov obsiahnutých v *RAP* požiadavke na spustenie. Druhá spočíva v zaradení

modulu do procesu spracovania dát pochádzajúcich zo zvoleného *listener* modulu. Zapojenie modulu do zrkadla a jeho inicializácia prebiehajú rovnako ako pri *RTSP* module. V závere inicializácie dátových štruktúr sa navyše vykonáva registrácia modulu do radov na spracovanie dát a získanie vzorky dát, podľa ktorej bude modul filtrovať prechádzajúce pakety. Registrácia zabezpečí len spojenie modulu s radom dát. Filtrovanie sa začne, až keď je modul explicitne zaradený do spracovania dát pomocou *RAP* požiadavky *PROCESS*. Implicitnou vzorkou je slovo *ping*.

Filtrovanie dát je relatívne jednoduchá operácia. V hlavnom cykle modulu sú dáta vybrané zo vstupného radu, porovnané so vzorkou pomocou funkcie *memcmp*[24] a v prípade zhody odfiltrované (maskované) funkciou *meta_mask_all*[25] volanou s maskou 0 v druhom parametri. Po nastavení masky sú dáta posunuté na ďalšie spracovanie a proces sa opakuje. Maska zabráni odoslaniu vybraných dát klientom a umožní plynulé prehrávanie. Použitie modulu *Filter* je nutné pre zabezpečenie celkového fungovania konceptu zrkadla ovládaného protokolom *RTSP*.

4.4 Testovanie

Testovanie *RTSP* modulu je rozdelené do dvoch základných kategórií. Do prvej kategórie patria testy overujúce fungovanie, stabilitu a kompatibilitu s klientskou implementáciou protokolu *RTSP* v prehrávačoch. Do druhej patria testy škálovateľnosti a dopadu použitia modulu na výkon celého paketového zrkadla *RUM2*. Procesorový modul *Filter* nie je testovaný zvlášť, dôkazom jeho fungovania je plynulé prehrávanie prichádzajúceho audia/videa v teste funkčnosti *RTSP* modulu.

Funkčnosť

Funkčnosť modulu je testovaná vzhľadom na prehrávač *VLC Player* (vo verzii 1.0.6), ktorý bol vybraný ako referenčná implementácia klienta protokolu *RTSP*. Pri úvodných testoch bol prehrávač spúšťaný bez grafického užívateľského rozhrania, aby bolo možné priamo sledovať postupné nadväzovanie spojenia s modulom a okamžite identifikovať prípadné problémy. Takto bol identifikovaný problém s *UDP* paketmi obsahujúcimi slovo *ping* spôsobujúci ukončenie prehrávania, ktorý je riešený procesorovým modulom *Filter*. V neskorších fázach testovania bolo použité grafické užívateľské rozhranie, aby

24. Štandardná funkcia porovnávajúca jednotlivé bajty v pamäti.
25. Funkcia z hlavičkového súboru *data.h* zrkadla *RUM2*.

bolo možné otestovať spúšťanie a zastavovanie prúdu dát pomocou grafických ovládacích prvkov.

Po úspešnom ukončení testovania modulu vzhľadom na referenčnú implementáciu klienta bola testovaná kompatibilita s prehrávačmi *MPlayer* (vo verzii 4.4.3) a *Windows Media Player* (*WMP*, vo verzii 12). Tieto prehrávače nie sú kompatibilné so súčasnou implementáciou *RTSP* modulu. Nekompatibilita je v prvom rade spôsobená odlišnosťami v klientskej implementácii protokolu *RTSP*. Napríklad implementácia *RTSP* v *MPlayeri* zasiela jednotlivé riadky *RTSP* správ postupne s určitým časovým odstupom a *RTSP* modul využívajúci *libev* na rýchle prepínanie medzi prichádzajúcimi správami ich interpretuje ako jednotlivé nekompletné správy. Okrem toho obom prehrávačom chýba podpora pre prehrávanie dát prichádzajúcich na lokálny *UDP* port.

Testovanie funkčnosti modulu bolo zakončené testami stability a odolnosti voči neočakávanému obsahu prijatých správ. Testy boli vykonávané pomocou jednoduchého testovacieho skriptu napísaného v jazyku *Python*[26], ktorý sa nachádza na priloženom médiu. Testovalo sa opätovné nadväzovanie spojenia, vypršanie časových limitov a reagovanie na nezmyselné alebo úmyselne podvrhnuté správy. Snahou bolo zabrániť pádom modulu v neočakávaných situáciách. Odhalené chyby boli opravené. Jednou z chýb bol napríklad pád modulu pri neočakávanom ukončení spojenia zo strany prehrávača *MPlayer*.

Škálovateľnosť

RTSP modul vykonáva výpočtovo nenáročné operácie s malými objemami dát vo forme *RTSP* správ a nezasahuje do toku dát cez zrkadlo. Výkon a škálovateľnosť *RTSP* modulu je možné posudzovať pomocou počtu klientov, ktorý je modul schopný obslúžiť. Vzhľadom na primárny účel paketového zrkadla - sprístupnenie skupinovej komunikácie pre relatívne malé množstvo spolupracujúcich strán [9] - by však testovanie veľkého množstva aktívnych spojení nemalo zmysel. Môžeme preto povedať, že použitie *RTSP* modulu má na výkon a škálovateľnosť paketového zrkadla *RUM2* zanedbateľný vplyv.

Modul *Filter* musí preskúmať každý paket prichádzajúci od zvoleného *listener* modulu. Jeho činnosť teda má určitý dopad na výkon a škálovateľnosť zrkadla. Proces skúmania paketu je však taký jednoduchý, že tento dopad sa na celkovom výkone prejaví len minimálne ($\leq 1\%$). Modul pracuje s ukazateľmi na dátové štruktúry (nenastáva kopírovanie), kontroluje len prvých n bajtov prechádzajúceho paketu (kde n je dĺžka vzorky) a okamžite ho posúva na ďalšie spracovanie.

26. Dostupný na http://www.python.org/download/

5 Záver

Cieľom tejto práce bola implementácia *RTSP* serveru vo forme modulu pre modulárne paketové zrkadlo *RUM2* v jazyku *C*, ktorý by užívateľom umožnil kontaktovať paketové zrkadlo priamo z multimediálneho prehrávača *VLC Player* a zjednodušil tak jeho používanie. V úvode som čitateľa uviedol do problematiky sieťovej distribúcie multimediálnych dát skupinám užívateľov. V teoretickej časti som stručne vysvetlil princípy fungovania a architektúru modulárneho paketového zrkadla *RUM2*, popísal základy protokolu *RAP* používaného na administráciu zrkadla a načrtol fungovanie protokolov *RTSP* a *SDP* vzhľadom na ich využitie v *RTSP* module pre paketové zrkadlo. Jadrom práce a mojím prínosom k problematike je analýza dostupných implementácií protokolu *RTSP* v serverových a klientských aplikáciách, vlastná implementácia modulu založená na získaných poznatkoch a príprava modulu na nasadenie do prevádzky v rámci distribúcie prednášok Fakulty informatiky Masarykovej univerzity v reálnom čase.

Implementácia prebehla úspešne a výsledkom je dvojica modulov pre paketové zrkadlo *RUM2*. Prvým modulom je *RTSP* modul patriaci do triedy *msg-interface*, ktorý je pôvodným cieľom tejto práce. Druhým modulom je doplnkový modul *Filter* patriaci do triedy *processor*, ktorý je potrebný pre bezproblémové prehrávanie distribuovaného audia/videa. Jeho vznik je podložený zisteniami rozoberanými v stati 4.1. Moduly sú funkčné, pripravené na nasadenie do skúšobnej prevádzky a začlenenie do projektu *RUM2*.

Modulárna architektúra paketového zrkadla poskytuje rozsiahle možnosti v oblasti rozširovania existujúcej funkcionality a dopĺňania nových modulov. Jednou z možností v tejto oblasti je vytvorenie modulu, ktorý zrkadlu umožní prichádzajúce *UDP* pakety transformovať na pakety protokolu *RTP*. V kombinácii s modulmi, ktoré sú výsledkom tejto práce, by tak bolo možné zabezpečiť kompatibilitu s väčšinou multimediálnych prehrávačov, ktoré podporujú prehrávanie kontrolované pomocou protokolu *RTSP*. Výsledkom by bolo značné rozšírenie základne potenciálnych klientov paketového zrkadla a sprístupnenie nových oblastí použitia.

Literatúra

[1] The Apache Software Foundation <http://www.apache.org/> *Apache Subversion [online].* 2009, dostupné na WWW: <`http://subversion.apache.org/packages.html`> [cit. 2010-05-10].

[2] CompLang.org <http://www.complang.org/> *Ragel State Machine Compiler [online].* 2009, dostupné na WWW: <`http://www.complang.org/ragel/`> [cit. 2010-05-10].

[3] Free Software Foundation, Inc. <http://fsf.org/> *GNU Libtool 2.2.6 [online].* 2002, dostupné na WWW: <`http://www.gnu.org/software/libtool/manual/libtool.html`> [cit. 2010-05-01].

[4] Free Software Foundation, Inc. <http://fsf.org/> *GNU Make [online].* 2006, dostupné na WWW: <`http://www.gnu.org/software/make/`> [cit. 2010-05-10].

[5] The GNOME Project <http://www.gnome.org/> *GLib, a general-purpose utility library [online].* 2009, dostupné na WWW: <`http://git.gnome.org/browse/glib/`> [cit. 2010-05-10].

[6] Handley, M., Jacobson, V., Perkins, C. *RFC4566 - SDP: Session Description Protocol [online].* The Internet Society, 2006, dostupné na WWW: <`http://tools.ietf.org/html/rfc4566`> [cit. 2010-05-01].

[7] van Heesch, D. *Doxygen, a source code documentation generator tool [online].* M.C.G.V. Stack <http://www.stack.nl/wiki/MCGV_Stack>, 2009, dostupné na WWW: <`http://www.stack.nl/~dimitri/doxygen/download.html#latestsrc`> [cit. 2010-05-10].

[8] Hladká, E., Holub, P., Denemark, J. *RAP - Reflector Administration Protocol [online].* Technická správa, CESNET z.s.p.o., 2003, dostupné na WWW: <`http://www.cesnet.cz/doc/techzpravy/2003/rap/rap.pdf`> [cit. 2010-05-01].

[9] Hladká, E., Holub, P., Denemark, J. *User Empowered Virtual Multicast for Multimedia Communication.* V *Proccedings of 3rd International Conference on Networking*, University of Haute Alsace, Colmar, France, 2004, ISBN 0-86341-325-0, dostupné na WWW: <`http://www.sitola.cz/papers/491740.pdf`> [cit. 2010-05-01].

[10] Hladká, E., Holub, P., Denemark, J. *Videokonference :: Projekt UDP zrcadla [online]*. CESNET z.s.p.o., 2005, dostupné na WWW: <`http://miro.cesnet.cz/about_the_project/reflector.cz.html`> [cit. 2010-05-01].

[11] Kernighan, B. W., Ritchie, D. M. *The C Programming Language*. Prentice Hall, Inc., druhé vydanie, 1988, ISBN 0-13-110370-9, [cit. 2010-05-01].

[12] The LScube <http://www.lscube.org/> *Feng - the RTSP/RTP streaming server [online]*. 2009, dostupné na WWW: <`http://www.lscube.org/feng/`> [cit. 2010-05-01].

[13] Masarykova univerzita <http://www.muni.cz/> *Nástroj ping na kontaktovanie zrkadla paketov [online]*. 2004, dostupné na WWW: <`http://www.video.muni.cz/faq.html`> [cit. 2010-05-01].

[14] The MPlayer Project <http://www.mplayerhq.hu/> *MPlayer [online]*. 2007, dostupné na WWW: <`http://www.mplayerhq.hu/design7/dload.html`> [cit. 2010-05-01].

[15] Schmorpforge Ta-Sa <http://software.schmorp.de/> *libev, a full-featured and high-performance event loop [online]*. 2009, dostupné na WWW: <`http://dist.schmorp.de/libev/`> [cit. 2010-05-10].

[16] Schulzrinne, H., Rao, A., Lanphier, R. *RFC2326 - Real Time Streaming Protocol (RTSP) [online]*. The Internet Society, 1998, dostupné na WWW: <`http://tools.ietf.org/html/rfc2326`> [cit. 2010-05-01].

[17] VideoLAN <http://www.videolan.org/> *VLC media player [online]*. 2009, dostupné na WWW: <`http://www.videolan.org/vlc/`> [cit. 2010-05-01].

A Príloha - Komunikácia RTSP modulu s klientom

Správy zasielané klientom sú označené červenou a odpovede modulu modrou farbou.

```
%% Klient žiada o zoznam podporovaných metód
OPTIONS rtsp://localhost:6666/ RTSP/1.0
CSeq: 1
User-Agent: VLC media player (LIVE555 Streaming Media)


%% Modul akceptuje a odpovedá zoznamom v položke Public
RTSP/1.0 200 OK
CSeq: 1
Public: DESCRIBE, SETUP, STOP, TEARDOWN, PLAY


%% Klient žiada o popisovač média vo formáte SDP
DESCRIBE rtsp://localhost:6666/ RTSP/1.0
CSeq: 2
Accept: application/sdp
User-Agent: VLC media player (LIVE555 Streaming Media)


%% Modul akceptuje a v tele správy zasiela SDP popisovač
RTSP/1.0 200 OK
CSeq: 2
Date: Sun, 09 May 2010 15:14:58 GMT
Content-Type: application/sdp
Content-Length: 183

v=0
o=fi.muni 2890844526 2890842807 IN IP4 127.0.0.1
s=FI Lectures
i=Live stream from D1/D2/D3
e=support@video.muni.cz
c=IN IP4 127.0.0.1
a=recvonly
m=video 1234 udp 33
```

```
%% Klient žiada o inicializáciu podľa položky Transport
SETUP rtsp://localhost:6666/ RTSP/1.0
CSeq: 3
Transport: RAW/RAW/UDP;unicast;client_port=1234-1235
User-Agent: VLC media player (LIVE555 Streaming Media)


%% Modul akceptuje a vygeneruje identifikátor relácie
RTSP/1.0 200 OK
CSeq: 3
Server: RTSP module for RUM2
Session: 1774427278;timeout=60
Transport: udp;source=127.0.0.1;server_port=1234;client_port=1234


%% Klient žiada o spustenie prehrávania
PLAY rtsp://localhost:6666/ RTSP/1.0
CSeq: 4
Session: 1774427278
Range: npt=0.000-
User-Agent: VLC media player (LIVE555 Streaming Media)


%% Modul akceptuje a zaradí klienta do zoznamu
RTSP/1.0 200 OK
CSeq: 4
Session: 1774427278


%% Klient žiada o ukončenie prehrávania
TEARDOWN rtsp://localhost:6666/ RTSP/1.0
CSeq: 5
Session: 1774427278
User-Agent: VLC media player (LIVE555 Streaming Media)


%% Modul akceptuje, odstráni klienta a ruší spojenie
RTSP/1.0 200 OK
CSeq: 5
Session: 1774427278
```

B Príloha - RAP správy spúšťajúce moduly zrkadla

Správy zasielané užívateľom sú označené červenou a odpovede zrkadla modrou farbou.

```
%% Užívateľ žiada o spustenie modulu UDP
%% listener na porte 1234
START RAP/1.0
Module: listener/udp
Port: 1234
```

```
%% Zrkadlo akceptuje
RAP/1.0 200 OK
Source: management/master
Module: listener/udp-0.0.0.0:1234
```

```
%% Užívateľ žiada o spustenie modulu TCP
%% msg-interface na porte 65001
START RAP/1.0
Module: msg-interface/tcp
Port: 65001
```

```
%% Zrkadlo akceptuje
RAP/1.0 200 OK
Source: management/master
Module: msg-interface/tcp-65001
```

```
%% Užívateľ žiada o spustenie modulu RTSP
%% msg-interface na porte 6666
START RAP/1.0
Module: msg-interface/rtsp
Port: 6666
```

```
%% Zrkadlo akceptuje
RAP/1.0 200 OK
Source: management/master
Module: msg-interface/rtsp-0.0.0.0:6666
```

```
%% Užívateľ povoľuje akceptovanie dát
%% od klientov vyhovujúcim 0.0.0.0/0
ACL RAP/1.0
Target: aaa/routing
Address: 0.0.0.0/0
Listener: reflector/listener
Access: rw

%% Zrkadlo akceptuje a oznamuje nastavenie práv
RAP/1.0 200 OK
Source: aaa/routing
Content-Length: 25
Content-Encoding: US-ASCII

Default access mode set

%% Užívateľ žiada o spustenie modulu
%% Filter processor
START RAP/1.0
Module: processor/filter

%% Zrkadlo akceptuje
RAP/1.0 200 OK
Source: management/master
Module: processor/filter-0

%% Užívateľ zaraďuje modul Filter
%% processor do spracovania dát
%% z modulu UDP listener
PROCESS RAP/1.0
From: 0.0.0.0/0
To: 0.0.0.0/0
Listener: listener/udp-0.0.0.0:1234
Processor: processor/filter-0

%% Zrkadlo akceptuje
RAP/1.0 200 OK
Source: processor/master
```

C Príloha - CD

Štruktúra priloženého CD nosiča:

- Adresár *build* obsahujúci vopred preložené súbory modulov.
- Adresár *docs* obsahujúci vygenerovanú dokumentáciu.
- Adresár *include* obsahujúci hlavičkové súbory zrkadla *RUM2*.
- Súbory *filter.c* a *filter.h* obsahujúce zdrojový kód modulu *Filter*.
- Súbory *Makefile* a *Makefile_FI* na preklad modulov pomocou *GNU Make*.
- Súbory *rtsp.c* a *rtsp.h* obsahujúce zdrojový kód modulu *RTSP*.
- Súbory *rtsp_eris_common.rl*, *rtsp_eris_headers.h*, *rtsp_eris_parser.rl* a *rtsp_ragel_request_line.rl* obsahujúce syntaktické analyzátory servera *Feng*.
- Súbor *RTSPTest.py* obsahujúci jednoduchý testovací skript v jazyku *Python*.
- Súbor *start_rtsp* obsahujúci *RAP* správy na ukážkové spustenie zrkadla s *RTSP* modulom.