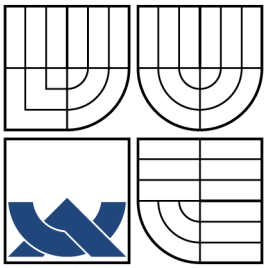
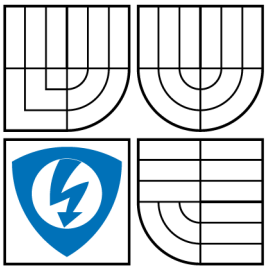


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

AUDIO A VIDEO VYSÍLÁNÍ S VYUŽITÍM REAL-TIME PROTOKOLU

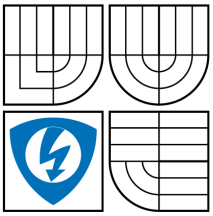
AUDIO AND VIDEO STREAMING USING REAL-TIME PROTOCOL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — TOMÁŠ KŘENEK
AUTHOR

VEDOUCÍ PRÁCE — ING. RADIM BURGET
SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Tomáš Křenek ***ID:*** 83542
Ročník: 2 ***Akademický rok:*** 2008/2009

NÁZEV TÉMATU:

Audio a video vysílání s využitím real-time protokolu

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s knihovnou AVcodec a popište ji. S jeho použitím implementujte vysílač, který bude načítat data z OGG souboru a transformovat je do formátu real-time protokolu. Připravte skripty pro převod různých formátů do formátu OGG. Dále implementujte přehrávač. Posuďte dosažené výsledkdy z pohledu licenncích podmínek pro připojené knihovny.

DOPORUČENÁ LITERATURA:

[1] SCHULZRINNE, H., CASNER, S., FREDERICK, R., JACOBSON, V.. RTP: A Transport Protocol for Real-Time Applications, Request for Comments 3550, Internet Engineering Task Force, 2003.
[2] An ffmpeg and SDL Tutorial, http://www.dranger.com/ffmpeg/tutorial01.html

Termín zadání: 9.2.2009 ***Termín odevzdání:*** 26.5.2009

Vedoucí práce: Ing. Radim Burget

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práve třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

ABSTRAKT

Tato diplomová práce je zaměřena na vysílání multimediálních dat přes počítačovou síť. V první části jsou podány podrobné informace o protokolech RTP a RTCP, pomocí nichž lze realizovat přenos. V dalších kapitolách jsou zmíněny a vysvětleny některé základní pojmy související s řešenou problematikou a následují informace o knihovnách použitých při vývoji aplikací. Ve druhé části práce je implementován přehrávač multimediálních souborů, který využívá kodeky FFmpeg a multimediální knihovnu SDL. Přehrávač je konzolovou aplikací, disponuje základními ovládacími prvky a je schopen reprodukovat multimediální obsah zadaného souboru. Další kapitoly praktické části se věnují problematice RTP komunikace. Jsou implementovány konzolové vysílače a přijímače přenášející data kódovaná pomocí Theory a Vorbisu. V závěru práce jsou shrnuty a zhodnoceny dosažené výsledky.

KLÍČOVÁ SLOVA

Přehrávač, FFmpeg, kodeky, SDL knihovna, ogg kontejner, RTP komunikace, klient, server, Xiph.Org Foundation, Theora, Vorbis, konzolová aplikace.

ABSTRACT

This diploma thesis focus on transmiting multimedia over computer network. There are detailed informations about protocols RTP and RTPC in a first part because a transmition over a network is realized by using these protocols. Some basic multimedia terms, FFmpeg codecs and SDL library are described in next chapters. A multimedia player using FFmpeg and SDL is implemented in a second part of thesis. The player is console application and it has basic user interface. The player reproduces video and audio from a given file. RTP communication is described in next chapters of the second part. RTP server and client are implemented there too. They are console aplications and they use data coded by Theora or Vorbis. There are summarized results in a conclusion.

KEYWORDS

Player, FFmpeg, codecs, SDL library, ogg container, RTP communication, client, server, Xiph.Org Foundation, Theora, Vorbis, console application.

KŘENEK T. *Audio a video vysílání s využitím real-time protokolu*. Brno: Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2009. 75 s. Diplomová práce. Vedoucí práce Ing. Radim Burget.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Audio a video vysílání s využitím real-time protokolu“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

Poděkování

Děkuji Ing. Radimu Burgetovi za metodické vedení a cenné připomínky, kterými přispěl k vypracování této diplomové práce.

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

V současné době dochází k neustálému nárůstu přenosových kapacit počítačových sítí, což umožňuje provozovat v těchto sítích velké množství služeb najednou, popř. nasazovat další druhy služeb, které dříve nebylo možné uskutečňovat. Také vysoký stupeň integrace elektronických prvků, spolu s jejich rostoucím výkonem, umožňuje, aby stacionární i mobilní terminály poskytovaly stále větší výkon a uživateli se tak nabízí možnost využít rostoucích kapacit sítí a jejich nových služeb v plném rozsahu. Relativně novou službou poskytovanou počítačovými sítěmi je příjem multimediálního obsahu (audio, video, popř. další doplňková data), který se postupně stává standardem i u mobilních zařízení jako jsou telefony či PDA[1]. Navíc v poslední době je snaha poskytovat uživatelům multimediální obsah ve vysoké kvalitě. Popularita počítačových sítí je na vysoké úrovni, přičemž svoji významnou roli zde sehrál i Internet. Sítě, založené na principu jedinečných číselných adres a komutace paketů, jsou dnes dostupné na mnoha veřejných místech a jejich počet neustále roste. Jednoduchost, snadná rozšiřitelnost, vysoká kompatibilita zařízení a možnost navyšovat přenosovou kapacitu, to jsou významná kritéria, která již dříve vedla k tomu, že pro počítačové sítě, prvotně určené pouze pro spolehlivý přenos dat, byl postupně vybudován protokolový aparát umožňující i nespolehlivý nepotvrzovaný přenos multimediálních dat. Dnes tak můžeme v počítačových sítích využívat i služby jako jsou např. videokonference, IP[2] telefonie, vysílání videa v reálném čase apod.

Pro správné zobrazování multimediálních dat uživateli musí být implementována podpora v síti i koncových zařízeních (osobní počítače, mobilní telefony, notebooky apod.). Zde je nutné zejména zajistit přítomnost dekodéru, nejčastěji softwarového, a také aplikací, které dokážou data po dekódování správně zobrazit. Nejčastěji jde o nějaký druh přehrávače. Postupem času bylo vyvinuto velké množství licencovaných i nelicencovaných kodérů a dekodérů videa i audia a ještě větší množství přehrávačů, které byly více či méně úspěšné a používané. Obliba kodeků a přehrávačů je – zejména v oblasti osobních počítačů – závislá především na jejich jednoduchosti, rychlosti, kvalitě zpracování dat a v neposlední řadě i na dostupnosti.

Tato práce se věnuje přenosu multimediálních dat pomocí protokolu RTP[3] přes počítačovou síť. V první části a jejích kapitolách je rozebrána teoretická stránka projektu. Je zde zmíněna problematika síťových protokolů, uvedeny základní informace o tvorbě audio a video signálu, popsán postup redukce datového toku i problematika dotýkající se kódování a způsobu ukládání multimediálních dat v počítačích. Na konci první části jsou také uvedeny informace o softwaru, který byl využit pro

[1] osobní digitální pomocník, kapesní počítač – Personal Digital Assistant

[2] datový protokol používaný pro přenos dat přes paketové sítě – Internet Protocol

[3] protokol pro přenos dat v reálném čase – Real Time Transport Protocol

tvorbu aplikace. Část druhá se věnuje implementaci programu, který je praktickým výstupem této práce. Jednotlivé kapitoly se zaměřují vždy na určitou část aplikace a korespondují s tím, jak byl program postupně vyvíjen. Část třetí obsahuje závěr, tj. shrnutí a zhodnocení dosažených výsledků.

Před samotným řešením daného tématu je nutné ještě uvést poznámky týkající se formální stránky práce. Výraz pakety bývá spojován se síťovou vrstvou TCP/IP[4] modelu. Na aplikační vrstvě se hovoří spíše o přenášení tzv. zpráv, avšak i samotné RFC[5] dokumenty používají taktéž pojem pakety. V následujícím textu je používáno v souvislosti s aplikační vrstvou obou pojmů. V širším slova smyslu je v následujících kapitolách pojem paket také spojen se získáváním a zpracováním dat z multimediálních souborů. V tomto případě je myšlena určitá posloupnost bitů. Obdobně je tomu u výrazu rámec, který slouží jako ekvivalent slova snímek.

[4] čtyřvrstvý síťový model využívaný ve většině dnešních sítí, založen na protokolech TCP a IP

[5] sada doporučení popisujících internetové protokoly, systémy aj.; sestavovány na základě praktických zkušeností – Request For Comments

1 SÍŤOVÉ PROTOKOLY, KONTEJNER, KODEK, PODPŮRNÝ SOFTWARE

Tato kapitola je věnována především teoretickým aspektům, které jsou spjaty s daným tématem. Následující text podá základní informace o přenosu multimediálních dat přes počítačovou síť (protokoly). Postupně zde bude také věnován prostor problematice komprese a archivace audia a videa v oblasti počítačové techniky. Na závěr kapitoly jsou uvedeny projekty z nichž bylo čerpáno (zdrojový kód, knihovny apod.) při tvorbě programu, který je praktickým výstupem této práce.

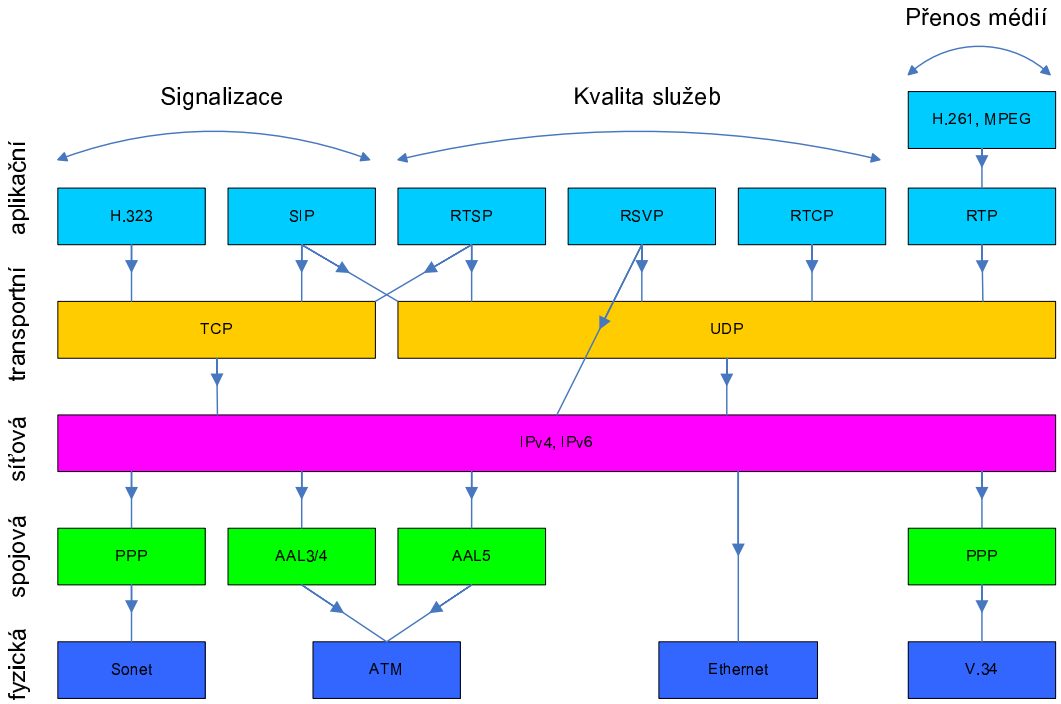
1.1 Síťové protokoly pro přenos multimediálních dat

Zejména s rozmachem Internetu je na počítačové sítě kladen postupně požadavek přenášet i multimediální data, jejichž datové toky se vyznačují některými vlastnostmi, které jsou odlišné od rysů „klasických“ dat. Např. pakety musí být v přijímači seřazeny tak jak byly vyslány, když přijdou mimo pořadí. Dále musí být detekována ztráta paketu a následně kompenzována, nejlépe bez znovu vysílání. Při přehrávání musí být zachován původní časový interval mezi pakety, stejný jako byl při vysílání. Navíc pokud existuje více datových toků (zvuk, video, text apod.) musí být zaveden mechanismus pro jejich vzájemnou synchronizaci, aby se některý z toků nezpožďoval či nepředbíhal oproti ostatním. V síti Internet je mnohdy požadavek také měnit styl kódování dle aktuální vytíženosti přenosové linky nebo změny šířky přenosového pásma. Proto byly pro přenos multimédií vytvořeny nové síťové protokoly, které lze spolu s ostatními protokoly pro přenos „klasických“ dat uspořádat do hierarchického modelu (obr. 1.1). Protože může existovat i více protokolů, které zajišťují stejnou úlohu v síti, zaměří se tato kapitola podrobněji zejména na nejužívanější protokoly, popř. na ty, které budou v praktické části použity pro vývoj aplikace.[17]

V podstatě lze protokoly spravující multimediální data v síti rozdělit na tři skupiny:

- pro přenos médií – doručují konkrétní mediální obsah,
- pro zajištění QoS[1] – monitorují datové toky, resp. jejich určité parametry,
- signalizační – realizují přenášení řídících dat mezi stanicemi.[17]

[1] soubor pravidel zajišťujících rozdělení síťových prostředků dle stanovených priorit – Quality of Service



Obr. 1.1: Hierarchická struktura protokolů podporujících přenos multimediálních dat.[17]

Nejdůležitějšími protokoly jsou ty, které zajišťují vlastní přenos dat. Nejznámějším je RTP, ale existují i jiné, např. firmy Microsoft a RealNetworks mají své vlastní patentované protokoly.[17]

Neméně důležité jsou protokoly, zajišťující požadovanou kvalitu služeb. Sem můžeme zařadit RTCP[2] a SNMP[3]. Jako výhodnější se jeví použití RTCP, protože oproti SNMP umožňuje rozlišovat mezi jednotlivými aplikacemi a navíc je přímo integrován v RTP.[17]

K rezervaci síťových prostředků dochází na základě sběru dat a jejich vyhodnocení, přičemž jsou uplatňována pravidla pro zajištění QoS. Při využití protokolu RSVP[4] mohou být síťové prostředky vyhrazovány buď pro jednotlivé mediální toky nebo pro celá spojení. Tokem zde rozumíme jedinou relaci vytvořenou mezi dvěma aplikacemi, např. sezení v případě videokonferencí. Relace je pak charakterizována zdrojovou a cílovou adresou, portem a protokolem.

Zajišťujeme-li rezervaci na úrovni toků, je možné pro každý z nich vyhradit určité předem definované parametry, což může probíhat dvěma způsoby:

[2]slouží k řízení RTP relace a ke sledování kvality toku – Real Time Control Protocol

[3]umožňuje průběžný sběr a vyhodnocování dat pro potřeby správy sítě – Simple Network Management Protocol

[4]protokol pro rezervaci zdrojů v síti – Resource reSerVation Protocol

- zajištění parametrů přenosu,
- řízení parametrů přenosu.

V prvním případě jsou hodnoty zvolených parametrů udržovány v určitých mezích, ve druhém není dodržování parametrů tak striktní, ale je dbáno, aby zvolené parametry neklesly pod určitou spodní hranici, jejíž překročení by mělo za následek znehodnocení poskytované služby.[17][1]

V oblasti signalizace lze provést dělení protokolů dle zacházení s multimediálním obsahem na tři skupiny:

1. Multimédia na požádání – zde se nejvíce využívá protokol RTSP[5]. Před zahájením přenosu musí vysílač vědět o přijímači a naopak. Přijímač si pak sám řídí relaci s vysílačem a dává mu příkazy (přehrát, pozastavit, rychlý přesun vpřed aj.). Tento druh přenosu multimédií přes síť je velmi citlivý na správnou signalizaci a stoupá tak režie v síti.

2. Internetová telefonie – podobá se prvnímu případu s tím rozdílem, že spojení je obousměrné a vysílání probíhá v reálném čase. Z toho vyplývá nutnost zajištění spojitého, pokud možno bezchybného, toku dat v obou směrech. Rozdílem oproti prvnímu případu je absence některých funkcí (pozastavení a převíjení). Používají se protokoly H.323[6] a SIP[7].

3. Skupinové vysílání – v tomto případě zdroj oznámí vysílání datového toku, na což reagují přijímače připojením do skupiny. Lze implementovat algoritmus, který zabrání vysílání, pokud se v síti nevyskytuje žádný přijímač. Obvykle použito s protokolem SAP[8].

Kromě signalizačních protokolů se ještě využívá protokolu pro popis datových toků s názvem SDP[9].

Výše uvedené rozdělení do jednotlivých kategorií se ve skutečnosti tak přísně nedodržuje. Je možná vzájemná kombinace. Např. „pozvání“ uživatele do skupinového vysílání lze realizovat pomocí SIP, Internetová telefonie může užívat RTSP pro hlasovou poštu apod.[17][1]

1.1.1 RTP protokol

Jedná se o nejužívanější protokol svého druhu. Definuje standardní formát paketu pro přenos multimediálních dat po síti. Poprvé byl tento protokol zveřejněn v roce

[5] protokol pro řízení doručování dat v reálném čase – Real Time Streaming Protocol
[6] je doporučení definující protokoly pro audio-vizuální relace komunikace v jakékoli paketové síti
[7] protokol pro inicializaci relací, přenos signalizace v IP telefonii – Session Initiation Protocol
[8] signalizační protokol pro řízení relací všesměrového vysílání – Session Announcement Protocol
[9] slouží k popisu relace a vyjednání parametrů spojení – Session Description Protocol

1996 jako RFC 1889. RTP používá pro přenos UDP[10] protokol z transportní vrstvy síťového modelu TCP/IP, ale může využívat i libovolný jiný protokol z nižší vrstvy. Použití UDP je vhodnější než TCP[11], protože aplikace, ve kterých je RTP nasazován, jsou málo citlivé na ztrátu paketu, ale velmi citlivé na zpoždění. RTP nemá stanoven standardní komunikační port, jediné pravidlo, které se v tomto ohledu dodržuje je takové, že nejbližší vyšší lichý port od RTP portu je vyhrazen pro protokol RTCP (viz 1.1.2). Pro RTP komunikaci se zpravidla volí port sudý[12] a RTCP pak má přidělen port o jednu vyšší, lichý. Jelikož komunikační port není pevně stanoven, bývá někdy u komunikace problém s průchodem bezpečnostními branami (firewally) v PC[13]. RTP protokol přenáší pouze multimediální data, neposkytuje mechanismy, které zajistí včasné a správné dodání paketů, ani nepřenáší žádné informace o charakteristikách spojení v reálném čase. K těmto účelům byly vyvinuty protokoly další. I přes některé nevýhody má však řadu výhod:[15][1][12]

1. Nezávislost na přenášených datech – lze jej využít pro všechny typy multimediálních dat, přičemž doplňkové informace např. o kódování dat jsou uvedeny ve zvláštní specifikaci.

2. Podporuje mixery a filtry – mixery umožňují sloučit více datových toků do jednoho a ten pak přenést. Filtry slouží ke konverzi toku dat a užívají se ke snížení požadavků na šířku pásma, aniž by zdroj multimediálního obsahu musel měnit svoji bitovou rychlost. To v praxi znamená, že přijímače s kvalitní linkou budou přijímat multimediální data ve vysoké kvalitě, stanice s pomalým připojením nebudou tímto objemem zahlceny a budou schopny data přehrát.

3. Jednoduchá implementace QoS – lze použít doplňkový RTCP protokol, který je v RTP integrován. Poskytuje přijímačům možnost informovat ostatní členy skupiny o kvalitě příjmu. Vysílače užívají tyto informace k nastavení svých bitových toků, přijímače zjišťují zda je problém s QoS lokální nebo v rámci celé skupiny.

4. Řízení sezení – opět s pomocí RTCP mohou účastníci sezení posílat své identifikační údaje, s jejichž pomocí lze monitorovat, kdo se sezení účastní. Tyto zprávy se posílají periodicky, a proto není nutné udržovat žádnou centrální databázi účastníků.

[10] protokol transportní vrstvy modelu TCP/IP, poskytuje nespolehlivou a nepotvrzovanou službu – User Datagram Protocol

[11] protokol transportní vrstvy modelu TCP/IP, poskytuje spolehlivou a potvrzovanou službu – Transmission Control Protocol

[12] z doporučeného rozsahu 16384–32767

[13] osobní počítač – Personal Computer

5. Bezpečnost – datové toky zasílané pomocí RTP mohou být kódovány. Bezpečnostní klíče je ale nutné před zahájením komunikace předat ostatním účastníkům, např. použitím protokolů SIP nebo SDP.[17][1][15][12]

Struktura RTP paketu

Základní strukturu lze rozdělit na čtyři části:

- IP hlavička,
- UDP hlavička,
- RTP hlavička,
- RTP zátěž (data).[1]

Celková délka hlaviček IP, UDP, RTP činí přibližně čtyřicet bajtů. Ty jsou přidávány k vlastním přenášeným datům v každém paketu. Tento způsob tvorby paketu však může být neefektivní, zejména v prostředích s nízkými přenosovými rychlostmi, proto se zavádí komprese, kdy se kóduje pouze rozdíl v IP, UDP a RTP hlavičkách mezi po sobě následujícími pakety. Tato metoda umožňuje redukovat velikost hlaviček na jeden až dva bajty, ale nevýhodou je citlivost na ztrátu paketů a funkčnost pouze v rámci jednoho skoku na spojové vrstvě. [17][12]

Výše zmíněné dělení je však velmi obecné, a proto je vhodnější, zejména pro účely vývoje aplikací, znát strukturu paketu podrobněji, jak ji naznačuje obrázek 1.2. Paket se pak skládá z těchto polí:

1. Pole Ver (verze protokolu – Version) – současná verze je 2 a velikost pole je 2 bity.
2. Příznak Pd (bezpečnostní rozšíření protokolu – Padding) – pokud je nastaven znamená to, že zátěž je doplněna o potřebný počet bajtů, určený pro případné zabezpečení. Velikost je 1 bit.
3. Příznak X (rozšíření – Extension) – indikuje přítomnost rozšířené hlavičky mezi hlavičkou fixní a datovou částí. Velikost pole činí 1 bit.
4. Pole CC (počet záznamů v poli CSRC – Count CSRC) – velikost 4 bity.
5. Příznak M (značkovač – Marker) – význam tohoto bitu je definován použitým profilem. Pokud je nastaven, znamená to v podstatě, že přenášená data v paketu mají nějaký zvláštní význam. Využívá se při zpracování rámců multimediálních dat. Např. v případě videa označuje, že je daný paket již poslední pro sestavení celého snímku.

6. Pole PT (druh zátěže – Payload Type) – definuje použité kódování pro přenášená data. Vyhrazeno je 7 bitů.

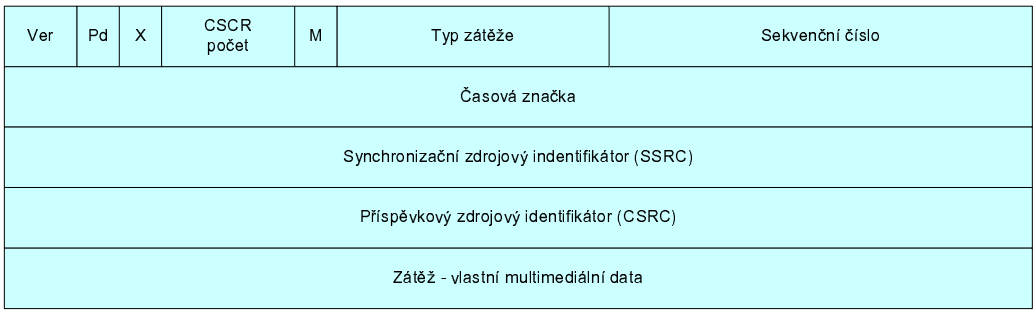
7. Sequence number – sekvenční číslo. Jeho hodnota je postupně zvyšována s každým odeslaným paketem a užívá se k detekci ztrát a obnovení správného pořadí paketů. Vyhrazeno je 16 bitů a počáteční hodnota může být náhodná.

8. Timestamp – časová značka. Toto pole je inkrementováno na základě vzorkovací frekvence přenášeného datového toku (signálu) a označuje, kdy byl paket vytvořen. Vyhrazeno je 32 bitů. Počáteční hodnota smí být náhodná.

9. SSRC (synchronizační zdrojový identifikátor – Synchronization Source identifier) – pomocí něj jsou rozlišeni uživatelé v rámci adresní skupiny. Umožňuje tak rozeznat jednotlivé toky, které pocházející ze stejného mixeru či translátoru, a identifikovat zdroje na základě zpráv přijímačů. Stane-li se, že dvěma uživatelům je přiřazen stejný identifikátor současně, jeden z nich si vygeneruje nový. Pro toto pole je vyhrazeno 32 bitů.

10. CSRC (příspěvkový zdrojový identifikátor – Contributing Source identifier) – pole obsahující seznam všech SSRC, které přispívají nějakým obsahem do vytvořeného paketu. Používá se v případě mixerů, které slučují mediální toky z několika zdrojů (např. videokonfrence). Seznam může obsahovat maximálně 15 položek, přičemž každá může mít velikost až 32 bitů.

11. Payload – zátěž (data). Pole může kromě vlastních multimediálních dat obsahovat i speciální hlavičku pro přenášená data.[17][15][1][12]



Obr. 1.2: Hlavička RTP paketu.[17]

Typy zátěže

RTP protokol je vytvořen tak, že je možné přenášet jakýkoliv multimediální obsah v libovolném kódování. Některé kodeky (viz 1.2.3) však mají dodatečné požadavky na přenášené informace. Pro podporu těchto kodeků je v RTP implementována specifikace přenášených dat. Konkrétní data jsou v paketu přenášena v části s názvem zátěž (Payload) a jejich typ je dále specifikován sedmibitovým číslem, které odpovídá přesnému názvu a kodeku. Vzájemný vztah mezi číslem a typem dat je definován tzv. mapováním hodnoty na odpovídající název. Mapování je dvojího druhu:

- statické,
- dynamické.

Statické mapování se vztahuje na kodeky a formáty definované v dokumentu profilů RTP. Např. hodnota 0 odpovídá zvukovému kodeku μ-law[14]. V důsledku velkého množství nově vznikajících kodeků a omezeného počtu možných číselných hodnot již nejsou definovány nové typy statických zátěží. Byla zvolena jiná varianta. Aplikace se na začátku komunikace musí dohodnout jaký typ dat odpovídá danému číslu typu zátěže. Pro toto dynamické mapování jsou vyhrazena čísla 96–127. Chceme-li tedy např. pod číslem 120 používat kodek H.263[15], je nutné, aby si komunikující strany vyměnily tuto informaci pomocí protokolu H.323 nebo SDP. Tímto způsobem je zajištěno, že pomocí RTP lze přenést prakticky libovolně kódovaná data. Statické typy zátěží byly definovány např. pro H.263, H.261[16], JPEG[17] a video kodeky MPEG[18].[15][17][12]

1.1.2 RTCP protokol

RTCP je sesterským protokolem RTP, je definován ve stejném RFC dokumentu. Podílí se na přenosu dat, ale sám není schopen přenášet žádná multimediální data. RTCP je v podstatě řídícím protokolem RTP, protože dává zdroji vysílání zpětnou vazbu v podobě informací o kvalitě poskytované služby. Pomocí RTCP protokolu můžeme získat informace týkající se:

- odeslaných paketů,
- ztracených paketů,

[14] algoritmus redukující dynamický rozsah audio signálu; užívá se v Japonsku a Severní Americe
[15] video kodek navržený pro videokonference s nízkými přenosovými rychlostmi
[16] video kodek navržený pro přenos obrazu telefonními linkami ISDN
[17] metoda ztrátové komprese pro statické obrazy – Joint Photographic Experts Group
[18] skupina standardů pro kompresi audiovizuálních dat – Motion Picture Experts Group

- doručovacího zpoždění,
- kolísání zpoždění aj.[12][1]

Na základě těchto obdržených hodnot může vysílací aplikace reagovat a zvýšit kvalitu poskytované služby např. použitím jiného kodeku či změnou datového toku. Pro RTCP protokol se doporučuje používat port vždy o jedno vyšší než je použitý RTP port. Obvykle tedy port lichý. Samotný protokol také není schopen poskytovat žádné zabezpečení či autentizaci. U RTCP protokolu můžeme rozlišit tyto typy paketů:[12][17]

1. SR (zprávy vysílače – Sender Report) – jsou generovány uzly, které vysílají multimediální obsah (jsou zdrojem RTP paketů). Tyto zprávy popisují celkové množství vyslaných dat, obsahují také údaje o absolutním čase, a tak umožňují vzájemnou synchronizaci datových toků.

2. RR (zprávy přijímače – Receiver Report) – jsou posílány přijímači RTP sezení. Každá zpráva obsahuje jeden blok dat pro každý RTP zdroj ve skupině. Zmíněné bloky popisují okamžitou a kumulativní ztrátu paketů, kolísání zpoždění od zdroje, obsahují i poslední časové razítko a zpoždění od příjmu poslední zprávy vysílače. Tímto způsobem lze ve zdrojích stanovit přibližnou vzdálenost od přijímačů. Zprávy typu SR a RR mají téměř totožnou strukturu paketu, která je zobrazena na obr. 1.3.

3. SDES (popisovače zdrojů – Source Description items) – tyto pakety jsou užívány pro řízení sezení a obsahují pole, která informují o vlastnostech zdrojů v RTP sezení. Nejvýznamnějším je pole s názvem CNAME[19], což je jedinečný identifikátor, jehož forma je podobná e-mailové adrese. CNAME se užívá pro řešení konfliktů s hodnotou SSRC a k přiřazení různých mediálních toků generovaných stejným uživatelem. Pakety SDES mohou poskytovat i další identifikační údaje o účastníkovi sezení, např.:

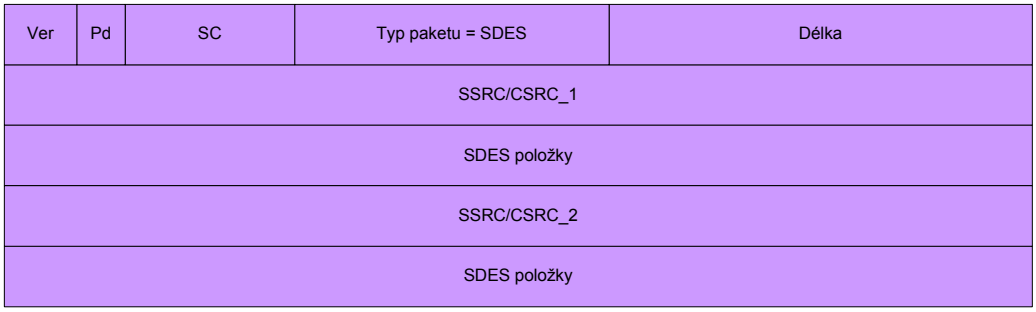
    - jméno,
    - e-mailová adresa,
    - telefonní číslo,
    - geografické umístění,
    - název použité aplikace.

[19] kanonické jméno – Canonical Name

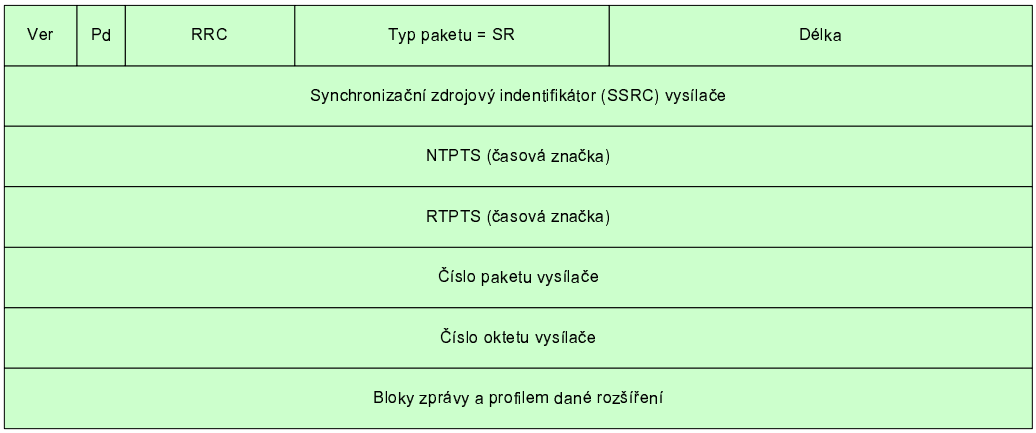
Klientské aplikace pak mohou v GUI[20] zobrazit tyto osobní informace. Účastníci tak mohou mít přehled o všech ostatních uživatelích v daném sezení a mohou také získávat kontaktní informace. Struktura paketu typu SDES je ukázána na obr. 1.4

4. BYE (indikátor ukončení spojení) – tuto zprávy posílá uživatel, který hodlá ukončit spojení.

5. APP (aplikačně specifické funkce – Application) – tento typ zprávy generují samotné aplikace a slouží k upřesnění nebo doplnění informací, např. důvod ukončení spojení apod.[11][17][15][12]



Obr. 1.3: Struktura RTCP paketu typu SR (typ RR je velmi podobný).[11]



Obr. 1.4: Struktura RTCP paketu typu SDES.[11]

Jelikož vysílač by měl mít neustále přehled o počtu vyslaných paketů a také musí hlídat připojení popř. odpojení účastníka, je třeba, aby pakety RTCP protokolu

[20] grafické uživatelské rozhraní – Graphic User Interface

byly vysílány periodicky. Ovšem v případě, že by účastníci sezení posílali pakety RTCP s fixní periodou, lineárně by narůstala obsazená šířka pásma přenosového kanálu s počtem účastníků sezení. To je nežádoucí. Řešením je, že si každý účastník zjistí počet dalších uživatelů ve skupině na základě posílaných RTCP zpráv. Perioda zasílání RTCP paketů od každého uživatele je pak nastavena lineárně v závislosti právě na počtu účastníků ve skupině. Tímto způsobem je zajištěno, že šířka pásma, kterou spotřebují RTCP pakety, bude fixní a nezávislá na počtu účastníků sezení. Velikost skupiny je zjišťována postupně sčítáním komunikujících uživatelů, proto nově příchozímu účastníkovi zabere jistý čas, než získá správný počet účastníků. V případě přihlašování velkého množství nových uživatelů do skupiny je zpravidla z počátku velikost skupiny detekována chybně. To způsobí „záplavu“ RTCP paketů, která může být odstraněna vytvořením vhodného algoritmu.[17][15][12]

1.2 Formáty souborů, kontejnery, kodeky

V této kapitole budou uvedeny a vysvětleny některé základní pojmy užívané v dalším textu a související se zpracováním a reprodukcí multimediálního obsahu v oblasti počítačové techniky.

1.2.1 Formát souboru

Protože vyvíjené aplikace jsou určeny pro OS[21] Windows, omezíme se na informace týkající se tohoto operačního systému. Formát souboru, někdy označován také jako typ souboru, určuje v podstatě význam dat uložených na disku počítače. Existuje velké množství různých formátů přizpůsobených pro ukládání rozmanitých typů informace, např. pro text, obrázky, hudbu, video atp. Často se vykytuje i více formátů pro reprezentaci jednoho typu dat (audio, video). Rozlišovat lze formáty určené pro jediný druh dat, jako je např. JPEG, který je určen výhradně pro statické obrázky, i takové, jenž slouží pro uložení více typů informace (viz 1.2.2).

Ve Windows je formát souboru posuzován dle přípony, která následuje za jménem souboru, resp. za následující tečkou. Přípona bývá z historických důvodů tvořena třemi znaky, přičemž se mohou vyskytovat číslice, malá i velká písmena. Jednotlivé tříznakové koncovky jsou pak v OS asociovány s aplikacemi, které si umí s daným typem dat v souboru poradit. Přiřazení k programům i příponu názvu souboru je možné měnit. V obou případech tak lze docílit změny aplikace, která soubor otvírá, ale typ dat, který je uložen v souboru, zůstává zachován.

[21] operační systém

1.2.2 Multimediální kontejner

Jedná se formát souboru, který v sobě sdružuje množství datových toků, což znamená, že může obsahovat jednu video stopu, k ní několik stop audia, případně i titulky. Jednotlivé typy kontejnerů se vzájemně liší podle svých schopností pojmout různá multimediální data. Některé mohou mít v sobě uloženy pouze omezenou množinu formátů (např. MPEG), jiné jsou tolerantní i k více proudům jednoho typu dat (např. ogg 1.3.3). Pro přehrání jednotlivých kontejnerů se používá program nebo zařízení s názvem demuxer (rozdělovač), které rozdělí datové proudy do různých kodeků a následně do výstupních zařízení. Kontejner sám neříká nic o vnitřní kompresi uložených dat, ta je určena použitým kodekem, avšak v souboru mohou být uloženy informace o tom, jaké kodeky lze pro jednotlivé stopy použít.[9][13]

1.2.3 Kodek

Název je poskládán z počátečních slabik slov **ko**dér a **dek**odér, resp. **ko**mprese a **dek**omprese. Jedná se o zařízení nebo počítačový program, který dokáže transformovat datový proud nebo signál. Kodeky ukládají data do zakódované formy za účelem přenosu, uchovávání nebo šifrování a používají se také pro přesné nebo přibližné obnovení původních dat, se kterými je následně nějakým způsobem manipulováno (zobrazování na monitoru apod.). Kodeky jsou základní součástí aplikací pro editaci multimediálních souborů (zvuk, obraz) a často se používají i pro videokonference a distribuci multimediálních dat v sítích.[8]

Kodek je někdy zaměňován s formátem audia nebo videa, který je daný svojí specifikací. Příkladem specifikace je standard MPEG-1 Audio Layer 3[22], který je využíván např. v kodecích LAME[23] a FhG[24]. V některých případech k záměně pojmů přispívá fakt, že se název kodeku shoduje s názvem formátu.

Jako kodek je také velmi často nazýván pouze dekodér, což je případ různých přehrávačů, které zpravidla nepoužívají pro svoji práci kodeky, poněvadž pouze reprodukují obsah souboru a k tomu postačí dekodér. V těchto případech je ovšem situace poněkud složitější, neboť již vznikají robustní aplikace umožňující kromě reprodukce i zachytávání videa a audia z různých zdrojů a k této operaci mají implementován i kodér.

Základní dělení kodeků lze provést na:

- ztrátové – při kompresi dochází k neobnovitelné ztrátě informace,
- bezeztrátové – kódováním nejsou data nevratně změněna.

[22] standard pro kódování zvuku (MPEG-1 Audio Layer 3)
[23] vysoce kvalitní MPEG Audio Layer 3 kodér uvolněný pod licencí LGPL (LAME)
[24] vysoce efektivní MP3 kodek – Fraunhofer-Gesellschaft (FhG)

Dále je možné dělit kodeky dle typu zpracovávaných dat, např. na audio a video.[8]

Jak vyplývá z předchozího textu je pojem kodek přesně vymezen, ale v jeho správném užívání se vyskytují jisté nedostatky i mezi odborníky. I přes to, že v následujících kapitolách bude respektována zmíněná definice, může být použití výrazu kodek někdy zavádějící.

1.3 Podpůrný software

Následující text je již úzce svázán s praktickou částí práce. Pro vývoj výsledného programu byly využity některé již existující volně dostupné softwarové prvky zabývající se zpracováním multimédií (např. knihovny statické i dynamické) a tato kapitola podává informace o projektech, z nichž byly některé jejich části převzaty. Zejména zde jsou uvedeny tyto informace: název projektu, stručný popis, možnosti využití.

1.3.1 FFmpeg

Jedná se o programové vybavení, které poskytuje možnost komplexního zpracování digitálního obrazu a zvuku. Tento balík je primárně vyvíjen pod OS Linux[25], ale některé jeho části lze zkompilovat i pro jiné OS. FFmpeg je šířen pod GPL[26] nebo LGPL[27] licencí, přičemž závisí na konkrétní části kódu. Nejsou vydávány oficiální stabilní verze, ale vývojáři doporučují použít vždy poslední „funkční“ verzi, která je dostupná na Internetu. Někteří z vývojářů FFmpegu se podílí i na tvorbě jiných programů, které více či méně využívají určité části z projektu FFmpeg. Samotný balík FFmpeg se skládá z těchto komponent:[2]

- ffmpeg – program pro příkazovou řádku pro převod mezi multimediálními formáty, podporuje také zachytávání v reálném čase, např. z televizní karty;
- ffserver – multimediální server pro skupinová vysílání;
- ffplay – jednoduchý multimediální přehrávač založený na SDL a knihovnách FFmpeg;
- libavcodec – knihovna obsahující všechny audio a video kodéry a dekodéry;
- libavfilter – knihovna schopná implementovat do multimédií různé filtry;

[25] vysvětlivky k názvům operačních systémů lze nalézt v části 3

[26] licence pro svobodný software; všeobecná veřejná licence – General Public License

[27] přísnější licence odvozená od GPL; zpravidla pro softwarové knihovny – Lesser GPL

- libavformat – je knihovna obsahující slučovače a rozdělovače pro kontejnerové formáty;
- libavutil – pomocná knihovna obsahující rutiny společné pro jednotlivé části kolekce FFmpeg;
- libpostproc – obsahuje algoritmy pro dodatečné zpracování videa;
- libswscale – umožňuje různou úpravu videa.[2]

Mezi hlavní výhody FFmpeg patří podpora téměř všech dnes užívaných formátů audia a videa, rychlost zpracování dat, podpora práce s kontejnery, možnost převodu mezi formáty a úprava multimédií. Značnou výhodou je i to, že celý balík je implementován v jazyce C[28].[5]

Obr. 1.5: Logo knihovny ffmpeg.[2]

1.3.2 SDL

Je multiplatformní multimediální knihovna, jednoduchá vrstva pro přímý přístup k médiím – Simple DirectMedia Layer, navržená tak, aby poskytovala nízkoúrovňový přístup k audio zařízení, klávesnici, myši i k 3D[29] zařízení. Kromě toho také podporuje 2D[30] operace s pixely, přístup k souborům, zpracování událostí, časování, vlákna atd. SDL knihovna je často používaná jako doplněk OpenGL[31] k nastavení grafického výstupu a poskytnutí vstupu z klávesnice a myši. Implementace SDL je provedena v jazyce C, v C++[32] běží nativně, ale je možné ji využívat i v mnoha jiných programovacích jazycích (Java, Python, Perl, Delphi atd.)[33] pomocí softwarových nadstaveb. SDL knihovna je šířena s otevřeným zdrojovým kódem, pod LGPL licencí, a často je užívána pro velké množství multimediálních aplikací od různých přehrávačů až po hry. Samotný princip funkce SDL je velmi jednoduchý.

[28] programovací jazyk vyvinutý v 70. letech 20. stol. pro potřeby OS Unix

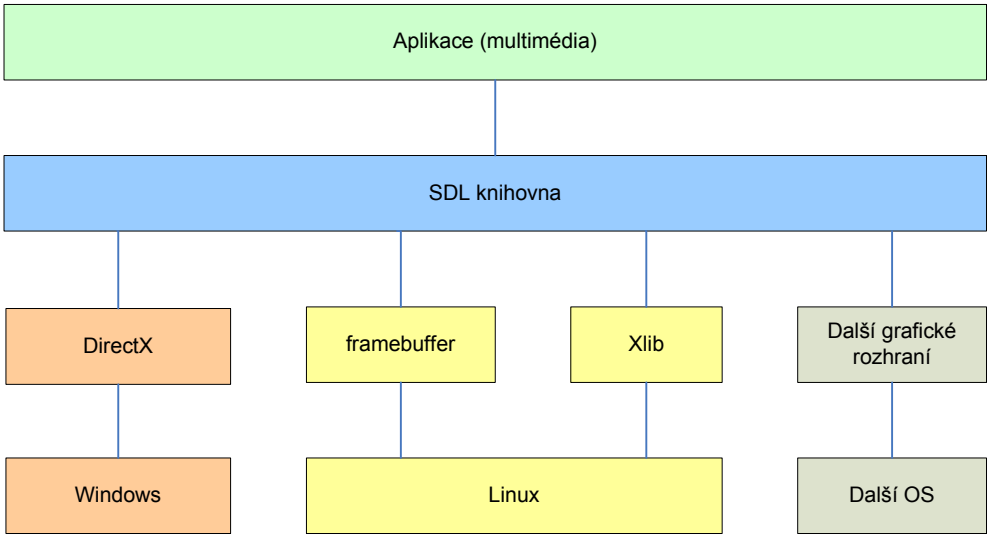
[29] obecná zkratka pro popis „světa“ pomocí tří souřadnic, odpovídá skutečnosti jak ji vnímáme

[30] obecná zkratka pro popis „světa“ pomocí dvou souřadnic; rovina, např. papír aj.

[31] standard specifikující multiplatformní rozhraní pro tvorbu aplikací počítačové grafiky – Open Graphics Library

[32] objektově orientovaný programovací jazyk, vznikl rozšířením jazyka C

[33] vysvětlivky k těmto názvům programovacích jazyků lze nalézt v části 3



Obr. 1.6: Umístění SDL v operačních systémech.[10]

Pracuje jako tenký obal nad funkcionalitou specifickou pro konkrétní OS, z čehož vyplývá, že knihovna je použitelná na mnoha OS (Windows, Linux, Mac OS, Solaris, BSD).[10]

Obr. 1.7: Logo SDL knihovny.[10]

1.3.3 Xiph.Org Foundation

Je neziskové sdružení, jehož cílem je vyvinout komponenty pro kódování a dekódování multimediálního obsahu, jenž budou svobodně dostupné a svobodně implementovatelné v dalších aplikacích. Různé části projektu jsou vyvíjeny pod BSD licencí a zamýšleny jako alternativy k nesvobodným standardům. Xiph poskytuje velmi dobrou dokumentaci, která napomáhá programátorům při vývoji vlastních aplikací, a je tvůrcem multimediálního kontejneru ogg a následujících kodeků:[13][20]

- Vorbis,
- FLAC,

- Speex,
- Theora.

Obr. 1.8: Logo projektu xiph.[20]

Theora

Principem vychází ze staršího původně licencovaného kodeku VP3[34], který byl později zbaven licenčních a patentových břemen. V roce 2002 pak začal samotný vývoj nástupce VP3, svobodného kodeku Theora, který je určen pro ztrátovou kompresi videa.[20]

Zakódovanou posloupnost snímků lze ukládat do vhodného kontejneru, přičemž je preferován ogg. Podobně jako jiné ztrátové komprese využívá Theora podvzorkování barvonosných složek, dělení obrazu na bloky 8 x 8 pixelů a pro transformační kódování je nasazena diskrétní kosinová transformace[35]. Dále je pak využito kompenzace pohybu v obraze na úrovni bloků a Theora podporuje také kódování rozdílů mezi jednotlivými po sobě jdoucími snímky, avšak pouze jednosměrně[36]. Datový tok vycházející z kodéru má proměnnou hodnotu. Standardní přípona souboru obsahujícího pouze data komprimovaná Theorou je ogv.[13][6][8][20]

Po několika letech testování a vydávání beta verzí byla na konci roku 2008 uveřejněna první stabilní verze 1.0, která by měla být konkurencí pro kodeky na bázi MPEG-4 Video[37]. Současný vývoj je zaměřen na stabilizování experimentální větve, která se má stát základem pro verzi 1.1. Kodek Theora je již podporován v mnoha knihovnách i aplikacích (např. VLC media player 2.2.2, GStreamer 2.2.2, FFmpeg aj.).[20]

[34] ztrátový video kodek vyvinutý společností On2 technologies
[35] velmi podobné kompresi JPEG
[36] analogie s kompresí H.261
[37] standard pro kódování videa, zastřešuje několik variant

Vorbis

Je ztrátový kodek určený pro kompresi zvuku (hudby). Jeho vývojem, který začal přibližně v roce 1998, měla být vytvořena alternativa k formátu MP3[38]. První stabilní verze Vorbisu 1.0 pochází z roku 2002 a poslední 1.2.0 byla uvolněna v červenci 2007. Vorbis si získal velkou oblibu v oblasti svobodného softwaru a je používán také pro kompresi zvuků u počítačových her. Oproti ostatním zvukovým kodekům tkví výhoda Vorbisu také v tom, že je schopen vytvářen soubory o menší velikosti při stejné nebo vyšší kvalitě. Zastaralejší formát MP3 se mu však ze spotřebního průmyslu doposud vytlačit nepodařilo.[20]

Kodek umožňuje nastavení kvality v deseti hladinách, kdy měněným atributem je hodnota průměrného datového toku, jehož rozpětí může být stanoveno v rozsahu 45–500 kb/s[39]. Jak plyne z předchozí informace, Vorbis je koncipován s variabilním výstupním datovým tokem kolísajícím okolo určité hodnoty. Pro zpracování je nutné vstupní data převést z časové domény do frekvenční oblasti pomocí modifikované diskrétní kosinové transformace. Získaný výsledek je následně rozdělen na dvě části. Oblast šumu a ostatní složky. Dále je provedeno kvantování a entropické kódování s využitím kódové knihy, která je založena na bázi vektorové kvantizace. Při dekompresi je posloupnost uvedených kroků opačná. Hladina šumu určuje charakteristické zkreslení kodeku, které je patrné zejména při nízkých bitových rychlostech. Zvukově působí zkreslení jako odrazy původního signálu od stěn místnosti.[4][8][20]

Přípona souboru, jenž obsahuje pouze data komprimovaná Vorbisem, může být dvojí ogg popř. oga. Jako kontejner je upřednostňován ogg. Ve specifikaci kodeku jsou podporována i metadata. Např. u hudby slouží k popisu skladby a interpreta. Co se týče podpory, tak existuje v softwarové (VLC media player, Windows Media Player, FFmpeg aj.) i hardwarové podobě (např. kapesní přehrávače značky iRiver). Jelikož Vorbis spatřil světlo světa již před poměrně dlouhou dobou, existují už i deriváty z něj odvozené.[13][6][20]

[38] viz MPEG-1 Audio Layer 3

[39] odvozená jednotka pro přenosovou rychlost, kilobity za sekundu

2 VÝVOJ SOFTWAROVÝCH APLIKACÍ

Dříve než zde bude popsáno, jak byla implementována aplikace plnící zadané funkce, je nutné uvést, že každá podkapitola je zaměřena na určitý úsek tvorby výsledného programu. Prostor je věnován především úskalím, která při vývoji vznikla a jak byla vyřešena.

Pro tvorbu programu bylo použito vývojové prostředí MVS[1]. Jedná se o komerční vývojové prostředí společnosti Microsoft, což přináší výhody i nevýhody. Výhodou je, že toto vývojové prostředí je velice kvalitní, má uživatelsky příjemné GUI, programátor může velmi detailně a pohodlně sledovat chování programu za běhu a je možné podrobně nastavit vlastnosti projektu, což může být někdy matoucí. Nespornou výhodou je i skutečně stabilní a bezproblémový běh MVS. Velkým nedostatkem je kompatibilita, která plyne i z komerčního původu vývojového prostředí. MVS je určeno pouze pro OS Windows, což v důsledku vede k tomu, že ne všechny zdrojové kódy je možné pod MVS sestavit do spustitelného souboru. Zejména jsou problémy se zdrojovými kódy vyvíjenými pod svobodnou licencí[2], zvláště pak pokud jsou výsledné programy určeny jako multiplatformní. Navíc společnost Microsoft nepřijala některé vydávané standardy (např. C99[3]), což může vést k dalším problémům s kompatibilitou.[14]

Výstupem této části práce bude aplikace plnící požadované funkce. Jelikož půjde o poměrně rozsáhlý program, je vhodné jeho vznik rozdělit na určitý počet na sebe navazujících úkonů. Navíc požadavky na výstupní aplikaci lze rozdělit do dvou skupin:

1. operace spojené s přehráváním multimédií,
2. činnost související s přenosem dat po síti.

Uvedené tématické oblasti lze s výhodou zpracovávat odděleně. Nejprve tedy bude vytvořen přehrávač. Dále dojde k realizaci síťového přenosu a po odladění obou vzniklých částí bude následovat provázání aplikací. Implementace programu tak bude probíhat přibližně podle následujících kroků:

- získání a zpracování dat z multimediálního souboru,
- zobrazení video obsahu,
- reprodukce audia,
- synchronizace datových toků,

[1] Microsoft Visual Studio 2008

[2] ani samotní vývojáři nejeví příliš velký zájem, aby jejich kód byl pod MVS sestavitelný

[3] standard ISO 9899:1999 vydaný v roce 1999 pro jazyk C, později přijatý i jako ANSI

- uživatelské rozhraní přehrávače,
- tvorba RTP serveru a klienta pro zvuk,
- tvorba RTP serveru a klienta pro video,
- provázání RTP komunikace a přehrávače,
- přenos videa i audia po síti se současným přehráváním.

2.1 Multimediální přehrávač

Jeho úkolem bude zprostředkovat uživateli data uložená v multimediálním souboru, z čehož lze odvodit, že základní požadavky na vytvořený program jsou:

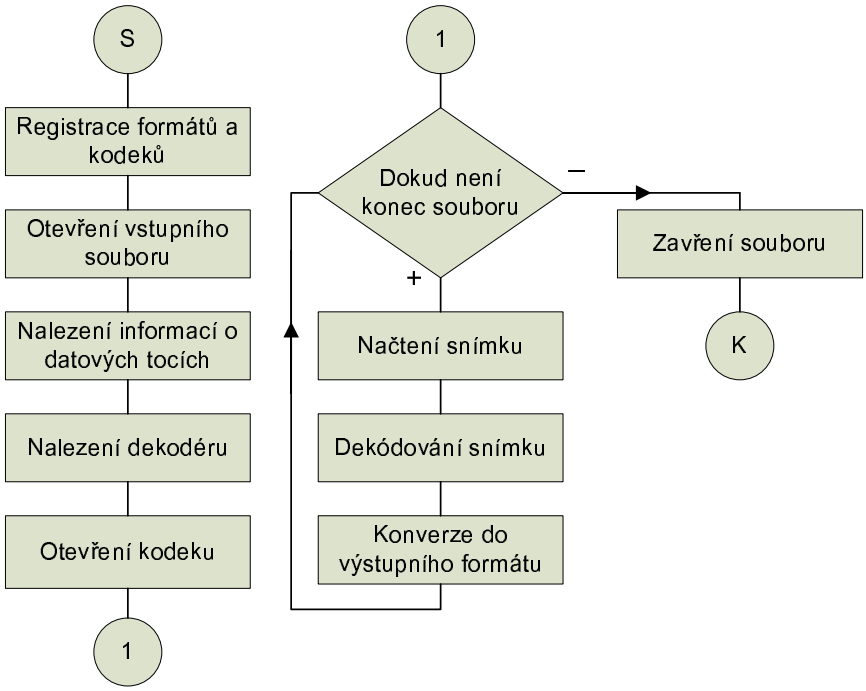
- zpracovat data ze souboru,
- korektní prezentace audia a videa,
- základní ovládací funkce (pauza, převíjení atp.).

Pro vývoj přehrávače byly využity multimediální knihovny FFmpeg a SDL. Ačkoliv je softwarový balík FFmpeg vyvíjen pod OS Linux, jsou části umožňující přehrávání multimediálního obsahu dobře zdokumentovány a z části kompatibilní i s OS Windows. Pro tvorbu přehrávače byly použity prvky z knihoven libavcodec, libavformat a libswscale. Dále bylo využito také některých vlastností knihovny SDL.[3][2][5]

Ze zdroje [14] byly získány statické knihovny FFmpeg, které jsou linkovány přímo ke zdrojovému kódu přehrávače. Přidělením knihoven k projektu tímto způsobem sice dochází ke zvyšování velikosti výsledného souboru, ale dekodéry potřebné pro přehrávání si přehrávač v podstatě „nosí s sebou“. Navíc aktuální verze FFmpeg jsou vydávány poměrně často a mohlo by docházet k tomu, že nové dynamické knihovny nemusí být zpětně kompatibilní se starými, popř. mohou obsahovat chyby. U knihovny SDL je situace jednodušší. Ze zdroje [10] byly získány knihovny pro vývoj a přiřazeny do projektu.

Využitím SDL a FFmpeg byl položen dobrý základ pro tvorbu přehrávače multimediálních souborů, ale knihovny samy o sobě jsou téměř bezcenné. Teprve jejich implementace a navázání na další zdrojový kód je činí užitečnými. Množství kódu, nutné k dotvoření, bylo značné. Tento fakt vedl k rozdělení postupu tvorby do sledu kroků, jimž odpovídají následující podkapitoly. Pokud bylo při tvorbě využito SDL nebo FFmpeg, je uvedeno jak a k čemu.

Implementace přehrávače byla provedena v programovacím jazyce C. Ten je dostatečně nízkoúrovňový, ale zároveň jednoduchý, výkonný a SDL i FFmpeg jsou v něm implementovány.



Obr. 2.1: Zpracování snímků.

2.1.1 Základní operace s multimediálním souborem a jeho obsahem

Dříve než budou informace uložené v souboru na paměťovém médiu reprezentovány uživateli, je nutné provést poměrně velké množství kroků, které připraví data na zobrazení. Také je nutné nejprve přiblížit, jakým způsobem jsou data v souboru organizována, resp. jak jsou zpracovávána pomocí FFmpeg funkcí (knihoven).

Soubor, v němž jsou data uložena, tvoří jakýsi „obal“. Nazýváme ho kontejnerem. Typ kontejneru určuje, kde vlastně hledat příslušná data, protože v každém souboru je zpravidla uloženo několik datových proudů představujících obraz a zvuk. Každý tok uložený v kontejneru může být zpracováván jiným typem dekodéru. Jednotlivé prvky datových proudů nazýváme rámce. Do rámců jsou dekódovány tzv. pakety, což jsou vlastně části toků obsahující vždy určitý počet bitů dat uložených v souboru. Vytvořený program pak manipuluje s rámci, kde platí, že každý načtený paket obsahuje kompletní rámec popř. několik rámců.[3][5][21]

Zjednodušený princip zpracování snímků pak zobrazuje vývojový diagram na obr. 2.1.

Při implementaci uvedených událostí byly využity funkce knihoven FFmpeg. Existujících knihoven bylo využito na doporučení, protože jen tvorba kodeků samot-

ných by svým rozsahem výrazně překročila rámec diplomové práce. Hlavní výhoda tohoto postupu tkví v tom, že není nutné řešit detaily týkající se kodeků, kterých je v současné době obrovské množství a lze se v řešení daného tématu posunout dále.

Problémem v této části vývoje se stala funkce:

```
int img_convert(AVPicture *dst, int dst_pix_fmt, const AVPicture
        *src, int pix_fmt, int width, int height),
```
[4]

která má za úkol převést obraz z nativního zdrojového do cílového formátu (RGB[5], YUV[6] apod.) a uložit jej na definované místo v paměti. Tato funkce neprošla překladem zdrojového kódu, přičemž přesný důvod neúspěchu nebyl zjištěn. Prototyp funkce je obsažen v hlavičkovém souboru tj. problém nastal nejspíše u knihovny FFmpeg. Bylo zjištěno, že původní funkci lze nahradit dvojicí funkcí:

```
SwsContext * sws_getContext (int srcW, int srcH, int srcFormat, int
  dstW, int dstH, int dstFormat, int flags, SwsFilter *srcFilter,
            SwsFilter *dstFilter, double *param)
```

a

```
  int sws_scale (struct SwsContext *ctx, uint8_t *src[], int
 srcStride[], int srcSliceY, int srcSliceH, uint8_t *dst[], int
                        dstStride[]),
```

které v podstatě realizují tutéž činnost a navíc, jak vyplývá z dokumentace, jsou tyto funkce z části FFmpegu, jež zavádí nové rychlejší a modulárnější rozhraní s lepším využitím multimediálně zaměřených části procesoru PC.[5]

2.1.2 Zobrazení videa

V této části vývoje přichází na řadu i knihovna SDL, která je použita pro zobrazování videa na obrazovce resp. monitoru. Než bude popsáno, jakým způsobem je video zobrazeno, je nutné zmínit několik teoretických aspektů.

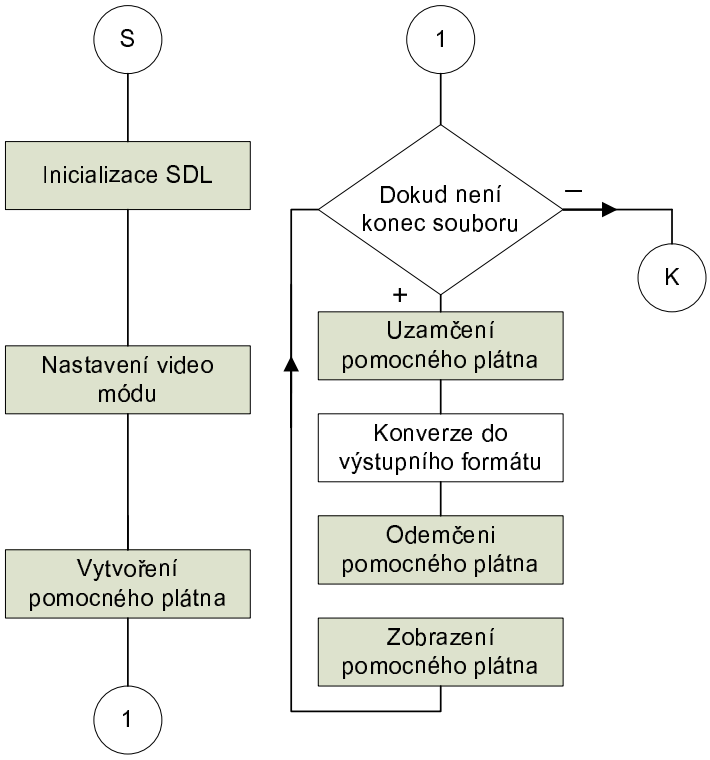
SDL umožňuje několik režimů zobrazování snímků, jeden z nich se nazývá YUV overlay. YUV[7] je způsob ukládání video snímků, kde je k popisu barvy využit tříprvkový vektor skládající se ze složek: Y – vyjadřuje jas, U, V – nesou barevné informace. YUV lze matematicky přepočítat na RGB a opačně. Oproti RGB má však jednu podstatnou výhodu. Při podvzorkování barvonosných složek dojde ke snížení šířky pásma potřebné pro přenos signálu, ale subjektivní kvalita obrazu zůstane zachována. SDL opět umožňuje použít několik druhů YUV formátů. Pro vyvíjenou

[4] funkce jsou uváděny dle prototypu z hlavičkových souborů FFmpeg.

[5] aditivní barevný model; pro vyzařovací zařízení; červená, zelená, modrá – red, green, blue

[6] barevný model; původně pro TV vysílání; tříprvkový vektor jasu a dvou barevných složek

[7] tento název je poněkud nepřesný, v oblasti digitalizovaného videa se užívá označení YCbCr

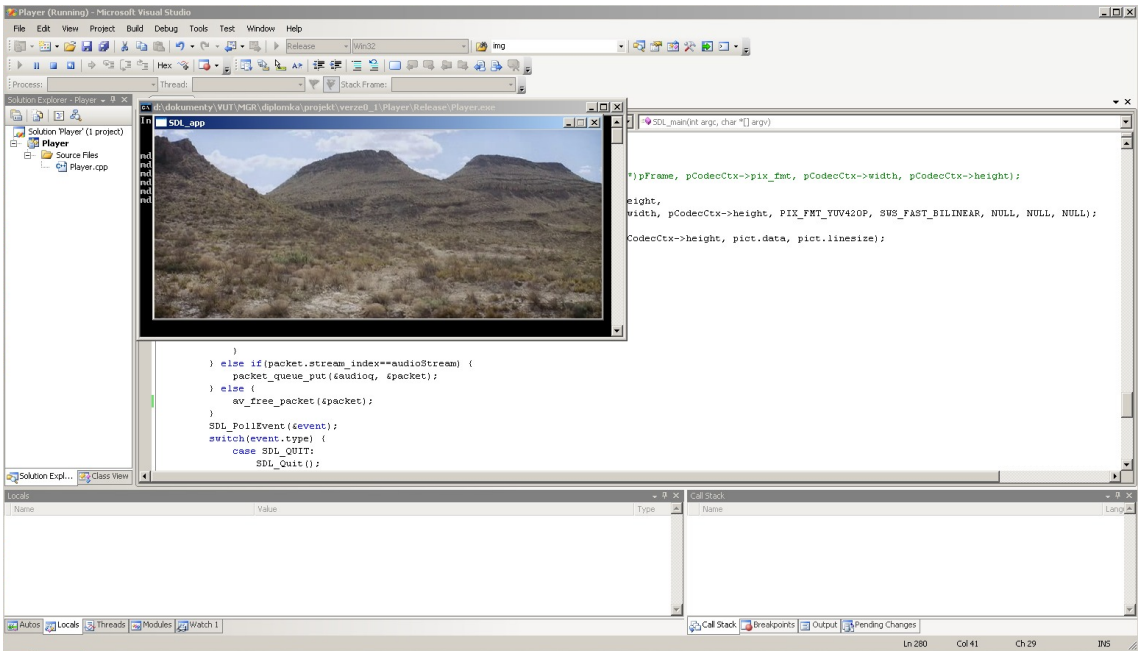


Obr. 2.2: Zobrazení videa.

aplikaci byl zvolen formát YUV420P. Číslo 420 značí, že podvzorkování složek signálu je provedeno v poměru 4:2:0, což prakticky znamená, že na čtyři prvky jasové složky připadá jeden prvek barevné složky. Písmeno P značí, že všechny tři složky jsou od sebe separovány. Zvolený formát zobrazení byl vybrán záměrně, protože většina obrazových toků je v tomto formátu ukládána nebo se do něj dají snadno převést (zajistí FFmpeg).[3][5][10][21][7]

Pro správné zobrazení videa bylo nutné do zdrojového kódu přidat posloupnost příkazů, jimž odpovídá vývojový diagram na obr. 2.2. Bloky s bílým pozadím vyznačují operace, které jsou totožné s vývojovým diagramem na obr. 2.1.„Pomocné plátno“ je vlastně paměť, kam se ukládá výstupní snímek ve formátu YUV420P. Ten je následně zobrazen na monitor.[5]

Obr. č. 2.2 ukazuje jaké kroky je třeba učinit pro zobrazení videa. Snímky jsou posílány na obrazovku tak rychle, jak to program zvládá, resp. jak rychle je možné je dekódovat a zobrazit. To znamená, že prozatím není zajištěn plynulý výstupní tok dat se správnou rychlostí. Na obrázku 2.3 je vidět vytvořená konzolová aplikace přehrávající nyní pouze obrazová data.



Obr. 2.3: Ukázka přehrávání videa.

2.1.3 Přehrávání audia

Digitální audio signál je složen z posloupnosti vzorků, přičemž každý reprezentuje určitou hodnotu audio signálu v daném čase. U digitalizovaného zvuku hovoříme o tzv. vzorkovací frekvenci (např. u CD 44 100 Hz), což je hodnota udávající počet vzorků získaných za sekundu. Důležitým parametrem je také počet audio kanálů. U stereo signálu jsou kanály dva, ovšem dnes je velmi častý i vyšší počet kanálů. V každém čase se pak musí zpracovat tolik přicházejících vzorků audio signálu, kolik je kanálů.[5][21]

Pro zpřístupnění audio dat byla opět využita knihovna SDL a princip zprostředkování zvukové stopy je vyřešen následujícím způsobem. Před začátkem přehrávání je třeba nastavit parametry audia (vzorkovací frekvence, počet kanálů, délka vyrovnávací paměti atd.). Po spuštění přehrávání zvuku je periodicky volána funkce zajišťující získání dat[8], která naplňuje definovanou vyrovnávací paměť audio daty. Jakmile jsou k dispozici data, zavoláme v programu další funkci[9], která zajistí obsluhu audio zařízení a pošle zvuk na výstup.[5][21][10][16]

Tento jednoduchý princip však skrývá dva zásadní problémy. Z datového toku videa jsou plynule načítány pakety s video daty, zároveň jsou ještě získávána audio data ze souboru, ale dochází také k pravidelnému volání funkce, která zvuk reprodu-

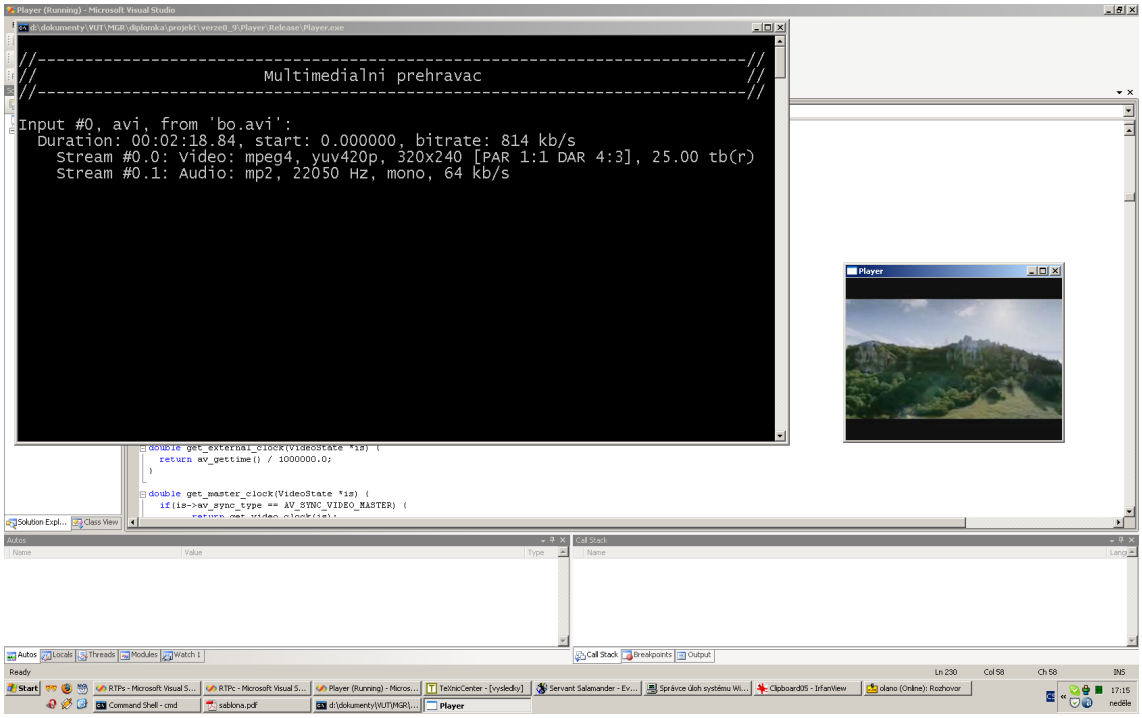
[8]při naplnění struktury informacemi o zvukové stopě je jedním z nastavovaných parametrů název funkce, která se bude periodicky volat

[9]zde už jde o způsob obsluhy vytvořený programátorem

kuje. Zde by mohly vznikat potíže se správným a včasným vysíláním audio a video dat na výstup (k uživateli). Řešením je vytvořit úložiště, do kterého budou načítány vzorky zvuku a později periodicky vyčítány a reprodukovány. V programovacích jazycích je ideálním řešením implementace ADT[10] fronta.[10]

Druhou překážku vytvořila samotná knihovna SDL, která zajišťuje přehrávání zvuku ve zvláštním vlákně. To na jednu stranu přináší vyšší výkon na architekturách s více procesory (jádry), ale na druhou stranu je třeba řešit další režijní záležitosti. Zejména je nutné správně určit kritické sekce (manipulace s frontami) a také vytvořit komunikaci mezi vlákny.[10][5][16]

V tomto bodě bylo dosaženo extrakce obrazu a zvuku ze souboru směrem k uživateli. Načítání a hlavně zobrazování video snímků probíhalo s přílišnou rychlostí. Získání vzorků audio signálu se sice také dělo s předstihem, ale byly ukládány do fronty. Zjištěním správné vzorkovací frekvence zvuku z datového toku se zajistilo volání funkce pro obsluhu zvukového vlákna ve správných časových intervalech. Zvuk pak byl reprodukován korektně. Obrázek 2.4 ukazuje současné přehrávání audia a videa. V konzoli je vidět, že jsou data načítána ze dvou datových proudů.

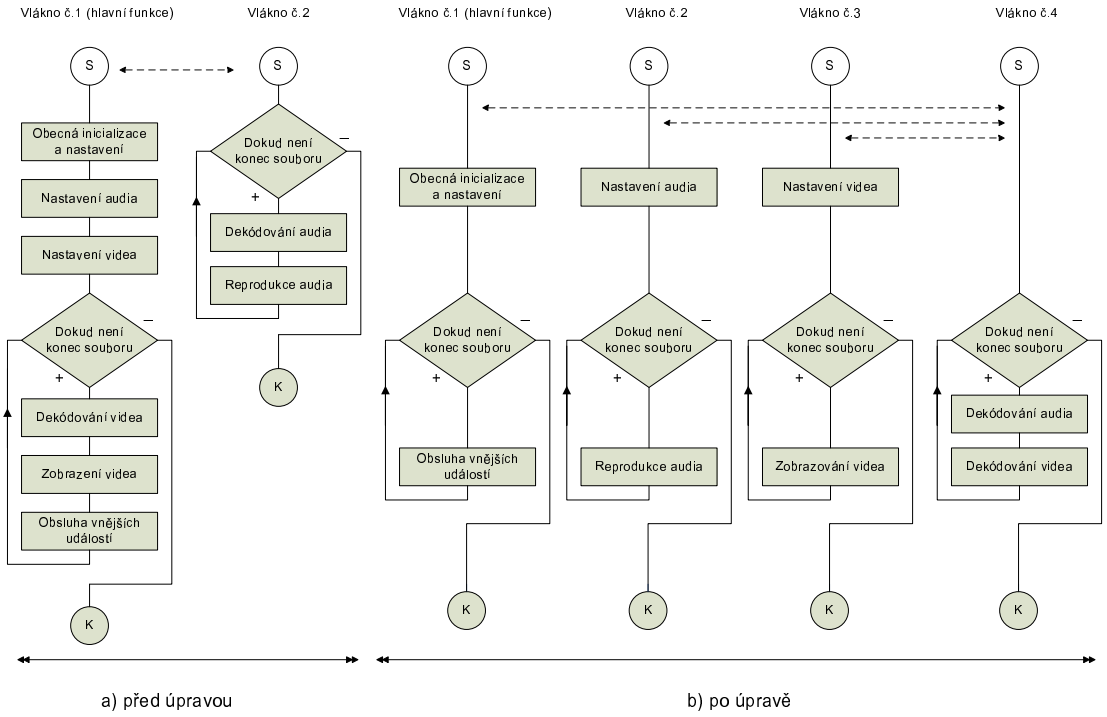


Obr. 2.4: Ukázka přehrávání obrazu i zvuku.

[10]abstraktní datový typ – Abstract Data Type

2.1.4 Vlákna

Kdyby byl vývoj aplikace zastaven v tomto bodě, existoval by již funkční přehrávač multimediálních souborů. Postačilo by jen vložit do hlavní smyčky programu vhodné zpoždění, aby video snímky byly posílány na obrazovku odpovídající rychlostí. Přehrávač vytvořený tímto způsobem by však byl málo odolný vůči špatně vytvořeným souborům a navíc struktura programu by byla značně nečitelná. Již zde obsahuje zdrojový kód stovky řádků. Bez dodržení jistých pravidel by se celý projekt stal nejprve nepřehledný a později neschopný dalšího rozvoje. Řešení této situace nabízí implementace vícevláknového přístupu. Jak již bylo řečeno v teoretické části práce, knihovna SDL podporuje dělení programu do vláken a v předchozí části vývoje už bylo této možnosti využito pro oddělení zobrazování videa a reprodukce audia. Technika vláken nejen zpřehlední zdrojový kód, ale pomůže mu také stát se modulárnějším. V důsledku tak bude usnadněna implementace synchronizace (viz 2.1.5). Obrázek 2.5 přibližuje, jaké změny nastanou ve struktuře programu po použití vláknového přístupu.[5][10][16]

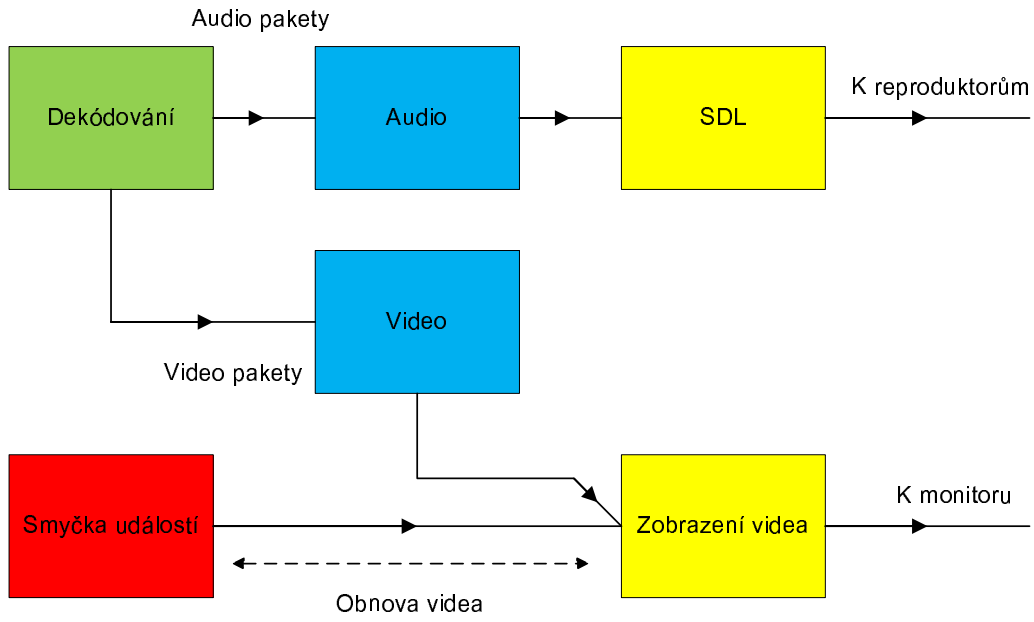


Obr. 2.5: Rozdíl ve struktuře programu při využití vláknového přístupu.

Prvním krokem pro úspěšnou implementaci vláken je sdružení všech uživatelských dat na jedno místo, resp. do jedné struktury. Ta pak bude poskytnuta jako parametr funkcím. Tak je zajištěno, že všichni mají dostupné to, co právě potřebují pro svoji činnost. Dále je vhodné ponechat co nejjednodušší hlavní funkci. V tomto

případě jsou do hlavní funkce včleněny jen inicializační záležitosti a smyčka obsluhující frontu SDL událostí. V programu již existuje fronta pro audio vzorky a vlákno pro obsluhu této fronty. Podobným způsobem lze vytvořit i vlákno obsluhující jinou frontu, která nyní uchovává video snímky. Pro naprosté oddělení získání audia a videa od jeho zpracování je nutné vytvořit ještě jedno vlákno, odpovídající za načítání multimediálních informací ze souboru a jejich ukládání do patřičných front.[5][16]

Jako poslední krok je nutná úprava té části kódu, jenž provádí zobrazování snímku. Tato úprava je prováděna naprosto cíleně, protože zajistí, že finální synchronizace audia a videa pak bude relativně snadnou úpravou zdrojového kódu. Celá záležitost spočívá v tom, že dochází vlastně k zobrazování snímku pokaždé vždy, když proběhne smyčka v hlavní funkci. Idea je taková, že dojde k dekódování snímku a jeho uložení na jiné místo. Dále je vytvořena uživatelská událost, kterou obsluhuje SDL. Při vyvolání této události se pak zobrazovaný snímek vymění za snímek následující (princip viz obr. 2.6). Tímto způsobem lze pohodlně kontrolovat, kdy a jak dlouho má být zobrazován určitý video snímek.[5][10][16]



Obr. 2.6: Detailnější popis upravené struktury programu.[5]

2.1.5 Synchronizace

Nyní již přehrávač zobrazuje video snímky a přehrává audio stopu. Zvuková, ale i obrazová data obsahují informaci o tom v jakém časovém intervalu vyslat další vzorek popř. snímek. U zvuku je podstatná vzorkovací frekvence a u videa tzv. fps[11].

[11] počet snímků za sekundu – frames per second

Jak již bylo řečeno, audio je reprodukováno správně, ale obraz výrazně předbíhá zvuk. Nyní je tedy nutné zajistit, aby i video bylo prezentováno korektně, což se neobejde bez úprav ve zdrojovém kódu.[5]

Nejjednodušší variantou jak zajistit správnou prodlevu mezi obrazovými snímky je prosté využití fps, což vlastně spočívá v počítání jednotlivých snímků a násobení fps. V případě, že vše bude pracovat bezchybně, lze tímto způsobem zajistit, že video snímky budou zobrazovány se správnou rychlostí a vzájemný posun mezi tokem audia a videa bude nulový. Bohužel počítače nejsou bezchybné a mnoho multimediálních souborů také ne. Při využití této metody se tedy může stát, že dojde k vzájemnému „rozjetí“ obrazu a zvuku. K zajištění správné prezentace obsahu multimediálního souboru je tedy vhodnější využít jinou techniku – vzájemnou synchronizaci datových toků. V podstatě je možné rozlišit tři druhy:[5]

- obraz ke zvuku,
- zvuk k obrazu,
- obraz i zvuk k externímu zdroji.

Synchronizace obrazu ke zvuku

Jako nejvýhodnější ze tří výše uvedených variant se jeví synchronizace obrazu ke zvuku a to především z těchto důvodů:

1. Zvuk je již přehráván správně, protože struktura užívaná při obsluze zvukové stopy vyžaduje při inicializaci vzorkovací frekvenci. Bez ní by reprodukce audia nefungovala. To znamená, že nebude třeba tak výrazná úprava zdrojového kódu při zvolení této varianty synchronizace.
2. Tento typ synchronizace je pro člověka subjektivně příznivější. Absence jednoho nebo několika málo obrazových snímků není mnohdy lidským vnímáním vůbec postřehnutelná nebo nepůsobí tak rušivě jako výpadek byť jen jediného audio vzorku.[12][5]

K implementaci synchronizace je možné využít DTS[13] a PTS[14], což jsou hodnoty, které obsahuje každý video paket načtený z datového toku. Název zmíněných dvou parametrů dostatečně vypovídá o jejich funkci, ale může vzniknout otázka, proč je třeba mít parametry dva. U některých formátů videa (např. MPEG) jsou totiž jednotlivé snímky v datovém proudu uloženy jinak, než jsou pak zobrazovány

[12] u zachytávání videa a audia v reálném čase se také doporučuje tento typ synchronizace
[13] dekódovací časová značka – Decoding Time Stamp
[14] prezentační časová značka – Presentation Time Stamp

při přehrávání. Tento fakt souvisí s tím, že pro dosažení redukce datového toku u videa se některé snímky vytváří pouze rozdílem od snímku předcházejícího, popř. od předcházejícího a následujícího.[5][22]

Načtením paketu z datového proudu jsou tedy získány hodnoty jeho DTS a PTS. Ve skutečnosti je požadováno PTS dekódovaného snímku, aby bylo možné jej ve správnou dobu zobrazit. Problémem je to, že po dekódování rámce získáme strukturu, která nemusí obsahovat použitelnou hodnotu PTS.[15] FFmpeg sice dokáže přeorganizovat pakety tak, že DTS paketu zpracovávaného dekódovací funkcí bude stejné jako PTS snímku, který funkce vrací, ale tyto informace nemusí být k dispozici vždy. Je tedy nutné zajistit i jinou metodu získání PTS ze snímku.[5][22]

Jako nejvýhodnější a nejjednodušší se jeví uložení PTS prvního paketu z rámce. Získáme tak vlastně PTS celého snímku, takže v případě, že nelze získat z datového proudu DTS, stačí využít uloženou hodnotu PTS. Který paket je první, můžeme zjistit s pomocí dekódovací funkce, protože kdykoliv paket obsahuje začátek nového snímku, dekódovací funkce alokuje část paměti snímku. Knihovny FFmpeg umožňují předefinování funkcí, které jsou v nich implementovány. Protože alokační funkce je součástí FFmpegu, postačí ji modifikovat a ukládat žádané PTS paketu.[5][22]

Zjednodušeně lze říci, že PTS dekódovaného snímku je rovno DTS posledního zpracovaného paketu. Pokud DTS tohoto paketu není vhodné, použije se PTS prvního paketu ve snímku.

Nyní, když je možné získat správnou hodnotu PTS, může být nastíněn princip, na kterém byla vytvořena synchronizace datových toků audia a videa. Po zobrazení aktuálního snímku je zjištěno, kdy, resp. za jak dlouho, má být použit snímek další a na základě toho je nastaven nový časový interval, po jehož uplynutí se obraz aktualizuje. Zároveň také dochází k testování hodnoty PTS z následujícího snímku se systémovými hodinami, aby byl nastaven správný časový interval. Tento princip začal fungovat po vyřešení dvou otázek:

1. Je nutné zjistit, kdy bude nová hodnota PTS. Přidáním časového intervalu na bázi fps k aktuálnímu PTS by byl tento problém vyřešen, ale jen částečně. Důvodem je to, že v některých videích mohou být stejné snímky opakovány několikrát. To by znamenalo, že další změněný snímek by byl prezentován příliš brzy. Strategie je tedy taková, že bude predikována hodnota, kdy má dojít k zobrazení dalšího snímku a předpověď bude provedena na základě časového intervalu mezi aktuálním a předchozím snímkem. Pokud se hodnota z nějakého důvodu změní, bude patřičně upravena.

[15] obsahuje sice proměnnou na uložení PTS, ale ta ne vždy obsahuje to, co je třeba, když získáme snímek

2. Musí být provedena synchronizace obou datových toků (viz výše). K tomuto účelu je nutná implementace tzv. audio hodin, které uchovávají aktuální hodnotu pozice v toku zvukových dat. Tato hodnota pak bude sloužit k porovnání, zda se obraz předbíhá či zpožďuje vzhledem ke zvuku. Pokud se obraz zpožďuje, jsou nové snímky vykreslovány s maximální rychlostí. V případě opačném jsou snímky zobrazovány s dvojnásobným zpožděním.[5]

Po implementaci výše uvedených kroků do zdrojového kódu je přehrávač schopen korektně zobrazovat video a přehrávat audio z multimediálních ogg souborů. Dále se aplikace dokáže vypořádat se špatnou synchronizací mezi jednotlivými datovými toky, což bylo cílem této kapitoly.

Synchronizace zvuku k obrazu

Jde o druhý ze tří možných typů synchronizace. Nyní je již možné korektně přehrávat videoklipy se zvukem popř. samostatnou zvukovou stopu, avšak bez dalších úprav si program neporadí s videem bez zvuku, protože je vyžadován hlavní datový proud, který je nastaven na audio. Cílem je tedy provést změny v programu tak, aby bylo možné přehrávat i samotné video soubory, resp. aby bylo možné zvolit, který datový proud bude hlavním.[16]

Pro implementaci synchronizace zvuku byl zvolen podobný postup jako v případě dorovnávání obrazu. Nejprve bylo nutné vytvořit tzv. video hodiny, jejichž úkolem je sledování toho, jak „daleko“ se nachází videoklip od svého počátku a jestli získaná hodnota souhlasí s hodnotou zvukové stopy. To znamená, že při přehrávání videa je nutné získat a neustále udržovat aktuální hodnotu časového ofsetu od začátku videoklipu. Jako nejjednodušší možnost řešení by se mohla jevit aktualizace s časovačem zobrazovaných snímků, tj. se současnou hodnotou PTS posledního zobrazeného snímku. Zde je však nutné vzít v úvahu, že časová prodleva mezi jednotlivými zobrazenými snímky je na úrovni milisekund relativně dlouhým intervalem. Z tohoto důvodu je vhodnější sledovat jiné hodnoty. Jsou jimi vlastně časy, ve kterých dochází k nastavování video hodin na PTS posledního zobrazeného snímku. Aktuální hodnotu video hodin lze pak získat dle jednoduchého vzorce (2.1):[5]

$$t_{\text{PTS\_posledniho\_snimku}} + (t_{\text{aktualni}} - t_{\text{nastaveni\_hodnoty\_PTS}}). \tag{2.1}$$

Samotná strategie synchronizace je velmi jednoduchá. Je měřeno, kde se nachází zvuková stopa a tato hodnota je porovnávána s hodnotou získanou z video hodin. Po jejich vzájemném porovnání se vyhodnotí, jaké množství audio vzorků je nutné vypustit nebo přidat, aby bylo audio reprodukováno synchronně k video snímkům.[5]

[16] při přenosu multimediálních dat po síti se volí jako hlavní datový proud video, zvuk je považován za doprovodnou informaci

Kromě principu synchronizace je nutné také správně určit, kdy a za jakých podmínek se bude spouštět synchronizační funkce. Nejvhodnějším řešením je volat synchronizaci vždy při zpracování vzorků audio signálu. Operace se vzorky audia je ale prováděna mnohem častěji, než v případě vzorků video signálu. Je tedy vhodné stanovit určité zpoždění, resp. podmínku, která zaručí, že nebude prováděn synchronizační algoritmus při každém volání synchronizační funkce, ale pouze jednou za určitý počet volání.[5]

Algoritmus synchronizace tedy předpokládá, že na jeho vstupu je určitý definovaný počet audio vzorků, které jsou „mimo synchronizaci“, což prakticky znamená, že posun od správné hodnoty je větší, než stanovený práh tj. vzájemný posun audio a video stopy se stává postřehnutelným pro uživatele. Protože každý vzorek může být teoreticky posunut o zcela jinou hodnotu, je nutné vypočítat hodnotu zpoždění[17] průměrováním. Obyčejný aritmetický průměr však nepostačuje, neboť hodnota zpoždění posledních příchozích vzorků je důležitější, než hodnota rozdílu patřící prvním vzorkům ve zpracovávaném bloku dat. Je tedy nutné zavést koeficient, kterým budou časové posuny násobeny a který zvýhodní zpoždění posledních vzorků. Výslednou hodnotu sumy posuvu (rozdílu) mezi datovými toky popisuje vzorec (2.2):[5]

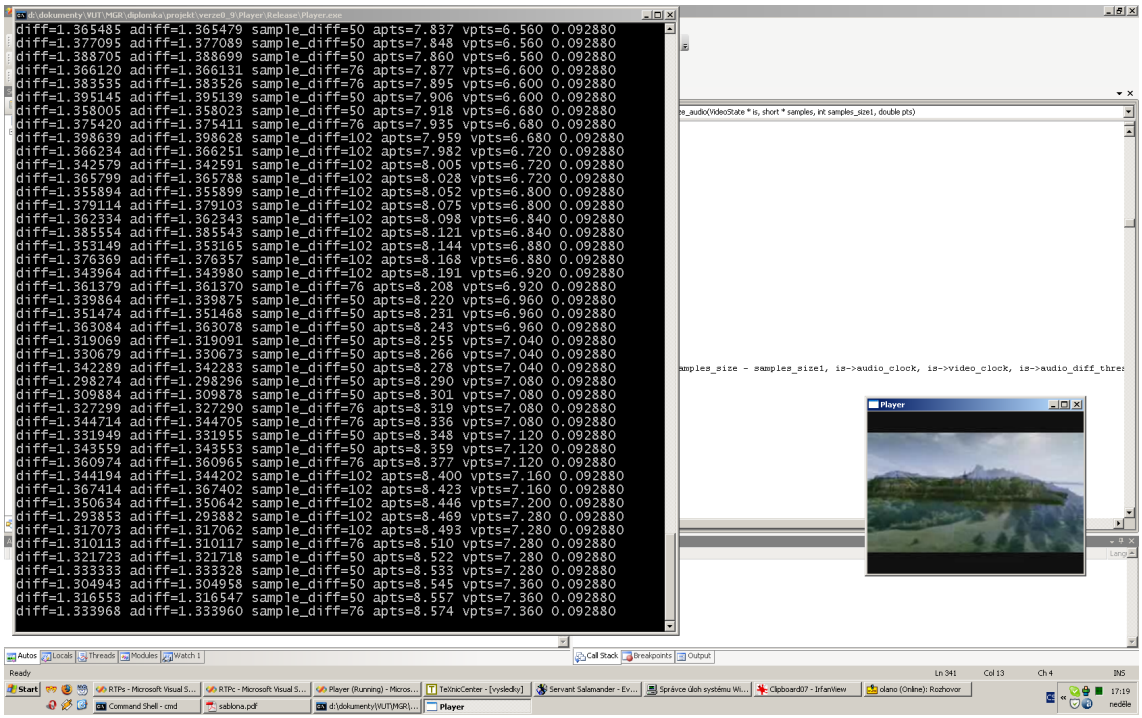
$$suma\_posunu_t = novy\_posun + suma\_posunu_{t-1} * koeficient. \tag{2.2}$$

Použitím koeficientu je získán místo aritmetického vážený průměr. Jakmile jsou splněny podmínky, lze jeho hodnotu spočítat dle vzorce (2.3):

$$prumerny\_posun = suma\_posunu * (1 - koeficient). \tag{2.3}$$

Použití průměrování nemá vliv na kvalitu synchronizace, protože jak již bylo řečeno, vzájemný posun audia a videa musí překročit určitý práh, aby byl pozorovatelný. V reálné situaci je navíc zpoždění po sobě jdoucích vzorků stejné, nebo alespoň velmi podobné, tj. nepatrné odchylky mezi sousedními vzorky uživatel nezaznamená.

Dalším krokem v synchronizačním algoritmu je úprava vzorků audio signálu. Ve zdrojovém kódu vytvořená funkce vrací vlastně novou hodnotu počtu vzorků audio signálu a úprava je provedena přidáním nebo odebráním vzorků. Z toho plyne, že je nutné také vypočítat jaké množství vzorků bude přidáno či odebráno. Dále je nutné určit limit, do jaké míry bude úprava provedena, aby zůstala zachována informační stránka datového toku a aby synchronizace nebyla příliš nepříjemná pro uživatele. Úprava zvuku je totiž pro člověka subjektivně rušivější než úprava videa. Samotná změna počtu vzorků je relativně snadná, ale efektivní. Pokud vzorky přebývají, jsou



Obr. 2.7: Ukázka synchronizace zvuku.

odstraněny. Je-li je vzorků nedostatek, jsou na potřebná místa kopírovány vzorky předcházející.[5]

Po vytvoření popsaného algoritmu a jeho následném testování docházelo k drobné chybě. I při sesynchronizování se zvuk neustále nepatrně zpožďoval za obrazem. Při opětovném prozkoumání funkce bylo odhaleno chybné nastavení meze *min_size* a *max_size*. Algoritmus tak v podstatě nemohl nikdy dorovnat zvuk úplně správně, vždy zůstal o něco pozadu. Po matematické úpravě výpočtu mezí vytvořená funkce pracovala správně.

Po implementaci výše popsané synchronizační funkce je možné přehrávat multimediální soubory obsahující audio i video stopu nebo pouze audio, resp. video. Před reprodukcí je třeba rozlišit hlavní datový tok dle konkrétního obsahu souboru (podrobněji viz 2.1.6). Nezbytným krokem je také ve zdrojovém kódu jasně definovat, které příkazy se budou provádět, když bude zvolen určitý datový proud jako hlavní. Při nedostatečném ošetření těchto podmínek by mohlo dojít k situaci, kdy se budou oba datové proudy, audio i video, synchronizovat navzájem a nemuselo by tak docházet ke korektní reprodukci obsahu souboru. Na obr. 2.7 je příklad synchronizace zvuku. V konzoli je vidět výpis synchronizačních informací. První položka ukazuje vzájemný posun audia a videa v sekundách. Je patrné, že zmíněný posun klesá.

[17] zde poněkud nepřesné, protože audio tok se může zpožďovat i předbíhat vzhledem k druhému proudu dat

2.1.6 Uživatelské rozhraní

Až doposud byly vytvářeny prvky přehrávače, které sice byly stěžejní pro správnou reprezentaci multimediálních souborů, ale uživatel je nemohl nijak ovlivnit z vnějšku. Pro ovládání programu musí existovat uživatelské rozhraní, jehož tvorbou se bude zabývat tato podkapitola. Jistým náznakem potřeby uživatelské interakce již byla v předchozí kapitole možnost volby hlavního datového toku. Protože požadavkem bylo, aby přehrávač pracoval v příkazové řádce, nebude uživatelské rozhraní programu grafické.

Dříve než bude popsáno, jaké možnosti ovládání poskytuje vytvořený program, je nutné uvést, jakým způsobem je realizováno zpracování uživatelských vstupů. Pro příjem a obsluhu vnějších událostí je částečně využita knihovna SDL. V hlavní funkci programu je realizována smyčka, k jejímuž opuštění dojde pouze při požadavku k ukončení programu. V této smyčce je umístěna funkce z SDL. Tato funkce čeká, dokud nedojde k události a její kód poskytne programu pomocí proměnné. Na základě předané hodnoty jsou pak učiněny posloupnosti příkazů, realizující požadovanou funkci např.:[5][10][16]

- posun v čase přehrávaného klipu,
- ukončení programu,
- režim celé obrazovky,
- pauza v přehrávání.

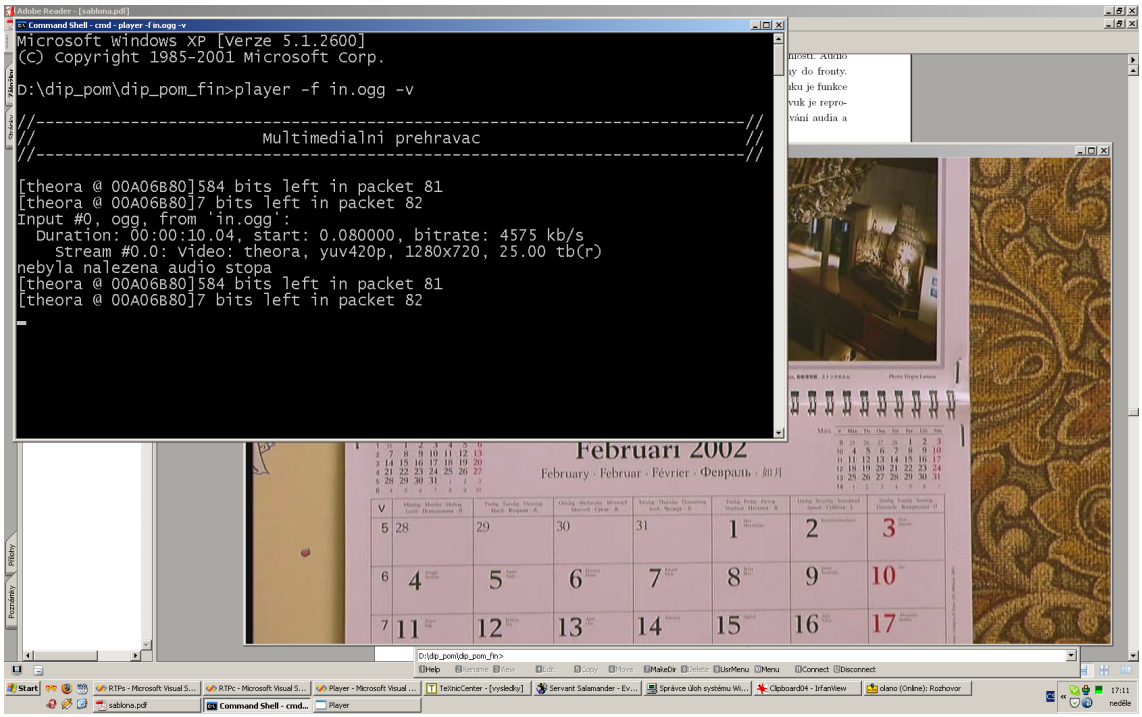
Vstupní parametry

Ačkoliv se nejedná přímo o uživatelské rozhraní, jsou vstupní parametry zmíněny v této kapitole, protože pomocí nich lze definovat chování programu ještě před samotnou reprodukcí multimediálních dat ze souborů. Přehrávači lze předat z příkazové řádky maximálně dva parametry, přičemž ani jeden z nich není povinný a jejich pořadí lze zaměňovat. Možné parametry jsou tyto:

1. -f [*název_souboru*]
   specifikuje vstupní soubor, jehož obsah bude přehrávač reprodukovat. Není-li uveden, otevře se soubor s názvem out.ogg popř. out.ogv (viz 2.3.4).
2. -v
   definuje, že řídícím tokem bude video. Při vynechání tohoto parametru je nastavena jako hlavní datový proud zvuková stopa.

Výše zmíněné parametry jsou předávány hlavní funkci při spuštění programu a zpracovány obslužným podprogramem. Jeho úkolem je procházet řetězec znaků následující za názvem spouštěného programu a části tohoto získaného řetězce jsou porovnávány popř. ukládány do proměnných, s nimiž se dále pracuje.[20]

Jelikož je přehrávač určen primárně pro reprezentaci multimédií, která jsou přijímána ze sítě, jsou stanoveny soubory, jejichž obsah bude zprostředkován uživateli. Tyto předdefinované soubory jsou vytvořeny RTP klienty. Podrobněji je o této problematice pojednáno v části zabývající se RTP komunikací 2.3.3. Praktická ukázka použití vstupních parametrů je na obr. 2.8. V konzoli lze opět nalézt příklad spuštění multimediálního souboru s tím, že hlavním datovým tokem je video.



Obr. 2.8: Ukázka zadávání vstupních parametrů pro přehrávač.

Přetáčení vpřed a vzad

Jedná se o jednu ze základních funkcí, kterou by měl být vybaven každý přehrávač multimédií, a proto vytvořený program nebude výjimkou. Poněvadž je vyvíjená aplikace určena pro práci v příkazové řádce, jsou zde poněkud omezené možnosti, jak realizovat posun po časové ose reprodukovaného obsahu. Přetáčení je tak uskutečněno pomocí tlačítek na klávesnici počítače, konkrétně jde o šipky, přičemž jsou využity všechny čtyři klávesy. Je možný přeskok o dva časové úseky, 10 a 60 sekund, a to vpřed (šipky nahoru a vpravo) i vzad (dolů a vlevo).

Programová realizace posuvů se pak opírá o obě z využitých knihoven. SDL disponuje funkcí, která dokáže po stisku klávesy vrátit odpovídající řetězec. Této vlastnosti je využito v hlavním programu, ve smyčce monitorující uživatelské události. Stlačením některé z šipek dojde k provedení odpovídající části zdrojového kódu a následně pak k posuvu. Dále pak jedna z funkcí knihovny FFmpeg umí vykonat část příkazů nutných k implementaci přetáčení a šetří tak programátorovi práci. I přes nasazení obou knihoven je nutné pro provádění korektních posuvů doplnit do zdrojového kódu nezanedbatelné množství příkazů a funkcí, jejichž význam bude popsán v následujícím textu.[5][10][16]

Po stisku klávesy je nejprve nutné rozlišit, o kterou z šipek se jedná. Následně je třeba získat hodnotu odpovídající aktuálnímu času, ve kterém se přehrávaný klip nachází při stisku klávesy, přičemž je nutné vždy pracovat s takovým datovým tokem, jenž byl zvolen jako hlavní. Jakmile je k dispozici požadovaná hodnota, přičte se k ní velikost posuvu a dojde k získání cílového času, kam se bude posouvat. Hodnoty multimediálních datových proudů však nejsou měřeny v sekundách, ale v počtu snímků. Z toho plyne, že je nutné cílovou hodnotu času správně přepočítat a následně předat funkci z FFmpegu, která zajistí přesun na správné místo ve všech aktuálně přehrávaných datových tocích.[5]

Po implementaci uvedených kroků je možné se přesunout na požadované místo v prezentovaných klipech, ale pro správnou funkci zbývá doplnit ještě jednu část. Jelikož je přehrávání video snímků a vzorků audio signálu relativně pomalou činností vzhledem k možnostem procesoru a vůbec celého počítače, jsou v programu části dekódování a přehrávání obsahu souboru odděleny do zvláštních vláken. Jak již bylo zmíněno dříve, tyto vlákna ukládají a načítají informace z front. Z uvedených faktů plyne, že aktuálně prezentovaný snímek či vzorek nemusí a zpravidla neodpovídá poslednímu záznamu ve frontách tj. dekódování je v předstihu před přehráváním. Z tohoto důvodu je nutné při přetáčení také vyprázdnit naplněné fronty, protože při přeskoku již nemusí obsahovat relevantní informace. Po zahození nepotřebných dat jsou fronty znovu naplňovány správnými informacemi.[5]

Při praktickém testování převíjení byl odhalen problém. Funkce přeskoku sice pracovala, ale při zadání požadavku z klávesnice došlo nejprve k časové prodlevě několika sekund. Teprve pak se odehrál vlastní přesun přehrávání na jiné místo prezentovaného klipu. Později byl problém vyřešen. Chybné porovnávání návratové hodnoty funkce pomocí logického operátoru „nerovná se“ (`!=0`) bylo nahrazeno porovnáním pomocí operátoru „menší“ (`<0`). Jednalo se o tuto funkci:

`int av_seek_frame(AVFormatContext *s, int stream_index, int64_t timestamp, int flags)`.

Po úpravě zdrojového kódu již posouvání fungovalo správně.

Pozastavení přehrávání

Pozastavení přehrávání je vedle převíjení pravděpodobně druhou nejužívanější funkcí v multimediálních přehrávačích a pracuje na následujícím principu. Podstatou je, aby bylo možné pozastavení prezentovaného klipu na libovolně dlouhou dobu a následně opětovné spuštění, přičemž aplikace je stále aktivní (pracuje) a zůstává zobrazen snímek, při jehož prezentaci byl zadán požadavek pro pozastavení přehrávání. Ponechání zobrazeného posledního snímku nepůsobí nijak rušivě, naopak podává uživateli informaci, že přehrávání nebylo ukončeno, nýbrž pouze pozastaveno. V případě audia je tomu jinak. Při pozastavení je vhodné, místo vysílání posledního zvukového vzorku, na výstup pustit „ticho“, protože pokud by se z reproduktorů linul po delší dobu neustále stejný zvuk, mohlo by to působit velice nepříjemně.

Implementace ve zdrojovém kódu je provedena následovně. Pozastavení i opětovné spuštění přehrávání lze provést pomocí stisku písmene „p“ na klávesnici PC. Musí existovat proměnná, v níž bude hodnota, která signalizuje, zda je nebo není vyžadováno pozastavení běhu programu. Zmíněná proměnná musí být viditelná v rámci celého kódu, jelikož bude ovlivňovat hned několik funkcí. Jak bylo již uvedeno, při pozastavení bude zobrazen poslední prezentovaný snímek obrazu, z čehož plyne, že je nutné pozastavit běh vlákna dekódujícího snímky z fronty snímků. Obdobně je tomu u zvuku, zde ovšem nelze vlákno pozastavit, jelikož není vytvářeno programátorem, ale přímo knihovnou SDL. Řešením je ukončit ve vlákně volanou funkci, což zároveň zajistí, aby na výstupu nebyl žádný zvuk po celou dobu pozastavení běhu programu. Vzhledem k tomu, že během pozastavení přehrávání nejsou z vytvořených front odebírány žádné prvky, není nutné dekódovat nové snímky a vzorky ze vstupního souboru. Dekódovací vlákno je také pozastaveno, resp. nedochází ke čtení nových informací ze souboru[18].[2]

Režim celé obrazovky

Pro komfortní sledování videoklipů je vhodné doplnit přehrávač o tento režim. Realizace je taková, že stiskem tlačítka „f“ na klávesnici PC je možné režim celé obrazovky zapnout i vypnout. Pro implementaci bylo využito vlastností knihovny SDL. Konkrétně se jedná o funkci nastavující videorežim, ve kterém budou zobrazovány snímky. Jedním z parametrů uvedené funkce je i přepínač, pomocí něhož lze vytvořit celoobrazovkový režim. Názornou ukázku celoobrazovkového režimu je možné vidět na obr. 2.9.[10][16]

Změnu rozměrů okna, v němž jsou zobrazovány video snímky, lze realizovat i jinou cestou, než je popsáno výše. V případě neaktivního režimu celé obrazovky je

[18]zde je využito funkcí z knihovny FFmpeg

Obr. 2.9: Ukázka režimu celé obrazovky.

možné požadovaného efektu docílit pomocí počítačové myši a to tak, že uživatel přesune kurzor na hranu video okna, stiskne levé tlačítko myši a tahem, při stisknutém tlačítku, zvětší okno na požadovanou velikost. Implementace ve zdrojovém kódu je provedena shodnou funkcí jako v případě režimu celé obrazovky.[10][16]

Ukončení programu

Zde jsou rovněž dvě možnosti, jak program uzavřít:

1. stiskem tlačítka „esc“ na klávesnici PC,
2. kliknutí pravým tlačítkem počítačové myši na křížek v pravém horním rohu okna, kde jsou zobrazovány video snímky.[19]

V obou případech je využito vlastností knihovny SDL.[10][16]

[19] standardní ukončování oken v OS Windows

Implementací všech doposud uvedených kroků byl vytvořen konzolový přehrávač umožňující prezentovat uživateli zvuk a obraz z multimediálních souborů. Program disponuje základními uživatelskými funkcemi (režim celé obrazovky, převíjení, pauza) a je schopen se vyrovnat se špatnou synchronizací datových toků.

2.2 RTP komunikace

Implementace síťové komunikace se jeví jako klíčový bod ve vývoji programu. Její zprovoznění umožní přenášet téměř jakákoliv data a bude tím zajištěna distribuce multimediálního obsahu i na jiné stroje připojené do počítačové sítě. Pro přenos dat byl zvolen RTP protokol, který poskytuje požadované funkce (viz 1.1.1) a zároveň jeho implementace není příliš složitá. Je požadován přenos pouze RTP paketů bez zpětné vazby či korekce QoS.

Jak se ukázalo v průběhu řešení práce, vytvoření RTP komunikace je nejen klíčovým, ale také velmi problémovým bodem zadání. V následujícím textu bude popsáno, jaké postupy byly zvoleny při tvorbě RTP komunikace a jakých výsledků bylo dosaženo.

2.2.1 FFmpeg

V kapitolách o tvorbě přehrávače již bylo popsáno, že FFmpeg provádí načítání informací z datových toků po paketech a další manipulace a zpracování se odehrává s celými snímky. Po prostudování dokumentace k projektu FFmpeg bylo zjištěno, že v těchto knihovnách již existuje řešení zabývající se síťovou komunikací. Jeho využití by bylo velmi výhodné, protože by docházelo ke zpracování a odesílání dat do sítě opět po celých snímcích. Navíc by byla výrazným způsobem usnadněna tvorba RTP paketu, jelikož by všechny informace o kodérech a typu zátěže již byly připraveny z přehrávače a spolupráce mezi přehráváním a tvorbou paketů by tak byla velmi elegantní. Zde je nutné podotknout, že tak jako je třeba pro přehrávání různých formátů (souborů, kontejnerů) použít různé dekodéry, tak je také nutné jiným způsobem sestavovat RTP paket, resp. hledat potřebné informace o datových tocích na jiných místech. Také již bylo dříve uvedeno, že problematika spojená s kodeky je nepříjemná, protože tvorba zdrojových kódů s tímto zaměřením je velmi náročná a zdlouhavá a v podstatě by neumožnila rozvoj daného tématu kupředu. Ve světle uvedených faktů se tedy jeví využití knihovny FFmpeg i pro síťové vysílání jako ideální řešení. Vyžadovalo by sice dotvořit mnoho dalších řádků zdrojového kódu, ale spolupráce mezi přehráváním a síťovou komunikací by byla „čistá“, přehledná a RTP komunikace by splňovala všechny zadané požadavky.[2]

Po prostudování dostupných materiálů se ukázalo, že v daném případě FFmpeg pravděpodobně nelze využít pro síťovou komunikaci, na čemž se podílí především tyto faktory:

1. Ve statických knihovnách využitých pro přehrávač nelze zpřístupnit funkce, které se starají o RTP komunikaci.

2. Nefunkčnost knihoven vedla na myšlenku využít místo nich modifikované zdrojové kódy projektu, které jsou volně dostupné. V tomto případě se však objevil problém s MVS. V tomto vývojovém prostředí nelze zkompilovat zdrojové kódy projektu FFmpeg. Bylo by nutné využít jiný překladač a to MinGW (kolekce hlavičkových souborů, knihoven a GNU kompilátoru pro OS Windows – Minimalist GNU for Windows).

3. Po zprovoznění MinGW byl zájem soustředěn na zdrojové kódy FFmpeg. Vzhledem k tomu, že prakticky neexistuje žádná dokumentace, jak pracovat se sítí s využitím FFmpeg, bylo cílem prostudovat popř. použít část zdrojových kódů k realizaci RTP vysílání. Bylo však zjištěno, že tento úkol vysoce překračuje především časové možnosti diplomové práce, protože se jedná o nastudování několika desítek vzájemně provázaných zdrojových souborů, přičemž každý obsahuje řádově stovky řádků zdrojového kódu v jazyce C.[2][14]

Z výše uvedených důvodů byla tato varianta řešení RTP vysílání zavržena.

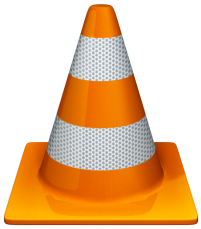
2.2.2 GStreamer a VideoLAN

Protože FFmpeg je svobodný software, který je navíc kvalitní, stabilní a jsou volně dostupné i jeho zdrojové kódy, je velmi často využíván pro multimediální aplikace vývojářů třetích stran. Jelikož pokus využít samotný FFmpeg k realizaci RTP vysílání skončil v podstatě neúspěchem, vznikla zde myšlenka pokusit se prostudovat projekty odvozené, resp. využívající knihovnu FFmpeg, zda by nebylo možné použít některých jejich částí. Jako nejslibnější se jevily projekty GStreamer a VideoLAN.[2]

VideoLAN

Je projekt, jehož výstupem je populární program s názvem VLC media player. Tento přehrávač je velmi kvalitním a silným nástrojem, který umožňuje provádět nejrůznější operace s multimediálními daty. Navíc se jedná o svobodný multiplatformní software a k dispozici jsou i jeho zdrojové kódy. Mimo jiné tento program umí i vysílat multimediální data do sítě.[19]

Úkolem bylo tedy zjistit, zda by nemohly být zdrojové kódy tohoto programu nějakým způsobem nápomocny při tvorbě RTP komunikace. VLC media player však



Obr. 2.10: Logo VLC media playeru.[19]

trpí podobnými nedostatky jako FFmpeg. Jeho zdrojové kódy jsou příliš rozsáhlé a navíc je nelze zkompilovat pod MVS, proto byla tato varianta řešení také zavržena.

GStreamer

Jedná se o knihovnu uvolněnou pod licencí GPL, jenž může být využita při tvorbě projektů pracujících s multimédii. Spektrum jejího nasazení je velmi široké. Najde své uplatnění při přehrávání, vysílání po síti, stříhání a míchání i při dalších operacích s multimediálními daty. Tato knihovna může být i rozšiřována pomocí jednoduchých zásuvných modulů, které mají definované rozhraní. Velmi podstatné je však to, že jsou k dispozici zdrojové kódy pro vývojáře, které lze zkompilovat v prostředí MVS. Projekt GStreamer se jevil jako velmi dobré řešení, které by mohlo napomoci při tvorbě RTP vysílání, avšak před jeho podrobným prozkoumáním byla podpora pro OS Windows přesunuta do ústraní a prakticky tak uvízla na mrtvém bodě. Pro další využití se tak stal GStreamer neperspektivní a byl zavržen.[18]

Obr. 2.11: Logo projektu GStreamer.[18]

2.2.3 Projekt xiph

Doposud byly zmiňovány projekty pracující na bázi FFmpeg knihoven. Již v teoretické části práce však bylo řečeno, že někteří z vývojářů pracují mimo FFmpegu i na dalších podobných projektech a xiph je jedním z nich. Při využití tohoto projektu pro tvorbu RTP komunikace by odpadly problémy se zkoumáním správného tvoření RTP paketů a obsluhování datových toků, ale síťová komunikace je v tomto případě omezena pouze na kontejner ogg. Ten je vyvinut pod záštitou Xiph.Org Foundation a je tedy zaručeně svobodný. Zmíněné omezení není až tak zásadním

problémem, neboť při tvorbě diplomové práce je také nutné dodržovat licenční podmínky a u množství dnes užívaných kodeků, formátů a kontejnerů by mohlo dojít k problémům právě s licencemi (MPEG, AVI[20] apod.). U produktů Xiph.Org Foundation tento problém nehrozí. Na základě uvedených skutečností byla část projektu xiph zvolena jako předloha pro tvorbu RTP komunikace. Protože produkty projektu xiph jsou schopny pracovat pouze s daty komprimovanými svobodnými kodeky, byly pro implementaci vybrány podklady související s kodeky Theora (video) a Vorbis (audio).[20][13]

2.3 Vývoj aplikací pro RTP komunikaci

Prvním krokem bylo prostudování dokumentace ze zdroje [20]. Dále byly z téhož zdroje získány části zdrojových kódů, které tvoří základ pro aplikace vysílání a přijímání RTP dat. Použité materiály se zabývají nízkoúrovňovými operacemi, které souvisí:

- s vytvořením RTP paketu,
- s řídícími prvky síťového spojení.

Hlavní důvod využití uvedených částí projektu xiph je ten, že úzce souvisí s použitými kodeky a jejich opětovná tvorba by vyžadovala zdlouhavé zkoumání problémů týkajících multimediálních kodeků a nebyl by pak možný rozvoj daného tématu kupředu. Aby vznikly aplikace plnící požadovaný úkol, tj. přenos multimédií pomocí RTP paketů, bylo nutné dotvořit nezanedbatelné množství zdrojového kódu v jazyce C, a to i přes nasazení již existujících funkcí.

2.3.1 Obecné informace

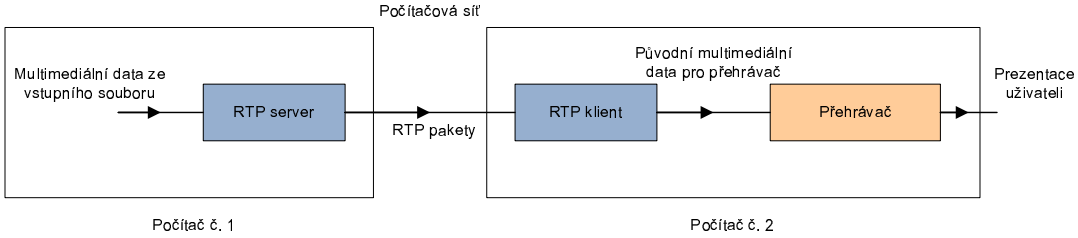
Podobně jako v případě přehrávače je vhodné rozdělit vývoj programu do několika fází. Implementace tak bude probíhat přibližně dle následujícího postupu:

- RTP přenos zvukových dat,
- navázání na přehrávač,
- RTP přenos video dat,
- navázání na přehrávač,
- propojení RTP aplikací pro přenos audio i video dat současně,

[20] multimediální kontejner vyvinutý firmou Microsoft – Audio Video Interleaving

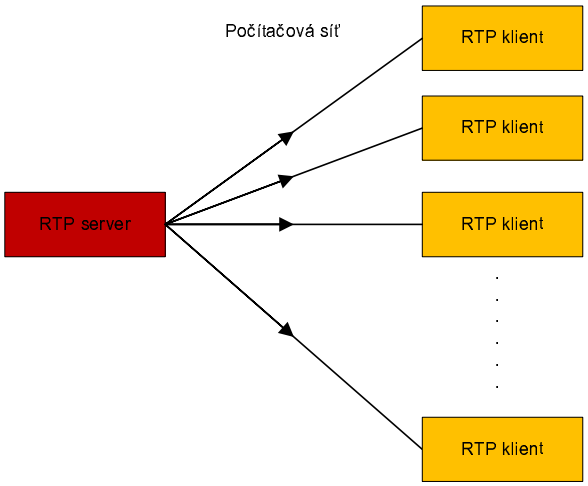
- navázání na přehrávač.

Jelikož se vysílač i přijímač RTP paketů bude nacházet obecně na zcela jiném místě v počítačové síti, musí být implementovány jako nezávislé aplikace. V této fázi vývoje již existoval také funkční přehrávač multimédií. V jedné z následujících podkapitol 2.3.4 je řešena také problematika komunikace RTP klienta s přehrávačem. Názorně je propojení (komunikace) aplikací naznačeno na obr. 2.12. Jak je vidět z naznačeného pracovního postupu, bod s názvem „navázání na přehrávač“ se v něm vyskytuje třikrát. Důvod je prostý. Správnost funkce RTP komunikace bylo nutné vždy ověřit pomocí přehrávače, který musel projít modifikací, poněvadž původně byl určen pro prezentaci souborů uložených pouze na disku počítače.



Obr. 2.12: Komunikace mezi aplikacemi.

Další obrázek 2.13 znázorňuje typ komunikace mezi RTP serverem a klientem resp. klienty. Jak je vidět, bude se jednat v podstatě o jednosměrnou komunikaci jednoho serveru k mnoha klientům pomocí kanálu UDP.



Obr. 2.13: Komunikace serveru s klienty.

Jak již bylo uvedeno, podkladem pro tvorbu RTP komunikace se stala část projektu xiph, která pomohla vyřešit problémy týkající se multimediálních kodeků a

licenčních smluv. Využité funkce jsou navrženy velmi sofistikovaně a univerzálně, což je výhodné, zejména pokud je třeba zpracovávat a přenášet audio i video. Hlavní výhoda tkví v tom, že v obou případech se pracuje s jedinou strukturou, přičemž je vždy využita její patřičná část. Z uvedeného faktu vyplývá, že aplikace pro síťový přenos zvuku budou strukturou velmi podobné aplikacím určeným pro distribuci videa. Popsaný přístup je nejen velmi elegantní, ale také umožní zjednodušit popis vývoje aplikací.[20]

V textu následujících kapitol bude postupně zmapována tvorba RTP klientů a serverů pro audio i video. Dále pak bude následovat část zabývající se propojením síťových komunikátorů s přehrávačem.

2.3.2 RTP server

Jde o aplikaci, která zajišťuje distribuci multimediálních dat do sítě, což znamená, že jsou na ni kladeny následující požadavky:

- nastavení síťových parametrů,
- otevření vstupního souboru,
- vytvoření paketů,
- odeslání paketů.

Samotná aplikace bude podobně jako přehrávač pracovat v konzoli OS Windows a v podstatě bude nezávislá na dalších vytvářených programech. Spojení s RTP klienty je realizováno přes univerzální rozhraní. Tím je zde počítačová síť resp. RTP pakety. Jediným pojítkem mezi serverem a klienty je nutnost podpory stejných formátů souborů, resp. kodeků (Vorbis a Theora).[20][6]

Síťové parametry

RTP server je program, který rozhoduje o tom, jaká data odesílat do sítě a komu, a proto je nutné před spuštěním nastavit několik základních informací. Jelikož tvořená aplikace pracuje v konzoli, bude nejvhodnějším způsobem definice parametrů jejich předání při spouštění programu. Importovaná data jsou zadána jako řetězec znaků za jménem programu, přičemž každý parametr je uvozen přepínačem, za nímž následuje předávaná hodnota (viz obr. 2.14). Možné parametry jsou tyto:

1. -i [*ip_adresa*]
   IP adresa, na kterou budou odesílány RTP pakety, vytvořené ze vstupního souboru. Přednastavena je hodnota 127.0.0.1[21].

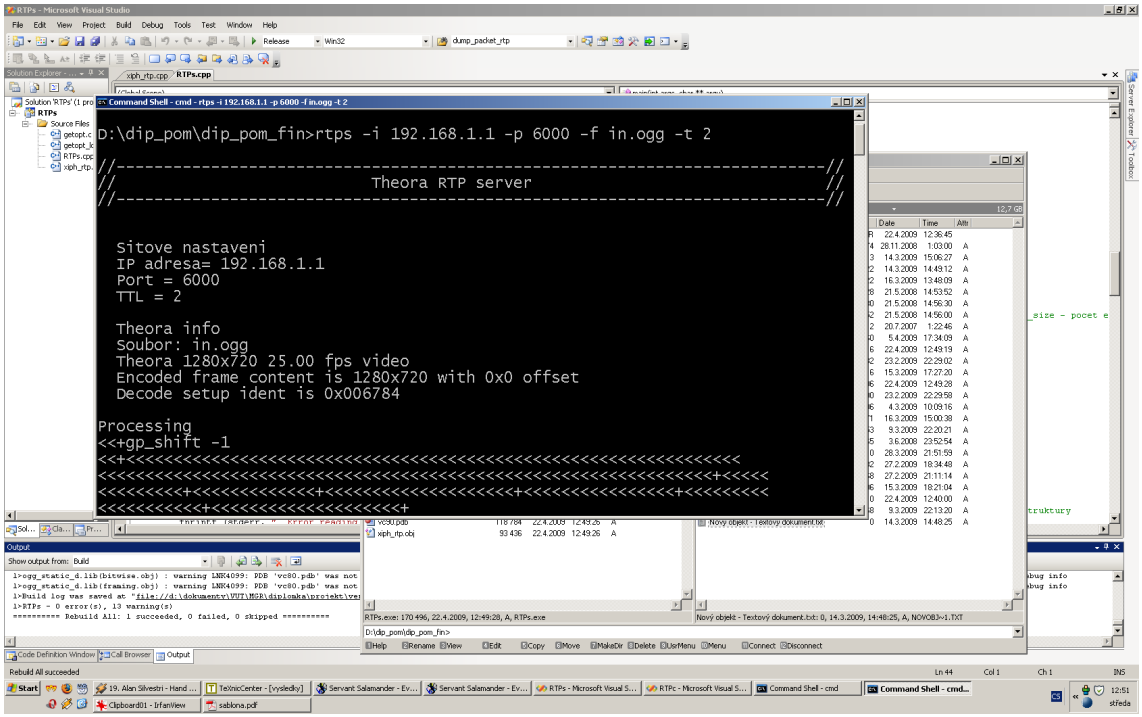
[21] lokální smyčka PC.

2. -p [*port*]
   definuje port, na kterém budou odesílány RTP pakety. Přednastavena hodnota 5000 (zvuk) popř. 4000 (obraz).

3. -f [*jméno_souboru*]
   specifikuje vstupní soubor, jehož obsah bude program zpracovávat.

4. -t [*ttl*]
   doba života paketu – Time To Live. Číselně udává přes kolik síťových uzlů může paket projít, než bude zahozen. Přednastavena hodnota 1.

Povinným vstupním parametrem je pouze název souboru. Pokud není specifikován, program se ukončí s chybou. Jestliže nejsou ostatní parametry při spouštění serveru definovány, použijí se interně specifikované hodnoty. Přednastavené hodnoty lze lehce změnit i ve zdrojovém kódu (programátor).[20]



Obr. 2.14: Ukázka spuštění a činnosti RTP serveru.

Princip zpracování vstupních parametrů je podobný jako v případě přehrávače. Postupně je procházen řetězec za názvem spouštěného souboru (programu) a jeho části jsou porovnávány a ukládány pro další použití. Také je možné zadat vstupní parametry v libovolném pořadí.

Na základě zadaných vstupních parametrů se pak realizují ve zdrojovém kódu serveru další operace. Nejprve je nutné inicializovat síťové rozhraní. Dále je vytvořen soket pro komunikaci po síti a pokud vše proběhne v pořádku, jsou na výstup

vytisknuty podrobnosti o síťovém spojení. Pro vytvoření těchto částí byly použity některé části z projektu xiph.[20]

Vstupní soubor

Pokud je bezchybně připravena síť, může být otevřen vstupní soubor, z něhož budou získávána data pro tvorbu paketů. Otevření i načítání dat je realizováno nízkoúrovňově, pomocí funkcí jazyka C.

Zpracování dat, pakety, ukončení vysílání

Než bude z načtených dat vytvořen paket, je nutné provést ještě několik kroků. Nejprve je třeba alokovat všechny struktury. Následuje inicializace kodéru a dále pak zjišťování, zda otevřený soubor (první načtená data) je skutečně zakódován pomocí Vorbisu nebo Theory. Pokud proběhnou uvedené kroky bez chyby, je nutné odeslat informaci naslouchajícím klientům, že budou posílána data „určená pro ně“. Toto se provede odesláním tří paketů, z nichž každý obsahuje tzv. „hlavičku“, která má v sobě informaci o použitém kodeku. Jelikož je komunikace prováděna jednosměrně, nečeká se na potvrzení od klientů a hned se pokračuje v posílání dalších dat.[20]

Odesílání spolu s načítáním dat ze souboru je implementováno v programové smyčce, která je ukončena, pokud program narazí na konec souboru. Nejdříve je získáno patřičné množství dat ze souboru. Dále je vytvořen RTP paket, který je následně odeslán do sítě. Při tvorbě paketu je nutné dbát na naplnění jednotlivých jeho částí správnými hodnotami (viz 1.1.1). Aby byl uživatel alespoň nějakým způsobem informován o odesílání paketů, je za každý odeslaný paket vypsán do konzole jeden znak[22], ukázka viz obr. 2.14, přičemž ve zdrojovém kódu může programátor povolit i podrobný výpis do konzole.[17]

Po dosažení konce souboru je ukončena programová smyčka a zbývá korektně ukončit aplikaci. Zavře se použitý soubor, je uvolněna alokovaná paměť PC a program se ukončí.

Nyní je k dispozici funkční vysílač RTP paketů. Jak již bylo řečeno dříve, uvedený postup implementace je téměř shodný pro zvuk i video. Odlišnosti jsou dvě. Při práci se zvukem jsou použity jiné části struktur, než je tomu v případě videa. V některých případech je také nutné místo funkcí pro kodek Vorbis využít funkce určené pro Theoru. Při tvorbě byly použity zdrojové kódy vyvinuté v rámci projektu xiph. Při práci s nimi je velmi nápomocná podrobně zpracovaná dokumentace. Na obrázku 2.14 je vidět praktická ukázka vysílání RTP paketů.[20]

[22]znaky jsou různé a mají svůj přesný význam, podrobněji viz zdrojový kód

2.3.3 RTP klient

Je jakýmsi protipólem serveru. Jeho umístění v síti je tam, kde bude požadován příjem RTP paketů a jejich následná reprezentace uživateli opět v podobě multimediálních dat. Ve finálním stádiu vývoje bude přímo navázán na přehrávač (viz 2.3.4). Požadavky na RTP klienta jsou následující:

- nastavení síťových parametrů,
- příjem paketů,
- zpracování paketů,
- komunikace s přehrávačem.

Nejprve bude RTP klient vyvíjen samostatně jako konzolová aplikace, která pouze přijímá a zpracuje pakety. Podporovány budou, jako v případě serveru, kodeky Vorbis a Theora. Rovněž zde platí, že struktura programu zpracovávajícího zvuk je téměř totožná se strukturou aplikace pro video.[20][6]

Síťové parametry

Obdobně jako v případě serveru lze při spouštění klientovi předat vstupní parametry. Zde však není žádný z parametrů povinný, všechny jsou přednastaveny na určitou hodnotu. Toto je provedeno záměrně, aby se na straně uživatele (klienta) minimalizovala režie související se spouštěním a nastavováním aplikací při RTP komunikaci. Přednastavené hodnoty lze samozřejmě změnit i přímo ve zdrojovém kódu. Předávané parametry mohou být následující:

1. -i [*ip_adresa*]
   IP adresa, ze které budou očekávány RTP pakety. Přednastavena je hodnota 227.0.0.1.

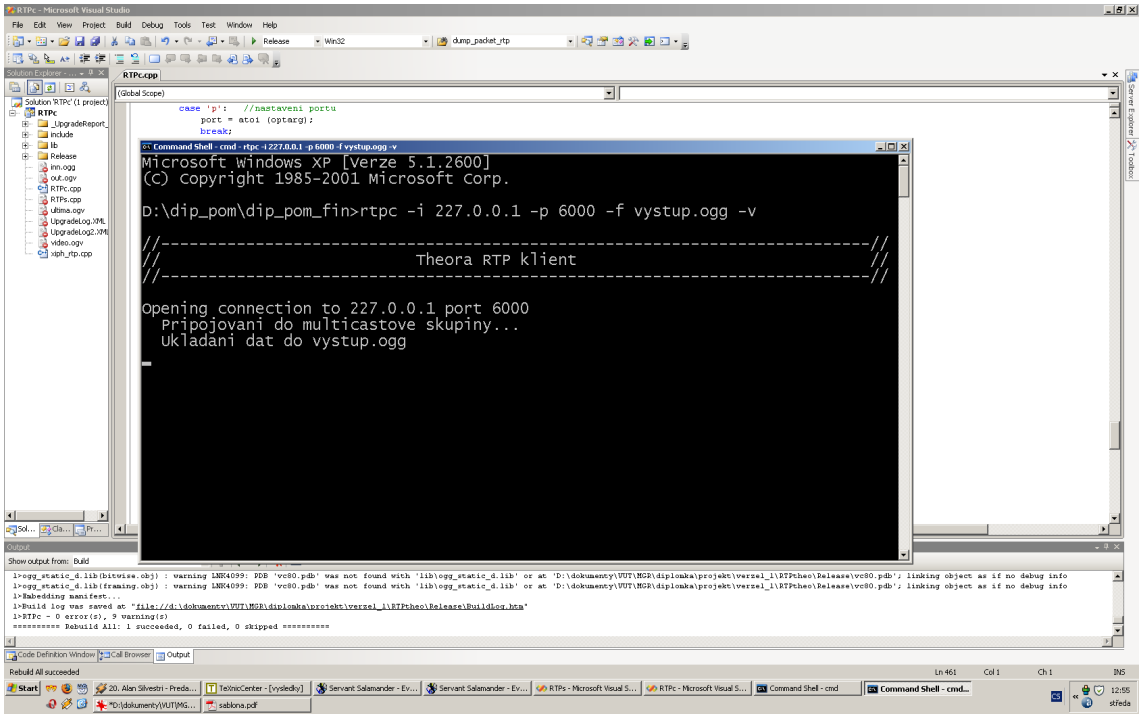
2. -p [*port*]
   definuje port, na kterém bude klient naslouchat. Přednastavená je hodnota 5000 popř. 4000.

3. -f [*jméno_souboru*]
   specifikuje výstupní soubor, do něhož budou ukládána přijatá data. Implicitně out.ogg popř. out.ogv[23].

4. -v
   umožňuje zapnout podrobný výpis o přijatém paketu do konzole.

[23] implementováno dodatečně, více viz další kapitola.

Princip zpracování a použití vstupních parametrů je totožný s RTP serverem. S jejich pomocí je následně otevřeno síťové spojení pro komunikaci. Ukázka zadávání parametrů při spouštění programu je na obrázku 2.15.

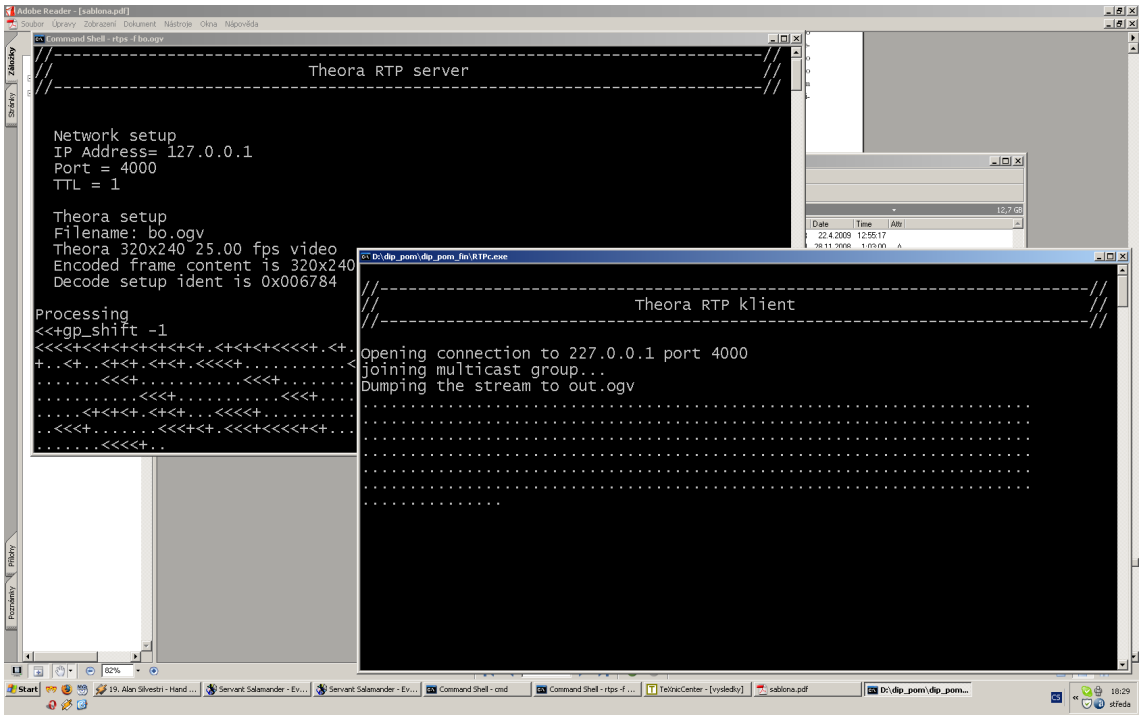


Obr. 2.15: Ukázka spuštění RTP klienta s parametry.

Příjem a zpracování paketů

Struktura hlavní funkce přijímače není tak bohatá jako v případě serveru. Po otevření sítě následuje programová smyčka. V té se čeká, dokud nepřijdou na definovaný port a soket pakety. V okamžiku příchodu paketů jsou tyto načteny do připraveného místa v paměti a mohou být dále zpracovány. Později byla v této smyčce implementována funkce, umožňující z paketů vytvořit multimediální data a ty uložit do souboru na disk počítače. Dále se v této smyčce vyskytuje funkce, která umí do konzole vypsat podrobné informace o přijatém paketu, což je podmíněno vstupním parametrem „-v“. Aby byl uživatel informován o činnosti programu, je s každým přijatým paketem vypsán do konzole jeden znak. Praktická je ukázka na obrázku 2.16.[20]

V tomto stádiu vývoje existují již aplikace realizující RTP komunikaci (obr. 2.16), přičemž je možné přenášet obraz i zvuk, prozatím však odděleně.



Obr. 2.16: Komunikace RTP klienta a serveru přes lokální smyčku PC.

2.3.4 Propojení s přehrávačem

Aby mohly být informace přenesené po síti prezentovány uživateli, je nutné nějakým způsobem provázat RTP klienta a přehrávač. Zde se nabízí dvě možnosti:

1. Integrace klienta do přehrávače (popř. naopak) a přímé zpracování a přehrávání dat ze sítě. Zde je výhodou existence jediná aplikace, plnící všechny funkce. Tak se minimalizuje režie na straně uživatele. Nevýhodou je, že přijaté informace jsou k dispozici jen jednou a v případě výskytu neočekávané události může dojít k jejich nenávratné ztrátě. Spojením aplikací pak dojde také do jisté míry ke ztrátě jejich modulárnosti.

2. Klient i přehrávač jsou autonomní aplikace propojené vhodným univerzálním prvkem. V tomto případě se zvyšuje režie pro uživatele, neboť musí spouštět vícero aplikací. Výhodou je pak zachování modularity a také možnost uchovat přijatá data, jelikož propojovacím prvkem může být např. soubor na disku počítače.

S ohledem na pohodlí uživatele byla nejprve zvolena varianta číslo jedna. Pokusy o integraci aplikací se však nesetkaly s úspěchem. Jak se ukázalo, hlavní překážkou se stalo použití rozdílných svobodných projektů pro RTP komunikaci a přehrávač. Knihovny FFmpeg, ač jsou naprogramovány čistě v jazyce C, pracují na „pseudo-objektovém“ principu, jenž je bližší spíše vyšším programovacím jazykům. Naproti

tomu části kódu z projektu xiph realizují všechny své operace na co nejnižší úrovni a využívají elementární funkce jazyka C. Sloučení těchto dvou přístupů se ukázalo být tak velkým problémem, že jej nebylo možné, především z časových důvodů, realizovat. Jednu z hotových částí práce by bylo nutné vytvořit v podstatě od začátku a zcela jinak. U přehrávače tato možnost téměř nepřipadá v úvahu, jelikož vytvořená aplikace je strohá funkcí, a přesto její zdrojový kód zabírá několik set řádků. V případě RTP komunikátorů je zase problém především s dokumentací a kodeky. Od varianty číslo jedna bylo později upuštěno, neboť přivedla vývoj do slepé uličky.

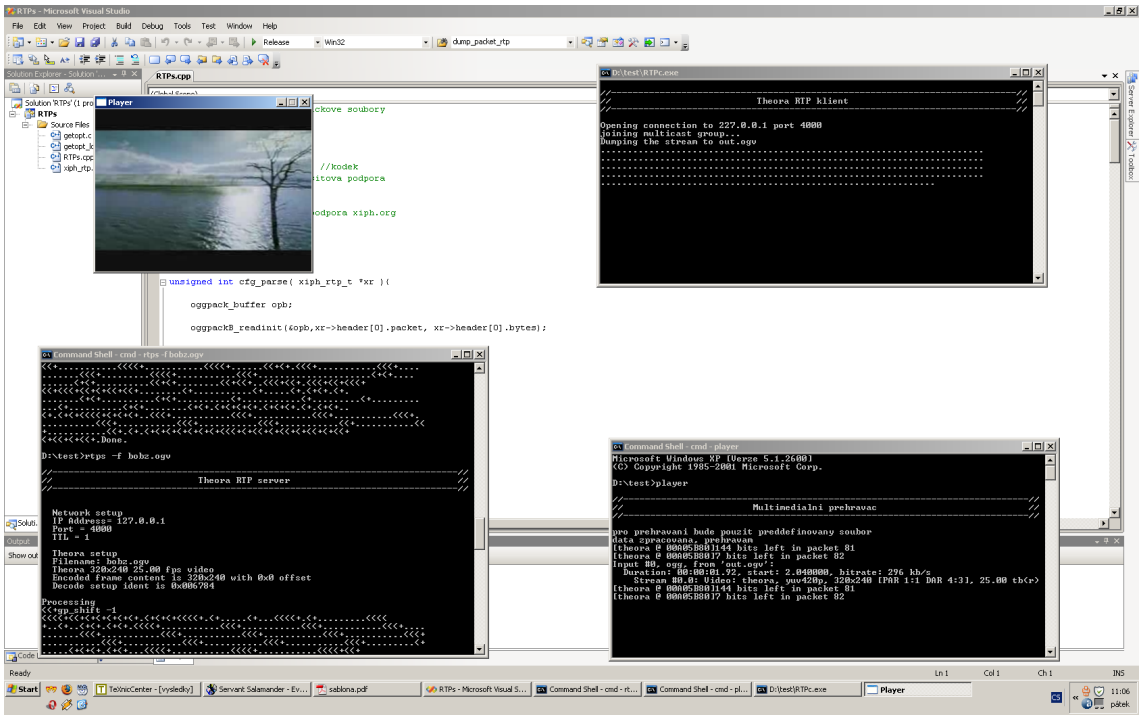
V případě autonomních aplikací nepatrně narůstá požadavek na uživatele, neboť je třeba spouštět dva programy. Jako propojovací článek mezi přehrávačem a RTP klientem byl zvolen soubor. Toto řešení poskytuje řadu výhod. Do souboru je možné ukládat jak přímo multimediální data tak např. pakety přijaté ze sítě a ty dále zpracovávat dle libosti, přičemž úprava zdrojového kódu je otázkou několika okamžiků. Další výhodou, plynoucí z použití souboru, je vysoká modularita vytvořených aplikací. Přehrávač lze použít zcela samostatně, jako jakýkoliv jiný multimediální přehrávač, nebo v návaznosti na vytvořený RTP komunikátor. Ten je možné rovněž využít samostatně, případně jej zaměnit za jiný, který umožňuje přenášet po síti i data zpracovaná jinými kodeky než je Vorbis a Theora[24]. Zmíněné řešení má však nevýhodu. Pokud použijeme jako spojovací prvek multimediální soubor, bude v něm přehrávač při otvírání hledat informace o datových tocích, aby mohly být použity patřičné dekodéry. Z toho vyplývá, že přehrávačem nelze otevírat prázdný soubor resp. soubor s nulovou velikostí. Je proto nutné zavést mezi příjmem prvních paketů a spuštěním přehrávače určité zpoždění, hodnotou odpovídající času potřebnému pro přenos základních informací o datových tocích po síti. Tento aspekt je vyřešen v rámci přehrávače vhodnou úpravou zdrojového kódu.

Nyní již tedy existuje přehrávač multimédií a RTP komunikátory pro audio a video. Vysílač na straně serveru otevře zadaný soubor, začne z něj načítat informace, z nichž postupně vytváří RTP pakety a odesílá je do sítě. Klientské aplikace dekódují obdržené pakety a ukládají získaná data do souboru. Z něho následně čte přehrávač, který již prezentuje uživateli obraz a zvuk (ukázka viz 2.17). Na straně přijímače je mezi uložením prvních dat do souboru a jejich zprostředkováním uživateli zavedeno prodlení v řádu několika sekund.

2.3.5 Společné vysílání audia a videa

Až doposud byly programy pro přenos audio a video dat po síti vyvíjeny odděleně. Tento postup byl zvolen záměrně, protože umožňoval snadnější odladění funkce jednotlivých komponent. Po síti se podařilo samostatně přenést audio i video. Zbývá

[24]jiný RTP komunikátor může být předmětem dalšího rozvoje zadaného tématu



Obr. 2.17: Přenos videa přes lokální smyčku PC se současným přehráváním.

tedy realizace té varianty, kdy bude zvuk a obraz vysílán a přijímám dohromady. Lze rozlišit dvě realizační cesty:

1. Audio i video jsou přenášeny přes síť jedním datovým proudem. Prakticky je vysílání realizováno na jednom síťovém portu.

2. Obraz a zvuk jsou vysílány a přijímány na odlišných portech. Existuje více síťových spojení od vysílače k přijímači.

Nejprve byla zvolena varianta číslo jedna. Vzhledem k tomu, že přehrávač je propojen s RTP přijímačem pomocí souboru, je nutné, aby v tomto případě byla data před i po přenosu uložena v jednom souboru. Splnění tohoto požadavku se ukázalo být velkým problémem. Zvuk i obraz jsou sice uloženy v jediném souboru (kontejner), ale ten prakticky obsahuje několik datových proudů uložených „vedle sebe“. Pro přehrávání audia i videa je tedy nutné nalézt v kontejneru počátky obou datových toků. Právě tato operace se stala úskalím, které se nepodařilo překonat. Aplikace vytvářející a zpracovávající síťová data totiž přistupují k souborům na nejnižší možné úrovni, jakou jazyk C poskytuje, a pomocí tohoto přístupu se nepodařilo nalézt další datové toky uložené v kontejneru. Po otevření souboru tak byl k dispozici pouze první datový tok uložený v kontejneru, v případě dvou datových proudů šlo o video. Jedním z řešení zmíněného problému je podrobně prozkoumat, jak multimediální kodeky pracují se vstupními soubory. Již dříve však bylo zmíněno,

že tato cesta vysoce převyšuje rámec diplomové práce zejména časovou náročností. Dané téma by tak uvízlo na mrtvém bodě.

I přes překonání problémů s nalezením počátku datových toků v souboru by mělo řešení dle bodu jedna své nedostatky. Při přenášení zpráv obrazu i zvuku v jediném informačním proudu by v případě výpadku spojení na příslušném portu přišel uživatel ihned o všechny informace (audio i video). Pokud jsou použita dvě síťová spojení, nastala by uvedená situace až v případě nefunkčnosti komunikace na obou portech. Dále pak použití jednoho aplikačního portu zvyšuje režii zpracování dat ve vysílači i přijímači, neboť pakety je nutné před odesláním značkovat a po doručení třídit dle jejich příslušnosti k audiu či videu. Na základě uvedených skutečností byla později vybrána pro řešení možnost číslo dvě.

I ve druhém případě, kdy jsou pro odesílání obrazu a zvuku zvoleny rozdílné porty, je však možné pracovat s jedním vstupním a výstupním souborem. Z toho vyplývá, že dříve se vyskytující problém s hledáním počátku jednotlivých informačních proudů se přenesl i sem. Jako řešení bylo nakonec zvoleno rozdělení jednotlivých datových toků z původního kontejneru do samostatných souborů. Tato volba poměrně elegantně odstraní vzniklou překážku, ale nepatrně zvýší režii obsluhy na straně serveru, protože je nutné z jednoho multimediálního kontejneru vytvořit více souborů[25]. Na straně klienta jsou po přenesení informací po síti vytvořeny také dva soubory, přičemž jeden obsahuje video a druhý audio data. Komfort obsluhy se však nesnižuje, jelikož s vytvořenými soubory si poradí přehrávač. Pro zachování vysoké modularity a přenositelnosti zůstávají komunikátory pro zvuk a obraz na straně serveru i klienta implementovány jako samostatné aplikace. Jejich sloučení do jednoho programu může být předmětem dalšího vývoje.

Úprava přehrávače

Původně byla aplikace určena pro čtení dat z jediného kontejneru obsahujícího oba datové proudy, zvuk i obraz. Nyní je požadována obsluha i dvou souborů, což si vyžádalo jisté úpravy ve zdrojovém kódu. Po prozkoumání struktury vytvořeného programu bylo zjištěno, že kýženého efektu lze docílit změnou dekódovacího vlákna. Původní verze funkce *decode_thread* zůstala zachována a přibyla nová procedura starající se o získávání informací ze dvou souborů. V hlavním programu je pak přepínač, který volí verzi vlákna.

Přehrávač je vytvořen tak, že nově implementovaná funkce je dostupná pouze v návaznosti na RTP komunikaci. Názvy dvojice souborů jsou přednastaveny na out.ogg (zvuk) a out.ogv (video). V případě použití samostatného přehrávače je možné zpracování pouze jediného multimediálního kontejneru. Název přehrávaného

[25]tento úkon je prováděn pomocí vybraného konvertoru, viz 2.3.6

souboru se pak zadává jako jeden ze vstupních parametrů při spuštění programu v konzoli.

Praktické testování

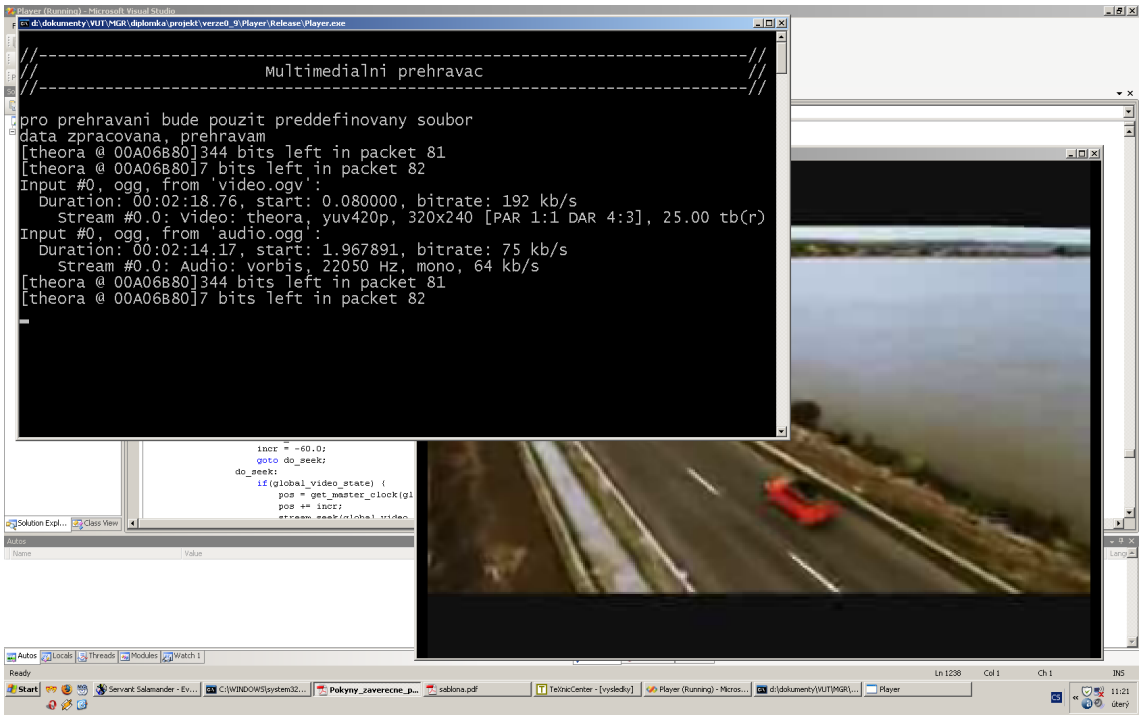
Při ověřování korektnosti běhu aplikací v této fázi vývoje se objevil problém, jehož odhalení nebylo dříve možné, protože až doposud přes síť putovalo pouze audio nebo video. Při společném přenosu a reprodukci obrazu a zvuku byla pozorována ztráta synchronizace mezi nimi. Po testování na mnoha videoklipech a podrobném zkoumání prováděných úkonů bylo zjištěno, že již při oddělování datových toků z původního kontejneru dochází k jevu, kdy je mezi začátek souboru a videoklipu vložena jistá časová „mezera“. Tato prodleva se vypisuje spolu s dalšími informacemi o datovém toku v souboru (viz obr. 2.18) a jistým způsobem pravděpodobně souvisí se ztrátou synchronizace mezi audiem a videem. Empiricky byly vypozorovány tyto vlastnosti:

1. Časový posuv je rozdílný pro audio i video.
2. Prodleva souvisí přímo s konkrétním klipem a je konstantní, tj. opětovným zpracováním původního souboru se vyskytuje stále stejná hodnota pro audio i video.
3. Kritickým bodem je přenos po síti, kdy se prodlevy radikálně mění a teprve pak dochází k problémům se synchronizací.
4. Nežádoucí intervaly se vyskytují jen u některých kodeků, např. Theora a Vorbis. U kodeků pracujících se specifikacemi MPEG-4 Video a MPEG-1 Audio Layer 3 nebyly zaznamenány.

Z důvodu omezeného časového úseku, který je určen pro řešení diplomové práce, nebylo již toto úskalí odstraněno. Jeho řešení může být součástí dalšího vývoje tohoto projektu. Přehrávač je vybaven funkcí, která udržuje posun mezi audiem a videem nepostřehnutelných mezích. Protože problémy se synchronizací vznikají až po přenosu dat po síti, ačkoliv nežádoucí prodlevy se vyskytují již dříve, hledaná chyba vzniká nejspíše v aplikacích zajišťujících síťovou komunikaci. Jednou z možností řešení by mohlo být i použití jiných kodeků. V tomto případě by však bylo nutné vytvořit jiné RTP vysílače a přijímače.

2.3.6 Multimediální konvertor

Již dříve bylo zmíněno, že v současné době existuje obrovské množství multimediálních kodeků pro video i audio a vytvořené RTP komunikátory pracují jen s daty



Obr. 2.18: Ukázka existence časových prodlev po rozdělení datových proudů do více souborů – položka *start* ve výpisu informací.

komprimovanými Vorbisem a Theorou. Aby tedy bylo možné distribuovat po síti libovolný multimediální klip, je třeba jej převést do formátu, který vytvořené programy umí zpracovat. Zmíněný úkol bude nejvhodnější uskutečnit pomocí některé z již existujících aplikací. Hledaný program by měl splňovat tyto požadavky:

- konverze na formát Theora a Vorbis,
- možnost nastavení kvality výstupu,
- konzolová aplikace,
- svobodný software,
- slučování a rozdělování datových toků.[13][14]

I přes přísné podmínky byla nakonec nalezena aplikace s názvem ffmpeg.exe, která splňuje všechny zmíněné požadavky. Jedná se o součást projektu FFmpeg a aktuální verzi lze získat ze zdroje [14].

3 ZÁVĚR

V rámci diplomové práce byl vytvořen přehrávač multimediálních souborů, který si poradí s velkým množstvím dnes užívaných kodeků. Aplikace přehrává samostatné audio, video i klipy obsahující obraz a zvuk. Program pracuje v konzoli OS Windows a je vybaven základními ovládacími prvky, tj. pozastavení přehrávání, posun vpřed a vzad (o 10 nebo 60 sekund), režim celé obrazovky. Přehrávač disponuje také funkcí, která zajišťuje synchronizaci mezi datovými toky při přehrávání. Jako hlavní tok lze zvolit audio, nebo video. Název souboru, který má být reprodukován, je předáván programu jako parametr při spuštění v příkazové řádce.

Dále byly implementovány aplikace zabývající se RTP komunikací v počítačové síti. Konkrétně se jedná o vysílač a přijímač a to pro audio i video. Úkolem vysílače je otevřít multimediální soubor, načíst data, zpracovat je do síťových zpráv a tyto odeslat. Přijímač pak zprávy přijme, extrahuje z nich multimediální obsah a ten uloží do souboru. Podobně jako v případě přehrávače lze programům při spuštění předat parametry z příkazové řádky. Jsou to např. tyto: IP adresa, aplikační port, název souboru, doba života paketu. RTP komunikátory pracují s daty komprimovanými kodeky Vorbis (zvuk) a Theora (video). Propojení s přehrávačem je provedeno pomocí nejuniverzálnějšího možného rozhraní a tím je soubor na disku počítače.

Přehrávač i RTP vysílače a přijímače jsou ponechány jako autonomní aplikace. Je tak zachována jejich vysoká modularita a přenositelnost. Při testování vytvořených aplikací se podařilo po sítí přenést a reprodukovat samostatné audio a video. Při pokusu o přenos zvuku a obrazu současně došlo k přenosu dat, ale při přehrávání byl zjištěn problém se synchronizací mezi datovými toky. Ten se již nepodařilo odstranit z důvodu omezených časových prostředků vyhrazených na řešení diplomové práce, bylo však zjištěno, že chyba se s největší pravděpodobností vyskytuje v algoritmech aplikací určených pro síťovou komunikaci.

Z hlediska licenčních podmínek se jako problematická jeví knihovna SDL, neboť je uvolněna pod LGPL licencí. Použité části z FFmpegu spadají pod GPL a prostředky z projektu xiph zastřešuje licence BSD.

Zadané téma diplomové práce je schopno dalšího rozvoje, protože některá úskalí prozatím nejsou dořešena. Především oblast RTP komunikace vybízí k dalšímu vývoji (podpora více kodeků, odstranění synchronizačních problémů atd.), ale i přehrávač by mohl být dále zdokonalován (podpora titulků, integrace síťové komunikace atp.). Je nepochybné, že i v budoucnu bude látka s tématikou blízkou zadání práce perspektivní, jelikož již delší dobu je snaha integrovat multimédia a komunikaci do IP sítí.

LITERATURA

[1] BANERJEE, K. *Introduction to Internet Multimedia*, [online]. c 2005, 22. březen 2006 [cit. 2008-11-06]. Text v angličtině. Dostupný z WWW: <http://www.geocities.com/intro_to_multimedia/index.html>.

[2] BELLARD, F., NIEDERMAYER, M., et al. *FFmpeg*, [online]. 200-?, 3. prosinec 2008 [cit. 2008-11-23]. Text v angličtině. Dostupný z WWW: <http://ffmpeg.mplayerhq.hu/>.

[3] BöHME, M. *Using libavformat and libavcodec*, [online]. 18.2.2004, 22.7.2008 [cit. 2008-11-28]. Text v angličtině. Dostupný z WWW: <http://www.inb.uni-luebeck.de/~boehme/using_libavcodec.html>.

[4] DAVID, P. *Identifikace audiosegmentů pro automatickou transkripci zpravodajských pořadů : Autoreferát disertační práce*. 2006. 32 s. Technická univerzita v Liberci, Fakulta mechatroniky a mezioborových inženýrských studií. [cit. 2009-04-03]. Referát. Dostupný z WWW: <http://www.fm.tul.cz/files/autoreferat_david.pdf>.

[5] DRANGER, S. *An ffmpeg and SDL Tutorial*, [online]. c 2003 [cit. 2008-12-1]. Text v angličtině. Dostupný z WWW: <http://www.dranger.com/ffmpeg/>.

[6] GONCALVES, I., PFEIFFER, S., MONTGOMERY, C. *Ogg Media Types, Request for Comments 5334*, [online]. Network Working Group, c 2008 [cit. 2009-03-23]. Angličtina. Dostupný z URL: <http://www.xiph.org/ogg/doc/rfc5334.txt>.

[7] JAHODA, R. *Formáty obrazu videa* [online]. 1998-2009, 3. říjen 2001 [cit. 2008-11-5]. Dostupný z WWW: <http://www.tvfreak.cz/art_doc-AF3799F3A349EE89C125727C0059F8E5.html>. ISSN 1802-1328.

[8] JAHODA, R. *Kodeky tajemství zbavené* [online]. 1998-2009, 13. říjen 2005 [cit. 2009-03-16]. Dostupný z WWW: <http://www.tvfreak.cz/art_doc-373A9DA2913B7BD3C125727C00592A37.html?lotus=1&Highlight=0,kodek>. ISSN 1802-1328.

[9] JAHODA, R. *Kontejner není kontejner* [online]. 1998-2009, 10.5.2005 [cit. 2009-03-15]. Dostupný z WWW: <http://www.tvfreak.cz/art_doc-7336C842E0DDDE25C125727C0059416E.html?lotus=1&Highlight=0,kontejner>. ISSN 1802-1328.

[10] LANTINGA, S. *SDL*, [online]. c 1998, 31. prosinec 2007 [cit. 2008-11-23]. Text v angličtině. Dostupný z WWW: <http://www.libsdl.org/index.php>.

[11] LEDVINA, J. *RTP — Real Time protocol: Přednášky z projektování distribuovaných systémů*. Plzeň: Západočeská univerzita, fakulta aplikovaných věd, katedra informatiky a výpočetní techniky, 2006. 69 s. Prezentace k přednášce. Dostupný z WWW: <http://www.kiv.zcu.cz/l̃edvina/Prednasky-PDS-2007/09a-rtp_VoIP.ppt>.

[12] PERKINS, C. *RTP: Audio and Video for the Internet*. 1. vydání. Addison-Wesley Professional, 2003. 438 s. ISBN 0-672-32249-8.

[13] PFEIFFER, S. *The Ogg Encapsulation Format Version 0, Request for Comments 3533*, [online]. Network Working Group, c 2003 [cit. 2009-03-06]. Angličtina. Dostupný z URL: <http://www.xiph.org/ogg/doc/rfc3533.txt>.

[14] POLLA, R., et al. *FFmpeg Windows Help*, [online]. 200-?, 3. prosinec 2008 [cit. 2008-12-3]. Text v angličtině. Dostupný z WWW: <http://ffmpeg.arrozcru.org/>.

[15] SCHULZRINNE, H., CASNER, S., FREDERICK, R., JACOBSON, V. *RTP: A Transport Protocol for Real-Time Applications, Request for Comments 3550*, [online]. Internet Engineering Task Force, c 2003 [cit. 2008-11-06]. Angličtina. Dostupný z URL: <ftp://ftp.isi.edu/in-notes/rfc3550.txt>.

[16] TUREK, M. *SDL: Hry nejen pro Linux* [online]. 1998-2009, 22.2.2005 [cit. 2009-03-15]. Dostupný z WWW: <http://www.root.cz/serialy/sdl-hry-nejen-pro-linux/>. ISSN 1212-8309.

[17] VRBA, K., NAGY, Z. *Multimediální služby.* Brno: Vysoké učení technické v Brně, ústav telekomunikací, [200-?]. 83 s. [cit. 2008-11-3]. Dostupný z WWW: <https://www.feec.vutbr.cz/et/skripta/utko/Multimedialni_sluby_S.pdf>.

[18] *gstreamer – open source multimedia framework*, [online]. c 2008, 28.11.2008 [cit. 2008-12-5]. Text v angličtině. Dostupný z WWW: <http://gstreamer.freedesktop.org/>.

[19] *VideoLAN – VLC media player*, [online]. c 2008, 10. prosinec 2008 [cit. 2008-12-5]. Text v angličtině. Dostupný z WWW: <http://www.videolan.org/>.

[20] *xiph.org*, [online]. c 1994–2008 [cit. 2008-12-5]. Text v angličtině. Dostupný z WWW: <http://www.xiph.org/>.

[21] *transcode_sample.c*, [online]. c 2007 [cit. 2008-12-5]. Text v angličtině. Zdrojový kód programu. Dostupný z WWW: <http://www.inf.ufsc.br/~leandro.coser/transcode_sample.c>.

[22] *Video PTS/DTS underflows*, [online]. c 2001 [cit. 2008-12-10]. Text v angličtině. Výňatek z internetové diskuze. Dostupný z WWW: <http://www.pxh.de/fs/svcd/DVB2SVCD/dts_pts_underflows_explanation.pdf>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ADT	abstraktní datový typ – Abstract Data Type
ANSI	americká standardizační organizace – American National Standards Institute
APP	aplikačně specifické funkce – Application
AVI	multimediální kontejner vyvinutý firmou Microsoft – Audio Video Interleaving
BYE	indikátor ukončení spojení
BSD	je jedna z nejsvobodnějších licencí, ale také OS odvozený z Unixu a distribuovaný Kalifornskou univerzitou v Berkeley, dnes existuje několik dalších odvozených verzí – Berkeley Software Distribution
C	programovací jazyk vyvinutý v 70. letech 20. stol. pro potřeby OS Unix
CC	počet záznamů v poli CSRC – Count CSRC
CD	kompaktní disk – Compact Dics
CNAME	kanonické jméno – Canonical Name
CSRC	příspěvkový zdrojový identifikátor – Contributing Source identifier
C++	objektově orientovaný programovací jazyk, vznikl rozšířením jazyka C
C99	standard ISO 9899:1999 vydaný v roce 1999 pro jazyk C, později přijatý i jako ANSI
Delphi	vývojové prostředí firmy Borland, tvorba aplikací v objektové nástavbě programovacího jazyka Pascal
DTS	dekódovací časová značka – Decoding Time Stamp
FhG	vysoce efektivní MP3 kodek – Fraunhofer-Gesellschaft
fps	počet snímků za sekundu – frames per second
GB	jednotka pro určení množství dat, gigabajt
GHz	odvozená jednotka kmitočtu, gigahertz

GNU	projekt zaměřený na svobodný software, inspirovaný operačními systémy unixového typu
GPL	licence pro svobodný software; všeobecná veřejná licence – General Public License
GUI	grafické uživatelské rozhraní – Graphic User Interface
Hz	je hlavní jednotkou frekvence (kmitočtu) v soustavě SI – Hertz
H.323	je doporučení definující protokoly pro audio-vizuální relace komunikace v jakékoli paketové síti
H.263	video kodek navržený pro videokonference s nízkými přenosovými rychlostmi
H.261	video kodek navržený pro přenos obrazu telefonními linkami ISDN
IP	datový protokol používaný pro přenos dat přes paketové sítě – Internet Protocol
ISO	Mezinárodní organizace pro normalizaci – International Organization for Standardization
ISDN	standard pro plně digitální telefonní síť, která umožňuje přenos hlasu, textu i obrazu; digitální síť integrovaných služeb – Integrated Services Digital Network
Java	objektově orientovaný programovací jazyk vyvinutý firmou Sun Microsystems, ceněný především díky své přenositelnosti
JPEG	metoda ztrátové komprese pro statické obrazy – Joint Photographic Experts Group
kb/s	odvozená jednotka pro přenosovou rychlost, kilobity za sekundu
LAME	vysoce kvalitní MPEG Audio Layer 3 kodér uvolněný pod licencí LGPL
LGPL	přísnější licence odvozená od GPL; zpravidla pro softwarové knihovny – Lesser GPL
Linux	jádro několika počítačových OS, někdy tak bývá označován celý OS, resp. celá rodina OS
M	značkovač – Marker

Mac OS	operační systém pro počítače Macintosh firmy Apple – Macintosh Operating System
MB	jednotka pro určení množství dat, megabajt
Mb/s	odvozená jednotka pro přenosovou rychlost, megabity za sekundu
MinGW	kolekce hlavičkových souborů, knihoven a GNU kompilátoru pro OS Windows – Minimalist GNU for Windows
MP3	viz MPEG-1 Audio Layer 3
MPEG	skupina standardů pro kompresi audiovizuálních dat – Motion Picture Experts Group
MPEG-1 Audio Layer 3	standard pro kódování zvuku
MPEG-4 Video	standard pro kódování videa, zastřešuje několik variant
MVS	Microsoft Visual Studio 2008
NTPTS	odpovídá času odeslání RTCP paketu typu SR
oga	přípona souboru jehož data jsou kódována ztrátovým zvukovým kodekem Vorbis
ogg	přípona souboru jehož data jsou kódována ztrátovým zvukovým kodekem Vorbis; označení pro multimediální kontejner vyvinutý nadací xiph
ogv	přípona souboru jehož data jsou kódována ztrátovým video kodekem Theora
OpenGL	standard specifikující multiplatformní rozhraní pro tvorbu aplikací počítačové grafiky – Open Graphics Library
OS	operační systém
P	vyjadřuje, že jsou jednotlivé složky barevného modelu Y,U,V přenášeny odděleně – Planar
PC	osobní počítač – Personal Computer
Pd	bezpečnostní rozšíření protokolu – Padding
PDA	osobní digitální pomocník, kapesní počítač – Personal Digital Assistant

Perl	interpretovaný programovací jazyk
PT	druh zátěže – Payload Type
PTS	prezentační časová značka – Presentation Time Stamp
Python	interpretovaný objektově orientovaný programovací jazyk
QoS	soubor pravidel zajišťujících rozdělení síťových prostředků dle stanovených priorit – Quality of Service
RFC	sada doporučení popisujících internetové protokoly, systémy aj.; sestavovány na základě praktických zkušeností – Request For Comments
RGB	aditivní barevný model; pro vyzařovací zařízení; červená, zelená, modrá – red, green, blue
RTCP	slouží k řízení RTP relace a ke sledování kvality toku – Real Time Control Protocol
RTP	protokol pro přenos dat v reálném čase – Real Time Transport Protocol
RTPTS	odpovídá času příjmu RTCP paketu typu SR
RTSP	protokol pro řízení doručování dat v reálném čase – Real Time Streaming Protocol
RR	zprávy přijímače – Receiver Report
RRC	počet přijatých zpráv bloků obsažených v daném RTCP paketu – Reception Report Count
RSVP	protokol pro rezervaci zdrojů v síti – Resource reSerVation Protocol
SAP	signalizační protokol pro řízení relací všesměrového vysílání – Session Announcement Protocol
SC	u paketu typu SDES protokolu RTCP jde o označení počtu SSRC/CSRC záznamů obsažených v daném paketu – Source Count
SDES	popisovače zdrojů – Source Description items
SDL	multiplatformní multimediální knihovna, jednoduchá vrstva pro přímý přístup k médiím – Simple DirectMedia Layer

SDP	slouží k popisu relace a vyjednání parametrů spojení – Session Description Protocol
SI	mezinárodně platná soustava jednotek fyzikálních veličin
SIP	protokol pro inicializaci relací, přenos signalizace v IP telefonii – Session Initiation Protocol
SNMP	umožňuje průběžný sběr a vyhodnocování dat pro potřeby správy sítě – Simple Network Management Protocol
Solaris	dříve SunOS, je operační systém unixového typu, vyvinutý společností Sun Microsystems
SR	zprávy vysílače – Sender Report
SSRC	synchronizační zdrojový identifikátor – Synchronization Source identifier
TCP	protokol transportní vrstvy modelu TCP/IP, poskytuje spolehlivou a potvrzovanou službu – Transmission Control Protocol
TCP/IP	čtyřvrstvý síťový model využívaný ve většině dnešních sítí, založen na protokolech TCP a IP
ttl	doba života paketu – Time To Live
UDP	protokol transportní vrstvy modelu TCP/IP, poskytuje nespolehlivou a nepotvrzovanou službu – User Datagram Protocol
UTP	typ strukturované kabeláže založený na principu kroucených párů – Unshielded Twisted Pair
U	vyjadřuje jednu z barvonosných složek v barevném modelu YUV; konkrétně rozdíl modré barvy a jasu
V	vyjadřuje jednu z barvonosných složek v barevném modelu YUV; konkrétně rozdíl červené barvy a jasu
Ver	verze protokolu – Version
VP3	ztrátový video kodek vyvinutý společností On2 technologies
Windows	řada grafických víceúlohových OS společnosti Microsoft
X	rozšíření – Extension

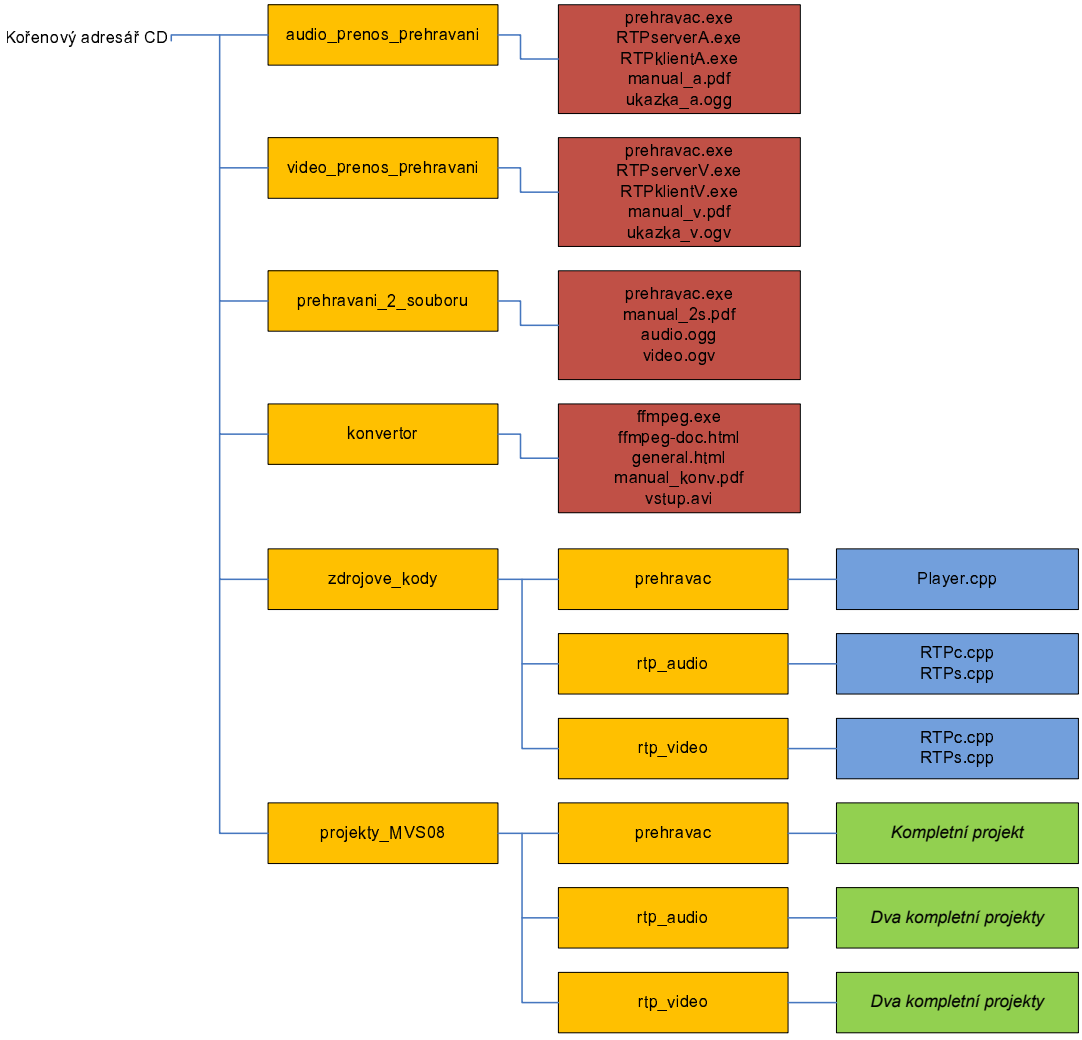
Y	vyjadřuje jasovou složku v barevném modelu YUV
YCbCr	modifikace původního YUV modelu pro digitální obraz a video; jas a dvojice barevných složek
YUV	barevný model; původně pro TV vysílání; tříprvkový vektor jasu a dvou barevných složek
-law	algoritmus redukující dynamický rozsah audio signálu; užívá se v Japonsku a Severní Americe
2D	obecná zkratka pro popis „světa“ pomocí dvou souřadnic; rovina, např. papír aj.
3D	obecná zkratka pro popis „světa“ pomocí tří souřadnic, odpovídá skutečnosti jak ji vnímáme

SEZNAM PŘÍLOH

A INFORMACE K PŘILOŽENÉMU CD

A.1 Struktura

Protože se na dodaném kompaktním disku kromě elektronické verze diplomové práce vyskytují také implementované aplikace, je jeho struktura naznačena na následujícím obrázku A.1. Veškeré návody k obsluze aplikací jsou viditelně označeny (manual_*.pdf) a uloženy ve složkách s aplikacemi.



Obr. A.1: Struktura souborů a složek na přiloženém CD; žlutá – složky, červená – soubory určené pro vyzkoušení, modrá – zdrojové kódy, zelená – projekty spustitelné v prostředí Microsoft Visual Studio 2008.

A.2 Testovací hardware

Vývoj aplikací proběhl v prostředí Microsoft Visual Studio 2008 Professional Edition (studentská licence) pod operačním systémem Windows XP Professional verze 2002, Service Pack 3. Testování bylo provedeno na dvojici počítačů připojených do jednoduché lokální sítě.

- Počítač č. 1 – notebook s procesorem Intel Core2Duo 1,83 GHz[1], 1 GB[2] operační paměti, pevný disk s 5600 otáčkami za minutu, operačním systémem je zmíněný Windows XP.
- Počítač č. 2 – stolní počítač s procesorem Intel Celeron 1,7 GHz, 512 MB[3] operační paměti, pevný disk s 7200 otáčkami za minutu, operačním systémem je zmíněný Windows XP.
- Lokální síť je propojena UTP[4] kabeláží kategorie 5. Aktivním prvkem je směrovač OvisLink eLive IP-3047. Maximální přenosová rychlost sítě je 100 Mb/s[5].

[1] odvozená jednotka kmitočtu, gigahertz
[2] jednotka pro určení množství dat, gigabajt
[3] jednotka pro určení množství dat, megabajt
[4] typ strukturované kabeláže založený na principu kroucených párů – Unshielded Twisted Pair
[5] odvozená jednotka pro přenosovou rychlost, megabity za sekundu