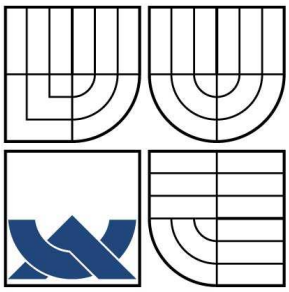
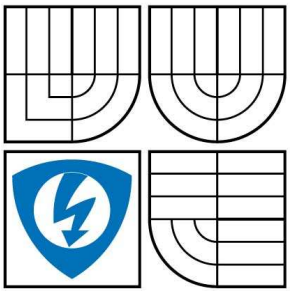


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

PŘENOS ZVUKOVÝCH DAT POMOCÍ PROTOKOLU RTP

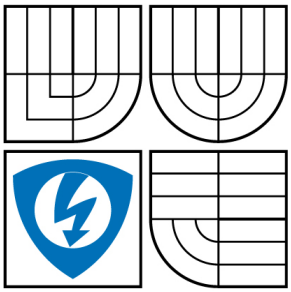
TRANSMISSION OF AUDIO DATA USING RTP PROTOCOL

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — ROMAN NOVOTNÝ
AUTHOR

VEDOUCÍ PRÁCE — ING. JIŘÍ KOUŘIL
SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Novotný Roman	***ID:***	77900
Ročník:	3	***Akademický rok:***	2007/2008

NÁZEV TÉMATU:

Přenos zvukových dat pomocí protokolu RTP

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte použití volně dostupných knihoven v jazyce C a C++ pro přenos zvukových dat v reálném čase pomocí protokolu RTP. Uveďte výhody a nevýhody u vybraných knihoven. Navrhněte a realizujte aplikaci typu klient-server pro přenos zvukových dat mezi dvěma počítači, ve které budou tyto knihovny použity. Aplikaci navrhněte pro platformu Windows a pro realizaci využijte programovacího prostředí C++Builder.

DOPORUČENÁ LITERATURA:

[1] PERKINS, C. RTP: Audio and Video for the Internet. Addison-Wesley Professional. 1st edition (June 11, 2003). 423. ISBN 978-0672322495.
[2] PUŽMANOVÁ, R. TCP/IP v kostce. KOPP : České Budějovice. ISBN 80-7232-236-2.
[3] SPORTACK, M. Směrování v sítích IP. Computer Press : Praha. c2004. ISBN 80-251-0127-4.

Termín zadání:	11.2.2008	***Termín odevzdání:***	4.6.2008

Vedoucí práce: Ing. Jiří Kouřil

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

UPOZORNĚNÍ:

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Roman Novotný
Bytem: Štefánokova 281, 66453, Újezd u Brna
Narozen/a (datum a místo): 19.7.1985, Brno

(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 244/53, 60200 Brno 2
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Ing. Kamil Vrba, CSc.

(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - ☐ disertační práce
   - ☐ diplomová práce
   - ☑ bakalářská práce

   jiná práce, jejíž druh je specifikován jako .....

(dále jen VŠKP nebo dílo)

Název VŠKP: Přenos zvukových dat pomocí protokolu RTP
Vedoucí/školitel VŠKP: Ing. Jiří Kouřil
Ústav: Ústav telekomunikací
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

- ☑ tištěné formě - počet exemplářů 1
- ☑ elektronické formě - počet exemplářů 1

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☑ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Tato práce se zabývá technologií multimediálního přenosu v reálném čase. Úvod práce je vyhrazen pro vrstvový model TCP/IP. Dále se bakalářská práce zaměřuje na teoretický návrh aplikace, která je schopna přenášet zvuková data minimálně mezi dvěma počítači pomocí protokolu RTP. Tuto část zahrnují protokoly pro přenos dat v reálném čase, zvláště pak protokol RTP a dále pak volně dostupné knihovny sepsané v jazyce C a C++, ve kterých je právě protokol RTP implementován. Z těchto poznatků nadále vychází závěr práce, ve kterém je teoretický návrh aplikace realizován, a to ve vývojovém prostředí Borland Builder C++.

Abstract

This project is about technology of multimedia transmission in the real time. Introduction of this work narrate model TCP/IP. Next the bachelor´s thesis is principally aimed on theoretical design of application, which is minimally able to transfer audio data between two computers by the help of the protocol RTP. This part is included in protocols for transmission of data in the real time, especially protocol RTP and then freely accessible libraries, which are written in language C and C++, in which is RTP protocol just implemented. End of this work comes out from these knowledges, in which is realized theoretical design and it is in developmental environment of Borland Builder C++.

Klíčová slova

real-time přenos, RTP, RTCP, klient, server, data, jazyk C, jazyk C++, knihovna

Keywords

real-time transmission, RTP, RTCP, client, server, data, C language, C++ language, library

NOVOTNÝ, R. *Přenos zvukových dat pomocí protokolu RTP*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 57 s. Vedoucí bakalářské práce Ing. Jiří Kouřil.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma "Přenos zvukových dat pomocí protokolu RTP" jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské 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 3. 6. 2008 **……..**

(podpis autora)

SEZNAM POUŽITÝCH ZKRATEK A SYMBOLŮ

ACK	Acknowledgement
ATM	Asynchronous Transfer Mode
CRTP	Compressed Real-time Transport Protocol
CSRC	Synchronization Source
DNS	Domain Name System
ECRTP	Enhanced Compressed Real-time Transport Protocol
FDDI	Fiber Distributed Data Interface
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IPsec	IP security
ISO/OSI	International Standards Organization / Open Systems Interconnection
ITU	International Telecommunication Union
LAN	Local Area Network
MCU	Multipoint Control Unit
MPEG	Motion Picture Experts Group
PC	Personal Computer
PCM	Pulse-Code Modulation
POP	Post Office Protocol
QoS	Quality of Service
RFC	Request For Comments
RR	Receiver Report
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
RTSP	Real-Time Streaming Protocol
S/MIME	Secure Multipurpose Internet Mail Extensions
SDES	Source Description Items

SIP	Session Initiation Protocol
SMTP	Simple Mail Transport Protocol
SNMP	Simple Network Management Protocol
SR	Sender Report
SRTP	Secure Real-Time Transport Protocol
SSL	Secure Sockets Layer
SSRC	Synchronization Source
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
TELNET	TELecommunication NETwork
UDP	User Datagram Protocol
WAN	Wide Area Network

OBSAH

SEZNAM OBRÁZKŮ

1 Úvod

Tato bakalářská práce je zaměřena na přenos zvukových dat v reálném čase. Tedy na sadu protokolů TCP/IP a na jeho transportní vrstvu, která má přenos dat v popisu práce. Sada protokolů TCP/IP má pro transportní vrstvu vyhrazeny dva protokoly, a to protokoly UDP a TCP. Tyto protokoly ale nejsou přizpůsobeny přenosu takových dat, které mají multimediální charakter, a pro které je velmi podstatné, v jakém čase budou doručeny až na místo svého určení. V poslední době byl proto v rámci rodiny protokolů TCP/IP vyvinut další protokol, který vychází vstříc multimediálním přenosům - přesněji přenosům, které jsou citlivé na fungování v tzv. reálném čase. Tímto protokolem je protokol jménem RTP (Real-Time Transport Protocol).

Praktickým úkolem bakalářské práce je navrhnout a realizovat aplikaci, která bude přenášet zvuková data minimálně mezi dvěma počítači. Dále je pak zaměřena na nastudování knihoven v jazyce C a C++, které podporují přenos zvukových dat právě pomocí protokolu RTP. Tyto knihovny pak budou v aplikaci využity.

Druhá až pátá kapitola popisuje vrstvový model TCP/IP, transportní protokoly TCP a UDP. Jsou zde představeny protokoly pro přenos dat v reálném čase a zmíněny mechanismy zabezpečení tohoto druhu přenosu.

V šesté kapitole jsou popsány volně dostupné knihovny v jazyce C a C++, pomocí kterých je možno zrealizovat požadovanou aplikaci.

Sedmá kapitola se zabývá návrhem aplikace typu klient-server, která přenáší zvuková data minimálně mezi dvěma počítači.

Osmá kapitola je věnována realizaci aplikace typu klient-server. Je zde popsán princip této aplikace, která se skládá z dvou dílčích aplikací (klient a server). Z náhledů na tyto aplikace je vysvětleno jejich ovládání.

V závěru je zhodnocena bakalářská práce, uvedeny výhody a nevýhody sestrojené aplikace.

2 TCP/IP

Tato kapitola se zabývá rodinou protokolů TCP/IP, která si koncem minulého desetiletí vydobyla výlučné postavení díky své otevřenosti a Internetu. Odpovídající znalost tohoto dnes bezkonkurenčně nejrozšířenějšího síťového protokolu je nezbytným předpokladem k pochopení problematiky přenosu dat v reálném čase pomocí protokolu RTP. RTP protokol se řadí do aplikační vrstvy síťového modelu TCP/IP. Je takzvanou nadstavbou protokolu UDP, který je na úrovni transportní vrstvy. Kapitola čerpá z literatury [1], [2], [3].

Protokol je průmyslovým standardem, který vznikl v roce 1969 na základě projektu amerického ministerstva obrany pro komunikace v rozsáhlých počítačových sítích. TCP/IP je nejvýhodnější protokol pro budování rozměrné a robustní sítě na větší vzdálenosti se složitějším směrováním, případně pro napojení na Internet. TCP/IP lze použít i v jednodušších sítích. Jeho nevýhodou je složitější implementace.

Zkratka TCP/IP se skládá ze jmen dvou nejpoužívanějších protokolů:

- TCP (Transmission control protocol) je protokolem transportní vrstvy.
- IP (Internet protocol) je protokolem síťové vrstvy, obhospodařuje adresování.

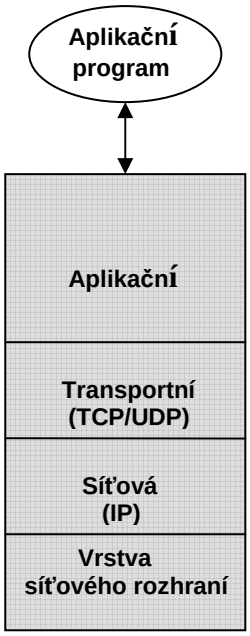
2.1 Vrstvový model TCP/IP

Vrstvový model popisuje, která činnost se má kde dělat. Rozděluje problém na dílčí úlohy, o které se starají jednotlivé vrstvy. Definuje rozhraní vrstev.

- Vrstva využívá služeb vrstvy nižší.
- Svoje služby každá vrstva nabízí vrstvě vyšší.
- Partnerem vrstvy N při vzájemné komunikaci je opět vrstva N.
- Spolupráce mezi entitami téže vrstvy je řízena komunikačními protokoly.

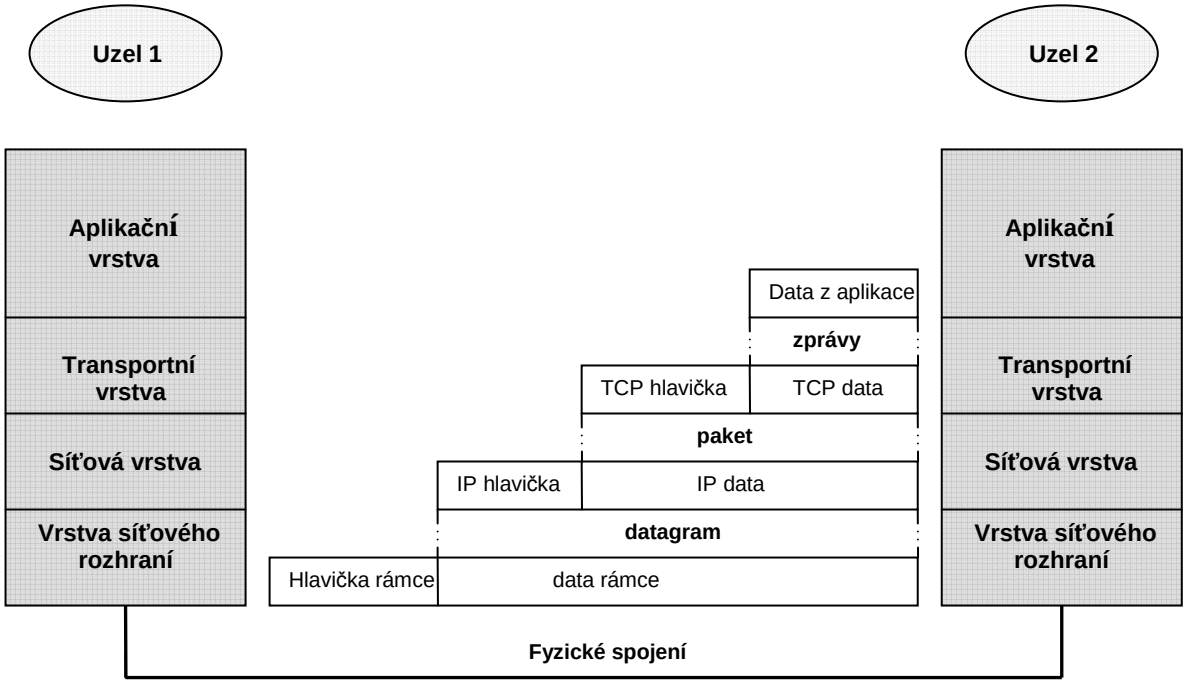
Model TCP/IP je 4 vrstvový:

- Aplikační – aplikační rozhraní
- Transportní (protokoly UDP a TCP) - předpokládá přímé spojení
- Síťová (Internet IP) – směrování
- Vrstva síťového rozhraní - přenos jednotlivých bitů



Obrázek 2.1: Síťový model TCP/IP

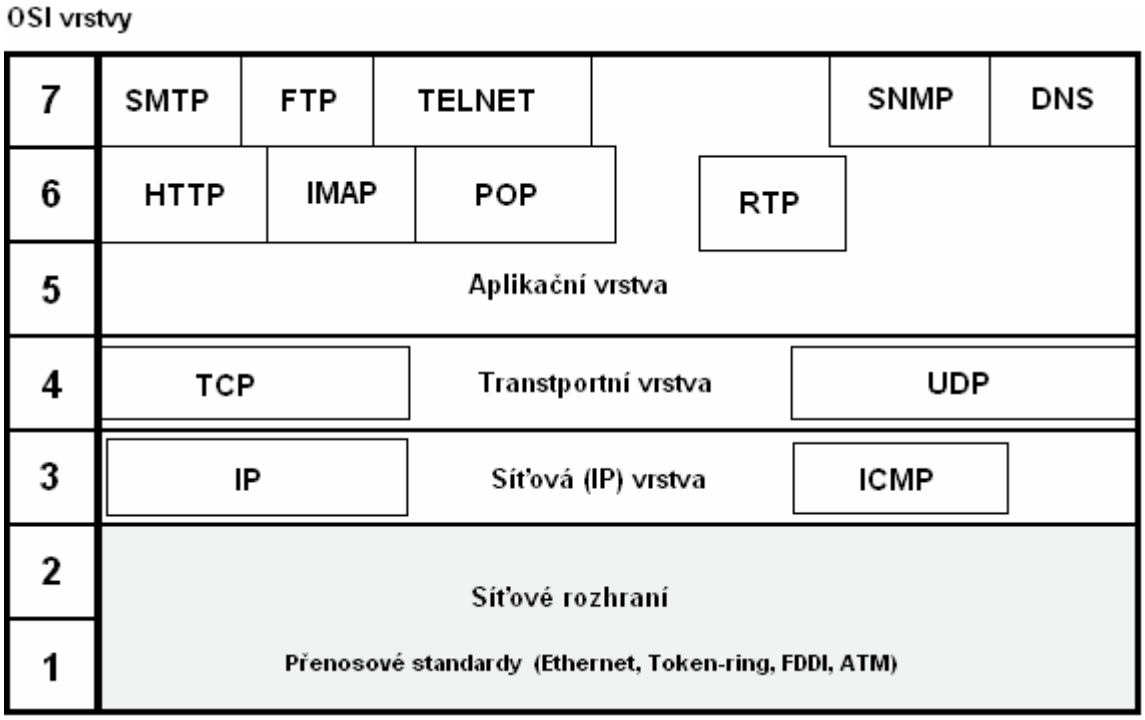
Na obr. 2.1 lze vidět jak každá vrstva při odesílání přidává svoji hlavičku.



Obrázek 2.2: Komunikace v modelu TCP/IP

2.2 Sada protokolů TCP/IP

Na obr. 2.3 nejsou uvedeny všechny protokoly, architektura TCP/IP jich zahrnuje mnohem více. Jak už bylo napsáno v kapitole 2.1 a jak vyplývá ze síťového modelu TCP/IP, sada protokolů architektury TCP/IP má čtyři vrstvy. Referenční model OSI/ISO se skládá ze sedmi vrstev.



Obrázek 2.3: Sada protokolů TCP/IP ve vrstvách architektury TCP/IP

2.2.1 Vrstva síťového rozhraní

Vrstva síťového rozhraní je vrstva nejnižší. Zajišťuje přístup k přenosovému médiu a řízení datového spoje. Podrobněji není v rámci TCP/IP architektury specifikována. Její implementace závisí na přenosové technologii, kterou používá příslušná síť, ať již je to přenosová technologie LAN (např. Ethernet), nebo přenosová technologie WAN (např. ATM). Znamená to, že TCP/IP může být implementována na jakémkoliv typu sítě z hlediska používané přenosové technologie, tzn. na sítích heterogenních.

2.2.2 Síťová vrstva

Internet Protocol (IP protokol) prakticky odpovídá síťové vrstvě. IP protokol přenáší tzv. IP datagramy mezi vzdálenými počítači. Každý IP datagram ve svém záhlaví nese adresu

příjemce, což je úplná směrovací informace pro dopravu IP datagramu k adresátovi. Takže síť může přenášet každý IP datagram samostatně. IP datagramy tak mohou k adresátovi dorazit v jiném pořadí, než byly odeslány.

Každé síťové rozhraní v rozsáhlé síti Internet má svou celosvětově jednoznačnou IP adresu (jedno síťové rozhraní může mít více IP adres, avšak jednu IP adresu nesmí používat více síťových rozhraní). Internet je tvořen jednotlivými sítěmi, které jsou propojeny pomocí směšovačů. Směrovač se anglicky nazývá router, můžeme se ale setkat i s označením gateway.

Protokol ICMP (Internet Control Message Protocol) podporuje protokol IP v tom smyslu, že informuje vysílací (zdrojový) uzel o určitých nekorektních situacích, které nastaly v průběhu přenosu datagramu. Jedná se tedy o zprávy režijní, které se v případě korektních přenosů nevysílají.

2.3 Transportní vrstva

Protokoly TCP a UDP odpovídají transportní vrstvě, která je vrstvou třetí. Protokol TCP dopravuje data pomocí TCP segmentů, které jsou adresovány jednotlivým aplikacím. Protokol UDP dopravuje data pomocí tzv. UDP datagramů.

Protokoly TCP a UDP zajišťují spojení mezi aplikacemi běžícími na vzdálených počítačích. Protokoly TCP a UDP mohou zajišťovat i komunikaci mezi procesy běžícími na témže počítači.

Rozdíl mezi protokoly TCP a UDP spočívá v tom, že protokol TCP je tzv. spojovanou službou, tj. příjemce potvrzuje přijímaná data. V případě ztráty dat (ztráty TPC segmentu) si příjemce vyžádá zopakování přenosu. Protokol UDP přenáší data pomocí datagramů (obdoba telegramu), tj. odesílatel odešle datagram a už se nezajímá o to, zda-li byl doručen.

Adresou je tzv. port. Pro pochopení rozdílu mezi IP adresou a portem se používá srovnání s poštovní adresou. IP adresa odpovídá adrese domu a port jménu a příjmení osoby, které má být dopis doručen.

Transportní protokoly budou dále rozebírány v kapitole 3.

2.3.1 Aplikační vrstva

Aplikační vrstva předepisuje v jakém formátu a jak mají být data přebírána/předávána od aplikačních programů. Aplikační protokoly TCP/IP odpovídají několika vrstvám ISO/OSI.

Absence prezentační vrstvy v síťovém modelu TCP/IP se řeší zavedením specializovaných „prezentačních-aplikačních“ protokolů, jako jsou protokoly SSL a S/MIME specializující se na zabezpečení dat.

Aplikačních protokolů je velké množství. Z praktického hlediska je lze rozdělit na:

- Uživatelské protokoly, které využívají uživatelské aplikace (např. pro vyhledávání informací v Internetu).
- Služební protokoly, tj. protokoly se kterými se běžní uživatelé Internetu nesetkají. Tyto protokoly slouží pro správnou funkci Internetu. Jedná se např. o směrovací protokoly, které používají směrovače mezi sebou, aby si správně nastavily směrovací tabulky. Dalším příkladem je protokol SNMP, který slouží ke správě sítí.

3 Transportní vrstva, transportní protokoly

V následující kapitole jsou popsány dva nejdůležitější protokoly z transportní vrstvy. Těmito protokoly jsou UDP a TCP. Pro problematiku přenosu zvukových dat pomocí protokolu RTP je důležitý protokol UDP. Pro přenos v reálném čase se protokol TCP nevyužívá. Informace z této kapitoly byly získány z literatury [2], [3].

Jak už bylo napsáno v předešlé kapitole, třetí vrstva TCP/IP je označována jako transportní vrstva, nebo též jako TCP vrstva, neboť je nejčastěji realizována právě protokolem TCP. Hlavním úkolem této vrstvy je zajistit přenos mezi dvěma koncovými účastníky, kterými jsou v případě TCP/IP přímo aplikační programy (jako entity bezprostředně vyšší vrstvy). Podle jejich nároků a požadavků může transportní vrstva regulovat tok dat oběma směry, zajišťovat spolehlivost přenosu, a také měnit nespojovaný charakter přenosu (v síťové vrstvě) na spojovaný. Přestože je transportní vrstva TCP/IP nejčastěji zajišťována právě protokolem TCP, není to zdaleka jediná možnost. Dalším používaným protokolem na úrovni transportní vrstvy je například protokol UDP, který na rozdíl od TCP nezajišťuje mj. spolehlivost přenosu - samozřejmě pro takové aplikace, které si to (na úrovni transportní vrstvy) nepřejí.

3.1 Protokol TCP (Transmission Kontrol Protocol)

Protokol TCP je spojovanou službou (connection oriented), tj. službou která mezi dvěma aplikacemi naváže spojení – vytvoří na dobu spojení virtuální okruh. Tento okruh je plně duplexní (data se přenášejí současně na sobě nezávisle oběma směry). Přenášené bajty jsou číslovány. Ztracená nebo poškozená data jsou znovu vyžádána. Integrita přenášených dat je zabezpečena kontrolním součtem.

3.1.1 TCP segment

Základní jednotkou přenosu v protokolu TCP je TCP segment. Někdy se také říká TCP paket. Aplikace běžící na jednom počítači posílá protokolem TCP data aplikaci běžící na jiném počítači.

Zdrojový port (source port) je port odesílatele TCP segmentu, cílový port (destination port) je portem adresáta TCP segmentu. Pětice: zdrojový port, cílový port, zdrojová IP adresa, cílová IP adresa a protokol (TCP) jednoznačně identifikuje v daném okamžiku spojení v Internetu.

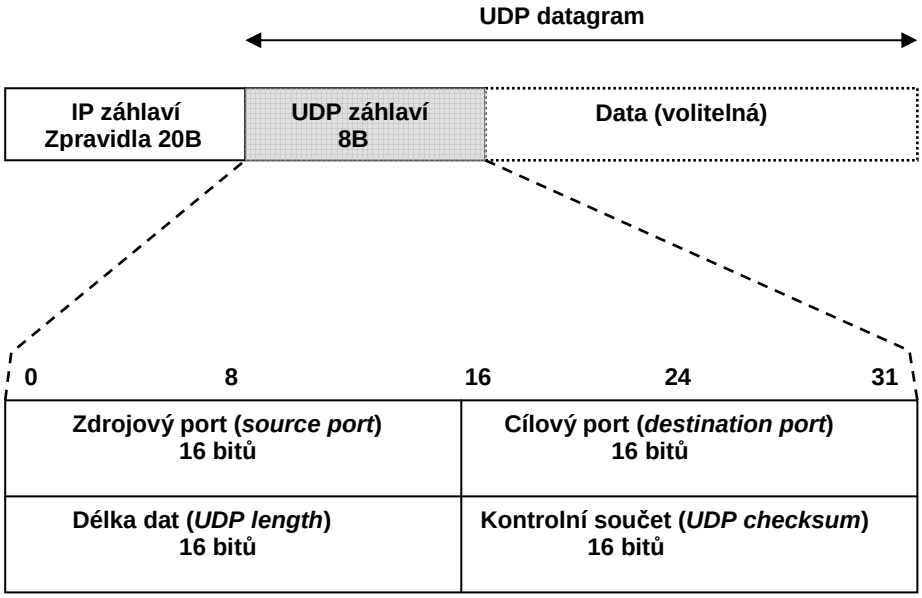
3.2 UDP protokol (User Datagram Protocol)

Protokol UDP je jednoduchou alternativou protokolu TCP. Protokol UDP je nespojovaná služba (narozdíl od protokolu TCP), tj. nenavazuje spojení. Odesílatel odešle UDP datagram příjemci a už se nestará o to, zda-li se datagram náhodou neztratil (o to se musí postarat aplikační protokol). UDP datagramy jsou baleny do IP-datagramu.

UDP jako jednodušší transportní protokol nezajišťující spolehlivé doručení paketů je vhodnější transportní mechanizmus pro provoz v reálném čase. Ale protože se mu nedostává některých vlastností potřebných pro specifika přenosu v reálném čase, je potřeba ještě další doplňující protokol nad UDP, a to RTP.

3.2.1 UDP datagram

Záhlaví UDP protokolu je velice jednoduché, viz obrázek 3.1. Obsahuje čísla zdrojového a cílového portu – což je zcela analogické protokolu TCP. Je třeba dodat, že čísla portů protokolu UDP nesouvisí s čísly portů protokolu TCP. Protokol UDP má svou nezávislou sadu čísel portů.

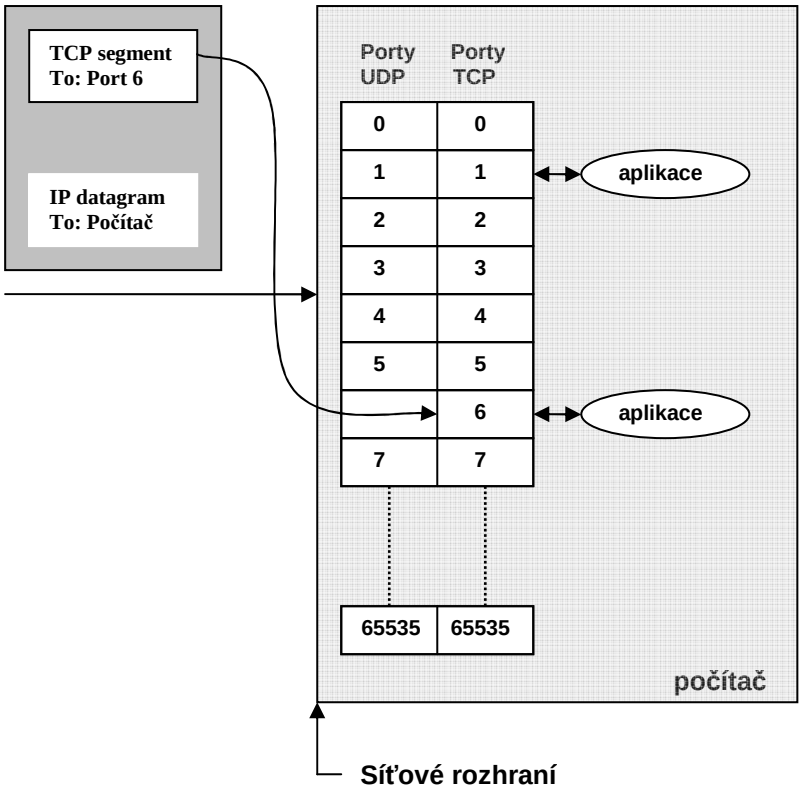


Obrázek 3.1: Záhlaví UDP datagramu

Pole délka dat obsahuje délku UDP datagramu (délku záhlaví + délku dat). Minimální délka je tedy 8, tj. UDP datagram obsahující pouze záhlaví a žádná data. Pole kontrolní součet nemusí být povinně vyplněné. Výpočet kontrolního součtu je tak v protokolu UDP nepovinný.

3.3 Porty TCP a UDP

Konce spojení ("odesílatel" a „adresát") jsou určeny tzv. číslem portu. Toto číslo je dvojbajtové, takže může nabývat hodnot 0 až 65535. U čísel portů se často vyjadřuje okolnost, že se jedná o porty protokolu TCP tím, že se za číslo napíše lomítko a název protokolu (tcp). Pro protokol UDP je jiná sada portů než pro protokol TCP (též 0 až 65535), tj. např. port 53/tcp nemá nic společného s portem 53/udp. Cílová aplikace je v Internetu adresována (jednoznačně určena) IP adresou, číslem portu a použitým protokolem (TCP nebo UDP). Protokol IP dopraví IP datagram na konkrétní počítač. Na tomto počítači běží jednotlivé aplikace. Podle čísla cílového portu operační systém pozná které aplikaci má TCP segment doručit.



Obrázek 3.2: Porty UDP a TCP

4 Protokoly pro přenos v reálném čase

Ve čtvrté kapitole je popsána problematika přenosu dat v reálném čase. Kapitola obsahuje protokoly, které přenos v reálném čase zaručují. Následně je s těmito protokoly seznámeno. Čerpáno z literatury [4], [5], [6].

Přenos datového souboru je sekvenční předávání jednotlivých paketů, do kterých je původní soubor určený k přenosu rozdělen, od vysílacího bodu do bodu cílového mezi směrovači tvořícími přenosovou trasu. Úkolem protokolu IP vrstvy síťové je:

- Vytvoření paketů na vysílací straně.
- Určení přenosové trasy.
- Znovusestavení paketů do originálního souboru na straně cílové.

U přenosů v reálném čase, tj. u přenosů proudových je nezbytně nutné ještě zajistit rovnoměrnost toku paketů, což znamená, že časové zpoždění mezi jednotlivými pakety má být konstantní. Jestliže se časové odstupy mezi jednotlivými přijímanými pakety výrazně mění, aplikace pracující v reálném čase má nekvalitní výstup nebo dokonce může dojít k jejímu selhání. Zachování stejných časových odstupů mezi jednotlivými pakety multimediálního proudu je zásadním požadavkem na multimediální přenos v reálném čase. Naproti tomu je připuštěna určitá tolerance pokud se jedná o ztráty některých z paketů, vytvářejících datový proud.

Multimediální data, jak již bylo uvedeno, jsou značně redundantní z hlediska lidského vnímání zvuku a obrazu. Pokud tedy během přenosu proudu paketů dojde k "rozumným" ztrátám, interpretovaná informace (tj. zvuk nebo obraz) na přijímací straně se může uživateli zdát méně kvalitní, nicméně akceptovatelná. Pokud ale dojde k porušení časového sledu paketů, obraz se "roztrhá" a zvuk ztrácí zcela srozumitelnost.

U sítí architektury TCP/IP, mezi něž náleží Internet, není možno zajistit základními protokoly síťové a transportní služby časově rovnoměrný transport paketů, příslušejících určitému datovému proudu. Pro přenosy v reálném čase byly vyvinuty protokoly aplikační vrstvy, které tvoří podporu aplikacím pro tento druh přenosu. Mezi tyto protokoly patří především:

- protokol RTP (Real-time Transport Protocol),
- protokol RTCP (Real-time Transport Control Protocol),
- protokol RTSP (Real-time Streaming Protocol).

4.1 Protokol RTP (Real-time Transport Protocol)

Původní standard TCP/IP měl pouze dva transportní protokoly: TCP a UDP. S příchodem multimédií se ukázalo, že tyto dva nestačí, a proto byl vytvořen protokol pro přenos v reálném čase RTP. RTP je vlastně jakási nadstavba nad UDP. RTP je protokol zajišťující na bázi IP protokolu podporu pro real-time multimediální přenosy. Nezaručuje doručení dat ani správné pořadí jednotlivých paketů, ale definuje jejich pořadová čísla, podle kterých mohou multimediální aplikace rozpoznat chybějící pakety. K multimediálnímu obsahu připojuje záhlaví, které kromě pořadového čísla paketu a jeho časové známky (pro časovou rekonstrukci datového proudu) obsahuje rovněž označení typu obsahu, tj. informaci o formátu multimediálního souboru, který tvoří obsah paketu. RTP protokol byl navržen jak pro přenosy multicastové, tak i pro přenosy typu unicast, a to pro jednosměrný i dvousměrný přenos. Je tedy použitelný pro aplikace videokonferenční i pro internetovou videotelefonii.

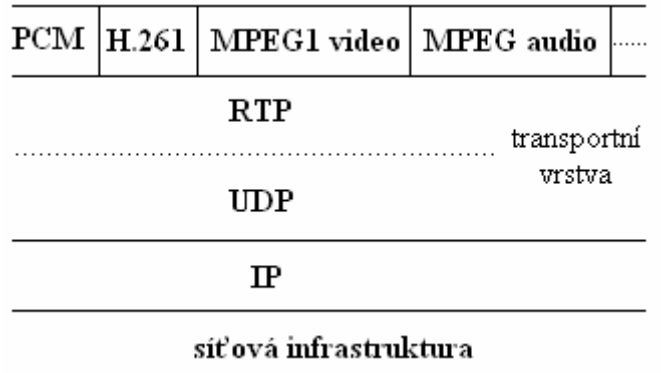
4.1.1 Druhy přenosů

- **Unicast** – odesílatel a příjemce komunikují na principu bod-bod. Datové pakety jsou adresovány pouze příjemci a žádný jiný počítač na síti je nebude muset zpracovávat.
- **Multicast** – síťová komunikace mezi jedním odesílatelem a několika příjemci. Technologie multicast se používají pro snížení zátěže sítě, když se na jeden zdroj videa (např. třeba i zvuku) chce podívat více uživatelů, protože dodávají stejný datový tok třeba i stovkám příjemců. Největší rozdíl v porovnání s unicastem je v tom, že video stream je třeba odeslat pouze jednou. Multicasting (IP Multicasting) je běžně používán ve spojení s RTP přenosy.
- **Broadcast** – vysílání od jednoho odesílatele všem bodům v síti. V LAN síti, jsou broadcasty obvykle omezené na určitý síťový segment a nepředstavují praktický způsob vysílání videa po síti.

Protokol RTP využívá transport protokolu UDP. Formát datagramu obsahujícího RTP paket je znázorněn na obrázku 4.1.

Obrázek 4.1: Datagram s paketem protokolu RTP

Formát RTP datagramu je jednoduchý a obecný, takže vyhovuje všem aplikacím pracujících v reálném čase (existuje pouze jeden typ zprávy RTP). Mezi typy dat RTP zprávy jsou G.721 audio, GSM audio, G.722 audio, MPEG audio, G.728 audio, H.261, MPEG-1 a MPEG-2 video.



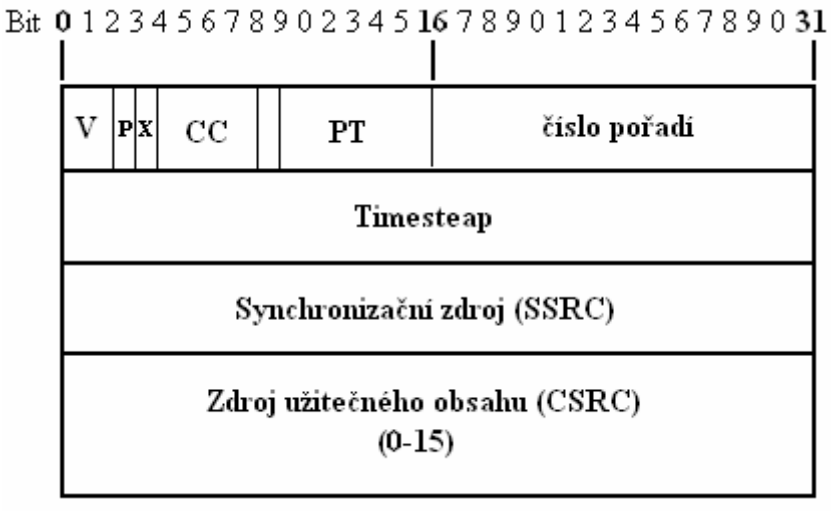
Obrázek 4.2: Protokolová architektura s RTP

4.1.2 Architektura RTP

Vytvoření RTP spoje je vlastně asociací skupiny aplikací, komunikujících s RTP. Spoj je identifikován síťovou adresou a párem portů. Jeden port je určen pro přenos dat a druhý port je určen pro RTCP data.

Účastníkem je jeden stroj, hostitel nebo uživatel účastnící se spojení. Účastí ve spojení může být jednak pasivní příjem dat, vysílání dat nebo dokonce obojí, tj. příjem i vysílání.

Každý rozdílný typ dat je přenášen jiným spojem. Například, pokud je při videokonferenci přenášen zvuk i obraz zároveň, je jeden spoj určen pro přenos audio dat a druhý spoj pro přenos video dat. To umožňuje účastníkovi výběr typu dat, který chce přijímat, např. pokud je někdo v místě s nízkou šířkou pásma, může zvolit pouze příjem audio dat z konference.



Obrázek 4.3: Formát záhlaví datového paketu RTP

Záhlaví datového paketu RTP obsahuje:

- Verzi RTP protokolu (**V**): 2 bity
- Doplnění (**P**): 1 bit. Pokud je tento bit nastaven, je na konci paketu jeden nebo více bytů, které nejsou součástí užitečného obsahu. Úplně poslední byte v paketu indikuje počet doplněných bytů. Doplnění je využíváno některými šifrovacími algoritmy.
- Rozšíření (**X**): 1 bit. Jestliže je tento bit nastaven, následuje za pevným záhlavím jedno rozšíření záhlaví. Tento mechanismus umožňuje vložení rozšiřujících informací do záhlaví RTP.
- Počet CSRC (**CC**): 4 bity. Počet CSRC identifikátorů, které následují za pevným záhlavím. Jestliže je počet SCRC nula, je zdrojem synchronizace zdroj užitečného obsahu.
- Záložka (**M**): 1 bit. Záložkový bit, definovaný konkrétním profilem média.
- Typ užitečného obsahu (**PT**): 7 bitů. Index z tabulky profilu média, který popisuje formát užitečného obsahu.
- Číslo pořadí: 16 bitů. Jedinečné číslo paketu, které popisuje pozici paketu v pořadí paketů. Číslo paketu je inkrementováno po každém odeslaném paketu.
- TIMESTAMP: 32 bitů. Vyjadřuje moment odebrání vzorku prvního bytu užitečného obsahu.
- SSRC: 32 bitů. Identifikuje synchronizační zdroj. Jestliže je počet CSRC roven nule, je zdroj užitečného obsahu zdrojem synchronizace. Jestliže je CSRC nenulové, SSRC identifikuje mixér.

- CSRC: 32 bitů každý. Identifikuje zdroje přispívající do užitečného obsahu. Počet přispívajících zdrojů je určen polem počtu CSRC (**CC**). Celkem může být 16 přispívajících zdrojů. Jestliže je více přispívajících zdrojů, je výsledný užitečný obsah sloučením těchto zdrojů.

4.1.3 Komprese RTP

Protokol RTP může úspěšně využít komprese při přenosu videa, jednak v datové části zprávy (RFC 2435 řeší formát dat RTP pro komprimované video v JPEG), a jednak v záhlaví (RFC 2508 specifikuje kompresi záhlaví IP/UDP/RTP pro pomalé spoje).

Komprimovaný RTP (CRTP, Compressed RTP) je určen pro omezení objemu záhlaví datových jednotek IP, UDP a RTP. Protokol je velmi účinný zejména na spolehlivých a rychlých dvoubodových spojích. V méně spolehlivých sítích s dlouhými zpožděními, ztrátami paketů a doručením paketů mimo pořadí ale CRTP nepracuje ideálně pro aplikace IP telefonie. Pro tento případ se používá jeho vylepšení ve formě ECRTP (Enhanced CRTP; RFC 3545).

4.1.4 Vlastnosti RTP

- RTP zajišťuje spojení mezi koncovými body pro real-time data.
- RTP neposkytuje žádný mechanismus, který zaručí doručení dat a jejich doručení včas a pro tohle potřebuje podporu nižších vrstev.
- RTP neví o síti pod sebou víc než že přenáší data v rámcích - je tedy možné jej implementovat i na další protokoly jako ATM AA5 a IPv6.
- RTP poskytuje časová razítka a číslování, pro sestavení dat.
- RTP/RTCP poskytuje mechanismu zjišťování a řízení parametrů spojení na základě informací z vysílačů a přijímačů.
- RTP je pouze rámcem, který slouží pro implementace jednotlivých aplikací a je otevřený pro nové formáty dat.

4.2 Protokol RTCP (Real-time Transport Control Protocol)

Zatímco RTP neposkytuje žádný mechanismus k zajištění včasného doručení nebo k poskytnutí záruky jiné kvality služeb (QoS), jsou tyto mechanismy poskytovány kontrolním protokolem (RTCP), který umožňuje sledování kvality distribuce dat. RTCP také poskytuje kontrolní a identifikační mechanismus pro přenosy RTP.

RTCP je řídící protokol určený pro spolupráci s protokolem RTP aby přenášel mezi účastníky spojení informace o jeho kvalitě. Je standardizován v RFC 1889 a RFC 1890. RFC 1889 definuje tyto typy RTCP paketů:

- **RR** (receiver report) - Jsou generovány účastníky, kteří nevysílají a obsahují informace o doručovaných datech jako nejvyšší číslo doručeného paketu, počet ztracených paketů, kolísání zpoždění paketů při průchodů sítí (jitter) a časové známky k výpočtu času doručení (round-trip time).
- **SR** (sender report) - Jsou generovány aktivními vysílači a jejich součástí oproti RR je navíc ještě sekce informací o vysílači, která obsahuje informace o inter-mediální synchronizaci, celkové počítadla paketů a zaslaných dat.
- **SDES** (source description items) - Obsahují informace popisující zdroje dat.
- **BYE -** Odhlášení se účastníka.
- **APP -** Specifická funkce aplikace.

RTCP pakety jsou slučitelné a jsou zasílány jako skupina minimálně dvou paketů (report paket a paket popisu zdroje).

4.3 Protokol RTSP (Real-time Streaming Protocol)

RTSP podporuje efektivní průběh proudových multimediálních přenosů v sítích TCP/IP. Jedná se o aplikační protokol typu klient-server, který definuje mezi stranami klient a server:

- Metody pro vytvoření spojení.
- Metody pro správu spojení.
- Metody pro odesílání požadavků na určité multimediální objekty.

Protokol RTSP lze po stránce funkčnosti přirovnat k protokolu HTTP. Protokol RTSP zajišťuje transport multimediálních zdrojů podobně jako HTTP zajišťuje transport HTML dokumentů.

5 Zabezpečení přenosu dat v reálném čase

Tato kapitola chce informovat o tom, že je tu jistá problematika bezpečnosti přenosu dat v reálném čase. Jsou zde uvedeny druhy útoků na přenášená data v reálném čase a mechanismy jak bezpečnosti docílit. Informace byly získány z literatury [7], [8].

Transportní a signalizační protokoly neměly v původní podobně žádné bezpečnostní mechanismy, které by zajišťovali, že se útočníci nedostanou k obsahu přenášených dat nebo k autentizačním údajům. Je nutné, aby informace, které si klienti mezi sebou vyměňují, měly tři základní vlastnosti, a to autentičnost, důvěryhodnost a integritu.

5.1 Druhy útoků

Útoky na obsah přenášených dat nebo na autentizační údaje mohou být následující:

- **Falešné informace** - útoky, které při komunikaci vytvářejí zavádějící nebo falešné informace. Jedná se o doručování zpráv, které obsahují chybnou informaci o identitě, oprávnění nebo obsahu.
- **Přerušení služeb** - útočník se snaží přerušit spojení např. zničením hardware zařízení, vymazáním programu, či úmyslným zahlcením sítě.
- **Nevyžádané spojení** - nevyžádané spojení je jakékoliv spojení, které vyžaduje předchozí potvrzení nebo které obejde zamítnutí.
- **Odposlouchávání** - jedná se o útoky, kdy se útočník snaží zachytávat signalizační nebo multimediální data a následně analyzovat jejich obsah. Při tomto útoku nedochází k modifikaci přenášených dat.
- **Zachytávání a modifikace** - při tomto útoku získá neoprávněná osoba nejen přístup k přenášeným mediálním a signalizačním datům, ale také tohoto přístupu využije k pozměnění přenášených dat.
- **Krádež služby** - útoky zaměřené na modifikaci účtovacích údajů na straně poskytovatele služeb. Zahrnují i nezákonné používání placených služeb.
- **Zneužití služeb** – jedná se o útoky, které zneužívají komunikačních služeb.

5.2 Zabezpečení RTP streamů

Hlasové (ale i obrazové) streamy se v IP sítích přenášejí pomocí protokolu RTP, který jako transportní protokol používá UDP. Aby měla vysílací strana informace o kvalitě přijímaného proudu, příjemce v pravidelných intervalech odesílá pomocí RTCP zprávy obsahující potřebné informace parametrech spojení. Protože audio a video přenosy jsou velmi citlivé na zpoždění (delay) a kolísání zpoždění (jitter), nesmí žádná metoda, kterou budeme zabezpečovat data (ať již z hlediska utajení nebo integrity) nijak významně ovlivňovat tyto parametry (tzn. zpoždění a jitter). Existují tři standardizované kryptografické protokoly, které vyhovují tomuto zadání. Jedná se o protokoly SRTP, IPsec a zRTP.

5.2.1 Protokol SRTP (Secure Real-Time Transport Protocol)

Tento protokol představuje rozšíření RTP. Základním cílem je pro RTP a RTCP pakety zajistit utajení, autentizace a ochranu proti teplat útokům (útok opakovaným přehraním zpráv). O utajení se stará moderní algoritmus AES v režimu CTR, který nijak nezvětšuje velikost přenášených dat. Data jsou na druhé straně zvětšena o autentizační hlavičku, která každý paket navýší o 10B.

5.2.2 ZRTP (Z Real-time Transport Protokol)

Protokol ZRTP nabízí mechanizmus vytvářející vysokou úroveň entropie při výměně klíčů (Diffie-Hellmanův algoritmus) na základě výpočtu hashí několika utajovaných hodnot. A to včetně krátkého autentizačního řetězce, který je nahlas přečten volajícími stranami. Spočtená společná sdílená utajovaná hodnota je použita pouze jednou, její část je však uložena pro budoucí spojení.

5.2.3 Sada protokolů IPsec (IP security)

Namísto použití protokolu SRTP nebo ZRTP lze k zabezpečení přenosu multimediálních dat použít sadu protokolů IP security (IPsec), které zabezpečují komunikaci na úrovni síťové vrstvy, tj. zabezpečují jednotlivé IP datagramy. Tyto protokoly jsou nezávislé na vyšších protokolech TCP/UDP. IPsec vznikl jako součást IPv6, ale později byl navržen i pro IPv4.

IPsec definuje dva režimy zabezpečení IP datagramů:

- Transportní režim - šifruje pouze datovou část IP datagramu.
- Tunelovací režim - V tunelovacím režimu je zapouzdřen a chráněn celý IP datagram.

6 Knihovny v C/C++ pro přenos zvukových dat pomocí protokolu RTP

V kapitole číslo sedm jsou rozepsány knihovny pro dané téma. Zvláště knihovna s názvem RTPlib je podrobněji rozepsána a na jejím případě naznačen princip inicializace spojení a následný přenos RTP dat. V této kapitole jsou rovněž uvedeny výhody a nevýhody jednotlivých knihoven. Informace k sepsání kapitoly byly získány výhradně z dokumentací vybraných knihoven. Literatura [9], [10], [11], [12], [13], [14].

6.1 Knihovna RTPlib

Knihovna s názvem RTPlib je knihovnou, která poskytuje rozhraní velmi vysoké úrovně pro vyvíjející se aplikace, které využívají protokol pro přenos dat v reálném čase. Knihovnu sepsali Henninge Schulzrinne a Ping Pan (Columbia University) a je volně dostupná.

6.1.1 Inicializace

Knihovna obsahuje velký počet funkcí. Jednou z nich je RTPCreate(), která stanovuje kontext. Většina knihovních funkcí považuje kontext za první argument. Jakmile je RTPCreate vyzvána inicializovat spojení, adresy pro spojení musí být už nastaveny. Knihovna podporuje typy přenosů: unicast, multi-unicast, multicast a hybrids. Jsou zde dvě nastavení adres, které knihovna RTPlib využívá. První je „send set“. Je to přehled unicastových a multicastových adres, čísel portů a ttl hodnot (ttl hodnoty jsou definovány pouze pro multicast). Jakmile si aplikace vyžádá poslat paket, knihovna bude přenášet tento paket na všechny adresy zaznamenané právě v „send set“. S tímto počítá unicast (tím, že dá jednu unicast adresu / hodnotu portu), multicast (tím, že dá jednu multicast adresu / portový pár), multi-unicast (nastavením více unicast adres / portových párů), a hybrids.

Knihovna má také „set defined“. Tato jednotka obsahuje jednu adresu / párový port, který může být typu multicast nebo unicast. Je to adresa, na kterou knihovna očekává, že přijme pakety. Chování je definováno takto:

- Jestliže adresa je unicastová, ale neodpovídá místnímu rozhraní, knihovna místo toho váže k INADDR_ANY, pakety se budou přijímat na jiném rozhraní.
- Jestliže adresa je unicastová a odpovídá místnímu rozhraní, knihovna bude jen přijímat pakety na tomto rozhraní.

- Jestliže je multicastová, knihovna se bude vázat k INADDR_ANY a pak se připojí k multicastové skupině. Touto cestou se budou přijímat balíky, které jsou jednak unicastové nebo multicastové pro daný port.
- Jestliže adresa je nulová, knihovna se bude vázat k INADDR_ANY.
- Jestliže port je nulový, knihovna bude používat dynamický port. RTP a RTCP porty musí být nepřetržité, knihovna bude zkoušet náhodné páry dynamicky přiřazených portů (nahoře 49152) než se podaří najít pár. Jestliže pár nemůže být rozdělen, vrátí chybu.

Všechny adresy jsou specifikovány jako řetězce. Musí mít formu "A.B.C.D", nebo „hostname". Jestliže řetězec je „hostname", knihovna se bude pokoušet přidělit název k adrese používané DNS.

Je zde velký počet i dalších inicializačních funkcí. Většina je nepovinná. Některé je ale obecně potřeba využít:

- RTPSessionSetBandwidth (). RTCP pakety jsou posílány rychlostí, která závisí na šířce pásma. Je to globální parametr.
- RTPMemberInfoSetSDES (). RTCP paketový typ, SDES, obsahuje informaci o každém uživateli. Uvede jméno, e-mail, a CNAME (název domény, kam se chce uživatel připojit). Je to povinnost aplikace dát tyto pole. Zvláště CNAME pole musí být vyplněno ještě předtím než se volá RTPOpenConnection.

Jakmile jsou uvedeny adresy spojení, je zavolána funkce RTPOpenConnection(). Tímto je možno zajistit připojení k multicastovým skupinám. Po připojení je knihovna připravena přijímat a odesílat pakety.

6.1.2 Odesílání a přijímání paketů

Posílání paketů je poměrně přímočaré. Funkce RTPSend() se používá k tomu, aby sdělila knihovně, že má být poslán RTP paket. Knihovna vezme vyrovnávací paměť, přidá RTP záhlaví, vykoná požadované operace a pošle paket. Knihovna automaticky pošle RTCP pakety. Počáteční časová značka a číslo sekvence jsou vybrány náhodně.

Přijímání paketů je trochu více komplexní. Musí se vědět, jestli je paket dostupný pro čtení. Protože knihovna nenařizuje toto opatření, je na uživateli stanovit, která data jsou

dostupná pro čtení. Knihovna dovoluje uživatelský přístup k přijmutí soketů. Jsou dva: jeden pro RTP, jeden pro RTCP, funkce RTPSessionGetRTPSocket a RTPSessionGetRTCPSocket.

Když je paket součástí soketu, aplikace by měla volat funkci RTPReceive (). Tato funkce převezme kontext, ukazatel na vyrovnávací paměť a ukazatel na hodnotu délky. Hodnota délky by měla být inicializována k množství místa ve vyrovnávací paměti. Knihovna bude číst a zpracovávat RTP nebo RTCP balík. Pro RTCP to bude vykonávat celou sbírku statistik. Obecně už nebude od uživatele vyžadována další akce. Pro RTP paket bude knihovna také aktualizovat některé statistiky a proměnné. Vyrovnávací paměť bude vyplněná s RTP/RTCP paketem, včetně záhlaví. Jakmile je RTPReceive kompletní, uživatel může chtít zpřístupňovat data v balíku. Nicméně balík v originální vyrovnávací paměti používané v RTPReceive ještě obsahuje záhlaví. Získání přístupu k polím záhlaví se provádí pomocí funkce RTPGetRTPPacket (). Tato funkce považuje vyrovnávací paměť za vstup a vrací ukazatel na RTP paketovou strukturu. Tento ukazatel je typu rtp_packet, a obsahuje pole pro různá RTP záhlaví. Uživatel pak může zpřístupňovat tyto pole pro kterékoliv účely.

6.1.3 Šifrování

RTPlib podporuje šifrování RTP a RTCP paketů. Knihovna podporuje tři režimy:

- Žádné šifrován – výchozí hodnota. RTP nebo RTCP pakety nejsou zašifrovány.
- Plné šifrování - RTP a RTCP pakety jsou úplně zašifrované.
- Částečné šifrování - RTP a non-SDES RTCP pakety jsou zašifrovány. SDES RTCP pakety zašifrovány nejsou.

6.2 RTP tools

Knihovnu rtptools tvoří množství malých aplikací, které využívají RTP data. Nejnovější je verse 1.18. Autory této knihovny jsou Henning Schulzrinne, Dorgham Sisalem.

6.2.1 Instalace

RTP tools by měly jít přeložit pod Windows/NT/95/98/2000 (Win32). Knihovna byla testována i na SunOS 4.1, SunOS 5.x (Solaris), linux, NT 4.0, SGI Irix, a HP-UX. Pro Sun 4.1.x je nutné používat sun4 architekturu a pro SunOS 5.x architekturu sun5, pokud se daná architektura nepoužívá, bude vypsána chyba systémového hlášení. Pro unixové systémy musí

být na začátku vlastního programu vepsáno:

```
./configure; make
```

Pokud je zamýšleno instalovat RTP tools na WIN32 počítači, je nutné při instalaci následovat tyto kroky:

*. dsp soubory jsou soubory projektu, *. dsw soubor je soubor pracovní.

Uživatel může otevřít *.dsw soubor a použít „batch compile“ ke zkompilování všech projektů.

- Ve Visual C + + 6.0 otevřeme pracovní soubor rtptools.dsw.
- Ve VC menu Build využijeme „Batch Build“ k vytvoření všech nástrojů.
- Celý rtptools bude vytvořený pod adresářem "debug\".

Knihovna RTPtools lze zkompilovat i v programu Borland C++ builder. Ve složce bcb jsou soubory open dump_bcb.bpr, play_bcb.bpr, send_bcb.bpr and trans_bcb.bpr. Po spuštění těchto souborů je možné knihovnu zkompilovat tlačítkem F9.

Síťové adresy mohou být typu multicast nebo unicast. Mohou být specifikovány v soustavě (např., 224.2.0.1) nebo jako „jméno hostitele“ (např., lupus.fokus.gmd.de). Čísla portu musí být dána v rozsahu od 1 do 65535. Síťové adresy jsou specifikovány jako cíl/port/ttl. Time-to-live (ttl) hodnota je nepovinná a platí jen u multicastového vysílání.

6.2.2 Aplikace

Rtpplay - přehrává RTP pakety zaznamenané RTPdumpem.

RTPsend - tvoří RTP pakety z textovém popisu, vytvořený ručně nebo RTPdumpem.

RTPdump - rozebírá a vytvoří RTP pakety, tvoří výstupní soubory vhodné pro RTPplay a RTPsend.

RTPtrans - RTP překladatel mezi unicast a sítěmi multicast, také překládá mezi DPH a RTP formáty.

6.3 Knihovna jRTBlib

Knihovnu vytvořil Jori Liesenborgs z výzkumného institutu Hasseltovy univerzity. Knihovna jRTPlib má počátky již v roce 1999 a do roku 2008 byla celkem 27krát vylepšena až do současné verze jRTPlib-3.7.1, která se, jak sám autor uvádí, blíží k dokonalosti. Autor povoluje užití knihovny takřka k jakémukoliv účelu pod podmínkou, že nebude popřen skutečný autor knihovny, respektive, že se za něj nebude někdo vydávat.

Dokumentace knihovny jRTPlib je vytvořena pomocí univerzálního nástroje Doxygenu, což je program, který dokumentaci generuje z komentářů, umístěných přímo ve zdrojovém kódu projektu. Dokumentace je přehledná a velmi dobře se v ní orientuje.

JRTPlib je objektově orientovaná knihovna psaná v jazyce C++, která pomáhá vývojářům používat RTP tak, jak je popsaný v RFC 3550. Starší verze knihovny ze série 2.x jsou založeny na RFC 1889. Knihovna umožňuje uživateli posílat a přijímat data, používat RTP bez toho, aby se musel obávat SSRC kolize. Lze ji využít v platformách:

- GNU/Linux
- MS Windows (Win32 a WinCE)
- Solaris

Poskytuje několik tříd, které jsou užitečné k vytvoření RTP aplikace. Většina uživatelů bude potřebovat k sestavení vlastní aplikace třídu RTPSession. Je to třída, na které je postavena knihovna jRTPlib. Obsahuje funkce nutné pro posílání RTP dat, ovládá i RTCP. Není ale zrovna nejvhodnější pro vytváření aplikací, jejichž součástí je mixér nebo translátor. Mixér je RTP vysílač přijímající streamy RTP paketů z jednoho nebo více zdrojů, které následně kombinuje a posílá nové RTP pakety k jednomu či více cílům. Mixer může změnit datový formát. Translátor je zařízení, které produkuje z každého RTP paketu, jeden nebo více RTP paketů. Translátor může změnit formát dat nebo použít jakýkoli protokol nižší vrstvy k přenosu z jedné domény do druhé.

Naopak je knihovna nakloněna ke sestavování aplikací, kde je potřeba využít mimo jiné například packet schedule (plánovač paketů). Ten řídí zasílání proudů dat. K funkci využívá soubory front a další mechanismy, jako je například časovač, atd. Plánovač paketů musí být k dispozici v místech, kde jsou pakety řazeny do front. Ve většině případů je každý tok dat řešen samostatnou frontou. Plánovač pak řeší, jak s jednotlivými frontami zacházet.

Verze 3.7.1 podporuje IPv4 i IPv6.

6.4 Knihovna RTPi

RTP/I byla prvně pouze implementace protokolu RTP, postupem času se vyvinul samostatný protokol RTP/I (a stejnojmenná knihovna). Autory této knihovny jsou Volker Hilt, Juergen Vogel a Martin Mauve. Lze ji využít v platformách Windows a Linux. Knihovna je původně sepsaná pro programovací jazyk Java, ale existuje i verze pro jazyk C++. Licence GPL – zdrojové kódy „zdarma“.

6.5 Knihovna ccRTP

Autory této knihovny jsou Erik Eliasson a Johan. Knihovna je sepsána především pro uživatele využívající operační systém win32. Sepsána je v jazyce C++. Poslední verze knihovny je 1.5.0. Od verze 1.4.x podporuje knihovna IPv6. Podporuje přenos zvuků a videí. CcRTP knihovna využívá RTP tak, jak je popsaný v RFC 3550. Knihovnu lze kompilovat i pod operačním systémem Linux. Dokumentace knihovny je vytvořena pomocí Oxygenu.

Nejdůležitějšími vlastnostmi knihovny ccRTP jsou:

- Velké spektrum možností jak využít knihovnu k vytvoření RTP přenosu.
- Umožňuje sestavovat spojení typu multicast, multi-unicast a unicast.
- Poskytuje kontrolní funkce RTCP.
- Samočinně se zabývá SSRC kolizemi a vykonává detekci smyčky.
- Realizuje přehodnocení časovače a přehodnocení zpáteční rychlosti.
- Umožňuje šifrovat data.

6.6 Knihovna oRTP

Knihovna byla sepsána firmou Linphone, která knihovnu využívá ve svém video-telefonu. Je sepsána v jazyce C, což se dá brát jako nevýhoda oproti knihovnám, které jsou naprogramované v objektově orientovaném programování. Obsahuje plánovač pro odesílání paketů. Podporuje určité telefonní události popsané v RFC 2833. Je uvolněna pod licencí GPL – volně dostupné zdrojové kódy. Lze ji využít pod operačním systémem Windows i Linux.

Knihovna je z dokumentovaná za pomoci univerzálního programu Doxygonu.

7 Návrh aplikace typu klient-server

Kapitola číslo osm vychází z poznatků předešlých kapitol. Zabývá se návrhem aplikace typu klient-server, která přenáší zvuková data minimálně mezi dvěma počítači. Před samotným začátkem sepisování této aplikace bylo zapotřebí zvážit a vybrat, která volně dostupná knihovna nejvíce vyhovuje požadavkům na tuto aplikaci. Proto první část této kapitoly je zaměřena na výběr nejvhodnější knihovny sepsané v jazyce C nebo C++, která využívá protokol RTP.

Builder C++ ve svém základním režimu neoplývá množstvím knihoven, které dokáží pracovat se zvukem. Jako příklad je možno uvést komponentu TMediaPlayer, která se zvukem sice pracuje, ale pouze ve smyslu přehrávání určitého zvukového formátu. Neumí však číst data z paměti. Bylo proto potřeba nalézt volně dostupnou knihovnu, která číst data z paměti dokáže. Knihovna, která se zvukem tímto způsobem umí pracovat, je v této kapitole taktéž popsána. Je to knihovna s názvem FMOD.

Dále je zde rozebrána síťová architektura typu klient-server. Kapitola se zmiňuje i o vývojovém prostředí Borland C++ Builder, ve kterém je požadovaná aplikace, za využití popsaných knihoven, sepsána. Bylo čerpáno z literatury [15], [16].

7.1 Výběr nejvhodnější knihovny s implementovaným RTP

Při výběru se mj. vychází z toho, že aplikace, která využívá knihovnu s implementovaným protokolem RTP, má být navrhnuta pro platformu Windows a pro realizaci má být využito programovací prostředí C++ Builder. Tímto se okruh knihoven zmenšil. Všechny knihovny zmíněné v kapitole sedm jsou využitelné pro sestavení zadané aplikace. Ne ale všechny jsou si rovny.

RTP tools není ani tak knihovna, jako spíše soubor aplikací, které se dají využít ke sestavení vlastní aplikace. Proto bylo RTP tools, jako možnost k sestavení požadované aplikace, zavrhnuto. Jedna věc je využít knihovnu, druhá věc využívat celé programy způsobem například takovým, že by se ve zdrojovém kódě vlastního programu vytvořil nový proces, kterému by se „řeklo“, aby se spustil jiný program. Jedna aplikace za pomocí RTP data vysílá, druhá je přijímá, třetí je rozkóduje atd.

Důležitým faktorem při výběru knihovny byla i kvalita dokumentace dané knihovny. Bez dobré dokumentace není orientace v knihovně vůbec jednoduchá. U knihovny RTP/I nebyla

takřka žádná. Význam a použití některých tříd by se muselo pouze odhadovat. Proto i tato knihovna byla ze seznamu nejvhodnějších vyškrtnuta.

C++ Builder je kompletně postaven na tzv. objektově orientované architektuře, což byl jeden z hlavních důvodů toho, že za nejvhodnější knihovnu byla vybrána knihovna jRTPlib. Mj. pro ni hovoří tyto výhody:

- Free-software (licence GNU GPL) - tzn. volně přístupné zdrojové kódy.
- Přehledné rozhraní pro C++ - vše je přehledně zapouzdřeno do tříd.
- Dobře navržený systém tříd - dobrá orientace.
- Přítomnost velmi dobré dokumentace.
- Bezproblémová implementace knihovny do vlastního programu.

7.2 Knihovna Fmod pro práci se zvukovým formátem

Knihovna FMOD, aktuální verze je 3.4, od skupiny FireLight Multimedia je určitě jednou z nejlepších a nejznámějších zvukových knihoven. S její pomocí tu je možnost do daného programu přidat kompletní realistické ozvučení.

S knihovnou FMOD můžeme pracovat ve vývojovém prostředí:

- Microsoft Visual C++ 5 & 6
- Borland C++ Builderu
- Delphi, Kylixu
- Visual Basicu

Rozhraní je multiplatformní - lze jej „rozchodit“ v:

- Linuxu
- Unixu
- Windows 9x-XP
- WinCE/PocketPC
- konzoli Box

Podporuje standardy:

- DirectSound,
- DirectSound3D (DirectX 6.1, DirectX 7.0 a DirectX8),
- A3D 3.0
- EAX 2.0 i 3.0

Toto rozhraní je schopné pracovat se soubory typu MOD, S3M, XM, IT, MID, RMI, SGT, WAV, MP2, MP3, OGG, WMA, ASF.

7.2.1 Licence

FMOD je velice kvalitní zvukový systém, který lze používat na profesionální úrovni. Tomu také odpovídají licenční podmínky. Licence je výhodná pro ty, kdo plánují tvořit freewarové programy. Ti mají licenci zdarma. Dále pak ale:

- Pro shareware ve Win32, CE, LINUX program se platí 100USD (jeden sharewarový program), 250USD(neomezeně) .
- Pro plně komerční využití 1250 USD(jedna licence), 3200USD(neomezeně).
- Pro konzole se cena pohybuje od 5000USD do 17000USD.

7.3 Princip a charakteristika síťové architektury typu klient-server

Klient-server je síťová architektura, která odděluje klienta a server. Jednotliví klienti komunikují se serverem, který obvykle běží na vzdáleném počítači. Klasickou ukázkou může být prohlížení webových stránek, kde webový prohlížeč je klient, který při požadavku uživatele na novou stránku kontaktuje vzdálený server a vyžádá si od něj patřičnou webovou stránku.

Server je aplikace charakteristická tím, že na požádání poskytuje určité služby. Naproti tomu klientem je druhá, která na popud uživatele komunikuje se serverem a nechává si od něj vykonávat konkrétní služby. Na vztahu klienta a serveru je charakteristické zejména to, že aktivita je vždy na straně klienta, zatímco server je vysloveně pasivní. Své služby sám nikomu nevnucuje, ale pouze pasivně čeká, až o ně nějaký klient požádá. To také naznačuje další zajímavý aspekt vztahu klienta a serveru. Jeden server může sloužit více klientům, praktické omezení je většinou dáno pouze jeho výkonností.

Charakteristika serveru:

- Pasivní
- Čeká na požadavky od klienta
- Při přijetí požadavku jej obslouží

Charakteristika klienta:

- Aktivní
- Posílá požadavky serveru
- Čeká na odpovědi

Výpočetní model klient-server sice vzniknul v prostředí počítačových sítí, ale dnes je chápán obecněji, jako určitá filosofie tvorby počítačových aplikací - filosofie vycházející ze snahy rozdělit aplikaci na dílčí části s jasně vymezenými kompetencemi a úkoly. Dnes již není vůbec nutné, aby server i klient běželi na různých uzlových počítačích a komunikovali prostřednictvím sítě. Klient i server mohou být provozovány i na jednom počítači.

7.4 Vývojové prostředí Borland C++ Builder

C++ Builder je vývojový nástroj, který umožňuje jednoduše navrhovat a efektivně vytvářet aplikace pod operačním systémem Windows. Poslední verze fungují pod všemi nejrozšířenějšími verzemi Windows (Windows 95, 98, 98SE, ME, 2000, XP). Jak už napovídá název C++ Builderu, je toto prostředí založené na jazyce C++.

C++ Builder je zkrátka vývojový prostředek nové generace, který v sobě spojuje řadu mocných vizuálních nástrojů pro vytváření všech částí aplikace s velmi výkonným kompilátorem. C++ Builder je kompletně postaven na tzv. objektově orientované architektuře.

8 Realizace aplikace typu klient-server

Osmá kapitola je věnována realizaci aplikace typu klient-server. Jsou zde popsány aplikace RTP klient a RTP server z hlediska funkčnosti. Z hlediska jejich ovládání je naznačen způsob, kterým se docílí přenosu zvukových dat mezi počítači, a to jak unicastově, tak multicastově.

8.1 Popis aplikace RTP server z hlediska funkčnosti

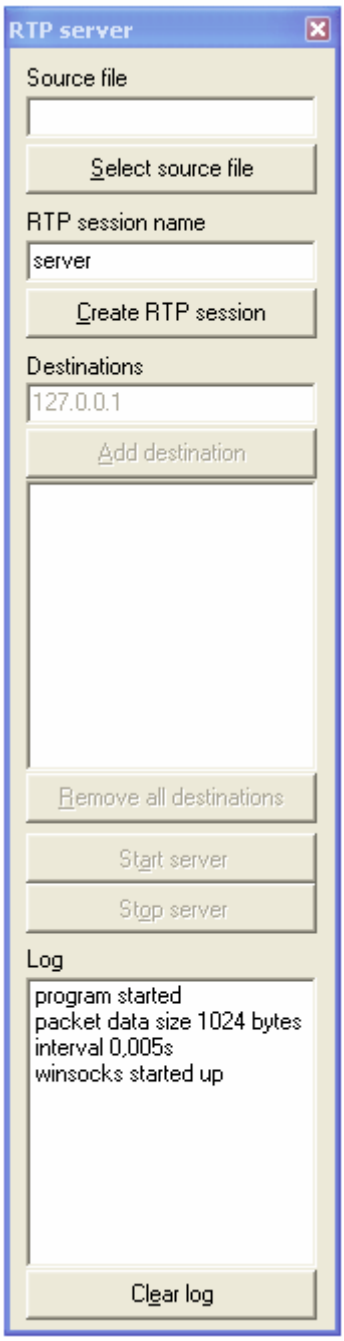
Nejprve byla potřeba inicializovat síť pro tuto aplikaci, a to ve Windows pomocí tzv. Windows sockets. Následovalo spuštění relace (session) RTP. Pomocí knihovny jRTPlib se prvně nastaví parametry pro protokol RTP a zavolá se členská funkce Create, která právě relaci vytvoří. Dále se přidají cíle pro odesílání dat (pomocí tlačítka Add destination – viz obr. 8.1). Cílů může být více a každému jsou odesílána stejná data. Poté se spustí server, konkrétně se vytvoří vlákno (pomocí winapi [windows application programming interface - windows programovací rozhraní] funkce CreateThread), které se stará o periodické odesílání dat. Cyklus odesílání dat probíhá tak, že se nejprve načtou všechna odesílaná data do paměti. Data se cyklicky a periodicky odesílají po částech s pevně stanovenou délkou (poslední data jsou samozřejmě kratší a mezi každým odesláním se čeká určitý interval).

Server lze zastavit dvěma způsoby:

- Tačíkem Stop server - chod vlákna, které zajišťuje odesílání dat, se přeruší.
- Po odeslání všech dat se server zastaví automaticky.

Po ukončení serveru se ukončí RTP relace. Pokud tedy chceme vysílat další data, musíme vytvořit novou RTP relace (pomocí tlačítka Create RTP session).

Při ukončení programu se provede deinicializace sítě (a pokud ještě server běží, před deinicializací sítě proběhne ukončení běhu serveru). Součástí ukončení běhu serveru je uvolnění paměti, využité k uchování dat pro odesílání.



Obrázek 8.1: Náhled na aplikaci RTP server

8.2 Popis aplikace RTP client z hlediska funkčnosti

Nejprve byla potřeba inicializovat síť pro tuto aplikaci, a to ve Windows pomocí tzv. Windows sockets. Dále pak inicializovat zvukovou knihovnu FMOD (vytvořit zvukový objekt FMOD a inicializovat ho). Následovalo spuštění relace (session) RTP. Pomocí knihovny jRTPlib se prvně nastaví parametry pro protokol RTP a zavolá se členská funkce Create, která právě relaci vytvoří. Od této chíle Klient vyčkává na příchozí informace, které jsou několika typů:

- Nový zdroj dat (OnNewSource) – Objeví se Server.
- Ukončení zdroje dat (OnRemoveSource) – Odpojí se Server.

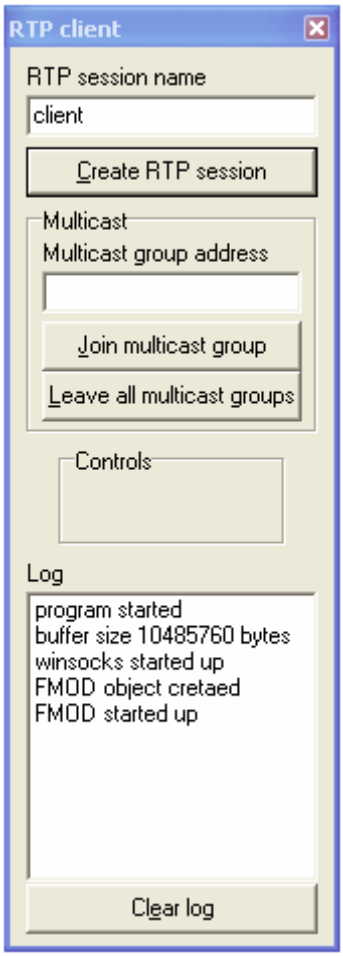
Příchozí pakety můžou být typu:

- RTP paket - přenášená data (v našem případě zvuková).
- APP paket - paket, který nese dodatečné informace o přenosu.
- BYE paket - posílá se těsně před odpojením serveru, po něm se vyvolá událost OnRemoveSource.

Pokud se objeví některá z těchto událostí, je vyvolána odpovídající metoda, která daný stav ošetří. Pokud se objeví událost OnNewSource, program vytvoří buffer pro příchozí data a začne čekat na data, která jsou do bufferu postupně zapisována (jak se objevuje událost OnRTPPacket - příchod paketu s daty).

Od této chvíle může uživatel v části Controls (obr. 8.2) kliknout na tlačítko s modrým kolečkem (načíst zvuková data z paměti) a zpřístupní se mu tlačítka pro přehrávání, a to tlačítko na přehrávání, tlačítko k pozastavení přehrávání (pauza) a poslední tlačítko slouží k uvolnění prostředků, které jsou využívány k přehrávání zvuku.

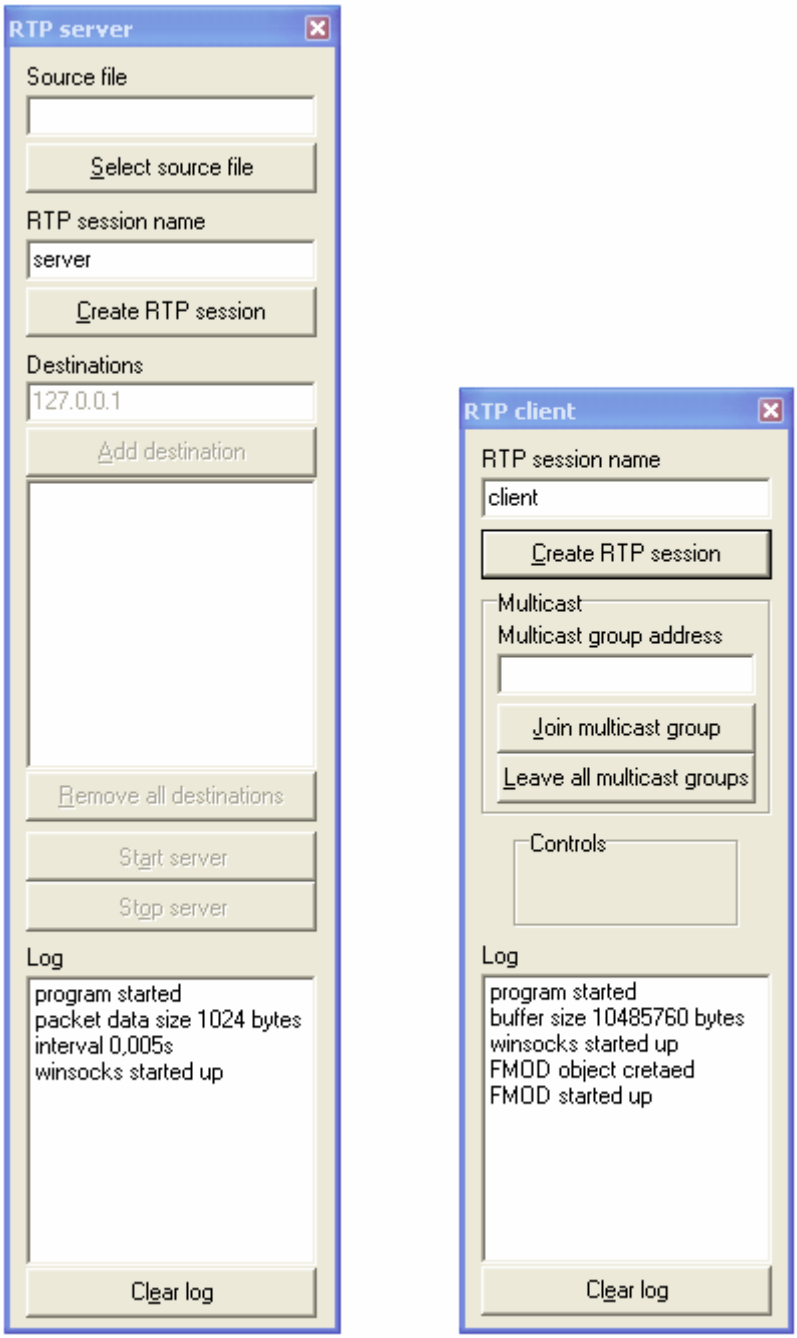
Při události OnRemoveSource (server ukončil posílání dat) se přestanou data ukládat do bufferu a audio data jsou kompletní - uživatel si je vyslechne do konce. Poté stačí aplikaci zavřít, aplikace se sama postará o uvolnění paměti využité pro příchozí data, o deinicializaci knihovny FMOD a knihovny jRTPlib a o deinicializaci sítě (Windows sockets).



Obrázek 8.2: Náhled na aplikaci RTP client

8.3 Ovládání aplikací RTP client a RTP server

Na obrázku 8.3 jsou zachyceny aplikace RTP klient a RTP server v takovém stavu, v jakém se uživateli představí při jejich spuštění. Ihned po spuštění se automaticky inicializuje síť (windows sockets). A to jak u RTP server, tak u RTP client. U RTP client se inicializuje i zvuková knihovna FMOD.



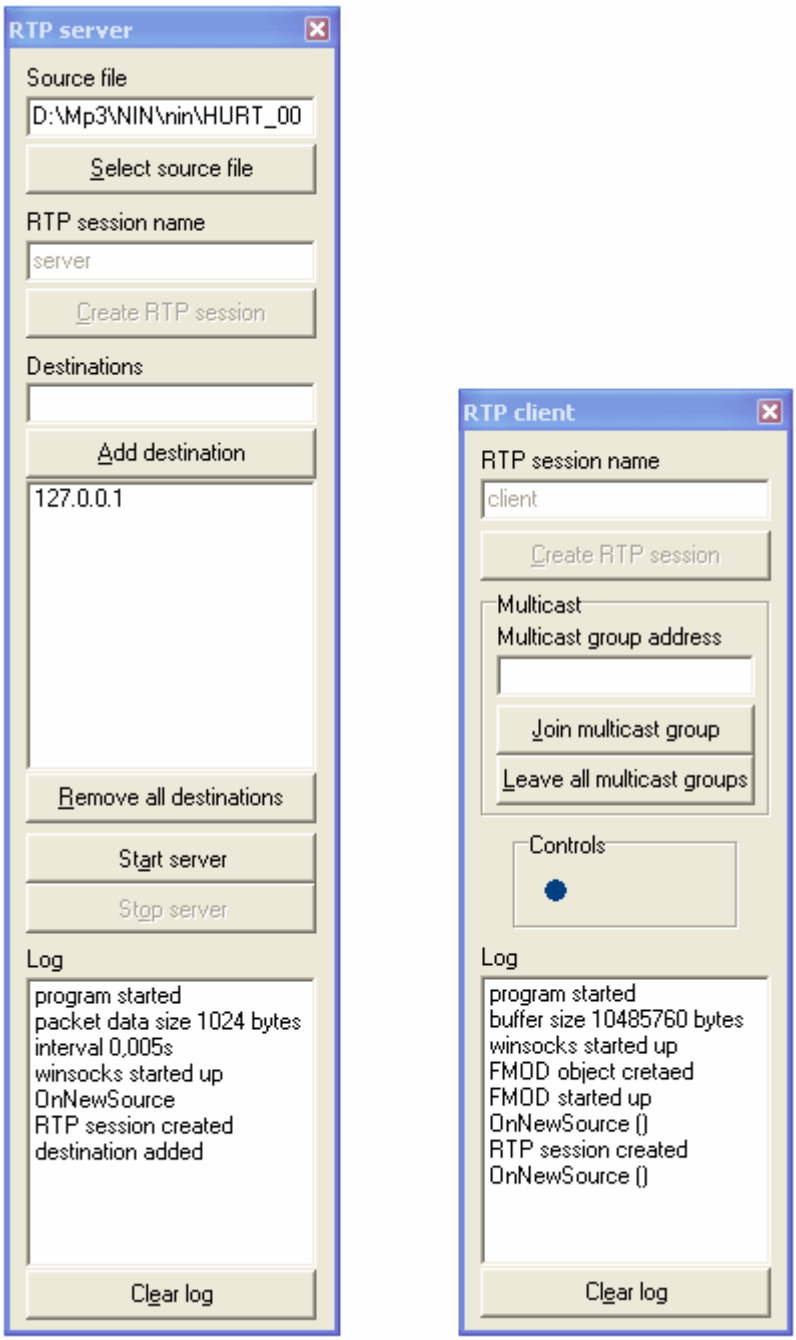
Obrázek 8.3: Náhled na aplikace RTP server a RTP client – fáze spuštění

Tlačítkem Select source file (RTP server) se zobrazí dialog, kde je možné vybrat zvukový soubor pro vysílání. Aplikace dokáže přenášet soubor libovolného typu, nemusí být ani zvukový. Klient však podporuje přehrávání pouze zvukových dat, smysl má tedy přenášet pouze je. Knihovna FMOD disponuje podporou velkého množství formátů zvukových dat, v dialogu pro výběr souboru máme možnost vybrat filtr pro zvukové soubory formátu wav a mp3 a dále filtr, který nám umožní vybrat soubor jakéhokoliv typu. Pokud si chceme být jisti

podporou určitého formátu, nahlédneme do dokumentace knihovny FMOD. Tlačítkem Create RTP session se vytvoří nová RTP relace (na toto tlačítko nutné kliknout na obou aplikací).

IP adresy, na které se budou vysílat data, se zadají do pole Destination. Může jich být uvedeno více. Všechny budou zobrazeny v seznamu pod tlačítkem Add destination, což je tlačítko, kterým se daná IP adresa potvrzuje. Po sestavení RTP relace je RTP client připraven přijímat RTP data. Po připojení serveru (událost OnNewSource) se aktivuje tlačítko s modrým kruhem, které slouží k načtení zvukových dat z paměti (aby bylo co načíst, musí se nejdříve začít vysílat data).

Na obrázku 8.4 lze vidět vybrání zvukového souboru, vytvoření RTP relace, zadání IP adresy, na kterou se zvuková data mají posílat a tlačítko ve tvaru modrého kroužku pro načtení dat.

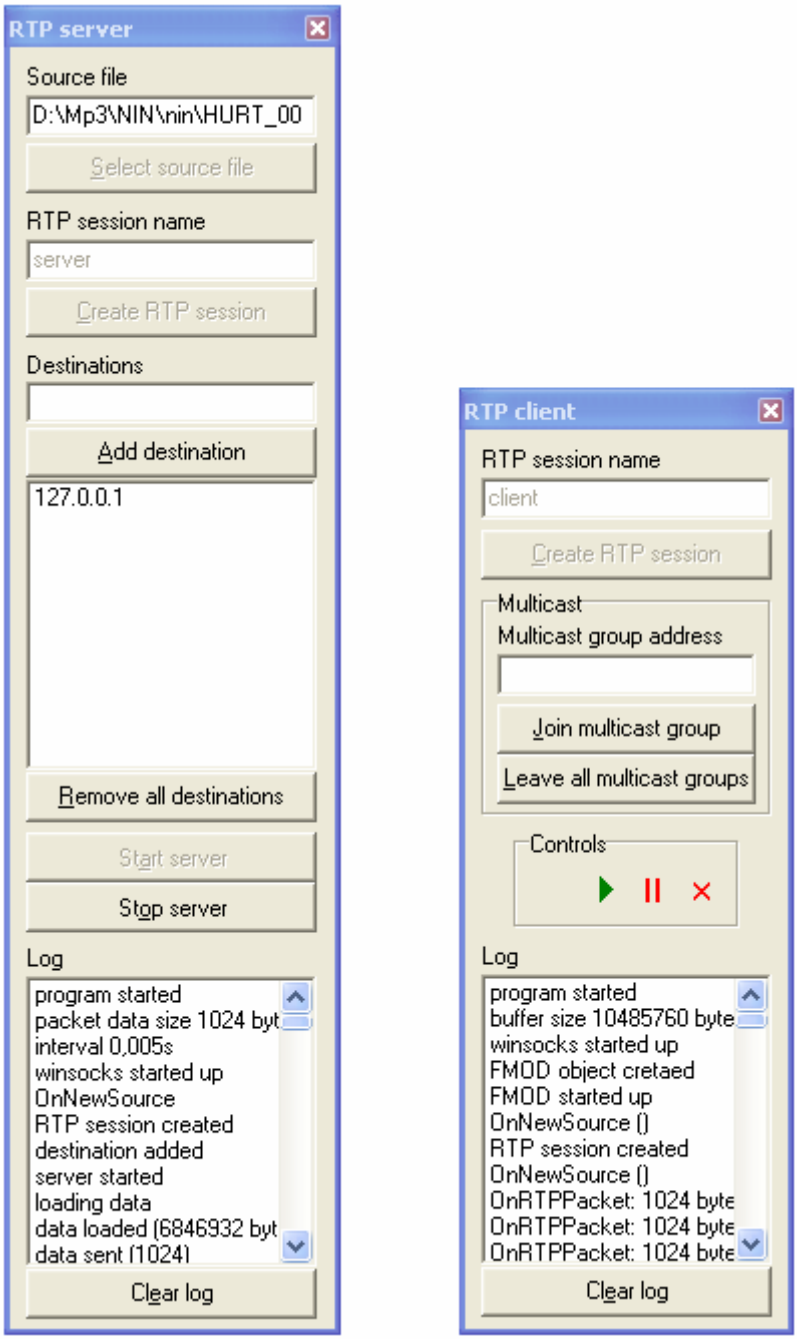


Obrázek 8.4: Náhled na aplikace RTP server a RTP client – fáze spojení

Vysílání dat zahájíme tlačítkem Start server. Vysílání dat lze kdykoli přerušit tlačítkem Stop server - zároveň se ukončí RTP relace. Pokud chceme odesílat další data, musíme vytvořit novou RTP relaci a případně vybrat jiný soubor.

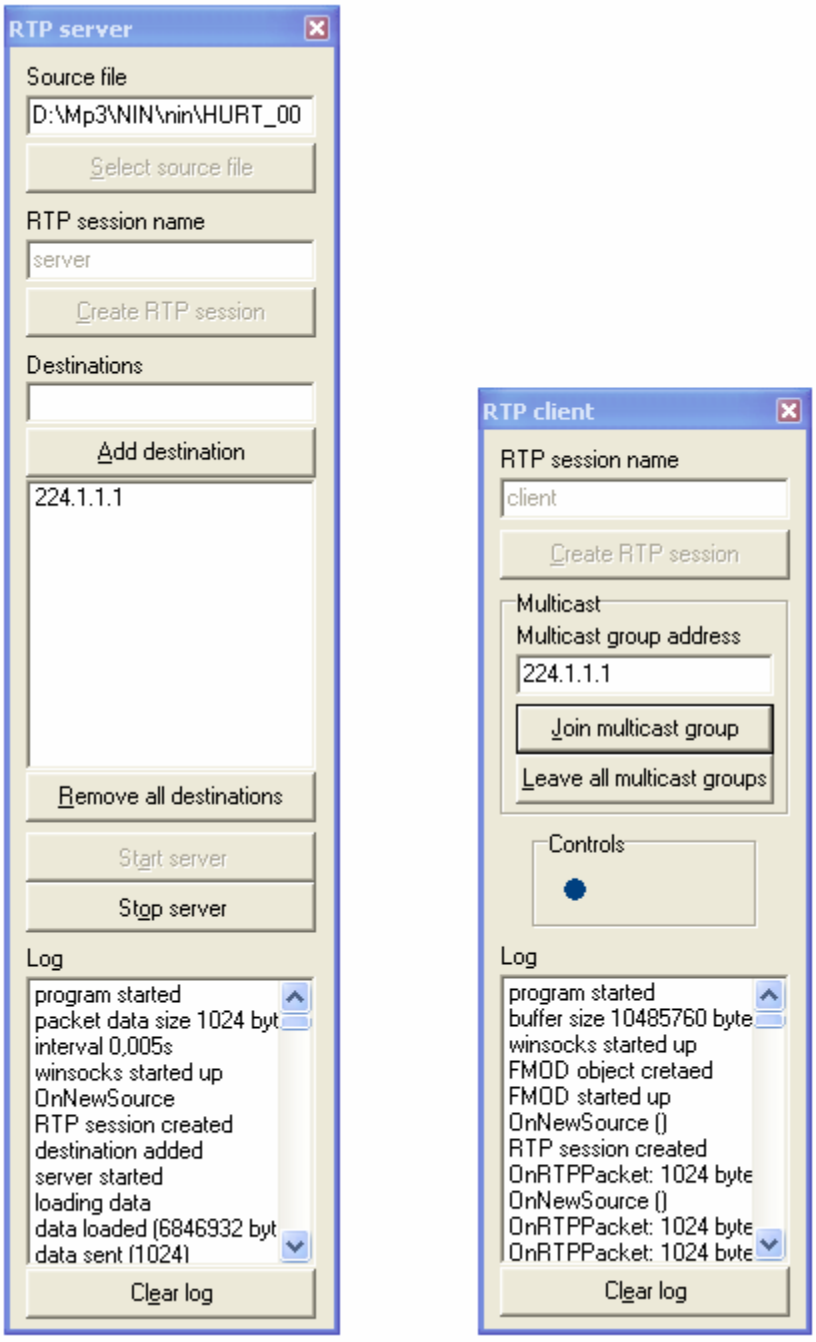
K ovládání přehrávání slouží tlačítka přehrát a pauza. Poslední tlačítko (s červeným křížkem) slouží k uvolnění paměti používané pro přehrávání zvuku. Práce programu je zachytávána v logu zpráv, který je v případě potřeby možno smazat tlačítkem Clear log.

Na obrázku 8.5 jsou aplikace zachyceny ve fázi, kdy se přenáší RTP data, které si klient může přehrát. U aplikace RTP client lze vidět tři tlačítka v poli Controls, kterými lze ovládat přehrávání zvukového souboru.



Obrázek 8.5: Náhled na aplikace RTP server a RTP client – fáze přehrávání

Pokud přidáme do seznamu cílů na serveru (Destinations) adresu typu D - tedy multicastovou adresu (rozsah IP adres je 224.0.0.0 až 239.255.255.255, přitom adresy v rozsahu 224.0.0.0 až 224.0.0.255 jsou určeny pro vyhrazené účely), budou data vysílána na skupinu počítačů. Klient se do multicastové skupiny může připojit pomocí tlačítka Join multicast group, opustit ji může tlačítkem Leave all multicast groups. Po připojení do multicastové skupiny klient začne odebírat data této skupině odesílané ze serveru. Multicastové vysílání je zachyceno na obrázku číslo 8.6.



Obrázek 8.6: Náhled na aplikace RTP server a RTP client – multicastové vysílání

9 Závěr

Tato bakalářská práce se zabývala přenosem zvukových dat. Cílem práce bylo navrhnout a zrealizovat aplikaci, která přenáší v reálném čase zvuková data. Byly prozkoumány volně dostupné knihovny, které se daly pro sestrojení požadované aplikace využít. Byla potřeba najít knihovna, která je volně dostupná, která má v sobě implementovaný protokol RTP. Za nejvýhodnější byla shledána knihovna s názvem jRTPlib. Za pomoci této knihovny a za pomoci zvukové knihovny FMOD, která dokázala pracovat se zvukovým formátem v požadovaném smyslu, byla ve vývojovém prostředí Borland C++ Builder zhotovena aplikace na principu síťové architektury klient-server.

Aplikace má své výhody a nevýhody. Za výhodu lze považovat to, že zvuková data lze přijímat na více klientech, aplikace umožňuje multicastové vysílání. Výhodou je i to, že se klient může připojit po zahájení vysílání dat. Aplikace využívá kvalitní knihovnu jRTPlib a velmi kvalitní zvukovou knihovnu FMOD. Další výhoda je ta, že jsou data přednačítána do paměti, pokud to datový tok dovolí. Mezi nevýhody patří poměrně velká spotřeba paměti (všechna odesílaná a přijatá data se ukládají do paměti najednou).

Jelikož vytvořené aplikace pracují se sítí, je potřeba znát dostatečně síťové rozhraní počítače, na kterém jsou tyto aplikace spouštěny. Je potřeba umět zjistit adresu IP počítače, protože je to jediný způsob, jak adresovat klienta. Aplikace mezi sebou komunikují na portech 4000 a 5000, proto se musí zajistit, aby tyto porty nebyly blokovány. Přednostně se musí zkontrolovat software pro kontrolu sítových rozhraní – tzv. firewall. Ten je v systému Windows buď ve formě standardního firewallu v jádře (Ovládací panely – Brána firewall systému Windows) nebo ve formě externího programu (např. Kerio Personal Firewall – pro domácí použití zdarma). Je potřeba zkontrolovat nastavení těchto programů nebo je zcela vypnout, aby nerušily chod programů. I některé antivirové programy zastupují funkci firewallu, tzn. kontrolují síťový provoz, a mohly by narušit chod programů. Proto se i tyto programy musí v případě použití sestrojené aplikace vypnout. Faktorů, které ovlivňují chod sítě je mnoho, toto jsou dva poměrně časté, proto se jejich konfigurace nesmí zanedbat (nevýhoda).

Sestrojená aplikace se nedá považovat za dokonalou, proto by nynější verze měla být brána spíše jako princip demonstrace přenosu zvukového formátu za pomocí protokolu RTP mezi dvěma počítači, tak jak je také stanoveno v zadání bakalářské práce. Aplikaci je možné shlédnout na přiloženém CD.

POUŽITÁ LITERATURA

[1] MUSIL, M. *Historie sítě Internet* [on-line], poslední aktualizace 21. 7. 2003. [cit. 9. 11. 2007]. Dostupné z WWW: <http://ihistory.webzdarma.cz/chap/sites/tcp_ip.php>.

[2] PUŽMANOVÁ, R. *TCP/IP v kostce*. KOPP : České Budějovice. ISBN 80-7232- 236-2.

[3] DOSTÁLEK LIBOR, ALENA KABELOVÁ. *Velký průvodce TCP/IP a DNS systémy*. Computer Press: Praha. ISBN 80-7226-323-4.

[4] PERKINS, C. RTP: *Audio and Video for the Internet*. Addison-Wesley Professional. 1st edition (June 11, 2003). 423. ISBN 978-0672322495.

[5] IETF. *RTP: A Transport Protocol for Real-Time Applications* [on-line]. 2003 [cit. 14.11.2007]. Dostupné z WWW: <http://www.ietf.org/rfc/rfc3551.txt?number=3550>.

[6] PUŽMANOVÁ, R. *Streaming media (4): transportní protokoly RTP/RTCP* [on-line], poslední aktualizace 1. 4. 2006. [cit. 14. 11. 2007]. Dostupné z WWW: <http://www.dsl.cz/clanky-dsl/clanek-60/Streaming-media-%284%29%3A-transportni-protokoly-RTP-RTCP>.

[7] VOIPSA, Inc. *VoIP Security and Privacy Threat Taxonomy* [on-line], poslední aktualizace 24. 9. 2005 [cit. 25. 4. 2008]. Dostupné z WWW: <http://www.voipsa.org/Activities/VOIPSA_Threat_Taxonomy_0.1.pdf>.

[8] SEDLÁK, M. *Bezpečnost systému pro VoIP*. Brno: Vysoké učení technické v Brně, Fakulta informačních technologií, 2006.

[9] Http://www.cs.columbia.edu/irt/software/rtplib/rtplib-1.0a1/rtp_api.html [online]. 1998 [cit. 19.5.2008]. Dostupný z WWW: <http://www.cs.columbia.edu/irt/software/rtplib/rtplib-1.0a1/rtp_api.html>.

[10] Http://www.cs.columbia.edu/irt/software/rtptools/ [online]. Poslední aktualizace 30. 1. 2008 [cit. 19.5.2008]. Dostupný z WWW: <http://www.cs.columbia.edu/irt/software/rtptools/>.

[11] Http://research.edm.uhasselt.be/~jori/page/index.php?n=CS.Jrtplib [online]. 2006 - 2008 [cit. 19.5.2008]. Dostupný z WWW: <http://research.edm.uhasselt.be/~jori/page/index.php?n=CS.Jrtplib>.

[12] Http://www.informatik.uni-mannheim.de/pi4/projects/RTPI/download.html [online]. 2007 [cit. 20.5.2008]. Dostupný z WWW: <http://www.informatik.uni-mannheim.de/pi4/projects/RTPI/download.html>.

[13] Http://data.gnutelephony.org/ccrtp/refman/html/classes.html [online]. Poslední aktualizace 18. 1. 2004 [cit. 21.5.2008]. Dostupný z WWW: <http://data.gnutelephony.org/ccrtp/refman/html/classes.html>.

[14] Http://download.savannah.gnu.org/releases/linphone/ortp/docs/ [online]. Poslední aktualizace 18. 1. 2004 [cit. 21.5.2008]. Dostupný z WWW: <http://download.savannah.gnu.org/releases/linphone/ortp/docs/>.

[15] SVOBODA, *FMOD - sound system* [on-line], poslední aktualizace 29. 11. 2001. [cit. 23. 5. 2008]. Dostupné z WWW: <http://www.builder.cz/art/cpp/fmod.html>.

[16] Http://www.fmod.org/ [on-line], poslední aktualizace 23. 5. 2008. [cit. 23. 5. 2008]. Dostupné z WWW: <http://www.fmod.org/>

PŘÍLOHA A

Seznam funkcí knihovny RTPlib

RTPCreate()	RTPNextMember()
RTPDestroy()	RTPReceive()
RTPOpenConnection()	RTPGetRTPPacket()
RTPCloseConnection()	RTPPacketGetCSRC()
RTPSend()	RTPSplitCompoundRTCP()
RTPSendVector()	RTPGetRTCPPacket()
RTPSessionAddSendAddr()	RTPGetReportBlock()
RTPSessionRemoveSendAddr()	RTPGetByeBlock()
RTPSessionSetReceiveAddr()	InitSDESItemIter()
RTPSessionGetReceiveAddr()	GetNextItem()
RTPSessionSetEncryption()	RTPMemberInfoGetStatus()
RTPSessionGetEncryption()	RTPMemberInfoSetSDES()
RTPSessionSetEncryptionFuncs()	RTPMemberInfoGetSDES()
RTPSessionSetRTPStampRate()	RTPMemberInfoGetRTPAddr()
RTPSessionGetRTPStampRate()	RTPMemberInfoGetRTCPAddr()
RTPSessionSetKey()	RTPMemberInfoSetNTP()
RTPSessionGetKey()	RTPMemberInfoGetNTP()
RTPSessionSetReconsideration()	RTPMemberInfoSetRTP()
RTPSessionGetReconsideration()	RTPMemberInfoGetRTP()
RTPSessionSetBandwidth()	RTPMemberInfoSetPktCnt()
RTPSessionGetBandwidth()	RTPMemberInfoGetPktCnt()
RTPSessionSetUserInfo()	RTPMemberInfoSetRTCPPktCnt()
RTPSessionGetUserInfo()	RTPMemberInfoGetRTCPPktCnt()
RTPSessionGetMemberList()	RTPMemberInfoSetOctCnt()
RTPSessionGetRTPSocket()	RTPMemberInfoGetOctCnt()
RTPSessionGetRTCPSocket()	RTPMemberInfoSetSSRC()
CSRC Handling Functions	RTPMemberInfoGetSSRC()
RTPSessionAddToContributorList()	RTPMemberInfoSetUserInfo()
RTPSessionRemoveFromContributorList()	RTPMemberInfoGetUserInfo()
RTPSessionAddToCSRCList()	RTPSenderInfoGetFirstReceiverReport()
RTPSessionRemoveFromCSRCList()	RTPSenderInfoGetNextReceiverReport()
RTPSessionGetUniqueIDForCSRC()	RTPSenderInfoGetLocalReception()
RTPSessionGetCSRCList()	RTPFindMember()
RTPSetUpdateMemberCallBack()	RTPMostRecentRTPTime()
RTPSetChangedMemberInfoCallBack()	RTPMostRecentRTCPTime()
RTPSetCollidedMemberCallBack()	RTPMostRecentRTPPerson()
RTPSetRevertingIDCallBack()	RTPMostRecentRTCPPerson()
RTPSetSendErrorCallBack()	RTPMostRecent()
RTPCurrentMember()	

PŘÍLOHA B

Seznam tříd knihovny jRTPlib

RTCPAPPPacket	Popisuje RTCP APP paket
RTCPBYEPacket	Popisuje RTCP BYE paket
RTCPCompoundPacket	Reprezentuje sestavený RTCP paket
RTCPCompoundPacketBuilder	Tato třída může být využita ke sestavení RTCP paketů
RTCPPacket	Základní třída pro specifické druhy RTCP paketů
RTCPPacketBuilder	Sestavení RTCP paketů ve vyšším levelu než je tomu tak u třídy RTCPCompoundPacketBuilder
RTCPRRPacket	Popisuje RTCP receiver report paket
RTCPScheduler	Tato třída určuje, kdy mají být sestavené RTCP pakety odeslány
RTCPSchedulerParams	Popisuje parametry využívané třídou RTCPScheuduler
RTCPSDESInfo	Třída RTCPSDESInfo je kontejnerem pro RTCP SDES informace
RTCPSDESPacket	Popisuje RTCP source description paket
RTCPSRPacket	Popisuje RTCP sender report paket
RTCPUnknownPacket	Popisuje RTCP paket neznámého typu
RTPAddress	Abstraktní třída, která využívá specifické cílové místo, multicastovou skupinu, atd.
RTPCollisionList	Tato třída zobrazuje seznam adres, ze kterých byly odhaleny SSRC kolize
RTPIPv4Address	Reprezentuje IPv4, IP adresu a port
RTPIPv6Address	Reprezentuje IPv6, IP adresu a port
RTPLibraryVersion	Poskytuje informaci o verzi knihovny
RTPMemoryManager	Správce paměti
RTPNTPTime	Jednoduchý obal pro slovo (MSW) a (LSW) NTP časové značky
RTPPacket	Reprezentuje RTP paket
RTPPacketBuilder	Vytváří SSRC identifikátor, časové značky a sekvenční číslo atd.
RTPRandom	Třída je využita ke generování náhodných čísel
RTPRawPacket	Slouží k ukládání přicházejících RTP a RTCP dat
RTPSession	Třída vysoké úrovně pro použití RTP
RTPSessionParams	Popisuje parametry využívané třídou RTPSession
RTPSourceData	Popisuje vstupní zdrojovou tabulku třídy RTPSources
RTPSources	Reprezentuje informační tabulku účastnického zdroje
RTPTime	Třída používána pro specifikaci zpoždění
RTPTransmissionInfo	Základní třída pro doplňující informace o vysílači
RTPTransmissionParams	Základní třída pro přenosový parametr
RTPTransmitter	Abstraktní třída, která odvozuje aktuální přenosovou komponentu
RTPUDPv4TransmissionInfo	Další informace o UDP přes IPv4 vysílač
RTPUDPv4TransmissionParams	Parametry pro UDP přes IPv4 vysílač
RTPUDPv4Transmitter	UDP přes IPv4 přenosovou komponentu
RTPUDPv6TransmissionInfo	Další informace o UDP přes IPv6 vysílač
RTPUDPv6TransmissionParams	Parametry pro UDP přes IPv6 vysílač
RTPUDPv6Transmitter	UDP přes IPv6 vysílač

PŘÍLOHA C

Seznam tříd knihovny ccRTP

AppDataUnit	RTCPCompoundHandler::NACKPacket
ApplicationHandler	RTCPCompoundHandler::ReceiverInfo
AVPQueue	RTCPCompoundHandler::RecvReport
ConflictHandler	RTCPCompoundHandler::RRBlock
ConflictHandler::ConflictingTransportAddress	RTCPCompoundHandler::RTCPFixedHeader
DestinationListHandler	RTCPCompoundHandler::RTCPPacket
DestinationListHandler::TransportAddress	RTCPCompoundHandler::SDESChunk
DualRTPChannel	RTCPCompoundHandler::SDESItem
DualUDPIPv4Socket	RTCPCompoundHandler::SenderInfo
DynamicPayloadFormat	RTCPCompoundHandler::SendReport
IncomingDataQueue	RTCPReceiverInfo
IncomingDataQueue::SyncSourcesIterator	RTCPReceiverReport
IncomingDataQueueBase	RTCPSenderInfo
IncomingRTPPkt	RTPApplication
Members	RTPApplication::ParticipantsIterator
MembershipBookkeeping	RTPBaseUDPIPv4Socket
MembershipBookkeeping::IncomingRTPPktLink	RTPDataQueue
MembershipBookkeeping::SyncSourceLink	RTPDuplex
OneThreadRTPSession	RTPPacket
OutgoingDataQueue	RTPQueueBase
OutgoingDataQueue::OutgoingRTPPktLink	RTPSessionBase
OutgoingDataQueueBase	RTPSessionBaseHandler
OutgoingRTPPkt	RTPSessionPool
Participant	SDESItemsHolder
ParticipantHandler	SingleRTPSessionPool
PayloadFormat	SingleThreadRTPSession
QueueRTCPManager	StaticPayloadFormat
RTCPCompoundHandler	SyncSource
RTCPCompoundHandler::APPPacket	SyncSourceHandler
RTCPCompoundHandler::BYEPacket	TRTPSessionBase
RTCPCompoundHandler::FIRPacket	

PŘÍLOHA D

Seznam funkcí knihovny oRTP

ortp_exit()	rtp_session_recv_with_ts()
ortp_global_stats_display()	rtp_session_recvm_with_ts()
ortp_init()	rtp_session_register_event_queue()
ortp_min_version_required()	rtp_session_release_sockets()
ortp_scheduler_init()	rtp_session_reset()
ortp_set_log_file()	rtp_session_resync()
ortp_set_log_handler()	rtp_session_send_dtmf()
ortp_set_log_level_mask()	rtp_session_send_dtmf2()
payload_type_destroy()	rtp_session_send_telephone_events_supported()
payload_type_set_recv_fmtp()	rtp_session_send_with_ts()
payload_type_set_send_fmtp()	rtp_session_sendm_with_ts()
rtp_profile_clear_all()	rtp_session_set_blocking_mode()
rtp_profile_clear_payload()	rtp_session_set_connected_mode()
rtp_profile_set_name()	rtp_session_set_data()
rtp_profile_set_payload()	rtp_session_set_dscp()
rtp_session_add_telephone_event()	rtp_session_set_jitter_compensation()
rtp_session_bye()	rtp_session_set_local_addr()
rtp_session_create_packet()	rtp_session_set_multicast_loopback()
rtp_session_create_packet_in_place()	rtp_session_set_multicast_ttl()
rtp_session_create_packet_with_data()	rtp_session_set_payload_type()
rtp_session_create_telephone_event_packet()	rtp_session_set_profile()
rtp_session_destroy()	rtp_session_set_recv_buf_size()
rtp_session_enable_rtcp()	rtp_session_set_recv_payload_type()
rtp_session_flush_sockets()	rtp_session_set_recv_profile()
rtp_session_get_current_recv_ts()	rtp_session_set_remote_addr()
rtp_session_get_current_send_ts()	rtp_session_set_remote_addr_and_port()
rtp_session_get_data()	rtp_session_set_scheduling_mode()
rtp_session_get_dscp()	rtp_session_set_send_payload_type()
rtp_session_get_last_recv_time()	rtp_session_set_send_profile()
rtp_session_get_local_port()	rtp_session_set_seq_number()
rtp_session_get_multicast_loopback()	rtp_session_set_source_description()
rtp_session_get_multicast_ttl()	rtp_session_set_ssrc()
rtp_session_get_profile()	rtp_session_set_symmetric_rtp()
rtp_session_get_recv_payload_type()	rtp_session_set_time_jump_limit()
rtp_session_get_recv_profile()	rtp_session_signal_connect()
rtp_session_get_send_payload_type()	rtp_session_signal_disconnect_by_callback()
rtp_session_get_send_profile()	rtp_session_telephone_events_supported()
rtp_session_get_stats()	rtp_stats_display()
rtp_session_new()	session_set_destroy()
rtp_session_read_telephone_event()	session_set_new()
rtp_session_recv_telephone_events_supported()	session_set_select()

PŘÍLOHA E

Obsah přiloženého CD

- Bakalarska_prace.pdf (elektronická verze bakalářské práce)
- Metadata.pdf (popisný soubor závěrečné práce)
- Složka „Project“ - obsahuje 4 další složky, a to:
    - bin (obsahuje spustitelné soubory)
    - src (obsahuje zdrojové kódy a projektové soubory klienta a serveru)
    - lib (obsahuje zkompilované statické knihovny)
    - include (obsahuje hlavičkové soubory knihoven FMOD, jthread a jRTPlib)