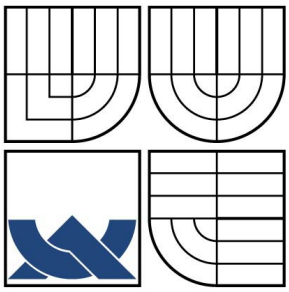
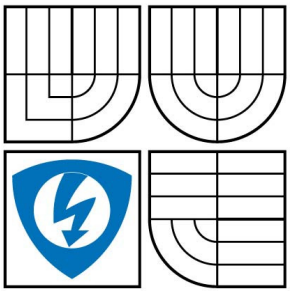


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

VYUŽITÍ SPOLEHLIVÉHO MULTICASTU PRO PŘENOS ŘÍDÍCH INFORMACÍ IPTV

USE OF RELIABLE MULTICAST FOR FEEDBACK TRANSMISSION IN IPTV

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — DAVID SOBOTKA
AUTHOR

VEDOUCÍ PRÁCE — ING. DAN KOMOSNÝ, PH.D.
SUPERVISOR

BRNO 2007

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení:

Bytem:

Narozen/a (datum a místo):

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií

se sídlem Údolní 244/53, 602 00, Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....

(dále jen „nabyvatel“)

Čl. 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: .....

Vedoucí/ školitel VŠKP: .....

Ústav: .....

Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v*:

- □ tištěné formě – počet exemplářů ……..
- □ elektronické formě – počet exemplářů ……..

* hodící se zaškrtněte

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

Anotace

Tato diplomová práce se zabývá částí síťové komunikace v systému IPTV za použití protokolu pro přenos multimediálních dat RTP, konkrétně jeho řídící části RTCP a jeho možné využití pro přenos hlasovacích výsledků. Dále zhodnocuje typy a možnosti použití spolehlivého multicastu a specifického multicastu (SSM).

Součástí práce je také vytvoření programu pro simulaci přenosu hlasovacích výsledků v programovacím jazyce C++ na platformě MS Windows. Výsledkem je tedy program na straně klienta, který přijímá anketu od serveru a odpovídá na ni aplikačním paketem protokolu RTCP. Na druhé straně server daný paket zachytí a zpracuje. Aplikace pracují v konzolovém módu, tedy bez grafického rozhraní. Uživatel v serverové aplikaci může tvořit anketu a zasílat ji klientovi a případně si prohlédnout výsledky hlasování. V klientské aplikaci provádí volbu hlasováním a případné zaslání dalších voleb jako simulaci připojení více klientů k serveru.

Klíčová slova

IPTV, RTP/RTCP, spolehlivý multicast, SSM, IP multicast, hlasování, anketa, C++

Abstract

This Diploma thesis is focused on a part of network communication in the system of IPTV using the protocol to provide transmission of real-time multimedia RTP, specifically using the control protocol RTCP to carry voting results. The thesis further shows variants and ways of using reliable multicast and specific multicast (SSM).

The main practical goal of this thesis is the program for the simulation of carrying voting results that I have created in the C++ programming language for MS Windows platform. The result is the client program receiving an inquiry from the server and reacting by application packets of the RTCP protocol. On the other side is the server program that picks up the packets and processes them. The programs run in the console mode, thus they have no graphical user interface. A user in the server program can create any inquiry and consequently send it to the client. Furthermore, the server can possibly view the current results of voting. In the client program a user can vote and possibly send another vote that will simulate other clients

Keywords

IPTV, RTP/RTCP, reliable multicast, SSM, IP multicast, vote, inquiry, C++

Citace

SOBOTKA, D. *Využití spolehlivého multicastu pro přenos řídících informací IPTV*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 50 s. Vedoucí bakalářské práce Ing. Dan Komosný, Ph.D.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma „*Využití spolehlivého multicastu pro přenos řídících informací IPTV*“ 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 ....
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Danovi Komosnému, Ph.D. za cenné rady a vstřícný přístup při vypracovávání bakalářské práce

Obsah

Seznam obrázků

1. Úvod

Rozvoj internetu prodělává stále nové změny v oblasti zanášení nových standardů a nabízí také rychle se rozšiřující nabídku stále nových služeb. Jednou z nejvíce se rozšiřujících oblastí je oblast multimediálních přenosů dat a to především díky velkému rozvoji a dostupnosti broadbandu. Právě díky vzrůstu vysokorychlostního připojení vznikají projekty s čím dál vyšší náročností na kapacitu linek a v oblasti multimedií se kladou stále větší nároky na zvýšenou kvalitu dat. Narůstající objem datových toků nestačí pokrývat pouze nárůst kapacity linek. Je třeba jej korigovat i vhodnými aplikacemi algoritmů pro kompresi či pro ekonomičtější způsob šíření sítí. Důsledkem toho probíhá mnoho inovací a rozšíření stávajících standardů a doporučení v oblasti protokolů pro přenos dat.

Jedním z projektů, který se zabývá vývojem nových technologií v oblasti komunikací v IP sítích, konkrétně pro multicast, je Multicast IPTV Research Group, který vznikl na Vysokém učení technickém v Brně, pod záštitou Ústavu telekomunikací.

Cílem semestrální práce je seznámit se problematikou přenosu multimediálních dat přes internet, konkrétně se týkajících IPTV. Mým úkolem bude se zaměřit na protokol používaný při přenosu v reálném čase tzv. Real-Time protokol, konkrétně na signalizační protokol využívající zpětnou komunikaci od klientů, přijímajících streamovaná data, k serveru, který je zdrojem vysílání multimediálních dat. Dále také poukázat na možnosti multicastového síření v IP sítích, zaměřit se na jejich možnou spolehlivost a případně porovnat jejich efektivitu, výkonnost a využití.

2. Popis IPTV

IPTV (Internet Protocol Television) je služba, která umožňuje na základě IP protokolu a protokolu pro přenos v reálném čase přenášet televizní vysílání přes internetovou síť digitální formou. Ta přináší oproti klasickému analogovému vysílání mnoho výhod a nadstandardních služeb. S šířením televizního vysílání (broadcasting) lze distribuovat i služby nelineárního charakteru jako např VoD (Video on Demand – video na vyžádání), PPV (Pay Per View – pořady za poplatek) apod. IPTV má velký potenciál pro individualizaci, jelikož každému příjemci se dopravuje individuální obsah po určitém kanálu, který je dopředný i zpětný. Tím se nabízí velký potenciál ve formě zpětné interaktivity, kdy je možnost nabízet takové služby příjemcům, které mají odezvu dle jejich přání (reklama, úprava programové nabídky, hlasování v anketách, filmech na přání...apod.), případně využít statistik, např. sledovanosti, za cílem ke zlepšení služeb zákazníkům.

Dalším zásadním rozdílem oproti ostatním způsobům vysílání je, že u IPTV k příjemci cestuje jeden vybraný program, čili výběr probíhá na straně serveru poskytovatele, s čímž je i spojena značná prodleva při přepínání programů (tzv. channel zapping). Nepřenáší se tak celý programový balík. Tato vlastnost má také za následek, že na jednu přípojku lze připojit pouze jeden set-top-box (dekodér digitálního signálu) a divák tak může v domácnosti s více přijímači sledovat pouze jeden program. Tyto důvody jsou ale pouze kapacitního charakteru, než principiálního a je jenom otázkou času kdy toto omezení bude překročeno (např. nasazením různých variant vysokorychlostních optických FTTx přípojek).

Další vlastností IPTV, bližší mé semestrální práci, je využití multicastingu, kde jeden program představuje právě jednu multicastovou skupinu, ke které se můžou potenciální příjemci přihlásit, případně odhlásit a přihlásit se k jiné. Právě toto přihlašování a odhlašování se ze skupin má za následek onu zpětnou prodlevu při přepínání programů, jak jsem se již zmiňoval výše.

2.1. Šíření multimediálních dat v reálném čase

Využití služeb pracujících s daty v reálném čase je mnoho. Jako příklad můžeme zmínit VoIP, videokonference, rozhlasové, televizní vysílání. Častým využívaným řešením pro služby v reálném čase je koncept server-klient, resp. jeden vysílací zdroj (streaming server) a mnoho účastníků ve skupině odebírající data ze zdroje. Dalším možným konceptem je, že ve skupině bude více vysílacích zdrojů (videokonference) případně se zdroje vysílaní budou měnit v rámci skupiny.

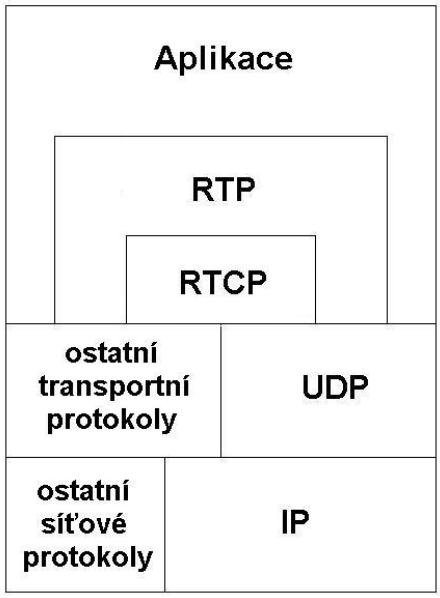
Jak již bylo naznáno, pro šíření multimediálních dat v reálném čase je třeba využít i patřičného protokolu. Takovýmto protokolem je RTP/RTCP (Real-Time Transport Protocol/Real-Time Transport Control Protocol), který je navržen výše uvedené šíření multimediálních dat v reálném čase.

2.2. Architektura protokolu RTP/RTCP

Protokol RTP/RTCP je nespolehlivý, čili nezaručuje že poslaný paket byl doručen k cíli korektně nebo zda vůbec dorazil a ani se nesnaží sjednat nápravu požadavkem o znovuzaslání jak je tomu u řady protokolů spolehlivých např. TCP (Transmmision Control Protocol). Protokol také nezajišťuje zda byl paket doručen ve správném pořadí. Protokol je navržen tak, že odděluje uživatelské data (RTP) od signalizace (RTCP).

Protokol přenosu v reálném čase je nezávislý na typu sítě a přenosovém protokolu, často však pracuje nad transportním protokolem UDP (User Datagram Protocol) a síťovým IP (Internet Protocol). Další možně protokoly nad kterými může pracovat jsou např. ST-II, IPX, ATM-AALx…

Obr. 2.1: Architektrura RTP/RTCP nad ostatními protokoly



2.3. Specifikace protokolu RTP/RTCP

Služba RTP před samotným zahájením přenosu dat zajišťuje identifikaci typu multimediálních dat a opatří je i patřičném pořadovým číslem. V případě více zdrojů se postará i synchronizaci jejich datových toků. Co však služba negarantuje je, že budou doručeny ve správném pořadí a proto se o to musí postarat aplikace na straně příjemce a rekonstruovat tak datové pakety pomocí jejich záhlaví, případně identifikovat chybějící.

Další důležitou chybějící vlastností RTP protokolu je, že neposkytuje záruku kvality služby (QoS – Quality of Service), která je pro přenos podstatná. Proto je nutno využít zde jiných protokolů (např. RVSP – Resource Reservation Protocol).

Kvalitu distribuce multimediálního přenosu dat obstarává protokol signalizační RTCP, který monitoruje provoz za pomocí kontrolních a identifikačních mechanismů. Tento protokol pracuje v úzkém spojení s protokolem RTP a jejich identifikace je dána párem portů, kde RTCP používá port o jedničku vyšší než RTP.

2.4. Popis protokolu RTP

Jak už bylo uvedeno, protokol RTP slouží pro vlastní přenos dat. Pro RTP protokol jsou typické tzv. relace (session), což jsou asociace mezi účastníky komunikující přes RTP, kdy účastník se může zúčastnit více relací v jednom čase. Relace je charakteristická cílovou transportní adresou (síťová adresa + port). Multimediální obsah tak není přenášen jako v rámci jedné relace, ale média jsou rozdělena do více relací se svým vlastním datovým tokem a vlastními RTCP pakety. Typickým příkladem je oddělený přenos zvuku a videa, kdy každý se přenáší zvlášť. Uplatnění to má především pro příjemce s nižší šířkou pásma, kteří si můžou zvolit např. pouze příjem audio toku (např. u videokonference).

V rámci RTP protokolu je třeba se ještě krátce zmínit dva pojmy, „mixer“ a „translator“. Úloha mixeru v přenosu je sloučit více zdrojů užitečného obsahu do jednoho a upravit a resynchronizovat tak datový tok pro příjemce který využívá užší přenosové pásmo než ostatní příjemci ve skupině, zabrání se tak neefektivní snižování přenosového pásma pro příjemce,co využívají rychlejší přenosové linky. Translator najde využití v případě, že příjemce se nachází za firewallem a není tak přímo dostupný přes IP multicast. Pak se před a za firewall instaluje translator, který pomocí tzv. tunelu přepraví data do sítě za firewallem. Užití translatoru je tedy například užitečné pro komunikaci, kde se spojují sítě využívající jiné protokoly nižších vrstev (např. IP/UDP a ST-II nebo ATM-ALLx…atp.)

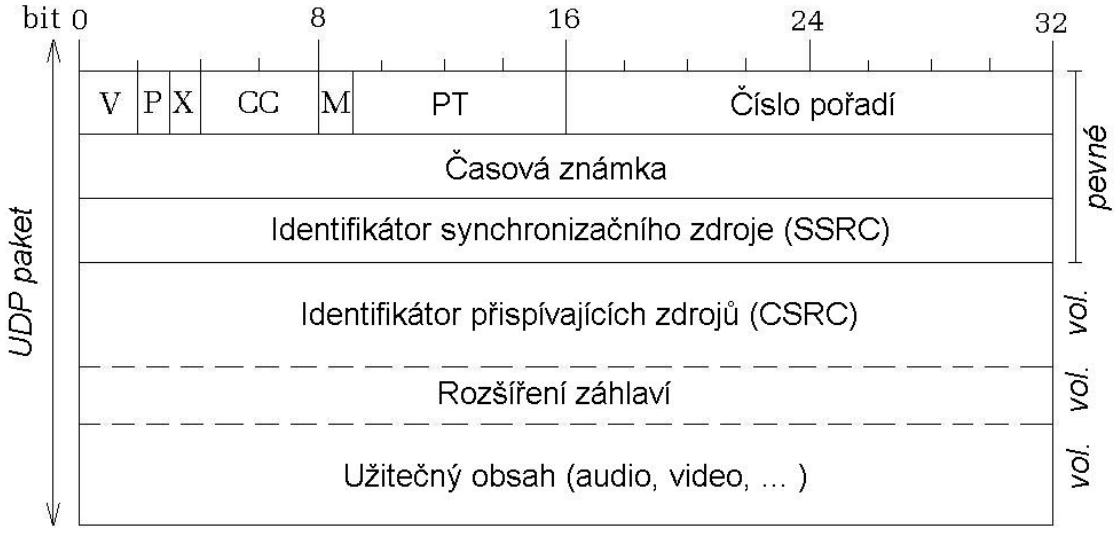
2.4.1. Formát záhlaví paketu RTP

Pro přehlednost následující obrázek znázorňuje příklad zapouzdření RTP paketu do paketu protokolů nižších vrstev.

Obr. 2.2: Možnost zapouzdření RTP paketu

Formát RTP záhlaví je zobrazen na obr. 2.3.

Obr. 2.3: Záhlaví datového RTP paketu



Prvních dvanáct oktetů tvoří pevné záhlaví RTP paketu a je přenášeno vždy. Pole CSRC identifikátoru je v záhlaví prezentováno pouze, když se v přenosu vyskytuje tzv. mixer. Pole záhlaví mají následující význam:

V (Version) – *2 bity* – udává o jakou verzi protokolu se jedná. Aktuální verze je V=2.

P (Padding) –*1 bit* – je tzv. doplnění, pokud je bit nastaven na 1, je na konci záhlaví paketu více oktetů (bytů), které nespadají k užitečnému obsahu (Payload) a jsou ignorovány. Z toho poslední oktet určuje kolik oktetů bylo doplněno. Doplnění se používá pro šifrovací algoritmy.

X (Extension) – *1 bit* – je tzv. rozšíření, pokud je bit nastaven na 1, následuje za pevným záhlavím další rozšíření záhlaví (viz následující kapitola)

CC (CSRC Count) – *4 bity* – určuje počet identifikátorů, následujících za pevným záhlavím (CSRC). Má-li toto pole nulovou hodnotu, je zdrojem synchronizace zdroj užitného obsahu.

M (Marker) – *1 bit* – je tzv. záložka, je definovaná konkrétním profilem media. Je určena k poskytování důležitých událostí, jako např. označení hranic rámce v datovém toku.

PT (Payload Type) – *7 bitů* – pole informuje o typu užitečného obsahu. Číslo PT je přiřazeno typu dat tabulkově a je specifikováno v RFC 3551.

Číslo pořadí (Sequence number) – *16 bitů* – je jedinečné číslo určující pozici pořadí paketu. Je voleno náhodně a inkrementováno s každým odeslaným paketem. Díky tomu je může příjemce detekovat ztrátu paketů.

Časová známka (Timestamp) – *32 bitů* – tzv. časová známka, vyjadřuje okamžik odebraní vzorku prvního oktetu datového paketu. Je odvozena z hodin, které se inkrementují lineárně v čase a zajišťuje synchronizaci a výpočet jitteru. Frekvence hodin je závislá na formátu přenášených dat (payload). Jsou-li RTP pakety generovány periodicky, je využito k časování vzorkovací frekvence a nevyužívá se systémových hodin. Aplikace vysílající už uložená data, používají časového formátu odvozeného z protokolu NTP (Network Time Protocol).

Identifikátor synchronizačního zdroje (Synchronization source identifier - SSCR) – *32 bitů* – je identifikátor synchronizačního zdroje (platí pro CC=0). Je volen náhodně a dva zdroje v jedné RTP relaci by neměly mít stejný SSCR identifikátor, případnou kolizi řeší patřičné mechanizmy.

Identifikátor přispívajících zdrojů (Contributing source identifiers - CSCR) – *0 až 15 položek po 32 bitech* – pole identifikuje zdroje, které přispívají do užitečného obsahu. Počet těchto zdrojů je dán CC polem. Celkem může přispívat až 15 zdrojů. Při vyšším počtu je nemožno je identifikovat. Užitečné obsahy zdrojů slučuje do jednoho zařízení zvané mixer

Užitečné data (Payload) – pole obsahující samotná data

2.4.2. Rozšíření záhlaví formátu RTP paketu

V případě nutnosti rozšíření stávajícího záhlaví je možné záhlaví rozšířit a implementovat tak nové typy formátu hlavičky RTP paketu přidáním dalších potřebných informací. K nastavení rozšířeného záhlaví slouží výše zmíněný bit rozšíření (X), nastavený na 1. Aplikace tohoto typu rozšíření není příliš častá, jelikož přídavné informace je většinou třeba přenášet v užitečném obsahu (payload).

Obr. 2.4: Formát rozšíření záhlaví RTP paketu

2.5. Popis protokolu RTCP

Již zmiňovaný protokol RTCP (Real-Time Transport Control Protocol) je nedílnou součástí funkce RTP jako transportního protokolu. Slouží jako signalizační protokol a sleduje kvalitu distribuce dat, čili monitoruje stav sítě. RTCP protokol tedy zajišťuje čtyři základní funkce:

- Poskytuje odezvu na kvalitu distribuce dat
- Nese tzv. kanonické jméno (CNAME), což je pevný identifikátor pro RTP zdroj. Jelikož SSCR identifikátor se může změnit při nějakém konfliktu nebo restartu programu, příjemce potřebuje jeho kanonické jméno k uchování cesty každého účastníka. CNAME je zahrnuto v popisu zdroje (SDES)
- Příjemci jej také potřebují k přidružení mnohonásobnému datovému toku od daných účastníků v relaci (např. k synchronizaci audia a videa), tzv. INTERMEDIA SYNCHRONIZATION
- Poslední funkcí je přenos minimální informace o řízení relace (např. RTCP paket typu BYE k opuštění relace účastníkem – nepovinná funkce)

Specifikace RTCP protokolu definuje několik typů paketů nesoucí určitý druh kontrolní informace. Typu jsou:

- **Sender report (SR)** – zpráva odesílatele (zdroje užitečného obsahu)
- **Reciever report (RR)** – zpráva příjemce (pro statistiky příjemců)
- **Source description items (SDES)** – popis zdroje
- **BYE** – indikace ukončení v relaci
- **Aplication-specific function (APP)** – specifikace aplikace

RTCP pakety posílají všichni účastníci spojení. Účastník vysílající RTP pakety zasílá i SR zprávu s obsahem počtu odeslaných paketů a oktetů a informaci užitnou k synchronizaci ostatních datových toků. Účastnící jako příjemci zasílají v pevných pravidelných intervalech RR zprávu pro ostatní zdroje, ze kterých přijímají. Interval posílání se liší dle počtu účastníků, z důvodu dodržení obsazení šířky pásma pro protokol RTCP, který je 5%. Zpráva obsahuje informaci o počtu nedoručených paketů, nejvyšší číslo pořadí a informaci o časově známce (Timestamp), které zdroj může použít k odhadu zpoždění v obou směrech a případně tomu přizpůsobit i datový tok.
Paket zprávy je vždy první a odesílá se i pokud nejsou odesílány žádné datové pakety. Všechny RTCP pakety musí obsahovat popis zdroje, pro identifikaci zdroje pomocí CNAME, případně další informace (viz kap. 2.5.1.)

2.5.1. Struktura RTCP paketu

RTCP zprávy se neposílají jako samostatné pakety, ale mohou být sloučen do tzv. složeného paketu libovolné délky, který zabalí se pošle v protokolu nižší vrstvy (transportní), např. UDP. Každý paket ze složeného paketu může být zpracován nezávisle bez požadavků na pořadí a kombinaci paketů. Nicméně vzhledem k funkci protokolu se zavádí následující omezení:

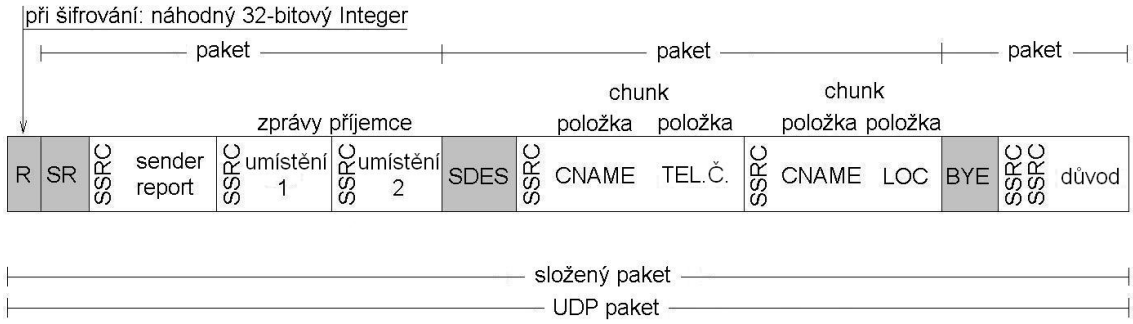
- Příjem statistik (v SR nebo RR zprávách) může být tak často zasílán jak to umožňuje omezení ze strany šířky pásma, proto každý složený paket musí obsahovat report paket.

- Nový příjemci potřebují přijmout SDES (CNAME) paket co nejdříve jak je to možné kvůli identifikaci zdroje a asociaci media (výjimku tvoří složené pakety rozdělené pro dílčí šifrování).
- Číslo typu paketu, které se může objevit ve složeném paketu nejdříve se musí týkat přičítání čísla konstantního bitu v prvním slově pro pravděpodobnost úspěšné validace RTCP paketu.

Proto všechny RTCP pakety musí být posílány ve složeném paketu o nejméně dvou samostatných paketech (RTCP zprávách) v následujícím formátu:

- Šifrovací prefix: týká se zašifrovaného složené paketu a musí mít 32-bitový prefix
- SR nebo RR paket: ve složeném paketu musí být vždy report paket, i za předpokladu že nebudou následovat žádná data a nebo, že za report paketem bude hned následovat BYE paket
- Doplňkové RR pakety: pokud počet zdrojů, pro které jsou přijímány zprávy statistik, překročí číslo 31, bude to číslo připojeno do dalšího doplňkového SR nebo RR paketu hned za tím prvním SR/RR paketem.
- SDES: paket obsahující popis zdroje (CNAME) musí být v každém složeném paketu (vyjma paketům určeným pro šifrování)
- BYE nebo APP: nakonec ostatní RTCP pakety, např. paket pro ukončení nebo definovaný aplikací

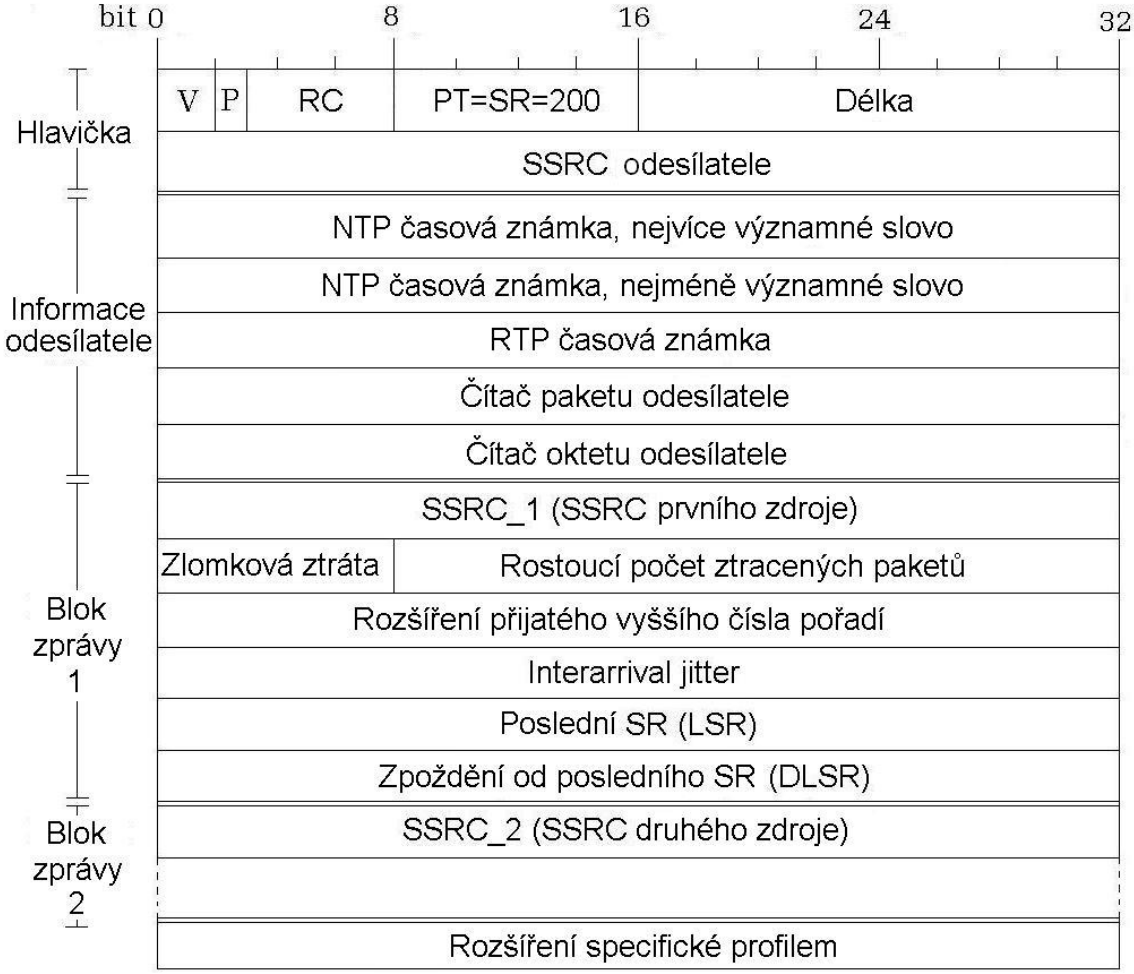
Obr. 2.5: Příklad struktury složeného RTCP paketu (compound packet)



2.5.2. Formát paketu zprávy odesílatele (Sender Report)

Paket zprávy odesílatele se skládá z tří sekcí následovanou čtvrtou rozšiřující profilem specifickou sekcí, pokud je definována. První sekce tvoří záhlaví paketu o velikosti osmi oktetů

Obr. 2.6: Formát RTCP paketu typu SR



Popis polí obsažených v SR paketu:

V (Version) – *2 bity* – Verze protokolu RTP (V=2), která je stejná jako i pro RTCP

P (Padding) – *1 bit* – je tzv. doplnění, pokud je bit nastaven na 1, je na konci záhlaví paketu více oktetů (bytů), které nespadají k řídícím informacím a mohou být ignorovány. Z toho poslední oktet určuje kolik oktetů bylo doplněno. Doplnění se používá pro šifrovací algoritmy. Ve složeném RTCP paketu je bit doplnění vyžadován v jednom individuálním paketu, protože složený paket je šifrován jako celek.

PT (Packet type) – *8 bitů* – určuje typ paketu pro SR je 200

RC (Reception report count) – *5 bitů* – Počet bloků zprávy obsažených v tomto paketu. Může mít nulovou hodnotu, ale pak je paket zbytečný

Délka (Length) – *16 bitů* – určuje délku tohoto RTCP paketu v 32-bitovým slově zmenšeným o jedničku. Zahrnuje i záhlaví a doplnění

SSRC odesílatele (SSRC of sender) – *32 bitů* – obsahuje identifikátor synchronizačního zdroje, pro původce tohoto paketu

NTP časová známka (NTP Timestamp) – *2x32 bitů* – indikuje čas kdy byl tento paket poslán a může být použit, v kombinaci s časovou známkou vrácenou v reportech příjemce od ostatních příjemců, k měření šíření paketu cestou tam a zpět

RTP časová známka (RTP Timestamp) – *32 bitů* – shoduje se s časem v NTP časové známce,ale ve stejné jednotce se stejným náhodným ofsetem jako RTP časová známka ve RTP data paketu. Shoda se využívá k interní synchronizaci medií (tzv. intermedia synchronization) viz kap 2.5.

Čítač paketu odesílatele (Sender's packet count) – *32 bitů* – celkové suma RTP data paketu, které byly odeslány od začátku vysílání do doby kdy byl vygenerován tento paket

Čítač oktetu odesílatele (Sender's octet count) – *32 bitů* – celkové suma oktetů užitečných dat (payload), nezahrnujících záhlaví a doplnění, které byly odeslány v RTP paketu od začátku vysílání do doby kdy byl vygenerován tento paket. Může být resetováno pokud odesílatel změní své SSRC

SSRC_n – *32 bitů* – SSRC identifikátor zdroje kterému informace v tomto bloku zprávy náleží (Blok zprávy 1)

Zlomková ztráta (Fraction lost) – *8 bitů* – ztracený zlomek RTP data paketu ze zdroje (SSRC_n) od doby poslání předešlého SR nebo RR paketu. Vyjadřuje se jako číslo s pevnou čárkou (fixed point number), která je v binární podobě v levém kraji pole. Tento zlomek je definován jako počet ztracených paketů (packet loss) děleno počet očekávaných paketů.

Rostoucí počet ztracených paketů(Cumulative number of packet loss) – *24 bitů* – celkový počet RTP data paketů ze zdroje (SSRC_n), které byly ztraceny od začátku přijímání. Číslo je definováno jako počet očekávaných paketů mínus počet přijatých paketů (včetně pozdně přijatých a duplikatů).

Rozšíření přijatého vyššího čísla pořadí (Extended highest sequence number recieved) – *32 bitů* – Spodních 16 bitů obsahuje přijaté vyšší číslo pořadí v RTP data paketu ze zdroje (SSRC_n) a více významných 16 bitů rozšiřuje číslo pořadí s odpovídajícím počtem cyklů, které mohou být udržovány podle algoritmu uváděného v RFC 3550 (Apendix A.1)

Interarrival jitter – *32 bitů* – je definován jako střední hodnota odchylky rozdílu v intervalu paketu, který porovnává přijímač pro dvojici paketů. [jitter - kolísaní změn zpoždění]

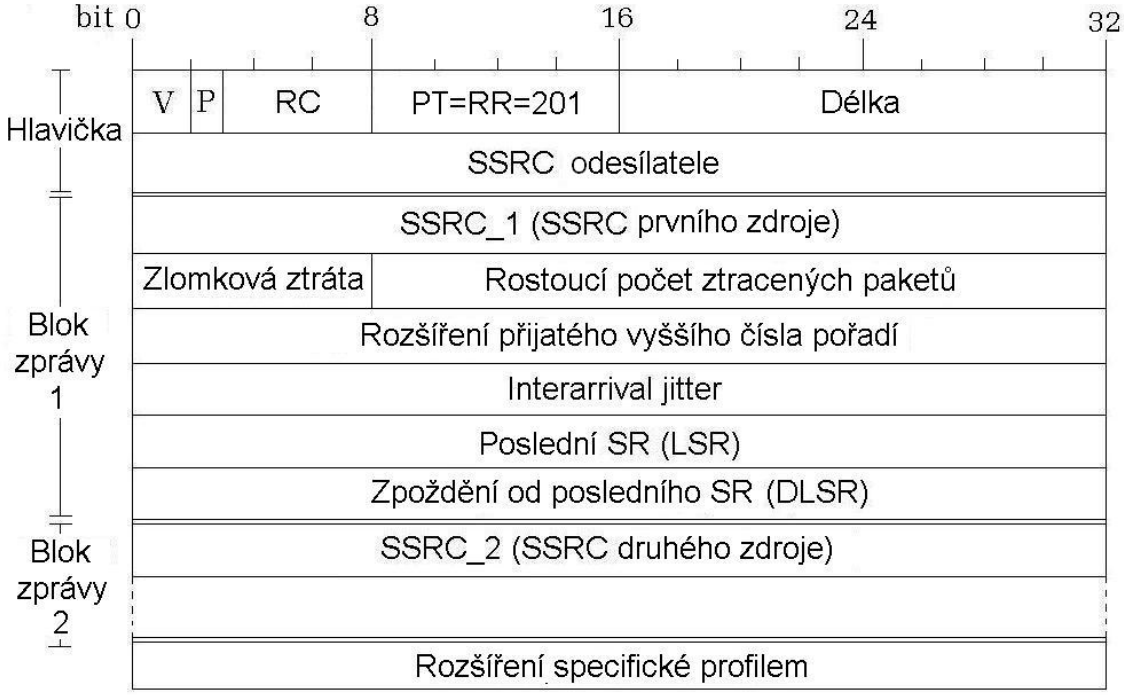
Poslední SR (Last SR) – *32 bitů* – středních 32 bitů z 64 v NTP časové známce přijatých jako část úplně posledního paketu RTCP zprávy odesílatele (SR) ze zdroje SSRC_n. Pokud ještě žádný SR přijat nebyl je nastavena nula.

Zpoždění od posledního SR (Delay since last SR - DLSR) – *32 bitů* – vyjadřuje jednotku o 1/65536 sekundy, mezi přijatým posledním SR (Last SR) paketem ze zdroje SSRC_n a posláním tohoto bloku ve zprávě příjemce.

2.5.3. Formát paketu zprávy příjemce (Reciever Report)

Formát paketu RR je stejný jako u SR paketu čili stejná pole mají stejný význam jako u SR paketu. Pouze pole typu paketu má konstantu 201 a sekce „Sender Info" o 20 oktetech je vypuštěno. Prázdný RR paket (pole RC=0) se vkládá hlavičky složeného paketu, pokud se nevysílají žádna data.

Obr. 2.7: Formát RTCP paketu typu RR

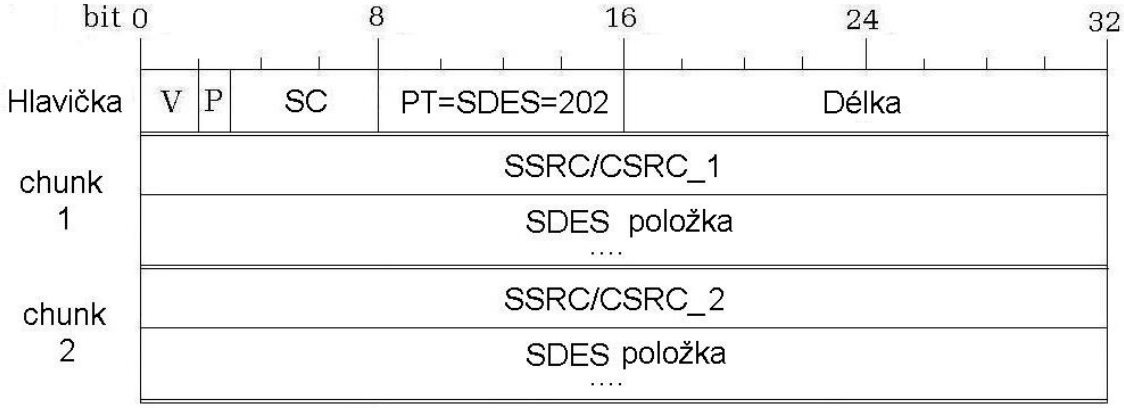


Popis polí je totožný jako u formátu SR paketu.

2.5.4. Formát paketu zprávy SDES (Source Description)

SDES paket má tří úrovňovou strukturu složenou ze záhlaví a nula a více tzv. „chunk“, kde každý z nich je tvořen položkami popisující zdroj popisovaný v „chunk“.

Obr. 2.8: Formát RTCP paketu typu SDES



Popis polí obsažených v SDES paketu:
V (Version) – *2 bity* – Verze protokolu (V=2)
P (Padding) – *1 bit* – stejný význam jako u záhlaví RTP protokolu
PT (Packet type) – *8 bitů* – určuje typ paketu pro SDES je 202
SC (Source count) – *5 bitů* – počet SSRC/CSRC „chunk“ obsažených v SDES paketu. Může mít nulovou hodnotu, ale pak je paket zbytečný

SDES položky se skládají z pole typu o velikosti oktetu, pole délky o velikosti oktetu a určující délku pole textu, který může být veliký až 255 oktetů. Položky vypadají následovně:

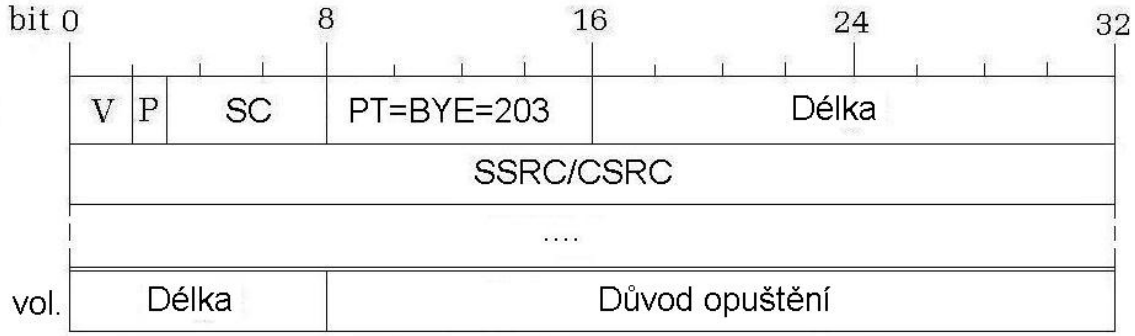
Obr. 2.9: Typy možných položek (SDES items) obsažených v „chunk“

8 bitů	8 bitů	Proměnlivé	
CNAME=1	Délka	Uživatel a jméno domény	···
NAME=2	Délka	Obecné jméno zdroje	···
EMAIL=3	Délka	Emailová adresa zdroje	···
TEL.Č.=4	Délka	Telefonní číslo zdroje	···
LOC=5	Délka	Geografická poloha stránky	···
TOOL=6	Délka	Jméno/verze aplikace zdroje	···
NOTE=7	Délka	Poznámka o zdroji	···
NOTE=7	Délka	Délka prefixu / Řetězec prefixu	···
....	Hodnota řetězce		···

2.5.5. Formát paketu zprávy BYE

BYE paket indikuje, že jeden nebo více zdrojů nejsou aktivní:

Obr. 2.10: Formát RTCP paketu typu BYE



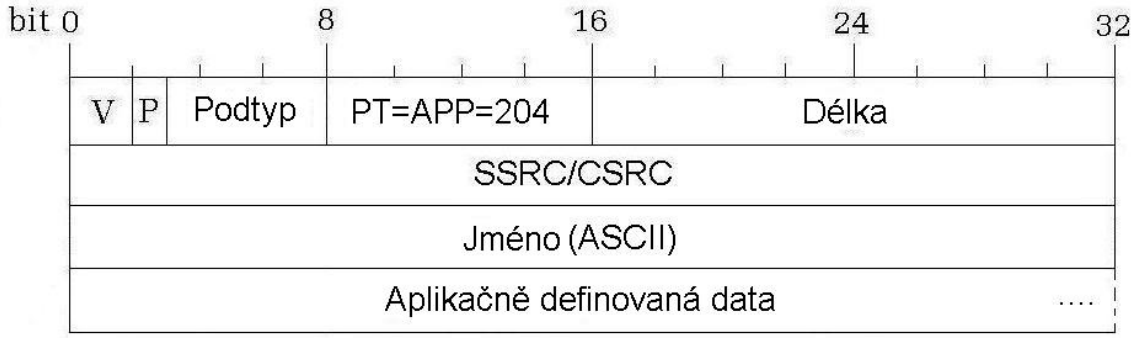
SC (Source count) – *32 bitů* – počet SSRC/CSRC identifikátorů obsažených v BYE paketu

Ostatní pole mají stejný význam jako v předchozích fomátech.

2.5.6. Formát paketu zprávy definované aplikací (APP)

Paket definovaný aplikací je určený k experimentálním účelům pro vývoj nových aplikacích a proto není třeba registrace typu pole (PT). Tyto pakety mohou být ignorovány. Paket je možné u organizace IANA, mající na starost správu, zaregistrovat bez polí „podtyp“ a „jméno“ po řádném otestování a oprávněném širším použití.

Obr. 2.11: Formát RTCP paketu typu APP



Popis polí obsažených v APP paketu:

V (Version) – *2 bity* – verze (V=2)

Podtyp (Subtype) – *5 bitů* – definuje podtyp APP paketů pod jedinečným jménem nebo může být určen pro libovolná data podmíněná aplikací

PT (Packet type) – *8 bitů* – obsahuje konstantu 204 identifikujících APP paket

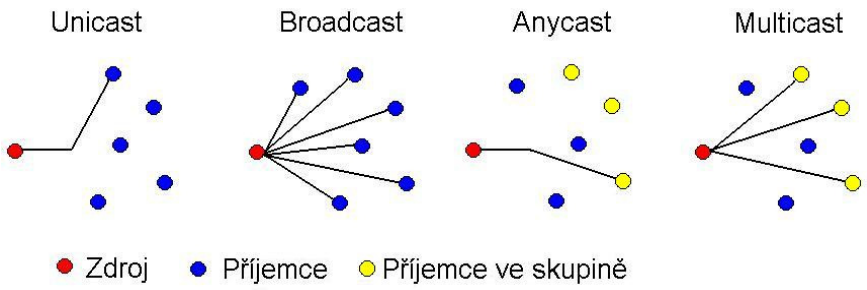
Jméno ASCII (name ASCII) – *32 bitů* – Jméno paketu je vybráno jako specifické pole a určuje jedinečnost APP paketu. Jeho zvolení je na tvůrci, stejně tak jako přidělený podtypu (Subtype) a je pak na aplikaci aby daný paket uměla zpracovat. Jméno je interpretováno čtyřmi ASCII znaky, kde se rozlišují malá a velká písmena

Aplikačně definovaná data (Application-dependent data) – *libovolná délka* – data nemusí být obsažena v APP paketu a jsou interpretována aplikací nikoli RTP samotným. Musí být násobkem 32 bitů.

3. Multimediální komunikace v IP sítích

V IP sítích se nejčastěji můžeme setkat s komunikací mezi dvěma procesy, které běží v různých nebo stejných podsítích. Komunikace tedy probíhá výměnou dat k jednomu cíli. Takovému nejznámějšímu a nejčastěji užívanému typu komunikace se říká „unicast“. Slovo je převzato z dalšího možné typu komunikace, tím je „broadcast“ neboli všesměrové vysílání, kdy se data posílají od zdroje ke všem účastníkům na síti. Dalším možným komunikačním typem kdy je požadováno posílat stejná data více účastníkům je „multicast“, kdy zdroj dat posílá data určité skupině příjemců. Posledním pojmem se kterým se můžeme setkat je „anycast“ což označuje komunikaci mezi zdrojem a jedním účastníkem z multicastové skupiny. Názorné pojmy graficky znázorňuje obr. 3.1.

Obr. 3.1: Možné formy komunikací mezi odesílatelem a příjemci



3.1. Úvod do IP Multicastu

Historie multicastu ve světě Internetu sahá do osmdesátých let minulého století, kdy student Stanfordské Univerzity Steve Deering pracoval na distribuovaném operačním systému, který využíval počítače spojené mezi sebou na nacházející se na více ethernetových segmentech. K realizaci bylo třeba použít skupinové vysílání v rámci IP paketů. Proto navrhl jak multicastové pakety adresovat, posílat sítí a také rozšířil směrovací protokol OSPF na bázi link-state o podporu multicastingu (MOSPF) a navrhl nový vektorový směrovací protokol na bázi RIP (DVMRP) a základy protokolu IGMP. Na jeho práci navazovali další a v roce 1992 vznikla experimentální IP síť Bone na principu multicastu.

V dnešní době není v komerční sféře multicast příliš podporován a využití najdeme spíše v akademických a výzkumných sítí. Nejčastější využití se jeví v přenosech multimediálních dat, jako distribuce vysílání audia, videa, videokonference, služby na bázi vyhledávání, distribuované simulace (např. síťové hry).

3.2. Adresace skupin IP Multicastu

Adresování multicastových skupin probíhá ve vyčleněné rozsahu adres D. Adresy jsou tedy v rozsahu 224.0.0.0 – 239.255.255.255, takže jejich první oktet začíná v binárním vyjádření posloupností 1110 a v IP paketu mohou být pouze jako adresa cíle.

Rezervace adres pro potřeby multicastového směrovaní je 224.0.0.0 – 224.0.0.255. šíření těchto paketů je omezeno na lokální segment pakety mají TTL=1 (viz kap. 3.3). Rezervované lokální adresy ukazuje následující tabulka 5.1.

Tab. 3.1: Přehled rezervovaných lokálních adres pro multicast

Lokální IP adresa	**Použití**
224.0.0.1	Všechny systému na segmentu
224.0.0.2	Všechny směrovače na segmentu
224.0.0.5	OSPF směrovače
224.0.0.6	OSPF vyhrazené směrovače
224.0.0.9	RIP verze 2 směrovače
224.0.0.10	EIGRP směrovače
224.0.0.12	DHCP servery
224.0.0.13	Všechny P1M směrovače
224.0.0.22	Směrovače s podporou IGMPv3
224.0.0.25	Směrovače-k-přepínačům (RGMP)

Další rozsah tzv. adresy s globálním rozsahem (Globally Scoped Address) je dán 224.0.1.0 – 224.0.1.255. Ty přiděluje organizace IANA (např. 224.0.1.1 pro NTP – Network Time Protocol). Zbývající rozsah 239.0.0.0 – 239.255.255.255 platí pro adresy s omezeným obsahem (Limited Scoped Address), které se mohou využít v nezávislých sítích vymezených konfigurací směrovačů.

3.3. Význam TTL v mutlicast paketech

TTL (Time To Live) v IP datagramu určuje jeho životnost, což znamená maximální počet průchodů přes směrovač. Jeho využití v komunikaci typu multicast je k omezení vzdálenosti jeho dostupnosti, avšak to není jediná cesta k omezování datagramů.
Omezení hodnotami TTL je následující:

- TTL = 0 … omezeno v rámci počítače
- TTL = 1 … omezeno v rámci jednoho segmentu (podsítě)

Při vyšších hodnotách parametru TTL se toto osmibitové číslo postupně postupným průchodem přes směrovače snižuje o jedničku.

Omezení má na starost směrovač, který má TTL hodnotu nadefinovanou a podle ní datagramy s nižší hodnotou zahazuje a s vyšším nebo stejným pouští dál.

3.4. Základy a princip přeposílání multicastu

U multicastového směrování je cílů kam musí doputovat stejná data ze zdroje více a je proto na směrovači, aby data replikoval, což pro něj přináší určitou zátěž. Dalším důležitým aspektem je zabránit duplikaci dat, aby příjemci nebyli zbytečně zahlcování stejnými daty.

Princip směrování je, že si směrovač si nastaví (případně ji nastaví ručně správce směrovače) směrovací tabulku (např. pomocí protokolu MBGP) a při příchodu datagramu mu sníží TTL o jedničku a není-li TTL hodnota nulová přešle paket dle směrovací tabulky na příslušné rozhraní dalšího směrovače (tzv. next hop). To se však komplice s multicastovou adresou, kde směrovači s cílová adresa nestačí a proto je důležité jakou má datagram zdrojovou adresu (unicast) a z jakého rozhraní směrovače dorazil (kvůli duplikaci datagramů). Ověření odkud multicastový datagram přišel se děje za pomocí unicastového datagramu, kde směrovač otestuje za pomocí tzv. „RPF check“ zda poslal tento datagram stejnou cestou na adresu, jakou měl multicastový paket uvedenou jako zdrojovou (proto si udržuje unicastovou směrovací tabulku). Jeli test v pořadku poračuje datagram k dalšímu zpracování v opačném případě je zahozen. Tento princip se nazývá princip přeposílání paketu (tzv. Reverse Path Forwarding).
Kvůli výkonu směrovačů se tento test neprovádí s každým datagramem, ale děje se tak s prvním datagramem a výsledek se uloží do paměti směrovače (tu je třeba obnovovat při změnách ve směrovacích tabulkách).

3.5. Směrovací principy pro multicast

V kap. 5.3 jsme uváděl využití směrovacích tabulek, ty se využívají jak unicastové tak i multicastové. Pro multicastové směrovací tabulky je třeba vyjít z určitých schémat, které určují jakým principem směrovat datagramy v IP síti, aby nedocházelo k jejich cyklování. Těmto schématům se říká „distribuční stromy“.
Existence distribučních stromů

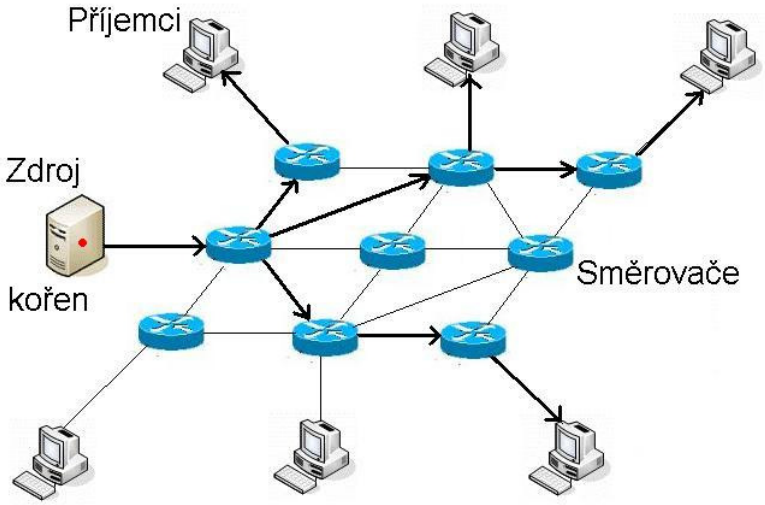
3.5.1. Distribuční stromy

Existují dva základní typy distribučních stromů:

- Zdrojový strom (Source Tree)
- Sdílený strom (Shared Tree)

Zdrojový strom, někdy též nazývaný „strom nejkratších cest“ (Shortest Path Tree - SPT), má svůj kořen právě ve zdroji multicastový dat a jeho listy jsou příjemci odebíraného datového toku ze zdroje. Ke značení stromu se užívá notace (S,G), kde S je adresa zdroje a G je adresa multicastové skupiny. Z obr. 3.2 je patrné, že pro různé zdroje v rámci stejné multicastové skupiny se tvoří další zdrojové stromy

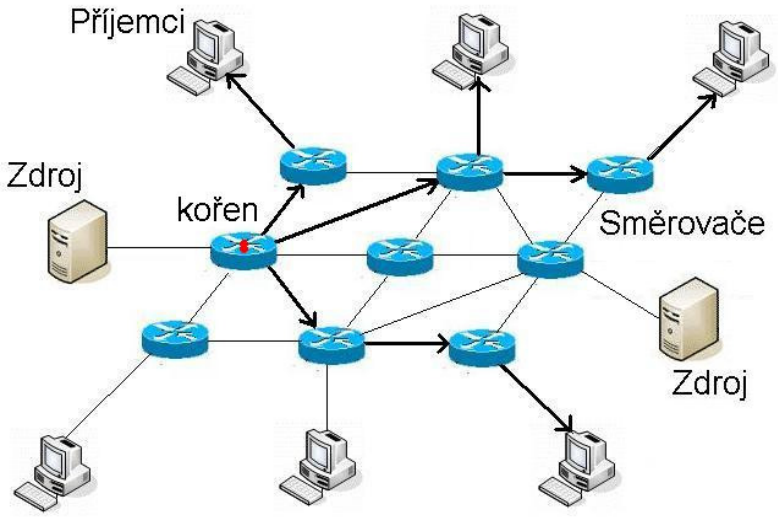
Obr. 3.2: Ukázka struktury zdrojového distribučního stromu



Sdílený strom se liší od zdrojového umístěním kořenu, který mají i v případě více zdrojů umístěn stále na jednom místě a je jím většinou samotný směrovač. Tomuto kořenu se tzv. „Rendezvous Point“ (RP). Jejich strukturu můžeme vidět na obr. 3.3. Sdílené stromy se ještě dělí na:

- Jednosměrné – zdroj pošle data unicastem ke kořenu a ten je rozdistribuuje
- Obousměrné – zdroj vysílá data směrem ke kořenu a zároveň po směru stromu k listům

Obr. 3.3: Ukázka struktury sdíleného distribučního stromu



3.5.2. SSM (Source-Specific Multicast)

Je multicastový model komunikace, který lze charakterizovat jako „jeden k mnoho“, čili jeden zdroj vysílající k mnoha příjemcům. V tomto způsobu přenosu je pevně daný datový zdroj. Vyskytuje-li jiný zdroj, který vysílá na stejnou multicastovou skupinu, bude ignorován, protože příjemci mají k určení zdroje nejen multicastovou adresu ale i unicastovou adresu zdroje. Z toho plyne ze chce-li účastník přijímat konkrétní data, zná zdresu zdroje (např. formou odkazu z webové stránky). To je také důvod proč nepotřebují RP (Rendezvous Point, viz kap. 3.5.1) a důvod proč nevadí, že dva zdroje vysílají do stejné multicastové skupiny, odpadá tu problém s alokací adres. Pro SSM je vyhrazen rozsah adres 232/8.

SSM má výhodu v nenáročnosti při realizaci aplikací a na síťový provoz, proto své hojné využití spatřuje v mnoha projektech zabývající se šířením multimediálního datového obsahu.

3.5.3. ASM (Any Source Multicast)

Druhým modelem je ASM, který se značí jako „mnoho k mnoho“, zdrojů je tedy v multicastové skupině více. Tento model se uplatňuje u videokonferencí, sdílení elektronických tabulí nebo u distribuovaných simulací a jeho mechanismus je složitý.

3.5.4. Základní skupiny směrovacích protokolů

Směrovací protokoly můžeme rozdělit do tří skupin:

- Dense mode (hustý režim)
- Spase mode (řídký režim)
- Link state

Dense mode protokoly

Využívají SSM a pracují na tzv. „push principu“, kde se předpokládá, že každý segment má aspoň jednoho příjemce a data jsou rozesílána na všechny podsítě. To však značně vytěžuje síť a směrovače, proto větvě stromu, kde se nenachází žádný příjemce vysílají tzv. „prune zprávu“ a daná větev je z distibučního

stromu „odříznuta“ (zpráva má časovou platnost a proto se musí obnovavat). Objeví-li se nový zájemce o příjem, zašle svému směrovači tzv. „graft zprávu“ a ten přestane „prune zprávu“ vysílat.

K dense mode protokolům patří DVMRP (Distance Vector Multicast Routing Protocol), PIM-DM (Protocol Independent Multicast – Dense Mode)

Spase mode protokoly

Tyto protokoly využívají sdílené stromy a používají tzv. „pull model“, který do sítě neposílá žádná data dokud si je někdo nevyžádá. Nový zájem o příjem tedy požádá svůj routek o připojení ke skupině tzv. zprávou „join“ a ten přepošle ke kořenu. Tím je sestavena větev stromu a datový tok je uskutečněn. Opět je třeba zprávu „join“ obnovovat po určité časovém intervalu. Tato metoda tolik nevytěžuje směrovače, jako předešlá.

K spase mode protokolům patří PIM-SM (Protocol Independent Multicast – Sparse Mode), CBT (Core Based Trees)

Link state protokoly

K distribuci používá zdrojové stromy, avšak nepoužívá „prune“ a „graft“ zprávy a namísto toho si vyměňuje informace o rychlosti stavu linek k příjemcům. Směrovač si tyto informace ukládá k sestavení stromu. Změna stavu linky vede k přepočítání v čemž tkví jeho značná nevýhoda.

Příklady zástupců jsou MOSPF (Multicast Open Shortest Path First), MSDP (Multicast Source Discovery Protocol), BGMP (Bordur Gateway Multicast Protocol).

3.6. Protokol IGMP (Internet Group Management Protocol)

Protokol je nejčastěji užíván klienty k signalizaci členství v multicastových skupinách, případně pro zprávy vektorových směrovacích protokolů (např. DVMRP). Protokol je popisován ve třech verzích, jak ukazují následující kapitoly.

3.6.1. IGMP verze 1

Tato verze už je poněkud zastaralá, avšak stále se najdou zařízení, které ho využívají. Formát IGMP zprávy ukazuje obr. 3.4.

Obr. 3.4: Formát IGMP zprávy verze 1

Význam jednotlivých polí:

Verze (Version) – *4 bity* – aktuální verze protokolu

Typ (Type) – *4 bity* – typ zprávy:

- dotaz na členství (Membership Query) [kód 1]
- report členství (Membership Report) [kód 2]

Nepoužito (Unused) – *8 bitů* – nastaveno na nulu, při příjmu ignorováno
Kontrolní součet (Checksum) – *16 bitů* – kontrolní součet IGMP zprávy
Adresa skupiny (Group address) – *32 bitů* – u dotazu na členství obsahuje multicastovou adresu, jinak nastaveno na 0.0.0.0

Význam a princip je patrný z popisu pole. Směrovač si kontroluje členství klientů po 60 sekundách zprávou „report členství“ na adresu 224.0.0.1. Odpověď zprávou „dotaz na členství“ nepřichází od všech klientů (kvůli úspoře přenosového pásma). Který klient to pošle je určeno náhodným číslem generovaným u klientů. Jakmile jeden pošle, ostatní už neposílají. Směrovač přestane posílat data klientům v případě, že 3x po sobě nepřijde žádný report.

3.6.2. IGMP verze 2

Formát IGMP zprávy ukazuje obr. 3.5.

Obr. 3.5: Formát IGMP zprávy verze 2

Význam jednotlivých polí:
Typ (Type) – *8 bitů* – typ zprávy:
- dotaz na členství (Membership Query) má dva podtypy
  - Obecný dotaz (General Query)
  - Dotaz specifické skupiny (Group-Specific Query) [kód 11] rozlišuje se obsahem pole „Adresa skupiny“
- report členství IGMPv1 (IGMPv1 Membership Report) [kód 12]
- report členství IGMPv2 (IGMPv2 Membership Report) [kód 16]
- opustit skupinu (Leave group) [kód 17]

Max. čas odezvy (Max resp time) – *8 bitů* – maximální čas k odeslání reportu v desetinách sekundy
Kontrolní součet (Checksum) – *16 bitů* – kontrolní součet IGMP zprávy
Adresa skupiny (Group address) – *32 bitů* – pro zprávy “report členství“, „opustit skupinu“ a „dotaz specifické skupiny“ obsahuje multicastovou adresu, jinak nastaveno na 0.0.0.0

Chce-li klient opustit skupinu zašle zprávu „opustit skupinu“ a směrovač reaguje zprávou „Dotaz specifické skupiny“ do skupiny. Max. čas odezvy je obvykle 1 sekunda. Je-li ještě někdo členem odpoví a směrovač posílá data dál.
Další novinka je, že dotazy (Query) posílá směrovač nejvyšším IP (po 125s). Směrovač s nižším čeká 400 sekund a pokud směrovač s vyšším IP po tu dobu mlčí začne vysílat. IGMPv2 je kompatibilní s první verzí.

3.6.3. IGMP verze 3

Třetí verze řeší stav, kdy je pro příjemce nežádoucí příjem od všech ostatních klientů ve skupině, proto se nepřihlašuje do skupiny (*,G), ale do skupiny (S,G), kde odebírá konkrétní data. Zprávy již nemají konstantní délku a pole „Max. čas odezvy“ se změnil a „Max. kód odezvy“ (do hodnoty 128 má stejný význam, nad 128 se mění význam).

Verze je opět kompatibilní s předchozími.

3.7. Transport IP multicastu na síťovém rozhraní

Jen krátce bych se zmínil o mapováni multicastu na MAC adresu v nižší síťové vrstvě.
Ethernetové multicastové rámce se rozlišují v cílové adrese nastavením bitu v nultém oktetu na hodnotu log. 1. IP multicast adresa začíná 25 bitovým prefixem, viz obr. 3.6.

Obr. 3.6: Ethernetový rámec s prefixem multicastové adresy

oktet 0	oktet 1	oktet 2	oktet 3	oktet 4	oktet 5
00000001	00000000	01011110	0xxxxxxx	xxxxxxxx	xxxxxxxx

Je patrné, že pro mapování IP adresy do MAC s prefixem o velikosti 25 bitů se používá zbývajících 23 bitů. Avšak IP adresa má 32 bitů. Z IP adresy můžeme odebrat první 4 bity prefixu multicastové IP adresy (1110). U zbylých 28 se odřízne posledních 5 bitů a zbytek se namapuje do MAC adresy, tím je možné, že jedna multicastová MAC adresa může odpovídat více IP multicast adresám, konkrétně 2^5, čili 32 IP adres.

3.8. Spolehlivý multicast

Komunikaci s garantovanou spolehlivostí známe například ze spolehlivého protokolu TCP, který ovšem pracuje na bázi unicastu. Spolehlivá komunikace na bázi multicastu je stále ve stavu experimentů a na vínku ji má nespočet projektů, přesto existuje mnoho protokolů ze stran různých firem, které jsou uzavřené. Organizace která se zabývá studiem spolehlivého multicastu je The Reliable Multicast Research Group (RMRG) a je právněna za standardizaci technik pro IETF (Internet Engineering Task Force).

Se spolehlivým multicastem je spojena řada problému, především jde o možné četné kolize v rozsáhlé síti resp. Internetu. Spolehlivý multicast má za následek poškozování dat komunikující na unicastové bázi (TCP). Co dále za hrozbu přestavují, je potenciální exploze paketů při řízení provozu (např. imploze potvrzení typu ACK, NAK, stavové zprávy). Další problém představuje přeposílání paketů k opravám poškozených dat na straně příjemců, který je v případě ztrát nedokáže uspokojit.

Jedna z možný metod jak řešit tyto problémy je omezení množství ACK/NAK na takový počet, kdy se bude stále zachována spolehlivost. Což se snaží řešit protokol RMTP (viz kap. 3.8.1.2).

Pro IETF zajišťuje RMRG, globální problém multicastu (feedback bandwight, repair latency), standard stále ve vývoji

3.8.1. Protokoly založené na zpětné vazbě

Zajišťují spolehlivost právě pomocí zpětné vazby

3.8.1.1 SRM (Scalable Reliable Multicast)

Protokol je založen na principu, kdy opravu případně přeposlání paketu může zajistit libovolný příjemce, který je nejblíže a má k dispozici požadovaná data. Členové skupiny v SRM posílají v relaci s nízkou frekvencí zprávy do skupiny,aby jejich soused se mohl dovědět jejich status, aby bylo možno měřit zpoždění mezi členy skupiny, dovídat se o členech skupiny a detekovat poslední pakety. Tyto zprávy jsou navrhnuty aby zabáraly pouze 5% z kapacity provozu v relaci.

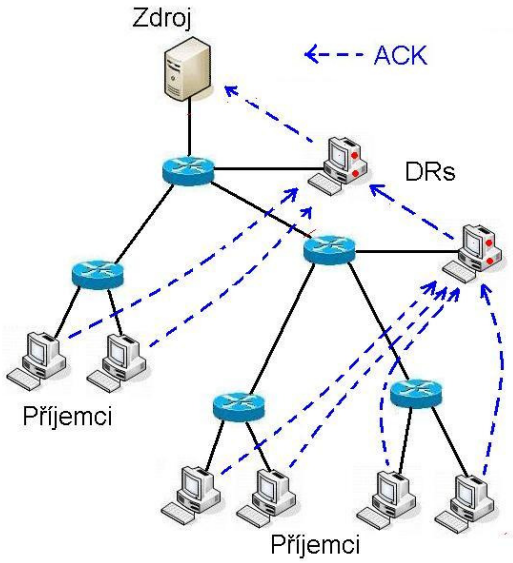
Příjemci , kterým chybí data čekají náhodně zvolenou dobu než vyšlou žádost na opravu. Čímž se potlačují duplikované požadavky. Stejný proces je pro posílání oprav. Požadavky na opravu mohou být zajištěny i odesílatelem (zrojem) dat, pokud je nablízku. Pokud má nějaký přijímač stejnou žádost jaká už byla právě na síť odeslána jiným příjemcem, svou žádost nepošle a vyčkává, kvůli duplicitě paketu žádosti. Až je dotyčný opraven, pošle tu svou žádost.

Tento protokol byl navržen původně pro data interaktivních konferencí, kde je jako nastroj tabule (tzv. whiteboard conferences)

3.8.1.2 RMT (Reliable Multicast Transport)

Jak už bylo řečeno protokol je navržen pro omezení provozu potvrzovacích paketů typu ACK/NAK. Protokol tak nabízí tzv. „vyhrazené přijímače“ (Designed Recievers - DRs), kteří sbírají stavové zprávy z uzlů v lokální RMTP doméně a poskytují tak opravy (přeposílání chybějících dat), pokud jsou dostupné. Příjemci tak vedou administrativní zprávy unicastem přímo k vyhrazenému přijímači (DRs). DRs tak poskytuje lokální opravy a udržuje kontrolu řízení s dalším DRs v hierarchii v případě, že žádosti na data nemůže vyhovět (viz obr. 3.7)

Obr. 3.7: Hierarchie a postupování při opravách v protokolu RMTP



3.8.1.3 PGM (Pragmatic General Multicast)

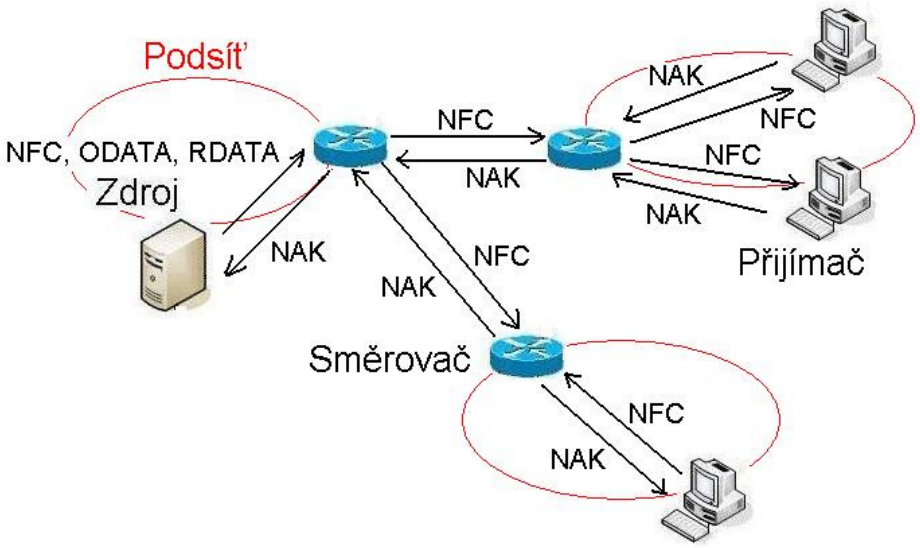
Tento protokol byl navrhnut v roce 1998, za účelem jednoduchosti a schopnosti optimálního vlivu směrovačů v síti. Je to příklad protokolu, , který obchází transportní protokol UDP a má rozhraní přímo na IP protokolu. PGM nenabízí pojetí ve stylu členství ve skupinách jako SRM protokol. PGM definuje několik typů datových paketů:

- ODATA: původní obsah dat
- NAK: negativní přijetí dat (unicast)
- NFC: NAK potvrzení
- RDATA: přeposlání (oprava)
- SPM: zpráva cesty zdroje

Každý PGM paket obsahuje TSI (*Transport Session Identifier*) k identifikaci relace a zdroje dat, takže mnohonásobné relace mohou být lehčeji identifikovatelné směrovači a příjemci. ODATA, NFC, RDATA SPM pakety putují po směru šíření dat (od zdroje) distribučním stromem a NAK pakety proti směry šíření (směrem ke zdroji).

PGM je navržen pro široké rozšíření a proto se hodí pro Real-Time aplikace. Čili je zde potřeba časová souslednost. Tato potřeba je zaopatřena vysílacím oknem, které definuje datové posuvné okno tak, že když zdroj nepřijme žádné NAK nebo určený lokální „přeposílač„ během doby , okno se zavře, data jednoduše nejsou k dispozici.
Dojde-li na ztracený paket, přijímač vyšle NAK na svůj směrovač, ten si přeposílané NAK ponechává dokud neobdrží NCF nebo RDATA, která indikují že oprávná data jsou poslána. Cesta směrem ke zdroji musí být stejná jako byla směrem dolů po distribuci stromu. To zajišťuje SPM paket, který je prokládaný s ODATA pakety při jeho šíření stromem, zajišťuje tak i relaci. Směrovače si cestu uchovávají k přeposílání NAK paketů. SPM paket také varuje příjemce, že data ve vysílacím okně jsou stará

Obr. 3.8: Topologie a princip protokolu PGM



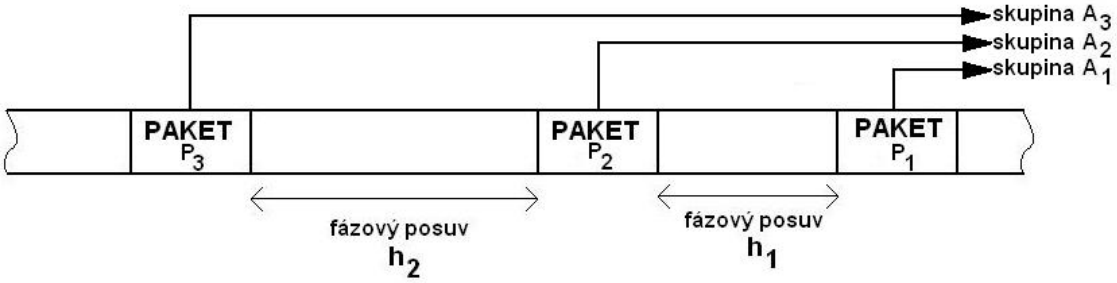
3.8.2. Projekt Multiecho

Jedná se o síťový projekt systému, který by umožňoval efektivní a spolehlivou distribuci velkých objemů dat bez nutnosti zpětné vazby. Jeho zaměření bylo především na oblast sítí s gigabitovou kapacitou a tak dostatečným volným pásmem pro přenos.

Tento projekt proběhl na Masarykově Univerzitě před šesti lety a jeho hlavní myšlenka byla představována jako tzv. opakovaná ozvěna (proto „Multiecho“), kdy se ten samý datový tok vysílá s určitým zpožděním v několika multicastových skupinách.

Organizace vysílání spočívá ve zvolení si x multicastových skupin, které by měly adresy A_1 až A_x, stanovení velikosti datového segmentu (segmentem se rozumí část určité délky z rozděleného datového toku) a jeho následné vysílání v samostatném paketu. Fázové posuvy mezi jednotlivými datovými toky, určenými pro příslušnou multicastovou skupiny, budou v počtě $x-1$ a budou nabývat hodnot h_1 až h_{x-1}. Hodnota h_i (kde i je od 1 do $x-1$) udává o kolik segmentů datového toku je zpožděno vysílání pro skupinu A_{i+1} oproti vysílání pro skupinu A_i. Vysílací aplikace vysílá zároveň pro všechny multicastové skupiny s patřičnými odstupy mezi nimi.

Obr. 3.9: Schéma multicastového vysílání v projektu Multiecho



Organizace na straně přijímací aplikace je následující. Přijímací aplikace se přihlásí přednostně do první vysílací skupiny A_1. Přijímané pakety obsahují pořadové čísla pro identifikaci ztráty některého paketu přijímačem. V případě takové identifikace se určí na základě známých hodnot fázových posuvů, která skupina bude vysílat zpožděná data nejdříve a přihlásí se do ní.

V rámci implementace se pro vysílací aplikaci předpokládalo, že frekvence vysílání se může měnit vlivem kolísaní zátěže pro vysílací aplikaci a proto se udržovaly odstupy v rámci stanovené tolerance mezi multicastovými skupinami. Přijímač byl složitější aplikací kvůli vícevláknové struktuře (jedno řídící vlákno, které inicializuje v rámci výpadku nové vlákno pro příjem dat s nejbližším fázovým posuvem a deaktivuje vlákna po jejich splnění úkolu).

Protokol Multiecho se funkčně testoval na lokální síti s uměle generovanými výpadky a proběhl úspěšně. Při testech na páteřní síti CESNET2 (vysílač v Praze, přijímače – Brno, Plzeň, Liberec, České Budějovice) projekt narazil na problémy ohledně funkčního příjmu na některých uzlech vlivem konfigurace směrování multicastu. Na některých uzlech byl příjem se ztrátovostí 80%, což není přípustné.
Z těchto důvodů byla metoda vhodná pouze např. k využití v akademických sítích, pro distribuci vekých dat (např. softwarové balíky), nikoli však k distribuci multimediálních dat, konkrétne IPTV. Otázkou je řešení v dnešní době, kdy technika směrování pokročila dopředu a podpora multicastu ve směrovačích je určitě na lepší úrovni.

Výhodou a důvodem proč jsem se o tomto systému zmínil je právě realizace vysílání dat generovaných v reálném čase, které právě metody založené na principu FEC (*Forward Error Correction*) neumožňují.

4. Popis řešeného systému IPTV

V následujících kapitolách se zabývám praktickým řešením navrhovaného systému.

4.1. Zpracování řešení přenosu specifických informací

Úkolem bylo vyřešit jakým způsobem přenést zpětně ke zdroji specifické data, obsahující informace ke zpracování. V našem případě se jednalo o přenos výsledků hlasování. Jelikož zpětnou vazbu (klient-server) tvoří signalizační protokol RTCP, který je nezbytnou součástí protokolu RTP a tak i přenosu v reálném čase, zaměřil jsem právě na možnost uskutečnit přenos specifických dat tímto protokolem.

Pro experimentální a vývojové účely zde slouží aplikačně definovaný paket (APP), jehož popis je v kap. 3.4.6.

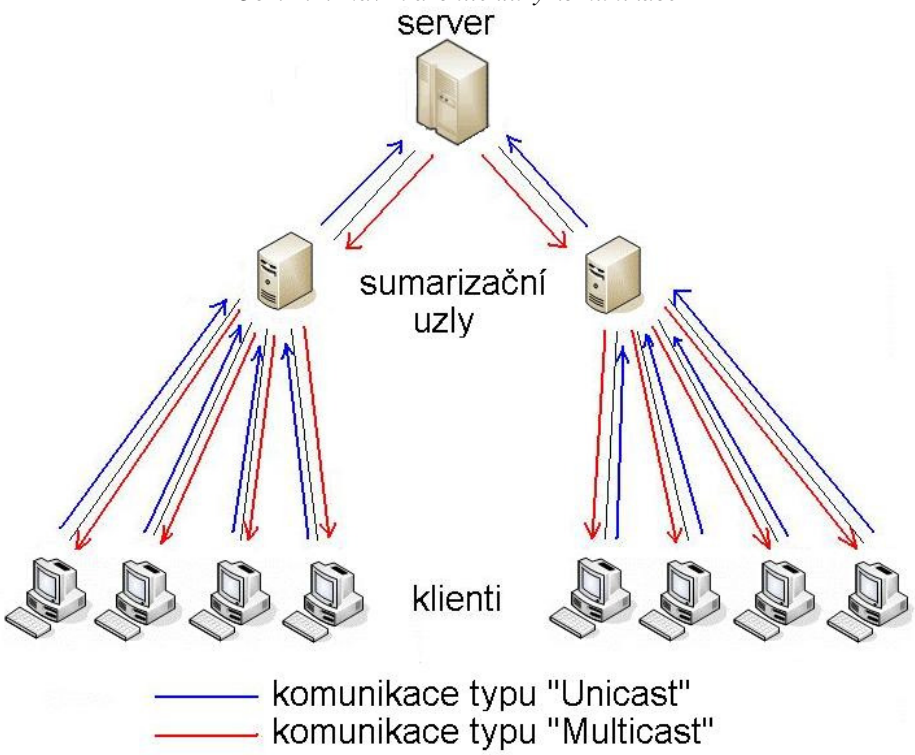
Pro návrh paketu je třeba brát na vědomí několik faktů:

- Jak bude vypadat struktura sítě a připojení uzlů. Návrh struktury sítě, kde je třeba počítat s existencí uzlů kde bude docházet k sumarizaci dat.
- Kolik možných příjemců se bude podílet komunikaci, případně kolik jich připadne na sumarizační uzel (V mém případě budu počítat s jednoduchým modelem, kvůli možné realizaci)
- Co je třeba zohlednit při návrhu obecného formátu paketu APP

4.1.1. Struktura sítě pro komunikaci

Struktura sítě se bude skládat ze serveru, který bude představovat zdroj multimediálního obsahu a ze kterého budou pomocí multicastu šířena data. Dalším prvkem bude sumarizační uzel, který bude mít za úkol rozesílat RTP data příjemcům, kteří budou mít o datový tok zájem a budou tak připojeni v příslušné multicastové skupině. Hlavní úlohou sumarizačního uzlu bude sběr signalizačních dat, konkrétně příjem APP paketů od příjemců a jejich následné sčítání hodnot výsledků z patřičného hlasování případně jiných možných statistických dat (např. statistika sledovanosti atp.) a zaslání na server pomocí komunikací typu unicast. Posledním článkem jsou právě koncový účastníci, kteří přijímají RTP data šířena multicastově a zpětně posílají jak zprávy pro monitoring tak i zprávy aplikačně definované pro výsledky hlasování prostřednictvím protokolu RTCP

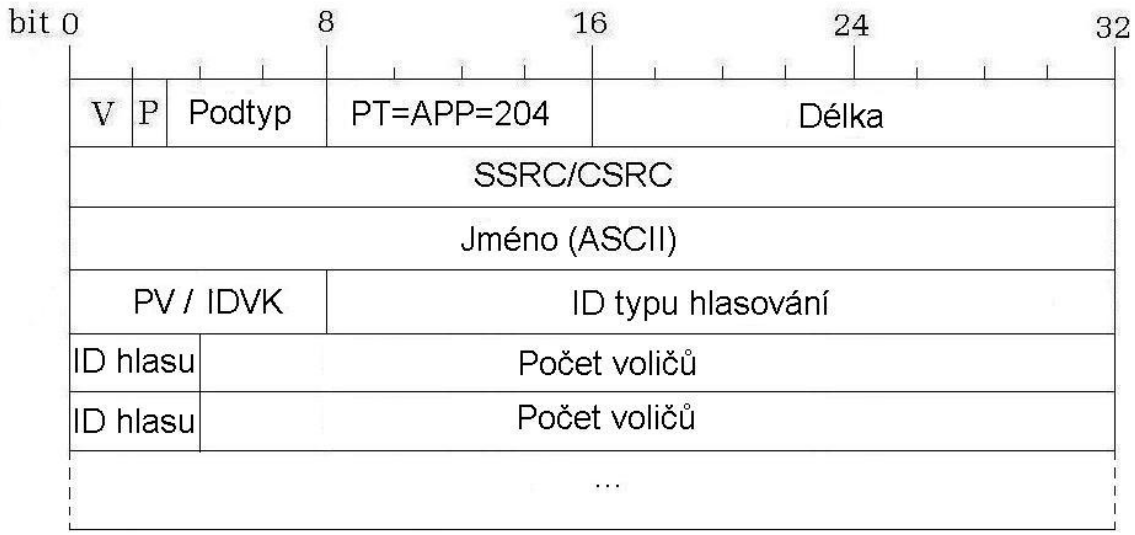
Obr. 4.1: Návrh architektury komunikace



4.1.2. Návrh RTCP zprávy typu APP pro přenos výsledku hlasování

V návrhu vycházím z faktu ze se budou přenášet směrem ke zdroji vysílání dva podtypy (subtype) těchto paktů. První typ je pro samotného koncového klienta, kde se bude Uskutečňovat příslušná volba hlasování, druhý typ bude odeslání paketu od sumarizačního uzlu, kde dochází k součtu výsledků příjemců spadající pod týž uzel.
Návrh paketu bude vypadat následovně:

Obr. 4.2: Formát RTCP paketu typu APP pro přenos hlasování



Podtyp (Subtype) – *5 bitů* – určuje zda se jedná o paket od koncového příjemce nebo zda se jedná o paket od uzlu kde dochází k sumarizací výsledku hlasování.
Podtyp – může nabývat těchto hodnot:
- 00001 (dekadická hodnota 1) – paket přichází od koncového účastníka (klienta), pole NOR má nulovou hodnotu
- 00010 (dekadická hodnota 2) – paket přichází od sumarizačního uzlu a pole NOR má přiřazenou hodnotu

Jméno ASCII (name ASCII) – *32 bitů* – pole slouží identifikaci a specifikaci. Přiřazením znaků do pole označíme, že se jedná o paket náležící aplikaci pro hlasování. Označení pole bude např. ASCII znaky HLAS, pak bude pole kódováno takto:

01001000 01001100 01000001 01010011

každému oktetu přísluší jeden ASCII znak (8 bitový)

IDVK (ID Volby Klienta) – *8 bitů* – typ odpovědi (hlasu) klienta. Pole je určeno pro koncového účastníka a určuje jak klient hlasoval. Pro hlasování je možný počet odpovědí v rozsahu abecedy, kde je každé odpovědi přiřazeno číslo od 0 do 32.

ID typu hlasování – *24 bitů* – obsahuje identifikační číslo, které určuje o jaké hlasování se jedná, resp. k jakému hlasování má výsledný hlas klienta přiřadit. Generování čísla závisí na aplikaci serveru. Může být generováno z hodin nebo jen postupně inkrementováno, záleží na tom jak dlouho bude celé hlasování aktivní a do kdy tedy bude mít toto pole platnost.

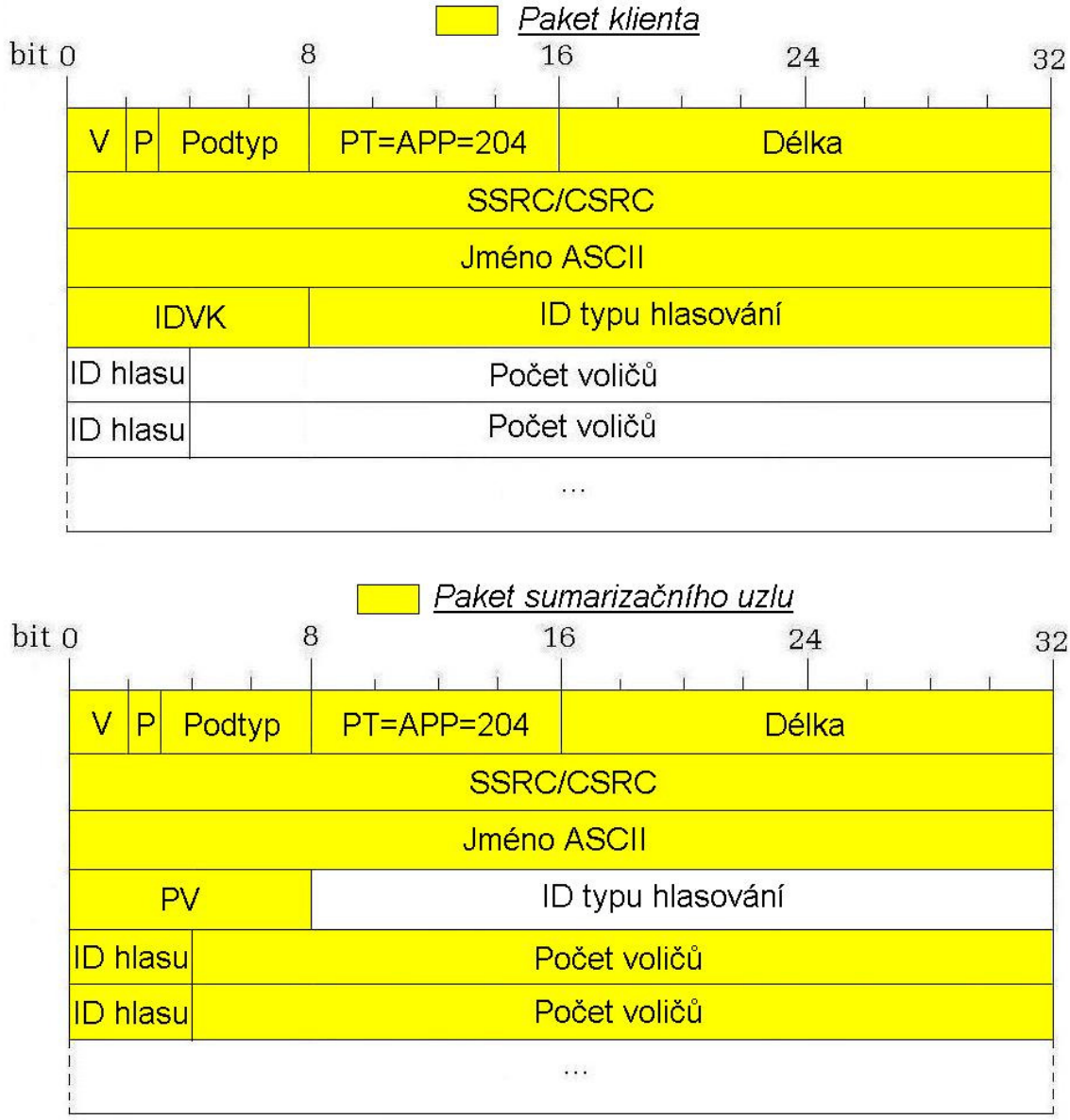
PV (Počet Výsledků) – *8 bitů* – Pole je určeno pro sumarizační uzel a udává počet typu odpovědí, které uzel obdržel během hlasování. Maximální počet odpovědí může je stejný jako u volby klienta (IDVK). Aby nebyly zbytečně alokovány všechny pole, čili počet všech možných odpovědí konkrétního hlasování (ID typu hlasování), bude se číslo PV zvyšovat postupně v závislosti kolik možných typů už od účastníků sumarizační uzel obdržel. Např. v situaci kdy je na výběr A,B,C,D možných odpovědí a všichni účastníci si vyberou pouze A nebo B, bude číslo PV rovno dvěma a rozšíří se paket o dvě slova (slovo má délku 32 bitů a v tomto případě obsahu pole „ID hlasu“ a „Počet voličů“).
Číslo PV se může pohybovat v maximálním rozmezí jako IDVK, proto i příslušné hlasování je omezeno maximálně 32 možnými odpovědi.

ID hlasu – *4 bity* – pole je určeno pro sumarizační uzel a slouží jako identifikace typu odpovědi, které se v hlasování vyskytlo (např. odpověď „A“, kterému je přiřazeno konkrétní číslo v tomto poli potažmo v poli „IDVK“)

Počet voličů – *28 bitů* – pole je určeno pro sumarizační uzel a udává počet odpovědí typu určeného v předešlém poli („ID hlasu“), které uzel přijal od klientů.

Pro přehled, jaké pole využívá klient a jaké sumarizační uzel, jsou pole vyznačená žlutě na obr.4.3.

Obr. 4.3: Formát RTCP paketu typu APP pro klienta a sumarizační uzel



4.1.3. Popis komunikace programů KLIENT a SERVER

K simulaci hlasování jsem vytvořil dva programy pro vzájemnou komunikaci. Program KLIENT, který má za úkol reagovat příslušným hlasem na přijatou anketu a program SERVER, který vytváří a zasílá ankety klientům a zpětně sbírá výsledky hlasování a ukládá je, aby bylo možné nahlédnou na statistiku voleb a příslušných anket.

Oba dva programy se spouští s následujícími parametry:

KLIENT: *UDP_client_v6.exe <nazev_pocitace> <port>*

např. : *UDP_client_v6.exe dsobotkapc 6970*

SERVER : *UDP_server_v6.exe <port>*

např. : *UDP_server_v6.exe 6970*

Pro přehled ještě uvádím seznam a popis funkcí, které má uživatel k dispozici v záhlaví programu, tzv. menu, pro ovládání aplikace a které vyvolává stiskem příslušné klávesy shodující se s písmenem v hranaté závorce.

SERVER:

- ***[A]nketa*** – vytvoření nové ankety
- ***[O]deslat*** – odeslání vytvořené ankety
- ***[K]onec*** – ukončení běhu programu
- ***[V]ysledky*** – zobrazení výsledků, sumarizace všech anket a počtu hlasování

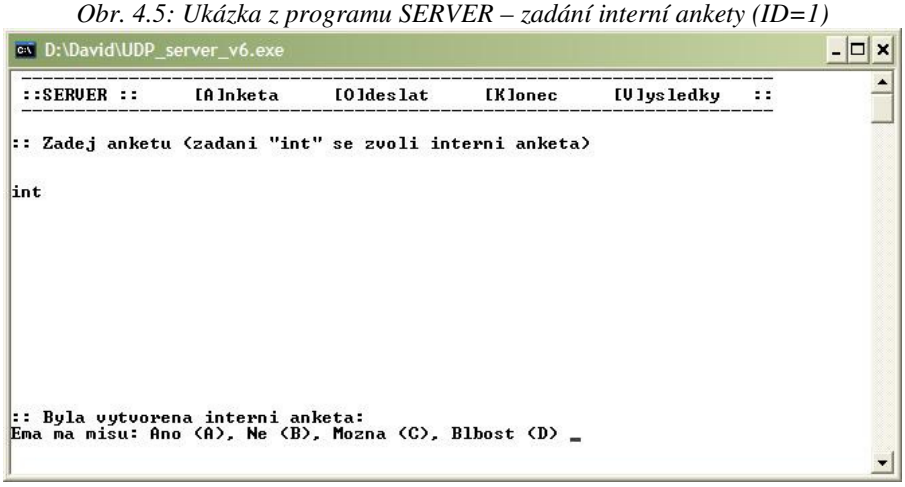
KLIENT:

- ***[K]onec*** – ukončení běhu programu
- ***[P]oslat*** – zaslaní dalších hlasování pro aktuální anketu (simulace dalších klientů)

Klient potřebuje ke svému chodu mít spuštěný Server, proto při spuštění klienta se zobrazí oznámení o nutném zapnutí serveru a poté se v klientovi pokračuje stiskem klávesy *ENTER*.

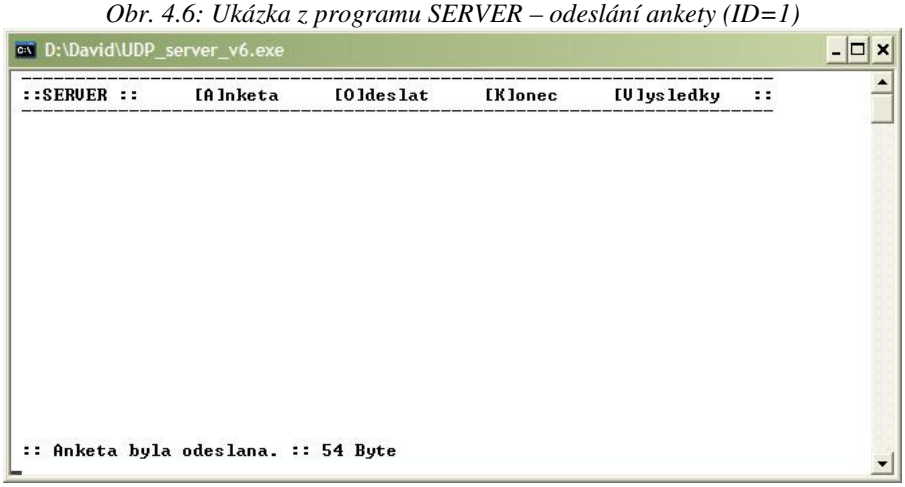
Pokud oba programy "běží" vytvoří se na serveru anketa stiskem klávesy „A" (příp. „a"). Zadání může být ručně, případně zadáním slova „int" se vytvoří přednastavená anketa(viz obr.4.5)

Obr. 4.5: Ukázka z programu SERVER – zadání interní ankety (ID=1)



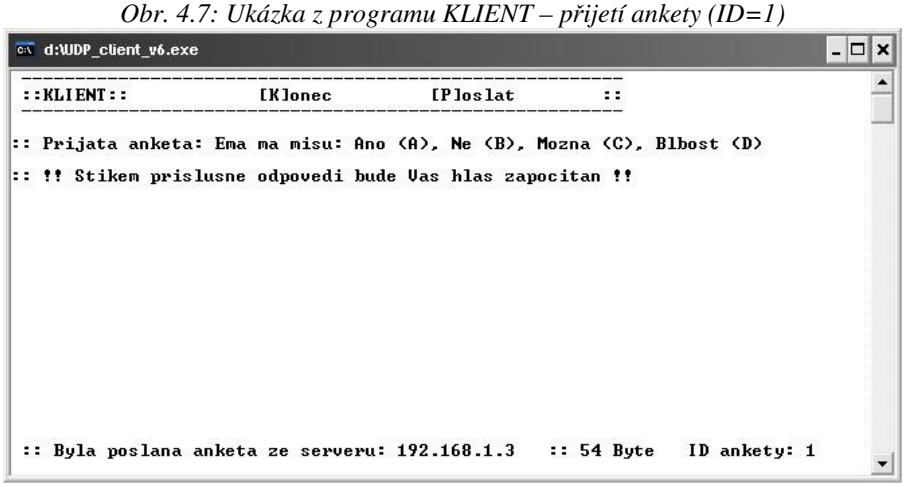
Dalším krokem je odeslaní ankety. To se provede stiskem klávesy „O" (příp. „o"). Na serveru se dole v okně objeví pouze status pro odeslaní, že data byla odeslána a jejich velikost (viz obr.4.6)

Obr. 4.6: Ukázka z programu SERVER – odeslání ankety (ID=1)



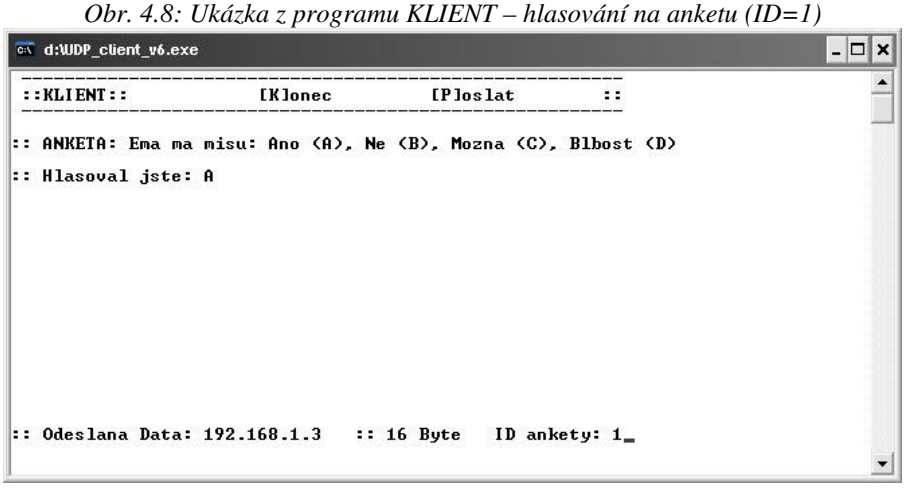
Současně na straně klienta se objeví upozornění o přijetí nové ankety, jejím zobrazení a výzva na hlasování stiskem příslušné klávesy pro příslušnou možnost (viz obr.4.7). Na spodu okna je opět stavový řádek, ukazující z jaké IP byla anketa přijata, kolik dat bylo přijato a jaké je ID ankety (identifikační číslo)

Obr. 4.7: Ukázka z programu KLIENT – přijetí ankety (ID=1)



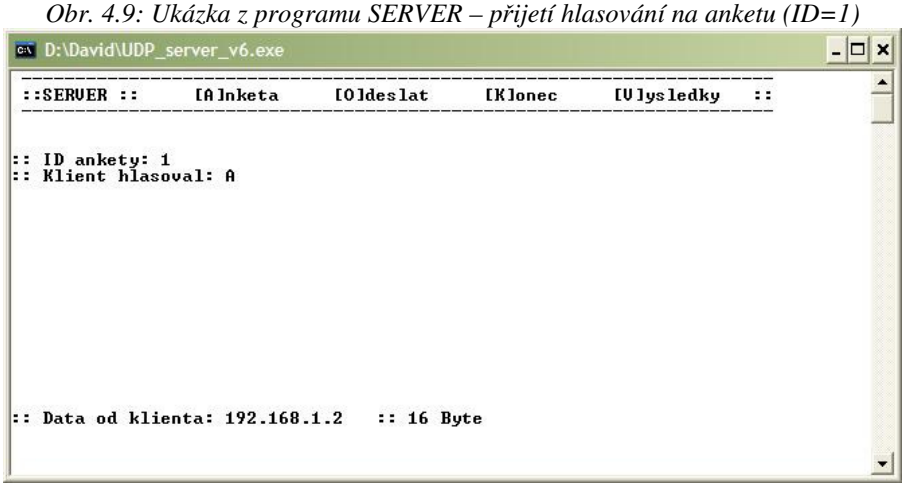
Další obrázek (viz obr.4.8) ukazuje hlasování (v našem příkladě stisk klávesy „A“ pro možnost A). Tuto informaci potvrdí opět klient hlášením jak jsem hlasoval. Ve stavovém spodním řádku je vidět, že paket byl odeslán zároveň na server (jeho IP), kolik byte bylo odesláno a k jaké anketě se hlasování vztahovalo.

Obr. 4.8: Ukázka z programu KLIENT – hlasování na anketu (ID=1)



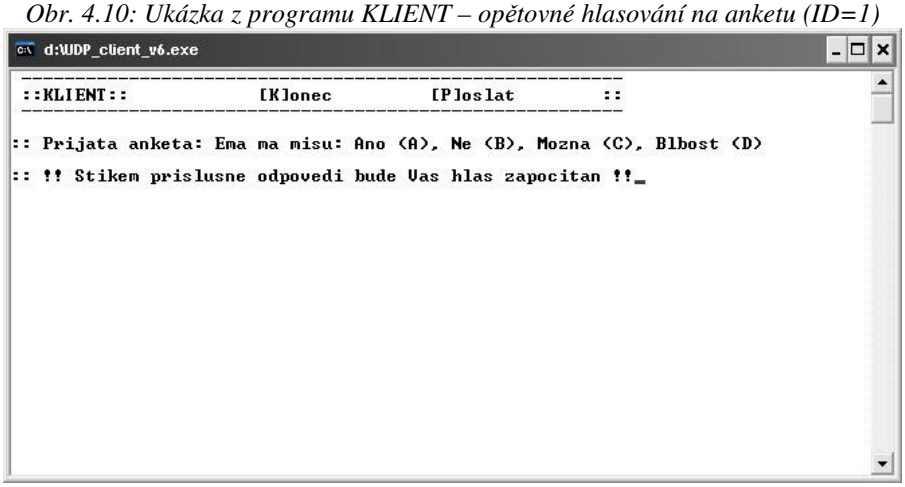
Serveru nám zase v zápětí zobrazí odpověď na anketu od klienta jehož příslušnou IP je možno vidět dole ve stavovém řádku spolu s počtem přijatých dat. Server také vypíše jak klient hlasoval a na jakou anketu bylo hlasováno (viz obr.4.9)

Obr. 4.9: Ukázka z programu SERVER – přijetí hlasování na anketu (ID=1)

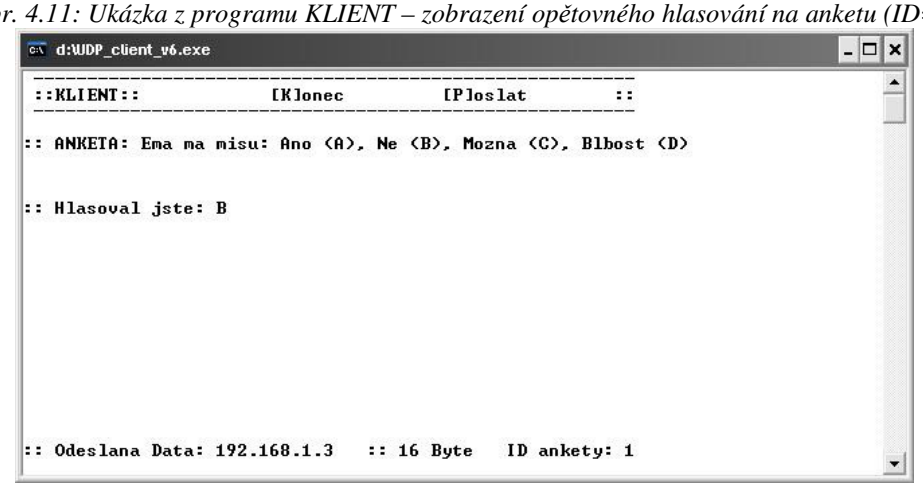


V případě této simulace je třeba na server poslat více odpovědí, aby se mohli zpracovat. Jelikož jsem neměl k dispozici možnost otestovat to na síti, má klient v nabídce položku „Poslat“, kde po stisku „P“ budu moci znovu volit k naposledy přijaté anketě. Tento postup je zobrazen v následujícím obrázku (viz obr.4.10) a opět jeho odpověď, v našem příkladě hlas „B“, stále pro anketu s ID = 1 (viz obr.4.11).

Obr. 4.10: Ukázka z programu KLIENT – opětovné hlasování na anketu (ID=1)

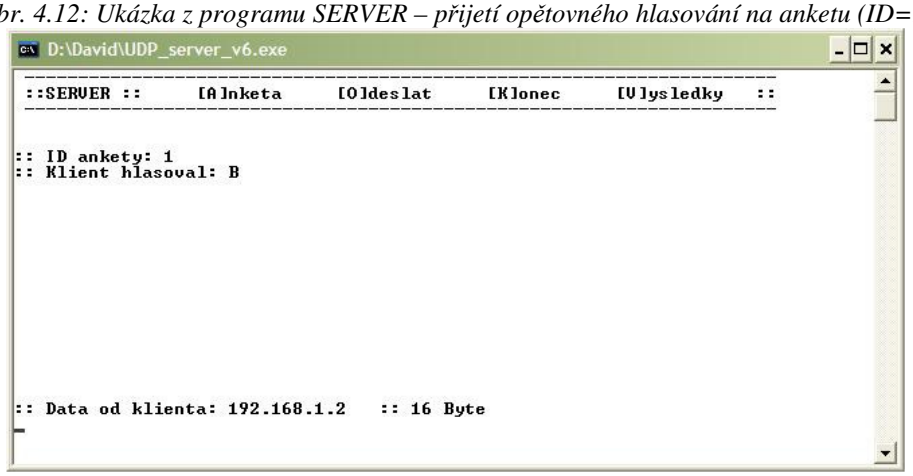


Obr. 4.11: Ukázka z programu KLIENT – zobrazení opětovného hlasování na anketu (ID=1)



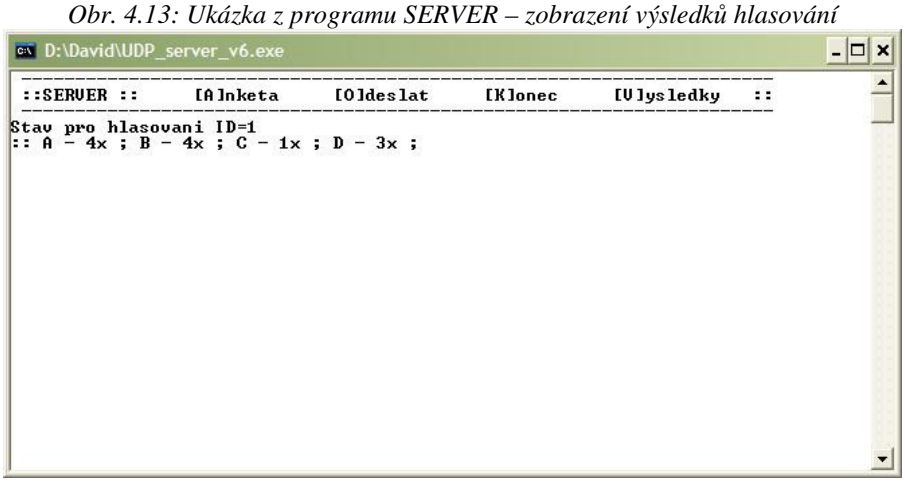
Opět tento druhý hlas byl přijat na straně serveru (viz obr.4.12)

Obr. 4.12: Ukázka z programu SERVER – přijetí opětovného hlasování na anketu (ID=1)



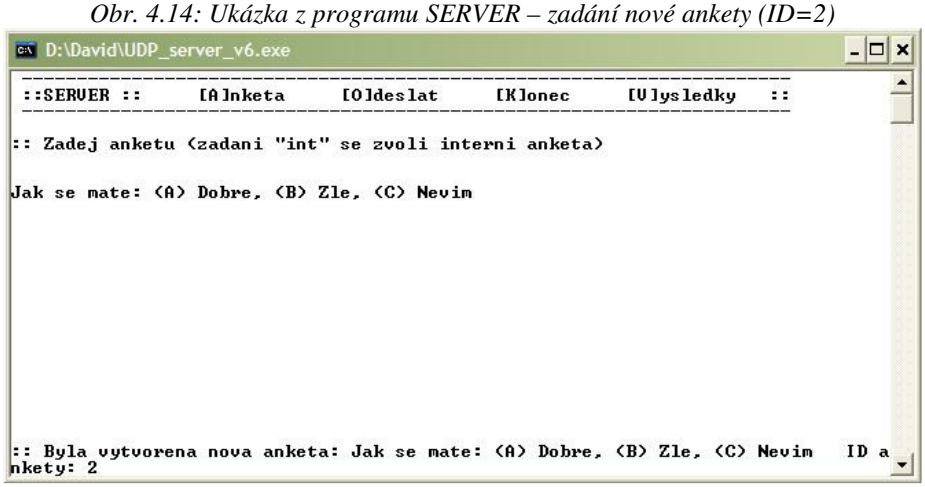
Po sléze jsem ještě odeslal ze stany klienta několik odpovědí, aby se výsledky mohly promítnout na serveru, kde se pro příslušnou anketu sumarizují hlasy. tuto sumarizaci je možno na serveru zobrazit stiskem klávesy „V“ kde nám server vypíše výsledky (viz obr.4.13). Pro tuto simulaci se výsledky ukládají do maticového pole, které má omezený počet prvků (konkrétné 10 možností odpovědí na anketu a 50 možných anket, což je pro ukázkovou simulaci více než postačující), je to z důvodu názornosti simulace. Pro rozlehlou IPTV by bylo samozřejmě příhodné použít databázi.

Obr. 4.13: Ukázka z programu SERVER – zobrazení výsledků hlasování



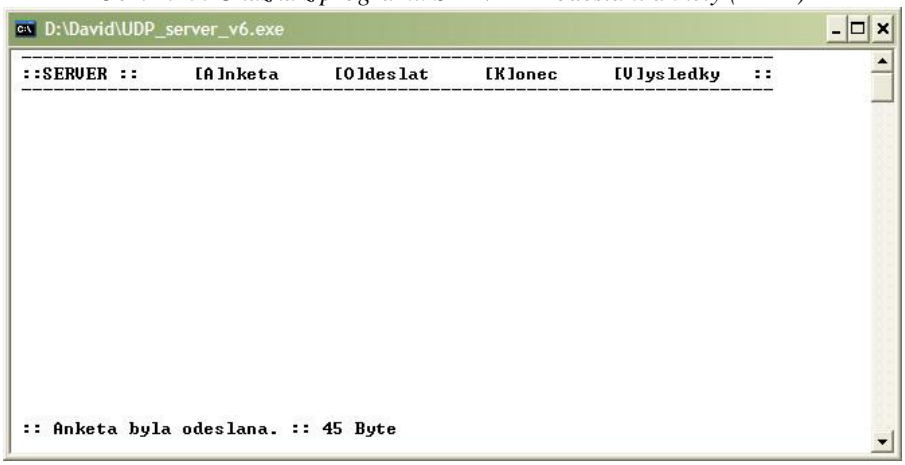
Dále si ještě vytvořím druhou anketu, aby bylo možno vidět, že simulace pracuje pro více „průzkumů pro veřejnost“. Nově vytvořenou anketu ukazuje následující obrázek (viz obr.4.14), ve stavovém řádku je možno vidět , že se jedná o anketu s ID=2.

Obr. 4.14: Ukázka z programu SERVER – zadání nové ankety (ID=2)



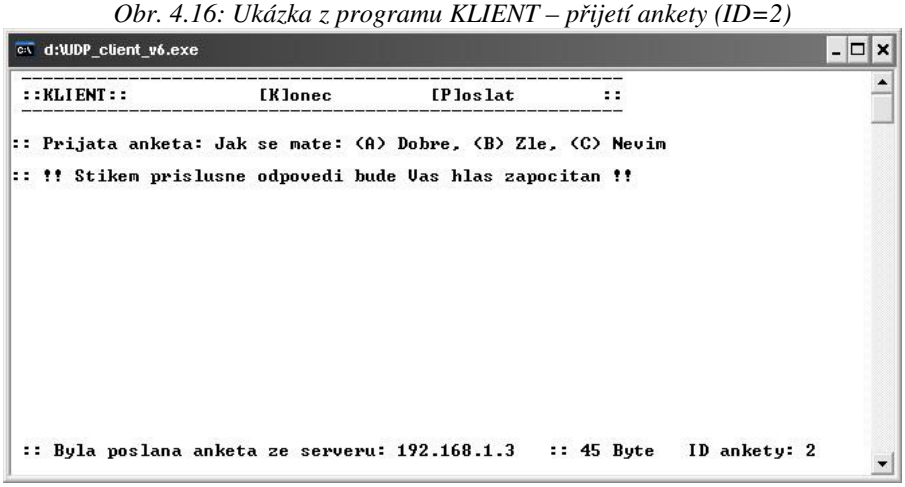
Dále je postup zcela stejný jako v případě předchozí ankety, čili anketu odešleme (viz obr.4.15), ve stavovém řádku opět vidíme velikost odeslaných dat (45 byte).

Obr. 4.15: Ukázka z programu SERVER – odeslání ankety (ID=2)



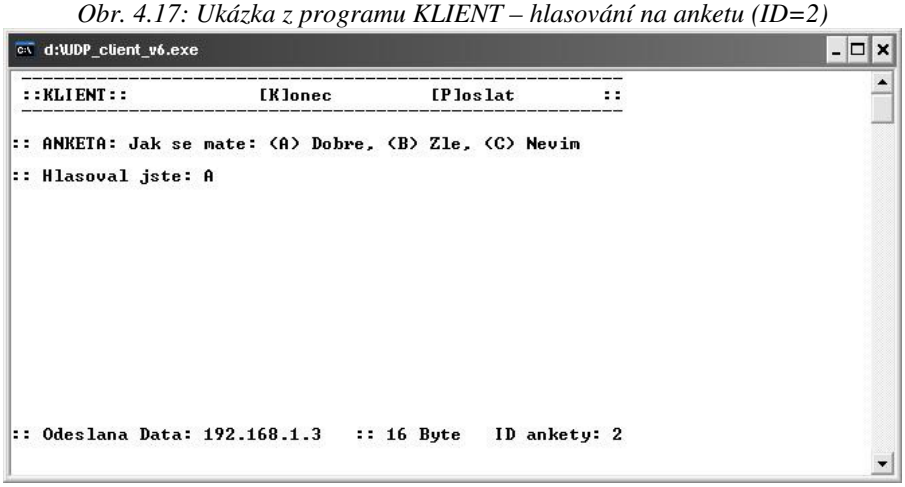
A zároveň s odesláním ze serveru, je anketa přijata na straně klienta (viz obr.4.16), opět ve stavovém spodním řádku vidíme informace o právě přijaté a zobrazené anketě.

Obr. 4.16: Ukázka z programu KLIENT – přijetí ankety (ID=2)



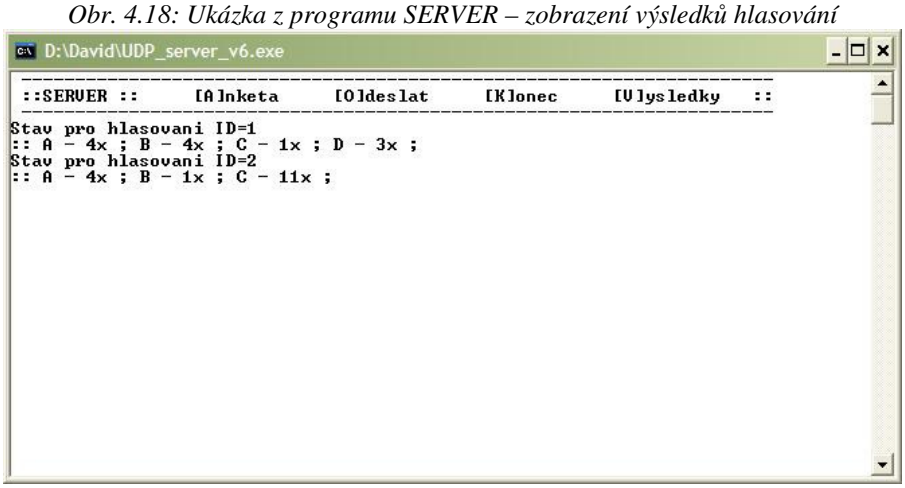
A opět hlasování pro anketu s ID=2 (viz obr.4.17)

Obr. 4.17: Ukázka z programu KLIENT – hlasování na anketu (ID=2)



V posledním obrázku můžeme vidět výsledky hlasování pro obě ankety. U druhé ankety bylo stejně jako v první posláno více hlasů pro názornější zobrazení výsledku statistik (viz obr.4.18)

Obr. 4.18: Ukázka z programu SERVER – zobrazení výsledků hlasování



4.1.4. Popis zdrojových kódů pro KLIENT a SERVER

Programy byly vytvářeny ve vývojovém prostředí *DEV-C++ verze 4.9.9.2*, pod operačním systémem MS Windows. Hlavním komunikační nástrojem je SOKET za jehož pomocí můžeme komunikovat libovolným protokolem, v našem případě UDP s nadstavbou RTCP. Vývojové diagramy obou aplikací jsou uvedeny v

Struktura RTCP protokolu pro aplikačně definovaný paket vypadá následovně:

```
typedef struct rtcp_hlavicka
{  u_int8_t  verze:2,
             doplneni:1,
             podtyp:5;
  u_int8_t  typ_paketu;
  u_int16_t length;
  u_int32_t SSRC;
  u_int32_t jmeno_ASCII;
  u_int8_t  IDVK;
  u_int16_t  ID_ankety;
} rtcp_hlavicka_t;
```

Naplnění hlavičky RTCP APP paketu:

```
rtcpHL.verze = RTCP_VERSION;
rtcpHL.doplneni = 0;
rtcpHL.podtyp = 1;
rtcpHL.typ_paketu = RTCP_PACKETTYPE_APP;
rtcpHL.length = sizeof(rtcpHL);
rtcpHL.SSRC = 123;
rtcpHL.jmeno_ASCII = 1396788296;
rtcpHL.IDVK = 0;
rtcpHL.ID_ankety = 0;
```

Při komunikaci jsem pro znázornění zachytával pakety na rozhraní síťové karty pomocí programu **Wireshark** (verze 0.99.6a (SVN Rev 22276)).

Popis zachyceného RTCP paketu, které ukazuje obr.4.19, je následující:

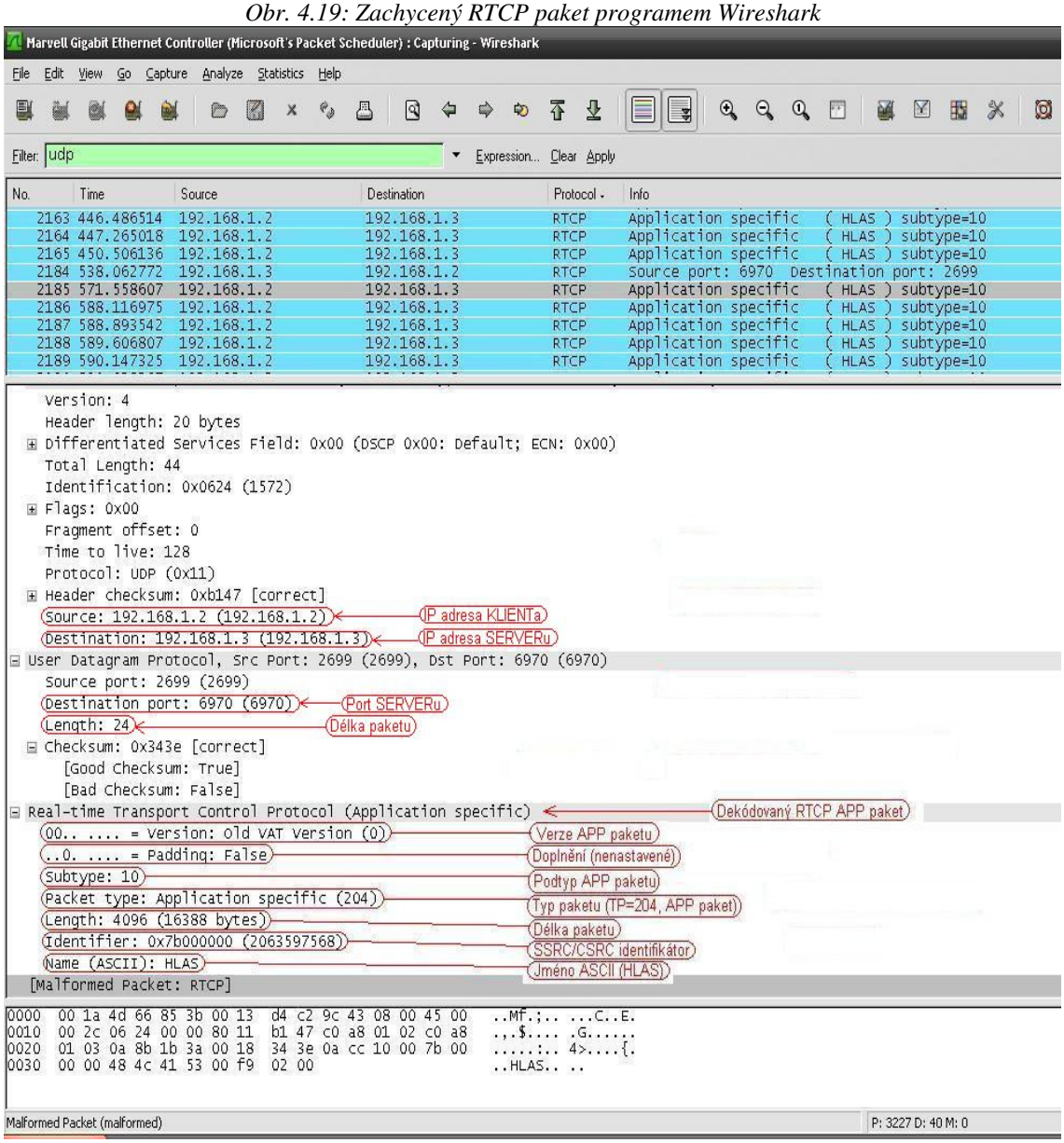
V části s modře zabarvenými řádky okna programu Wireshark jsou vyobrazeny zachycované pakety. Šedě označený řádek ukazuje paket, který je detailně rozebrán ve spodní části okna s bílým pozadím. Zde se odlišují jednotlivé protokoly patřičným vrstev ISO/OSI modelu šedým podbarvením řádku. Na obrázku můžeme vidět část protokolu IP síťové vrstvy, dále pak kompletní část protokolu UDP (*User Datagram Protocol, Src Port: 2699 (2699), Dst Port 6970 (6970)),* patřící do transportní vrstvy a jako poslední část pro nás nejzajímavější protokol RTCP (*Real-time Transport kontrol Protocol (Aplication Specific)).*Červeně doplněné odkazy popisují jednotlivé položky, které popisují:

UDP
- *Source port: 2699 (2699)* – uvádí port zdrojové adresy (klienta)
- *Destination port 6970 (6970)* – uvádí port cílové adresy (serveru)
- *Length: 24* – uvádí délku paketu v bytech
- *Checksum: 0x343e [correct]* – kontrolní součet v hexadecimální podobě, v hranatých závorkách je výsledek kontroly (correct = bezchybný)

RTCP
- *00.. …. = Version: Old VAT Version (0)* – první dva bity uvádějící verzi protokolu hodnota nula ukazuje použití starší verze
- *..0. …. = Padding: False* – doplnění v třetím bitu je nulové a proto rozšíření hlavičky paketu nenásleduje
- *Subtype: 10* – hodnota podtypu námi definovaného uvádí, že se jedná o paketu od klienta (binární hodnota)
- *Packet type: Application Specific (204)* – typ paketu 204 označuje APP
- *Length: 4096 (16388 bytes)* – uvádí délku v bytech v závorce
- *Identifier: 0x7b000000 (2063597568)* – náhodné číslo SSRC
- *Name (ASCII): HLAS* – řetězec identifikují paket jako hlasovací

Detailněji popsané položky RTCP protokolu jsou uvedeny v kapitole 2.5.6.

Obr. 4.19: Zachycený RTCP paket programem Wireshark



4.1.4.1 Popis programu KLIENT

Klient si před spuštěním hlavní funkce main() definuje čtyři další funkce na které se v programu odkazuje. Jsou to: menu_vypis(), oznText(), my_delay (int Sekundy), Data_ke_cteni (SOCKET &Sock).

- oznText() – je zobrazení informace o ukončení v případě výskytu nějaké chyby v běhu programu a čekání na stisk klávesy ENTER k jejímu ukončení
- menu_vypis() – se vícekrát se opakující funkce, která vypisuje na záhlaví obrazovky výběrové uživatelské menu s výčtem funkcí, které si může uživatel zvolit při běhu programu
- my_delay (int Sekundy) – je funkce vracející celočíselný typ v podobě sekund. tato funkce je vyložené efektová a užívá se při ukončení běhu programu uživatelem
- Data_ke_cteni (SOCKET &Sock) – je funkce která vrací adresu dat v případě že se na rozhraní societu nějaká nachází resp. se jedná o data která jsou přijata od serveru a která se budou dále zpracovávat.

Hlavní tělo programu pracuje následovně:

V prvé řadě je podmínka syntaxe, zda byla dobře zadána cesta programu s příslušnými parametry, pokud tomu tak nebylo program vypíše Chybu a zobrazí obecnou syntaxi jak má správné zadání vypadat. Poté následuje příprava a inicializace soketu (nástroje neboli mechanizmu, který je určen pro síťovou komunikaci) a ověření zda to proběhlo v pořádku, v opačném případě program opět vypisuje chybové hlášení a ukončuje se. V dalším kroku se ověřuje zda zadaný port v parametru je z rozsahu 1 - 65535, pokud ne vypíše chybové hlášení a ukončí se. Pak následuje převod jména PC serveru na IP adresu, proto je také možné zadávat v parametru serveru jeho název a nikoli jen IP. V případě neexistence příslušné IP serveru vypíše program chybové hlášení a ukončí se. Dále přichází na řadu vytvoření soketu ze strany klienta, funkce určuje, že se bude jednat o protokol z rodiny TCP/IP, konkrétně o datagramovou službu (UDP). V případě chyby při vytváření vypíše opět chybové hlášení a ukončí se. V dalším kroku zaplňuji strukturu *sockaddr_in* a charakterizuji kam budu právě vytvořený soket navazovat (IP a port serveru). Za pomocí funkce connect() se připojuji ke vzdálené straně. Po naplnění parametrů RCTP paketu (výše uvedených) následuje funkce, která posílá naplněný APP paket vypadá následovně:

```
sendto(id_Socket, (char *)&rtcpHL , rtcpHL.length, 0, (sockaddr*)&srvrSock, sizeof(srvrSock))
```

Nyní ještě než začne hlavní cyklus programu, vyšlu na server první tzv. inicializační paket o navázaní spojení. Je to z důvodu, že server nemůže vysílat (a tím poslat anketu) pokud nemá jako první navázané a potvrzené spojení ze strany klienta.
V hlavním cyklu, který je ukončen stiskem klávesy „K“ případně „k“, testuji zda nedošlo k zmáčknutí nějaké klávesy (podmínka if (kbhit())) nebo zda nedošla ze serveru anketa (podmínka if (Data_ke_cteni(id_Socket))).

V případě stisku klávesy testuji zda nebyla stisknuta klávesa „P“ případně „p“. Pokud byla stisknuta provede se, za předpokladu, že ze serveru již došla anketa, zaslání dalšího hlasu, který se vztahuje k právě aktuální anketě. To má za úkol simulovat zasílání paketů od více klientů.

Při komponování paketu s hlasováním, který se posílá k serveru se modifikují následující datová pole:

IDVK (ID Volby Klienta) – obsahuje hlas volby klienta (znakový řetězec), vybrané části zdrojového kódu, která naplňují toto pole jsou následující:

```
hlasABCD = toupper(_getch());
rtcpHL.IDVK = hlasABCD-65;
```

ID typu hlasování – posílá se identifikační číslo příslušné ankety, ke které se vztahuje volba klienta. Vybrané části zdrojového kódu, která naplňují toto pole jsou následující:

```
ID_anketa_s.copy(ID_anketa_ch, ID_anketa_s.length(), 0);
ID_anketa_ch[(ID_anketa_s.length())+1]='\0';
ID_anketa = atoi(ID_anketa_ch);
rtcpHL.ID_ankety = ID_anketa;
```

Vývojový algoritmus klienta je uveden v příloze 1.

4.1.4.2 Popis programu SERVER

Server obdobně jako klient si před spuštěním hlavní funkce ***main()*** definuje čtyři další funkce na které se v programu odkazuje. Tyto funkce jsou zcela obdobné a jejich výčet a popis je uveden u popisu klienta

Serverový program pracuje na obdobném principu jako klient, jen má více uživatelsky volitelných položek v menu, jejichž výčet a popis je v úvodu popisu komunikace KLIENT/SERVER. Postup práce algoritmu je podobný až na pár detailů a

tak ho uvedu pouze ve zkratce. Ještě třeba podotknout, že následující podmínky a ověření vrací v případě neúspěchu chybové hlášení a program ukončí.

- ověření podmínky pro správnou syntaxi zadání cesty programu a jeho parametru <port>

```
if (argc != 2)
```

- příprava a inicializace soketu serveru

```
if (WSAStartup(wVersionRequested, &wsa_data) != 0)
```

- ověření portu, zda je z platného rozsahu 1 - 65535

```
if ( port < 1 || port > 65536 )
```

- vytvoření soketu, stejně jako u klienta

```
if ((hlavniSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == INVALID_SOCKET)
```

- naplnění struktury sockaddr_in, která charakterizuje kam budu soket navazovat
- naplnění soketu – identifikátor soketu, ukazatel na adresu s portem, a délka předávané struktury

```
if (bind(hlavniSocket, (sockaddr *)&sockName, sizeof(sockName)) == SOCKET_ERROR)
```

- příjem paketu, tzv. inicializačního, který zasílá klient, aby server mohl prvně vyslat anketu

```
if ((size = recvfrom(hlavniSocket, (char*)&rtcpHL, rtcpHL.length, 0, (sockaddr *) &infoKlienta, &addrlen)) == SOCKET_ERROR)
```

- hlavní cyklus kde testuji nepřišla-li data od klienta

```
if (Data_ke_cteni(hlavniSocket))
```

nebo nebyla stisknuta nějaká klávesa

```
if (kbhit())
```

- v případě přijetí hlasování od klienta, se výsledek zobrazí na obrazovce a uloží se do maticového pole, kde se vede statistika výsledků pro jednotlivé ankety. Tomu předchází podmínky, které testují zda je to paket od klienta a zda se jedná o tzv. hlasovací paket nebo-li zda je položka jméno ASCII (name ASCII) rovna znakům „HLAS“.

```
if (rtcpHL.podtyp == 1)
if (rtcpHL.jmeno_ASCII == 1396788296)
```

- v případě stisknutí příslušné klávesy, odpovídající položce v menu se provede daný úkon, který je popsán na začátku

Vývojový algoritmus serveru je uveden v příloze 2.

4.2. Řešení spolehlivého multicastu metodou PGM

Jaké nejvíce vhodnou metodu jsem z možných prostudovaných metod zvolil PGM (Pragmatic General Multicast) protokol, protože jako jediný může být použit pro distribuci dat v reálném čase díky tzv. „Advanced Transmitt Window“. Tento model řešení spolehlivého multicastu je popisován v kap. 5.7.1.3. Blíže je jeho specifikace popisována v RFC 3208.

4.3. Srovnání využití spolehlivého multicastu a SSM multicastu

Vzhledem k několika nedostatkům spolehlivých protokolů, jejich problémovost s nasazením pro velké sítě, kde jsou provozovány spolehlivé protokoly komunikací typu

unicast, bych spíše volil pro distribuci multimediálních dat IPTV Source-Specific Multicast a pro jeho případnou signalizaci, jako zpětnou odezvu, odesiláni typem komunikace unicast. SSM je tímto osvědčený a jednoduchý na realizaci. V rámci požadavku na spolehlivý multicast např. pro korektní doručení dat služeb IPTV, kterou by případná ztrátovost paketů značně omezovala, bych navrhoval PGM (Pragmatic General Multicast) protokol s případným FEC (Forvard Error Correction), který se významně osvědčí s velkým počtem klientů na přijímací straně, nejlépe řádově tisíce.

5. Závěr

Cílem bakalářské práce bylo seznámit se s systémem IPTV, potažmo protokolem RTP/RTCP, který se používá pro přenos a řízení toku multimediálních dat a navrhnout řešení pro simulaci přenosu hlasování. Dále také navrhnout možnost využití spolehlivého multicastu a jeho aplikaci na komunikaci.

Vzhledem k mé nezkušenosti s programovacími jazyky jsem si pro návrh simulace vybral jazyk C++, protože v minulosti jsem se setkal pouze s jazykem Pascal a jako nejschůdnější mi přišlo navázat právě na jazyk, který mu má z mého pohledu nejblíže co se týče syntaxe a obecnému pojetí. Programy jsou vytvořeny pouze pro platformu operačního systému MS Windows a proto nedoporučuji jejich spouštění z jiného prostředí. Z důvodu multiplatformního nasazení by bylo optimálnější použití jazyka Java.

Po tvorbě softwaru jsem se potýkal stále s přílišným vytížením procesoru při spuštění klienta nebo serveru a tento neduh se mi nepodařilo odladit. Domnívám se, že je to zapříčiněno nepříliš dobrou optimalizací kódu a ke správnému doladění by byl zapotřebí vetší rozsah znalostí a praxe v programování.

Pokud bych měl svou práci shrnout, pak za hlavní přínos považuji seznámení se s distribucí multimediálních dat pomocí protokolu pro přenos dat RTP a jeho signalizačním doplňkem RTCP, kde jsem měl navrhnout možnost přenosu specifických dat daných aplikací, konkrétně přenos výsledků hlasování. Dále jsem se seznámil s přehledem směrovacích protokolů a protokolových typů užívaných pro Multicasting. Pro mne nejvíce přínosnou částí bakalářské práce bylo osvojit si znalost programování v jazyce C++ a seznámit se s prací se sokety pro síťovou komunikaci.

Vytvořené aplikace by se dali po mnoha stránkách optimalizovat a stále vylepšovat. Avšak jako simulační ukázka pro přenos hlasování jsou postačující a a nejen vystihují danou problematiku, ale odráží i mé dosavadní znalosti v tvorbě softwaru, které jsem si osvojil při práci na projektu.

Po absolvování této práce je pro mě hlubší znalost v programování v jazyce C++ do budoucna výzvou a rád bych se tomu věnoval nadále.

6. Seznam literatury

[1] PUŽMANOVÁ, Rita. TCP/IP v kostce. 1. vyd. České Budějovice : Kopp, 2004. 607 s. ISBN 80-7232-236-2.

[2] 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.

[3] DOSTÁLEK, Libor, KABELOVÁ, Alena,Velký průvodce protokoly TCI/IP a systémem DNS, 2. aktual. vyd., Computer Press Praha 2000, ISBN 80-7226-323-4

[4] Dan Komosný, Nové směry vývoje protokolu RTP/RTCP pro rozsáhlé konference v Internetu, http://www.elektrorevue.cz/clanky/04052/

[5] Open Source Code Search Engine, http://www.koders.com/

[6] Network Sorcery, Inc., http://www.networksorcery.com/enp/protocol/rtp.htm

[7] Computer Science at Columbia University, 2004, http://www.cs.columbia.edu/~hgs/rtp/

[8] Ondřej Filip, Úvod do IP Multicastu, 2004, http://www.lupa.cz/clanky/uvod-do-ip-multicastu/

[9] Petr Hrubý, Source Specific Multicast - nástroj pro zajištění multicastingu v Internetu, 2002, http://www.isdn.cz/clanek.php?cid=4461

[10] Petr Grygarek, IP Multicast, http://www.cs.vsb.cz/grygarek/SPS/lect/multicast/multicast.html

[11] Informační server o programování, http://www.builder.cz

[12] C. Kenneth Miller, StarBurst Communications, Reliable Multicast Protocols and Applications, 1998, http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_1-2/reliable_multicast.html

[13] Cesnet, Užití IP multicastu pro přenosy velkých objemů dat, 2002, http://www.cesnet.cz/doc/zprava2001/multicast.html

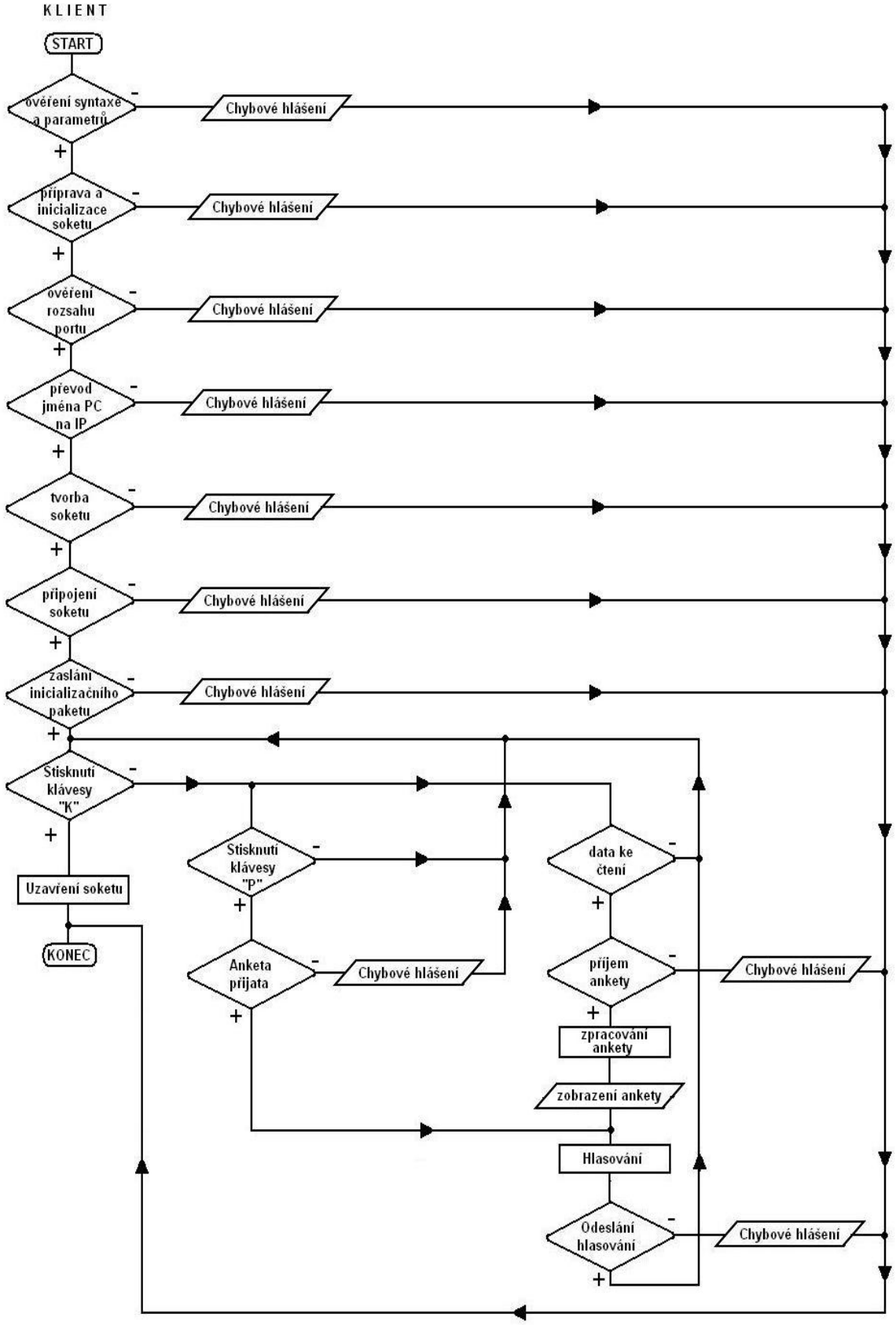
[14] Sanjoy Paul, RMTP: A Reliable Multicast Transport Protocol, 1997, http://www.bell-labs.com/project/rmtp/

[15] Teruji Shiroshita, Tetsuo Sano, Osamu Takahashi, Reliable Multicast Transport Protocol (RMTP), 1997, http://www.trl.ibm.com/projects/rmtp/

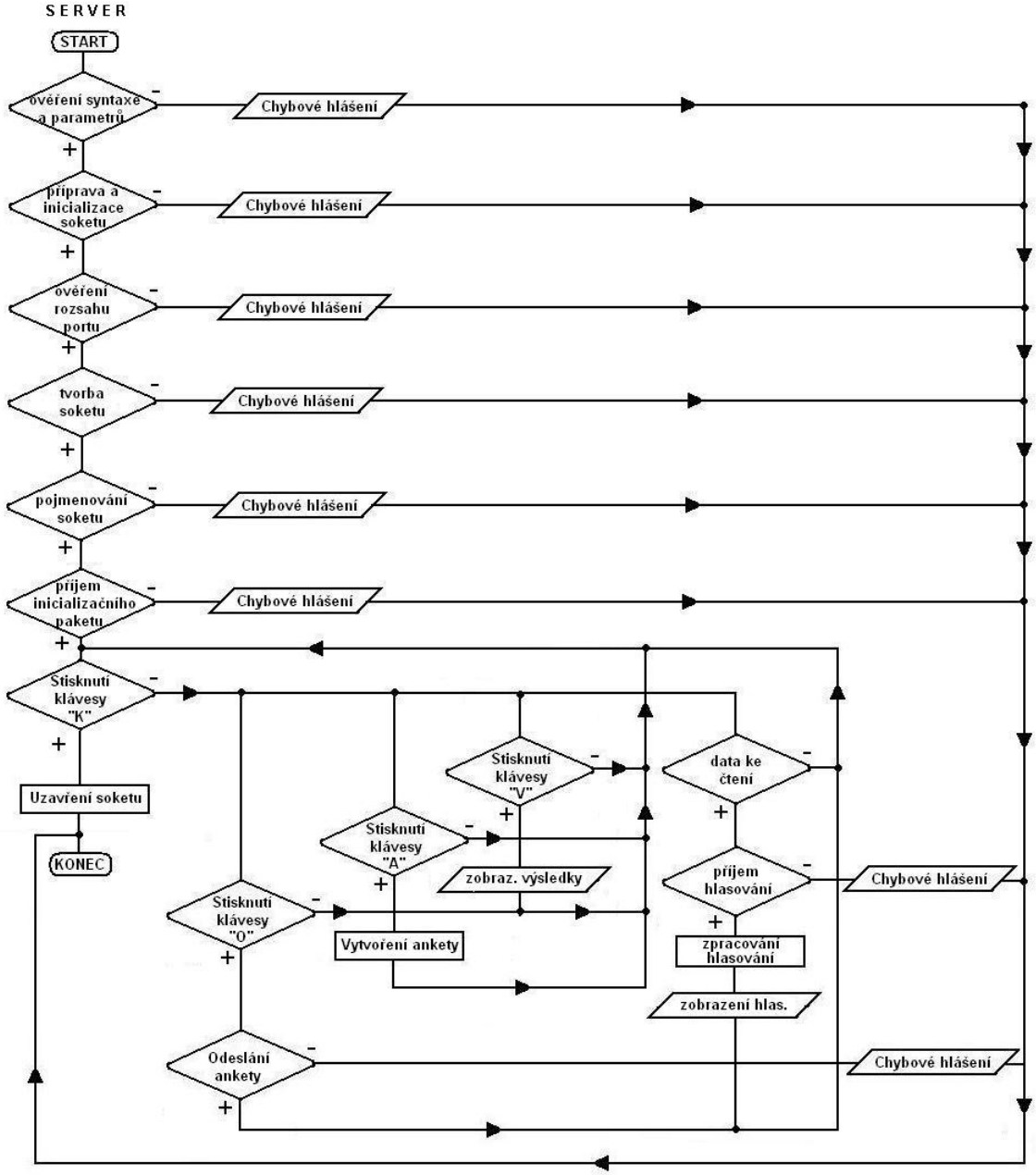
[16] Elias G. Khalaf, Scalable reliable multicast using multiple multicast groups, 2000, http://www.ic-icomp.org/IC2004/ConfMan/SUBMISSIONS/18-lidoynykha.pdf

7. Přílohy

Příloha 1: Vývojový algoritmus – klient



Příloha 2: Vývojový algoritmus – server



Příloha 4: CD s vytvořenými aplikacemi KLIENT a SERVER