VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

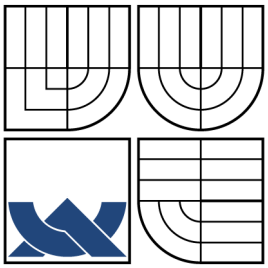
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

HLASOVACÍ SLUŽBA PRO INTERNETOVOU TELEVIZI

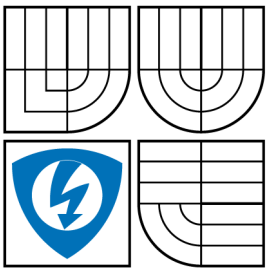
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE TOMÁŠ MENCLÍK
AUTHOR

BRNO 2010



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

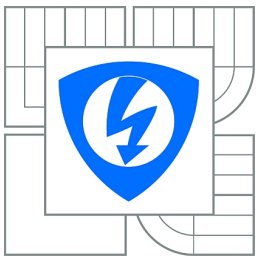
HLASOVACÍ SLUŽBA PRO INTERNETOVOU TELEVIZI
VOTING SERVICE FOR IPTV

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ MENCLÍK
AUTHOR

VEDOUCÍ PRÁCE — ING. JAKUB MÜLLER
SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Tomáš Menclík	***ID:***	106633
Ročník:	3	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Hlasovací služba pro internetovou televizi

POKYNY PRO VYPRACOVÁNÍ:

Student implementuje službu pro hlasování v následujících krocích:
1.zvolí vhodnou grafickou úpravu vysílačů pro odesílání hlasovacích zpráv a následnou práci s nimi(vytvoření tlačítek),
2. realizuje hlasovací paketu v Jave uvnitř simulačního prostředí IPTV,
3. zajistí funkčnost přenosu hlasovacích zpráv (při stisku tlačítka dojde k naplnění paketu a odeslání k nejbližšímu FT) (function: sendToClosestFT),
4. vhodným způsobem realizuje zpracování a zobrazení všech příchozích paketů.

DOPORUČENÁ LITERATURA:

[1] MÜLLER, J.; KOMOSNÝ, D.; BURGET, R.; MORÁVEK, P.; Advantage of Hierarchical Aggregation. International Journal of Computer Science and Network Security, 2008, roč. 2008, č. 8, s. 1-7. ISSN: 1738-7906.
[2] BURGET, R.; KOMOSNÝ, D.; MÜLLER, J.; Best Effort Hierarchical Aggregation Tree for IPTV Signaling. International Journal of Computer Science and Network Security, 2008, roč. 2008, č. 8, s. 1-5. ISSN: 1738-7906.
[3] BURGET, R.; KOMOSNY, D.; Real-time control protocol and its improvements for Internet Protocol Television. International Transaction on Computer Science and Engineering, ISSN 1738-6438, 2006, roč. 2006, č. 31, s. 1 - 12.

Termín zadání:	29.1.2010	***Termín odevzdání:***	2.6.2010

Vedoucí práce: Ing. Jakub Müller

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

ABSTRAKT

Tato práce se zabývá návrhem a realizací hlasovací služby pro internetovou televizi. Realizace této služby je provedena v programovacím jazyce Java a využívá již hotových programových struktur, které jsou určené pro simulování vlastností stromového přenosového protokolu. Mimo návrh a realizaci hlasovací služby jsou zde prezentovány protokoly, které pro svou komunikaci využívá internetová televize a jejichž vlastnosti využívá i v této práci realizovaná hlasovací služba. Jelikož realizace přesně nedodržuje návrh hlasovací služby jsou zde uvedeny i rozdíly mezi navrženou a realizovanou hlasovací službou.

KLÍČOVÁ SLOVA

IPTV, TTP, RTP, RTCP, hlasovací služba, hlasovací paket

ABSTRACT

This thesis deals with a concept and implementation of voting service for IPTV. Implementation of this service is executed in programming language Java and uses programming structures, which were already done. These structures are intended for simulation of propertys of tree transmition protocol. Outside of the concept and implementation of voting service, there are presented protocols, which IPTV uses for communication. And also voting service, implemented in this thesis, uses their propertys. There are also mentioned differences between outlined and implemented voting service as the implementation does not keep strictly the concept of the voting service.

KEYWORDS

IPTV, TTP, RTP, RTCP, voting service, voting packet

Menclík T. *Hlasovací služba pro internetovou televizi*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 52 s. Vedoucí bakalářské práce Ing.Jakub Müller.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Hlasovací služba pro internetovou televizi“ 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

Rád bych poděkoval vedoucímu bakalářské práce Ing. Jakubu Müllerovi za pomoc při zpracování této práce a za cenné rady během během jejího řešení.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Internetová televize je založena na poskytování služeb, v tomto případě televize, pomocí internetového protokolu přes počítačové sítě. Hlavním rozdílem mezi klasickým televizním vysíláním a internetovou televizí je zpětná vazba od koncového uživatele, v tomto případě diváka. To znamená, že se divák může v některých pořadech přímo účastnit například hlasováním nebo jeho přijímač může informovat zdroj vysílání o zhoršené kvalitě přijímaného obrazu či zvuku. S rostoucím počtem uživatelů internetu roste i zájem o multimediální služby, jakou je například internetová televize. Tento rostoucí zájem znamená to, že je třeba nabízet divákovi něco víc než jen sledování televizních pořadů a filmů. Proto vznikají nové doplňkové služby, které interaktivně zapojují televizního diváka do dění. Jednou z těchto služeb je i hlasovací služba.

Tato práce se věnuje právě návrhu a realizaci hlasovací služby pro internetovou televizi. Pro návrh a realizaci je využito vlastností nově vznikajícího stromového přenosového protokolu, jehož vlastnosti a části budou v této práci také popsány.

V první části je uvedeno srovnání klasického televizního vysílání s internetovou televizí. Poté se čtenář seznámí s protokoly, který využívá internetová televize ke komunikaci. Jedná se o protokol, který je určený pro přenos v reálném čase a k němu přidružený signalizační protokol. Dále bude popsán nově vznikající stromový přenosový protokol. Jsou zde popsány jeho hlavní části a pakety, které využívá.

Dále je v této práci proveden již zmíněný návrh hlasovací služby pro internetovou televizi a je zde popsána její následná realizace.

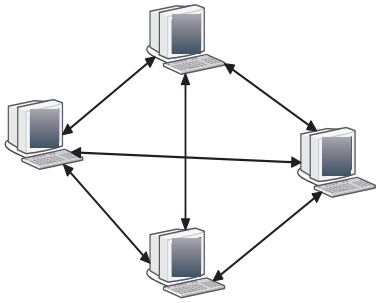
1 ÚVOD DO IPTV

Internet Protocol TeleVision(IPTV) je služba poskytující televizní vysílání přes počítačovou síť založenou na internetovém protokolu(IP). Tento přenos vyžaduje převedení televizních dat například do digitálního formátu MPEG-2 nebo MPEG-4(Moving Picture Experts Group). U klasického televizního vysílání uživatelé signál pouze přijímají, to znamená, že je toto vysílání pouze jednosměrné, z čehož vyplývá, že vysílač nemá vůbec ponětí kolik má aktuálně příjemců nebo kdo se na co dívá. Kdežto v IPTV se mohou diváci aktivně zúčastnit některých pořadů nebo si například vyžádat video na přání(VoD - Video on Demand). Klasické televizní vysílání neomezuje počet vysílaných stanic a k jejich výběru dochází přímo v přijímači. Klasické televizní vysílání využívá toho, že přenosová cesta od vysílače k divákům je dostatečně dimenzovaná, aby přenos více kanálů zvládla. Naopak IPTV je omezena počtem přijímaných kanálů a k výběru kanálu dochází na vzdáleném zařízení. Počet najednou přijímaných kanálů je omezen z důvodu nedostatečné šířky pásma přístupových sítí, proto v IPTV lze přijímat v reálném čase pouze jednu stanici. Oproti tomu nabízí IPTV interaktivní služby, jako je třeba již zmíněné hlasování, soutěže či různé ankety. Více o rozdílu klasického televizního vysílání a IPTV je v [4].

Komunikace v IPTV je založena na Real-time Transport Protokolu(RTP) a na Real-Time Control Protokolu(RTCP). Což jsou protokoly určené pro komunikaci v reálném čase, kde RTP je protokol distribující multimediální data a RTCP je signalizační protokol. Tyto protokoly byly původně navrhnuty pro služby s menším počtem koncových účastníků, jako jsou například videokonferenční hovory. Proto je zřejmé, že pro rozsáhlé relace, jakou IPTV bezesporu je, budou muset být použity další protokoly či metody, které přizpůsobí použití těchto protokolů, zejména RTCP protokolu, pro službu IPTV. Tyto protokoly a metody budou popsány dále v textu.

2 RTP/RTCP PROTOKOLY

Hlavními částmi těchto dvou protokolů jsou přijímač a vysílač. Toto rozdělení je vhodné pouze pro služby s menším počtem účastníků. Pro IPTV je toto rozdělení nedostačující, protože se předpokládá velký počet přijímačů a to v řádech statisíců či milionů. S takovým počtem koncových uživatelů by byla zbytečně zatížena poskytnutá šířka pásma redundantními informacemi o stavu kvality služeb jednotlivých uživatelů, protože RTP/RTCP jsou primárně specifikovány pro komunikaci pomocí Any Source Multicast(ASM) architektury, což zjednodušeně znamená, že každý může být zdrojem a všichni posílají informace o kvalitě služeb ostatním účastníkům relace. Tento způsob komunikace zbytečně zatěžuje šířku pásma. Tento způsob komunikace je zobrazen na obr. 2.1.



Obr. 2.1: Komunikace typu ASM

2.1 RTP protokol

K přenosu dat v reálném čase se používá RTP protokol, který se opírá o spojově neorientovaný a nespolehlivý protokol transportní vrstvy UDP. RTP nezaručuje kvalitu poskytované služby, pouze posílá dat přes multicastový kanál k přijímačům. V IPTV se používá multicastová architektura Source Specific Multicast(SSM), kde zdroj posílá data pomocí multicastu k přijímačům a ty mu posílají pomocí unicastu informace o kvalitě poskytovaných služeb. RTP protokol poskytuje konstantní přenosovou rychlost a spotřebuje 95% šířky pásma pro svůj provoz.

2.2 RTCP protokol

Real-Time Control Protocol(RTCP) protokol je signalizační protokol, který je určen pro monitorování kvality služeb probíhající relace. Aby signalizace zbytečně nezatěžovala šířku pásma, je jí alokováno pouze 5% z celkové šířky pásma. V těchto 5%

se musí přenést veškerá signalizace jak ve směru od koncových uzlů ke zdroji, tak od zdroje ke koncovým uzlům. 5% z celkové šířky pásma pro RTCP se dále dělí na dvě části a to Receiver Report(RR), což jsou signalizační informace směřující od přijímačů ke zdroji a Sender Report(SR), což je signalizace směřující od zdroje k přijímačům. Pro RR je alokováno 75% z šířky pásma určené pro RTCP a pro SR 25%. RTCP používá mechanismus na kontrolu frekvence zasílání RTCP paketů a algoritmus, který se snaží udržet šířku pásma pro RTCP pakety na 5% z celkové šířky pásma. Více o protokolu RTCP v [7] nebo rfc 3550.

2.2.1 Omezení protokolu RTCP

V relacích s velkým počtem koncových účastníků vyvstává u RTCP protokolu problém se zpětnovazebním intervalem pro doručování zpráv o kvalitě služby jednotlivých koncových účastníků. Perioda zasílání zpětnovazebních zpráv je lineárně závislá na počtu přijímačů, proto by toto zvětšování intervalu pro přenos zpětnovazebních zpráv vedlo k tomu, že by zdroji mohli docházet zastaralé informace, které by opožděně ovlivňovali funkci celé relace. Proto je ve specifikaci RTCP nastavena minimální perioda zasílání zpětné vazby na 5 s. Z tohoto důvodu je nutné udržet zpětnovazební interval na tak nízké hodnotě, jak je jen možné. Proto tento interval limituje počet účastníků relace.

Pro službu jako je IPTV nemohou protokoly RTP/RTCP zajistit kvalitu služeb pro velký počet koncových uživatelů, například v řádech miliónů. Proto se tyto protokoly používají ještě s dalšími protokoly a metodami. Jedná se například o nově vznikající Tree Transmition Protocol(TTP), který využívá metod lokalizace, sumarizace a agregace. Tento protokol rozdělí koncové uzly do několika částí a každé části je zaručen 5 s interval pro přenos zpětné vazby. Tento protokol a metody, které využívá budou popsány dále v textu.

2.3 Pakety používané protokoly RTP/RTCP

RTP/RTCP protokoly definují několik typů paketů. V této části budou popsány ty nejpoužívanější pro přenos dat mezi vysílačem a přijímačem.

2.3.1 RTP paket

RTP paket obsahuje mimo jiné informace o typu přenášených dat, verzi použitého protokolu, časové značkování(Time-Stamp), přenášená data a informaci o zdroji synchronizace, což je náhodně generovaná hodnota a musí být jedinečná v celé relaci.

2.3.2 Sender Report paket

Pomocí Sender Report(SR) paketu informuje vysílač všechny účastníky relace o kvalitě služby. Tento paket se dostane ke koncovým uzlů pomocí source specific multicast(SSM) architektury, což je multicástová architektura, kde je pouze jeden zdroj a více přijímačů. SR paket paket obsahuje například pole zdroj synchronizace(SSRC-Synchronization SouRCe), což je identifikátor zdroje SR paketu. NTP(Network Type Protocol) timestamp je časová značka, která určuje, kdy byl SR paket vygenerován. Sender's packet count je celkový počet přenesených RTP paketů od doby, kdy začal přenos až do doby, kdy byl vygenerován tento SR paket. Sender's octet count je celkový počet oktetů v zátěži přenesených RTP pakety od doby, kdy začal přenos do doby, kdy byl vygenerován SR paket. Podrobnější popis SR paketu a jeho polí je v rfc 3550.

2.3.3 Receiver Report paket

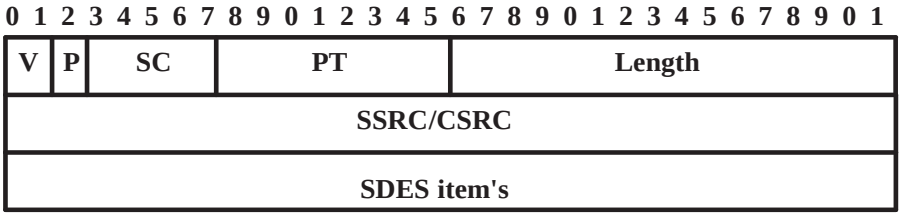
Přijímače generují periodicky Receiver Report(RR) paket a posílají ho pomocí unicastu přímo vysílači. RR má s SR paketem některé bloky shodné, ale navíc obsahuje například třeba Fraction Lost, což označuje ztrátovost RTP paketů od doby, kdy byl poslán první RR paket. Cumulative number of packets lost je celkový počet ztracených RTP paketů od doby, kdy začal přijímač přijímat RTP pakety. Podrobnější popis RR paketu a jeho jednotlivých polí je v rfc 3550.

2.3.4 Receiver Summarization Information paket

Tento paket slouží pro sumarizaci RR paketů v TTP protokolu, ale také pro sumarizaci RSI paketů, které pochází od sumarizačních uzlů z nižších úrovní. Základními poli Receiver Summarization Information(RSI) paketu jsou velikost skupiny a časová značka. Dále obsahuje doplňující bloky jako ztrátovost paketů, kolísavé zpoždění(jitter), kumulativní ztrátovost, šířku pásma a další obecné statistiky.

2.3.5 Source Description paket

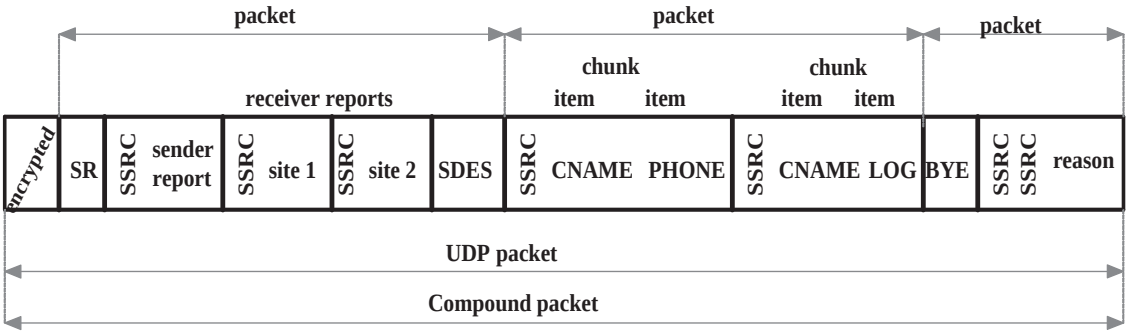
Source DEScription(SDES) paket se skládá z hlavičky a buď žádného nebo více chunků v zátěži. Každý chunk je složen z částí popisující zdroj v chunku. Source Count(SC) udává počet chunků v zátěži SDES paketu. Pole SSRC/CSRC(Synchronization SouRCe/Contributing SouRCe) je identifikátor zdroje RTP paketu. Zátěž pak může obsahovat jednotlivé části(itemy), například EMAIL, což je emailová adresa zdroje RTP paketů. PHONE, což je telefoní číslo zdroje RTP paketů. Což se dá využít například při video hovorech. Příkad SDES paketu ja ne obr. 2.2.



Obr. 2.2: SDES paket

2.3.6 RTCP Compound paket

Všechny výše popsané pakety nejsou přenášeny každý zvlášť, ale jsou slučovány do tzv. sloučeného paketu. Tento paket může obsahovat více RR paketů následovaných SDES pakety. Příklad jak může vypadat Compound paket je na obr. 2.3.



Obr. 2.3: Příklad Compound paketu

3 TREE TRANSMITION PROTOCOL

Tree Transmition Protocol(TTP) by se měl používat tam, kde je potřeba přenést stejná data od velkého počtu klientů přes úzký kanál. Tento protokol implementuje metody sumarizace, lokalizace a hierarchické agregace. Je založen na spolupráci mezi správcem sumarizačních uzlů, sumarizačními uzly navzájem a klienty. Tyto části vytváří jistý druh zpětnovazebního stromu. Tento protokol by se měl používat hlavně v rozsáhlých aplikacích, které pracují v reálném čase, jako například IPTV. TTP protokol nabízí novou cestu ve sběru dat zpětné vazby. Místo zasílání informací o kvalitě služeb přímo zdroji multimediálních dat se tato informace zasílá k nejbližšímu sumarizačnímu serveru, který provede sumarizaci zpětnovazebních dat a sumarizovanou zprávu pošle sumarizačnímu serveru do vyšší úrovně stromu. TTP protokol rozděluje koncové uzly do skupin a každá skupina spadá pod jeden sumarizační server. Toto rozdělení pomáhá řešit problém s intervalem zasílání zpětné vazby, protože počet uživatelů v každé skupině je omezen tak, aby všechny koncové uzly zaslaly svou zpětnou vazbu v nejkratším možném intervalu. Více o TTP protokolu je v [1], [3], [5], [6] nebo [10].

3.1 Metody použité v TTP

3.1.1 Sumarizace

Sumarizace je určena k tomu, aby síť nebyla zbytečně zatěžována nadbytečnými informacemi. Jelikož je pravděpodobné, že většina koncových uzlů bude zasílat stejné informace o kvalitě služeb, je zbytečné, aby ji každý přenášel přímo vysílači. Proto se tyto informace sumarizují v uzlech určených k tomuto účelu do jednoho formátu, kterým je Receiver Summarization Information(RSI) paket. Vysílač ve výsledku obdrží jeden paket, ve kterém je obsažena informace o kvalitě služeb celé relace.

3.1.2 Hierarchická agregace

Hierarchická agregace se používá k rozdělení koncových uzlů do skupin podle jejich relativní pozice. Jednotlivým skupinám je zaručen nejmenší možný zpětnovazební interval pro zasílání informací o kvalitě služeb. Dále má každá skupina nad sebou tzv.agregátora, kterému posílají zpětnovazební data a ten je dále posílá vysílači. Hierarchická agregace může být více úrovňová, to znamená, že „agregátoři“ mají nad sebou ještě jednu vrstvu „agregátorů“. Více informací o hierarchické agregaci je například v [1],[3] nebo [6].

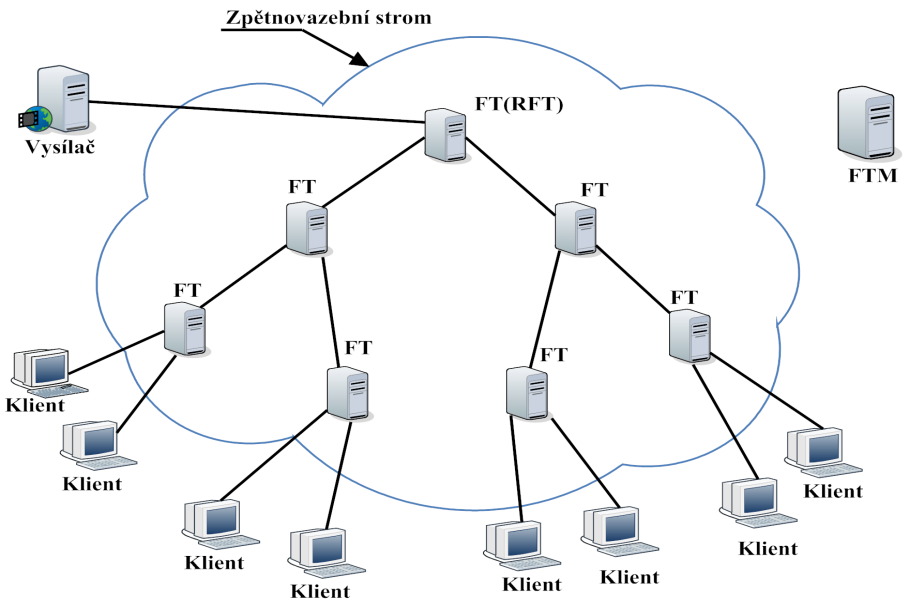
3.1.3 Lokalizace

Pomocí lokalizační metody se určují relativní pozice jednotlivých členů relace. Lokalizace je důležitá z toho důvodu, aby si koncové uzly vybraly nebo byly přiděleny, při vytváření stromové struktury, sumarizačnímu uzlu, u kterého budou zaslaná data dosahovat nejnižšího zpoždění.

3.2 Části zpětnovazebního stromu

Příklad zpětnovazebního stromu je zobrazen na obr. 3.1. Zpětnovazební strom se skládá z těchto hlavních částí:

- Vysílač(sender)
- Feedback Target
- Feedback Target Manager
- Přijímač(reciever)
- Lokalizační server(LandMark)



Obr. 3.1: Příklad zpětnovazebního stromu

3.2.1 Vysílač

Tento zdroj multimediálních dat využívá k přenosu Source Specific Multicast(SSM) architekturu, pomocí které se vždy šíří stromem jedna kopie RTP paketu. Před začátkem vysílání RTP paketů požádá vysílač Feedback Target Managera(FTM) o vytvoření zpětnovazebního stromu. Po jeho vytvoření vysílač obdrží od FTM jedinečný identifikátor tohoto stromu(Tree ID).

3.2.2 Feedback Target

Ve zpětnovazebním stromu plní funkci sumarizačního serveru. Zná přesný počet uzlů, které spravuje a přijímá od nich data, které následně sumarizuje a pomocí RSI paketu je zasílá FT do vyšší úrovně stromu. RSI paket je generován periodicky a to v intervalu 5 s. Může se nacházet na jakékoli úrovni ve stromu kromě té nejnižší, kde jsou koncové uzly. Když se nachází na vrcholu stromu, tak se nazývá Root Feedbac Target(RFT) a provádí finální sumarizaci. Pokud je to vyžadováno, může simultánně pracovat v několika stromech najednou.

3.2.3 Feedback Target Manager

FTM plní funkci správce stromu. Zná relativní polohu všech účastníků a je obeznámen jejich parametry. Na požádání vysílače vytváří strom a spravuje ho. Po vytvoření stromu posílá vysílači jeho jedinečný identifikátor. Na základě relativní polohy určuje v jaké úrovni stromu budou jednotlivá FT a v případě potřeby jejich umístění změní či FT úplně odstraní ze stromu. Při náhlých změnách počtu koncových uzlů odstraní FT nebo do stromu přidá další. Informace o stromu posílá všem účastníkům relace pomocí Feedback Target Specification(FTS) paketu.

3.2.4 Lokalizační server(LandMark)

Tento lokalizační server má svou stabilní polohu a slouží k určení polohy koncových uzlů a FT. Každý koncový uzel si na základě ping žádostí vytvoří zpožďovací tabulku, podle které zjistí, který lokalizační server je nejblíže.

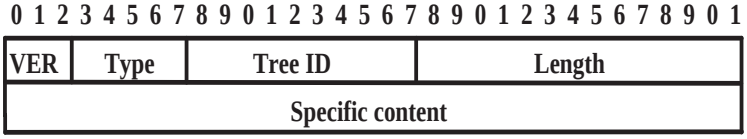
3.2.5 Přijímač

Po vytvoření zpětnovazebního stromu hned přijímače začnou posílat informace o kvalitě služeb, aniž by je zajímal stav zpětnovazebního kanálu. Pomocí IP adresy z FTS paketu najdou nejbližší FT, který pro ně reprezentuje virtuální vysílač. Přijímače musí neustále sledovat FTS pakety, kdyby nastala náhlá změna ve stromu.

Přijímač si také zjišťuje počet účastníků ve skupině(clusteru), aby podle tohoto počtu mohl určit svůj zpětnovazební interval pro zasílání zpráv. Přijímače se mohou kdykoli připojit nebo odpojit od stromu či změnit strom.

3.3 Druhy paketů používané TTP

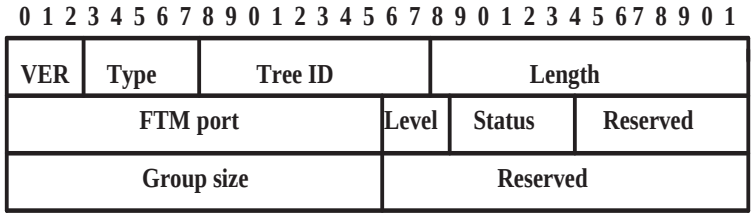
Obecná hlavička všech paketů, které TTP používá, má velikost 32 bitů. Pole VER určuje verzi protokolu TTP. Pole Type určuje typ paketů, jeho velikost je 5 bitů, což znamená celkem 32 typů paketu. Pole Tree ID je jedinečný identifikátor stromu a určuje do kterého stromu patří FT. Polem Length je definována velikost paketu. Obecná hlavička je zobrazena na obr. 3.2. Bližší popis jednotlivý paketů je v [1].



Obr. 3.2: Obecná hlavička paketu

3.3.1 Feedback Target Definition paket

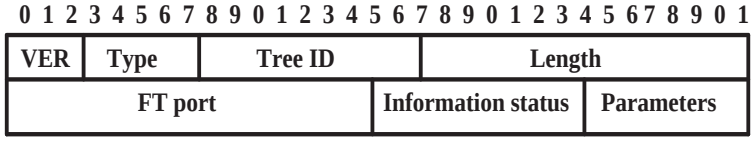
Feedback Target Definition(FTD) paket je generován FTM a používá se pro nastavení specifických parametrů FT. Pole Level označuje úroveň zpětnovazebního stromu, na které je FT využíváno. Pole status určuje jestli je FT nejvýše ve stromu, tedy plní funkci RFT nebo jestli pracuje jako lokalizační uzel. Pole Group size definuje počet koncových uzlů ve skupině, kterou FT spravuje. Tvar FTD paketu je zobrazen na obr. 3.3. Obrázek převzat z [1].



Obr. 3.3: FTD paket

3.3.2 Feedback Target Information paket

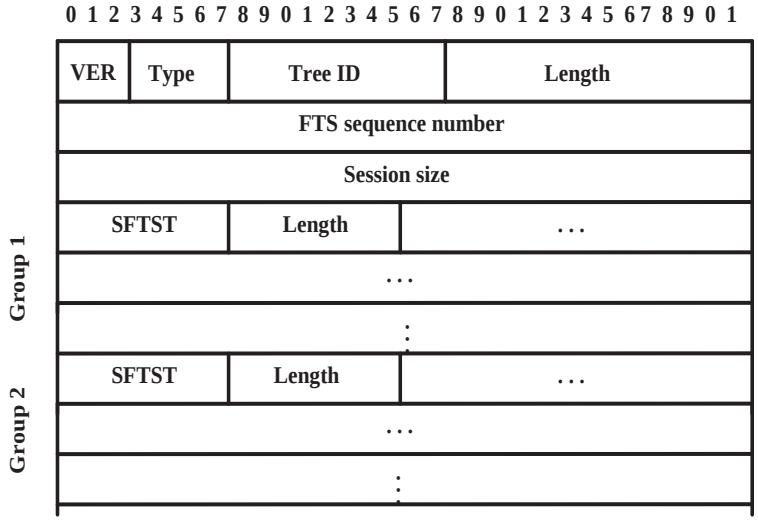
FT používá Feedback Target Information(FTI) paket jako reakci na FTD paket. Při komunikaci s FTM ho FT posílá jako potvrzení či odmítnutí parametrů, které mu FTM zaslal. FTM tento paket používá k odstranění FT ze stromu. FTI paket je zobrazen na obr. 3.4. Obrázek převzat z [1].



Obr. 3.4: FTI paket

3.3.3 Feedback Target Specification paket

Feedback Target Specification(FTS) paket(obr. 3.5) je generován FTM a pomocí multicastového kanálu je poslán všem členům relace, tedy všem FT a koncovým uzlů. Obsahuje informace o všech parametrech všech FT a jejich lokalizaci. Po přijmutí paketu jsou tedy všechny koncové uzly a FT schopni nalézt nejbližší sumarizační uzel ve vyšší úrovni. Velikost FTS paketu je závislá na počtu FT. FTS paket se posílá periodicky v intervalu 5s. Každý FTS paket obsahuje subbloky. Existují dva typy subbloků. První definuje parametry FT a druhý definuje parametry lokalizačního serveru.



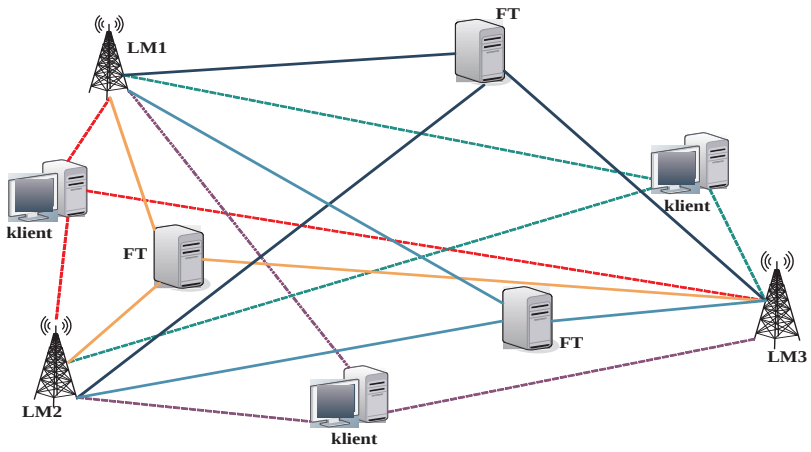
Obr. 3.5: FTS paket

FTS paket obsahuje tyto hlavní bloky :

- FTS Sequnce number určuje číslo paketu
- Session size definuje aktuální počet klientů
- SFTST určuje typ subbloku
- Length definuje velikost subbloku

3.4 Vznik zpětnovazebního stromu

Před vytvořením zpětnovazebního stromu musí všichni členové znát svoji relativní polohu, kterou zjistí tak, že zašlou deset žádostí každému lokalizačnímu serveru a čekají na odpověď. Těchto deset žádostí se negeneruje ve stejný čas, aby se zbytečně nezatěžovaly lokalizační servery. Čas změřený po odeslání žádosti a přijetí odpovědi se označuje jako Round Trip delay Time(RTT) vektor. Hodnotu tohoto vektoru všichni účastníci posílají FTM, aby měl představu o rozložení členů v síti. Princip určování relativní pozice jednotlivých členů relace je znázorněn na obr. 3.6.



Obr. 3.6: Příklad určení relativní polohy

Po té, co FT zjistí svoji relativní polohu a zašle ji FTM, se ještě musí zaregistrovat u FTM. Nejdříve FT zašle FTM registrační paket, na tento registrační paket FTM odpoví žádostí a čeká na odpověď. Tímto procesem si FTM vytvoří vlastní RTT tabulku o poloze jednotlivých FT, na základě které může vytvořit strom. Toto se provádí z důvodu toho, aby FTM věděl, které FT je mu, respektive vysílači, z hlediska zpoždění nejblíže, protože umístit FT s největším zpožděním na vrchol stromu

by mohlo negativně ovlivnit celou relaci. Po vytvoření stromu pošle FTM vysílači jedinečný identifikátor stromu(Tree ID).

Před tím, než začne IPTV vysílat, požádá vysílač FTM o vytvoření zpětnovazebního stromu, takže zpětná vazba od koncových uzlů bude sdružována a kolektivizována vytvořeným zpětnovazebním stromem. FTM vybere podle počtu koncových uzlů sadu FT a zformuje je do stromu.

3.5 Výhody/nevýhody TTP protokolu

Výhodou TTP je určitě to, že by měl zvládnout rozsáhle relace s velkým počtem koncových uzlů. Zdá se být jednoduše spravovatelný pomocí FTM. Dokáže rychle reagovat na změny počtu koncových uzlů přidáváním či odebíráním FT. Určitou nevýhodou může být zvyšování počtu úrovní, protože s tímto zvyšováním by rostla i velikost RSI paketu a mohl by zde být problém s alokovanou šířkou pásma pro zpětnovazební přenos.

4 HLASOVACÍ SLUŽBA

Hlasovací služba by měla být doplňkovou službou IPTV, která nabídne divákovi možnost interaktivně vstoupit do televizního vysílání. Tato služba, jako jedna z mála, představuje viditelný rozdíl mezi klasickým analogovým či digitálním terestriálním vysíláním a IPTV stejně jako například služba VoD(Video on Demand).

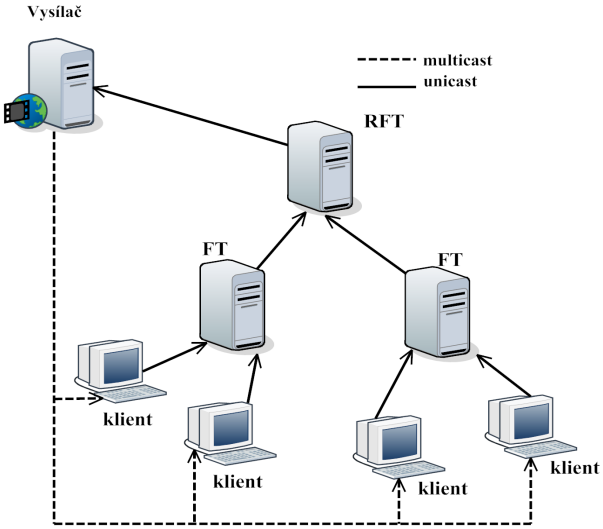
Tato služba by měla nabízet divákovi možnost interaktivně vstupovat do některých pořadů a to především do vědomostních nebo soutěžních, kde by televizní diváci mohly například poradit soutěžícímu se správnou odpovědí. Hlasování by mohlo probíhat za pomoci čtyř barevných tlačítek, které jsou dnes již standardní součástí televizních ovladačů. Jednotlivé barvy, respektive barevná tlačítka na ovladači, by měly být přiřazeny k jednotlivým možným odpovědím na položenou otázku a tak by divák nemusel udělat nic jiného, než že by v teple svého domova zmáčkl jedno ze čtyř tlačítek a přímo tak ovlivnil dění v televizi.

V této části bude dále diskutován návrh paketu pro hlasovací službu, sumarizačního paketu a závěrem i návrh vlastní komunikace hlasovací služby.

4.1 Návrh hlasovací služby

Jedna z možností jak navrhnout hlasovací službu je, že po zahájení hlasování v televizi by vysílač zaslal pomocí multicastu všem koncovým uzlům informaci o tom, že začalo hlasování. Po obdržení a zobrazení této informace na obrazovce televizoru by mohly diváci začít hlasovat. Informace o tom, jaké divák zmáčkl tlačítko by se vložila do hlasovacího paketu a pomocí unicastu by se poslala vysílači. Ten by pak vyhodnotil a zobrazil výsledky hlasování.

Pro zasílání hlasovacích paketů zpět ke zdroji multicastu by se mělo využít zpětnovazebního stromu a vlastností TTP protokolu jako je sumarizace a agregace. Takže v každém FT, které spravuje skupinu koncových uživatelů, by mohla probíhat sumarizace hlasovacích paketů. Sumarizace by probíhala tak, že by FT přijímalo hlasovací pakety, z nich by vyčlenilo informaci o tom, jaké tlačítko bylo zmáčknuto. Tato informace by se v FT kumulovala do doby, než by FT zaslalo sumarizační paket do vyšší úrovně zpětnovazebního stromu, kde by se opět provedla sumarizace a tato činnost by se opakovala, dokud by se nedosáhlo nejvyšší úrovně stromu, kde by RFT(Root Feedback Target) provedlo finální sumarizaci a zpracované výsledky hlasování by nepředalo vysílači. Zjednodušený model hlasování využívající zpětnovazební strom je zobrazen na obr. 4.1, kde čárkovaná čára značí zaslání informace o začátku hlasování prostřednictvím multicastové architektury SSM a plná čára značí unicastově zaslané hlasovací pakety.



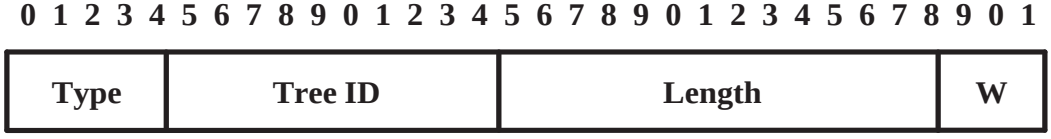
Obr. 4.1: Obecný model hlasování

4.2 Návrh paketu pro hlasovací službu

Tento paket by měl sloužit k přenosu informace o tom, jaké tlačítko stiskl divák sedící u obrazovky. Z tohoto důvodu bude paket přenášet ve své zátěži pouze informaci o tom, jaké tlačítko bylo stisknuto. Bude se tedy jednat o informaci, jestli bylo stisknuto červené(red), zelené(green), žluté(yellow) nebo modré(blue) tlačítko. Přesně v tomto pořadí jsou tlačítka seřazena na ovladačích.

4.2.1 Obecná hlavička interakčních paketů

Obecná hlavička interakčních paketu je znázorněna na obr. 4.2. Jednotlivá pole hlavičky jsou seřazena tak, aby při průchodu jednotlivými síťovými uzly ve zpětnovazebním stromu bylo nejdříve zjištěno, o jaký typ paketu se jedná, do jakého stromu patří, jakou má délku a na základě typu paketu na závěr ještě nějaké doplňující informace. Tuto hlavičku by v budoucnosti mohly využívat například služby určené pro přenos krátkých textových zpráv mezi koncovými uživateli nebo podobné služby zajišťující interaktivitu.



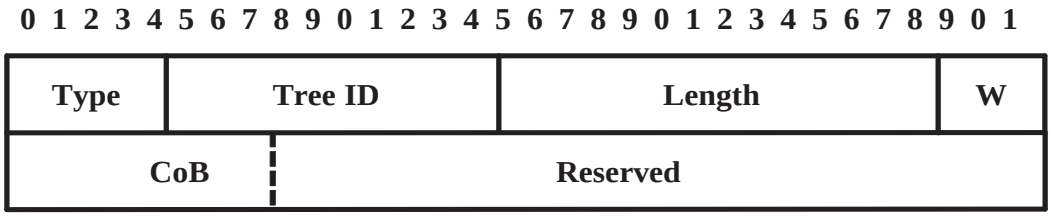
Obr. 4.2: Obecná hlavička interakčních paketů

Popis jednotlivých polí hlavičky interakčních paketů :

- Pole Type určuje typ paketu, tedy i to, jaká polr bude obsahovat zátěž. Velikost tohoto pole je 5 bitů, což znamená, že tuto hlavičku může využívat až 32 typů paketů. Paket pro hlasovací službu bude nastavena na typ 1, tedy pole Type bude obsahovat sekvenci bitů 00001.
- Tree ID je jedinečný identifikátor stromu, ve kterém bylo iniciováno hlasování a ve kterém se bude ve zpětné vazbě provádět sumarizace paketů hlasovací služby. Toto pole má velikost 10 bitů stejně jako u paketů, které využívá TTP protokol pro komunikaci ve zpětnovazebním stromu.
- Length je pole určující celkovou velikost paketu v bytech. Blok má velikost 14 bitů, což znamená, že paket může dosáhnou maximální velikosti 16384 bytů.
- Blok W(winning) má velikost 3 bity a rozšiřuje možnosti paketu. Určuje například, zad se jedná o hlasování, kde se dá něco vyhrát. Blok W by měl být defaultně nastaven na hodnotu 0, což znamená, že paket ve své zátěži bude obsahovat pouze bloky, které budou určeny pro přenos informace o vybrané odpovědi na položenou otázku. Nastavení bloku například na hodnotu 1, by znamenalo, že prvních 32 bitů zátěže by mohl být třeba identifikátor koncového uzlu na jehož základě by se provádělo určení výherce soutěže.

4.2.2 Hlasovací paket

Jak je vidět, tak oproti hlavičce interakčních paketů, kterou by měl využívat hlasovací paket přibyly dva bloky. Které budou popsány dále v textu. Kompletní hlasovací paket je zobrazen na obr. 4.3



Obr. 4.3: Hlasovací paket

Blok CoB(Color of Button) by měl přenášet informaci o tom, jaké barevné tlačítko bylo stisknuto a tedy jakou odpověď považuje divák za správnou nebo pro jakou možnost hlasuje. Velikost tohoto pole je 8 bitů, což je víc, než by bylo potřeba pro přenos informace o tom, které ze čtyř tlačítek bylo stisknuto. Velikost pole je

navržena s rezervou pro budoucí využití, protože není dopředu jisté, jestli v budoucnu na ovladačích nepřibudou další tlačítka, která by se dala využít k hlasování nebo jestli se nebudou k hlasování používat i tlačítka pro výběr televizních kanálů. Přiřazení tlačítka k možnostem by mělo být následující :

- červené tlačítko představuje možnost a) a hodnota v poli CoB by měla být 0
- zelené tlačítko představuje možnost b) a hodnota v poli CoB by měla být 1
- žluté tlačítko představuje možnost c) a hodnota v poli CoB by měla být 2
- modré tlačítko představuje možnost d) a hodnota v poli CoB by měla být 3

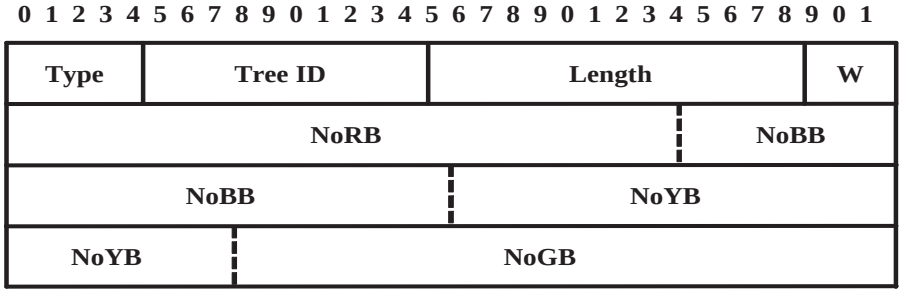
Přiřazení barev k možnostem odpovídá umístění tlačítek na ovladačích.

A poslední pole paketu je pole Reserved, což je pole, které je určeno pro budoucí využití. Celková délka hlasovacího paketu včetně hlavičky je tedy 8 bytů.

4.2.3 Možné další typy paketů

Sumarizační paket

Protože komunikace hlasovací služby by měla využívat zpětnovazebního stromu a jeho vlastností, je zde potřeba ještě navrhnout paket, který by měl sumarizovat hlasovací pakety z jednotlivých skupin koncových uzlů nebo od sumarizačních uzlů z nižších úrovní stromu. Návrh tohoto sumarizačního paketu hlasovací služby je zobrazen na obr. 4.4.



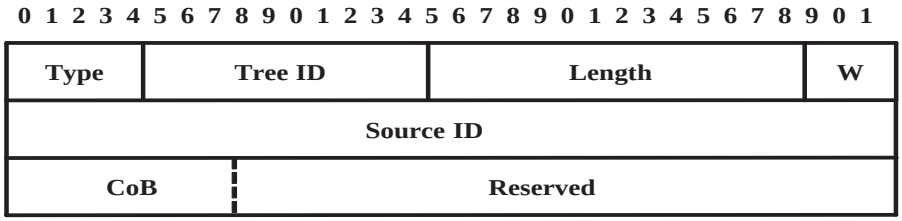
Obr. 4.4: Sumarizační paket hlasovací služby

Tento paket využívá stejnou hlavičku jako hlasovací paket, ale hodnota v poli Type nebude nastavena na 1, jak tomu je u hlasovacího paketu, ale na hodnotu 2. Jednotlivé bloky zátěže sumarizačního paketu přenášejí informaci o tom, kolikrát bylo zmáčknuto to či ono tlačítko. Tyto bloky jsou v zátěži čtyři a každý má velikost 24 bitů. To znamená, že blok může informaci maximálně o 16 777 215 poslaných

hlasech. Pro zemi jako je Česká republika je tento počet na jednu možnost příliš velký, ale jsou zde i větší země a je lepší, použít větší číslo než, aby při hlasování nastali nějaké komplikace. Je také nutno vzít v úvahu, že paket bude použit například i pro finální sumarizaci, takže i proto byla zvolena taková velikost těchto polí. Navíc diváci určitě všichni nebudou hlasovat ve stejný okamžik, a proto by měly sumarizační servery generovat tento paket každých 5 s jako RSI paket.

Výherní paket

Jak už bylo řečeno v popisu jednotlivých polí hlavičky interakčních paketů, tak pole W(Winning) rozšiřuje vlastnosti paketu, například by se mohlo jednat o výherní hlasování. Při nastavení tohoto pole na hodnotu 1 by v zátěži přibylo pole Source ID. Toto pole by identifikovalo klienta, který hlasoval, tento identifikátor by mohla být například IP adresa. Jak by mohl vypadat takový výherní paket je zobrazeno na obr. 4.5.



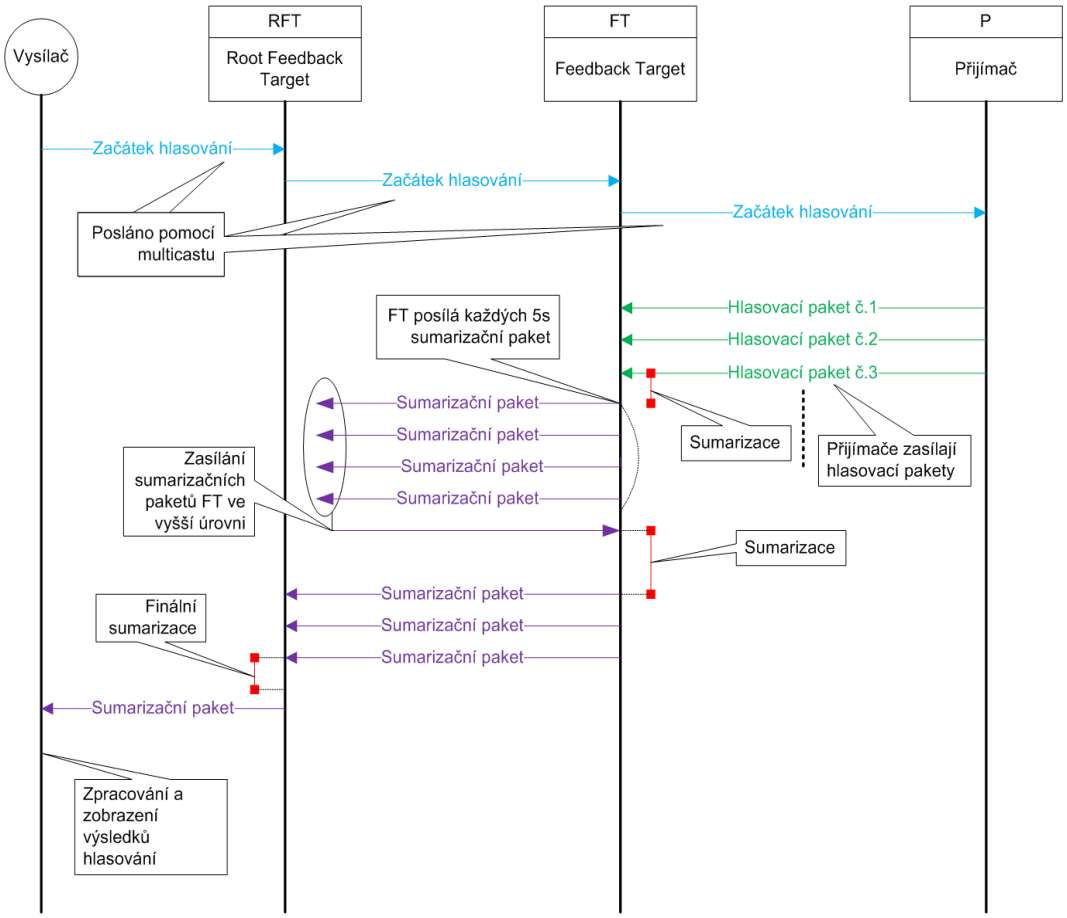
Obr. 4.5: Výherní paket

Výherní hlasování by probíhalo zhruba tak, že vysílač by poslal informaci, že započalo hlasování, respektive soutěž o ceny, a na základě toho by se pole W nastavilo na hodnotu 1, aby byla přenesena informace o zdroji hlasovacího paketu. Pro sumarizaci by se ale museli navrhnout algoritmy, které by rozlišovaly, kdo se zúčastnil hlasování, a kdo ne. Na závěr by byla ještě potřeba databáze, kde by k jednotlivým identifikátorů koncových uzlů byly přiřazeny informace o klientovi. Dále by pak bylo nutné omezit počet hlasů z jednoho koncového uzlu na nutné minimum, nejlépe však na jeden hlas z každého koncového uzlu. Návrhem tohoto druhu hlasování se dále tato práce dále zabývat nebude.

4.3 Návrh komunikace hlasovací služby

Jak již bylo v úvodu této kapitoly řečeno, hlasování by mělo probíhat tak, že vysílač pošle multicastově zprávu o zahájení hlasování a koncové uzly by na ni měly zareagovat tak, že pošlou hlasovací paket nejbližšímu sumarizačnímu serveru(FT),

ten provede sumarizaci hlasovacích paketů a tuto sumarizační zprávu pošle do vyšší úrovně zpětnovazebního stromu. Tyto operace se budou opakovat, dokud se nedosáhne nejvyšší úrovně stromu. Jak by měla vypadat komunikace hlasovací služby je znázorněno na časovém diagramu na obr. 4.6.



Obr. 4.6: Časový diagram komunikace hlasovací služby

Jak z časovém diagramu vyplývá, tak jako první by se mělo zahájit hlasování. Takže vysílač nejprve pošle pomocí multicastu informaci o tom, že hlasování začalo. Po obdržení informace o začátku hlasování by měli přijímače tuto informaci zobrazit na televizní obrazovce a diváci by mohly začít hlasovat. Tato část se zobrazením a hlasováním diváků není součástí časového diagramu. Jakmile začnou diváci hlasovat, jednotlivé přijímače by měli začít generovat hlasovací pakety a posílat je nejbližšímu FT, který spravuje skupinu koncových uzlů, do které daný přijímač patří. FT by tedy mělo přijímat hlasovací pakety od jednotlivý přijímačů a mělo by provádět sumarizaci těchto hlasovacích paketů. FT každých 5 s vygeneruje sumarizační paket, který je určen pro tento účel, a pošle ho FT do vyšší úrovně stromu. FT ve vyšší úrovni zpětnovazebního stromu provede sumarizaci přijatých sumari-

začních paketů a vygeneruje vlastní sumarizační paket, který pošle RFT do nejvyšší úrovně stromu. RFT provede finální sumarizaci a zašle sumarizační paket vysílači. Ten by po příjmu sumarizačního paketu měl zpracovat informace o tom, kolik diváků hlasovalo pro tu či onu odpověď.

Jelikož hlasovací služba bude, stejně jako RTCP protokol, využívat pouze 5% z celkové šířky pásma je zbytečné zatěžovat tuto minimální šířku pásma vytvářením spojení a potvrzováním příchozích dat. Proto by komunikace hlasovací služby měla probíhat na bázi spojově neorientovaného a nespolehlivého transportního protokolu UDP. Tento protokol se pro tuto službu hodí více, než spolehlivý a spojově orientovaný protokol TCP.

4.4 Možné nevýhody návrhu

Již dnes je zcela běžné, že televizní stanice vysílá svůj program na dvou odlišných kanálech na jednom v klasickém rozlišení a na druhém ve vysokém rozlišení. Kdyby se tyto praktiky aplikovaly na internetovou televizi, nastal by zde problém s rozlišením těchto dvou kanálů se stejným programem. Problém by nebyl v tom, že by pro jeden televizní program musely být vytvořeny dva zpětnovazební stromy, ale v rozlišení příchozích hlasů. Jelikož každý zpětnovazební strom má svůj jedinečný identifikátor, tak by po skončení hlasováni mohly vzniknou dva výsledky, které by však patřily jednomu hlasování. Tím pádem by se ještě muselo zajistit slučování těchto výsledků v jeden konečný, což by zejména při vysílání ze dvou různých zařízení znamenalo potřebu dalšího zařízení, které by provádělo sloučení.

Řešením by mohl být další identifikátor, který by ještě určoval, o který program se jedná. Což by znamenalo, že kdyby z příchozích paketů FT zjistilo vysílání jednoho programu ve dvou kanálech, respektive, že hlasování pro jeden program probíhá ve dvou navzájem nezávislých zpětnovazebních stromech, tak by sumarizační pakety posílal pouze v jednom vybraném stromu a tím by se docílilo pouze jednoho výsledku bez potřeby dalších zařízení. Znamenalo by to pouze větší softwarovou náročnost na FT v nejnižší úrovni stromu, které by prováděli rozpoznáváni na základě identifikátoru programu.

5 JAVA A ECLIPSE

Před tím, než bude popsána samotná realizace hlasovací služby, bude zde uvedeno pár základních informací o programovacím jazyku Java a jeho možnostech pro síťové komunikace. Dále zde bude zmínka o vývojovém prostředí Eclipse IDE, v němž probíhá samotná realizace hlasovací služby.

5.1 Programovací jazyk Java

Java je objektově orientovaný programovací jazyk, který vyvinula firma Sun Microsystems. Díky svým vlastnostem se stal jedním z nejpoužívanějších programovacích jazyků na světě. Mezi vlastnosti tohoto jazyka patří modularita programu a řídících instrukcí, silná typová kontrola, multithreading, ošetření vyjímek, správa paměti, silná podpora pro databáze, XML a síťové operace. Přenositelnost tohoto jazyka umožňuje vytvářet programy, které mají pracovat na různých systémech počínaje čipovými kartami, přes mobilní telefony, aplikace pro stolní počítače až po rozsáhlé distribuované systémy pracující na řadě spolupracujících počítačů.

5.2 Síťování v Javě

Význam datových přenosů a síťové komunikace obecně v posledních letech stoupá a tak i Java věnuje této oblasti mimořádnou pozornost. A z tohoto důvodu je také Java často nasazována ať už na stranu serveru nebo klienta.

Pro práci se sítí slouží v Javě balíky java.net, java.io a java.nio. Z těchto balíků zde budou popsány pouze ty třídy a metody použité pro realizaci hlasovací služby.

5.2.1 Balík java.net

V tomto balíčku, který podporuje komunikaci na bázi rodiny protokolů TCP/IP, jsou umístěny třídy určené pro síťovou komunikaci. Obsahuje prostředky pro spojově orientovanou a datagramovou komunikaci a také možnost komunikace typu multicast. Následující informace pochází z [8] a [9].

Třída java.net.InetAddress

Tato třída reprezentuje IP adresu a definuje metody pro její získávání. Některé ze základních metod této třídy jsou uvedeny v tabulce Tab. 5.1.

Základní metody z třídy java.net.InetAddress	
getByAddress(byte[] addr)	Vrací adresu na základě pole bajtů. 4 bajty pro IPv4, 16 bajtů pro IPv6.
getByName(String host)	Vrací adresu podle hostname.
getLocalHost()	Vrací adresu pro localhost.

Tab. 5.1: Základní metody třídy java.net.InetAddress

Třída java.net.Socket

Sokety jsou jedny z nejdůležitějších částí síťových aplikací, jejichž prostřednictvím probíhá veškerá síťová komunikace. Sokety reprezentují spojově orientovanou komunikaci na straně klienta. V tabulce Tab. 5.2 jsou uvedeny některé základní metody této třídy.

Základní metody z třídy java.net.Socket	
connect(SocketAddress endpoint)	Připojí se k serveru. Při neúspěchu vyhodí výjimku SocketTimeoutException.
getInputStream()	Vrací vstupní proud soketu, ze kterého čteme příchozí data.
getOutputStream()	Vrací výstupní proud soketu, do kterého data zapisujeme.
close()	Uzavře soket.

Tab. 5.2: Základní metody třídy java.net.Socket

Třída java.net.DatagramSocket

DatagramSocket je třída určená pro komunikaci prostřednictvím UDP protokolu a pro všesměrovou komunikaci. Jsou v ní implementovány metody pro odesílání datagramů, pro jejich příjem a pro naslouchání na určitém portu. V tabulce Tab. 5.3 je opět uvedeno několik základních metod této třídy. Jistým překvapením je fakt, že tato třída, která je určena ke spojově neorientované komunikaci prostřednictvím protokolu UDP, obsahuje funkci connect(). Tato metoda se ve skutečnosti nikam nepřipojuje, slouží pouze k dosažení vyšší bezpečnosti.

Ukázka metod z třídy java.net.DatagramSocket	
connect(SocketAddress addr)	Připojí se ke vzdálené adrese.
send(DatagramPacket p)	Odeslání datagramu.
receive(DatagramPacket p)	Příjem datagramu.
close()	Ukončení práce se soketem.

Tab. 5.3: Ukázka metod třídy java.net.DatagramSocket

Třída java.net.DatagramPacket

Pomocí této třídy lze nastavovat data, která budou vyslána pomocí třídy DatagramSocket a na přijímací straně umožňuje tato třídy přístup k těmto datům. Obsahuje metody pro nastavení cílového portu a adresy na kterou se bude datagram posílat. Dále obsahuje metody, které umožňují práci s daty přenesenými datagramem.

5.3 Eclipse IDE

Eclipse je open source vývojové prostředí určené pro programování v jazyce Java. Open source znamená, že se jedná počítačový software s otevřeným zdrojovým kódem. V základní verzi Eclipse obsahuje pouze prostředky pro psaní standardního kódu v Javě, další rozhraní jako třeba pro návrh grafického uživatelského rozhraní aplikací nebo aplikační server je nutné dodat formou různých pluginů.

6 REALIZACE HLASOVACÍ SLUŽBY

V následující kapitole bude popsána realizace hlasovací služby, která však nebude úplně ctít výše popsaný návrh. V této práci bude realizace omezena pouze na odeslání hlasovacích paketů k vybranému FT, které provede následnou sumarizaci těchto informací a odešle je prostřednictvím sumarizačního paketu k FTM, kde bude provedeno zpracování těchto paketů a následné zobrazení výsledků.

Komunikace tedy bude probíhat tak, že se nejdříve v okně, které představuje přijímač, stiskne tlačítko představující jedno ze čtyř barevných tlačítek ovladače. Informace o stisknutém tlačítku se vloží do hlasovacího paketu, který bude následně odeslán k vybranému FT. V tomto FT se provede vyčlenění informace o barvě tlačítka a její následné zpracování. Po sléze se provede odeslání pomocí sumarizačního paketu k FTM, kde se opět provede vyčlenění informace o počtech zmáčknutí jednotlivých tlačítek. Zobrazení výsledků hlasování bude provedeno tak, že se v okně, které představuje FTM, stiskne tlačítko a po jeho stisknutí se zobrazí okno, které bude obsahovat aktuální výsledky hlasování, respektive informace o tom, kolikrát bylo zmáčknuto to či ono tlačítko.

Hlasovací služba je realizována pod strukturou testovací sítě, která slouží pro simulaci vlastností nově navrhovaného TTP protokolu, který je určen pro komunikaci ve zpětnovazebním stromu služby IPTV. Tato testovací síť je realizovaná v programovacím jazyce Java ve vývojovém prostředí Eclipse IDE.

6.1 Realizace hlasovacího paketu

Pro realizaci hlasovacího paketu byl v projektu TTP4 vytvořen balíček s názvem cz.vutbr.feec.voting.packet, ve kterém se nachází třída PaketHeader, kde je, jak už název napovídá, realizována hlavička hlasovacího paketu, respektive obecná hlavička interakčních paketů. Tento balíček dále obsahuje třídu VotingPacket a třídu SumarizationPacket, v prvně jmenované je provedena realizace hlasovacího paketu a v druhé sumarizačního paketu hlasovací služby.

6.1.1 Třída PaketHeader

Tato třída obsahuje několik metod pro porovnání obsahu dvou objektů. V těchto metodách se po přetypování objektů, které jsou předávány těmto metodám jako vstupní parametr, na instanci třídy PaketHeader provádí porovnání s jednotlivými poli hlavičky hlasovacího paketu. Mezi tyto metody patří metoda *compareTo()*, která vyšetřuje rovnost, případně druh nerovnosti dvou hlaviček hlasovacího paketu, respektive proměnných představující jednotlivá pole hlavičky, a vrací hodnoty -1, 0,

1 v závislosti na tom, je-li hlavička představující vstupní parametr metody menší, rovna nebo větší než hlavička deklarovaná ve třídě PaketHeader.

Další metoda určená k porovnávání je metoda *equals()*. Tato metoda vyšetřuje pouze rovnost dvou instancí třídy PaketHeader. Takže tato metoda vrátí true pouze, jsou-li si všechna pole hlavičky rovna. S metodou *equals()* se setkáme i dále v textu, protože je použita při testování realizovaného paketu.

Mezi jednu z důležitých metod této třídy patří metoda *generate()*. V této metodě probíhá generování celé hlavičky na straně vysílače. Část kódu této metody provádějící generování hlavičky paketu je zobrazena na obr. 6.1.

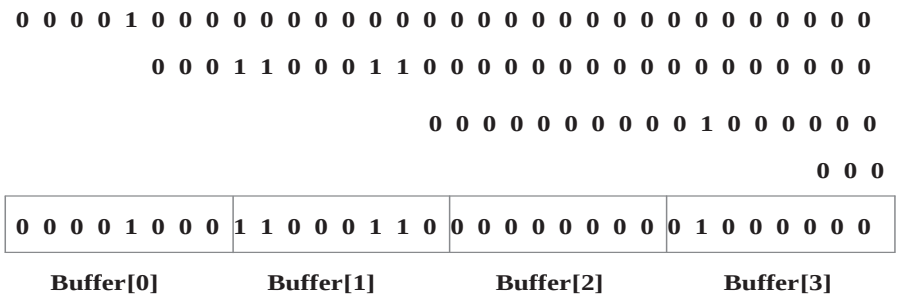
```
long fullHeader = ((packetType << 27) & 0xF8000000 | (idTree << 17) & 0x7FE0000
        | (length << 3) & 0x1FFF8 | winning & 0x7);


buffer[offset] = (byte)((fullHeader & 0xFF000000) >> 24);
buffer[offset + 1] = (byte)((fullHeader & 0xFF0000) >> 16);
buffer[offset + 2] = (byte)((fullHeader & 0xFF00) >> 8);
buffer[offset + 3] = (byte)(fullHeader & 0xFF);
```

Obr. 6.1: Část kódu metody *generate()*

Mezi vstupní parametry této metody patří proměnná *buffer*,do které se uloží vygenerovaná hlavička hlasovacího paketu. Další vstupním parametrem je *offset*, který ukazuje na místo v paměti, na kterém začne vkládání hlavičky do paměti. Generování hlavičky probíhá tak, že se nejdříve vezme pole *packetType*, které má velikost 5 bitů a pomocí operace bitového posunu se posune o 27 bitů vlevo, takže se dostaneme na celkovou délku hlavičky 32 bitů. Po bitovém posuvu se nad celou délkou hlavičky provede maskování, což znamená, že se pomocí logického součinu zamezí náhodnému výskytu nějaké informace v bitech, do kterých zatím žádná informace vložena nebyla. Zajistí se tím tedy, že informace o typu paketu bude zatím jediná vložená informace, která se vyskytuje v prvních pěti bitech a dalších 27 bitů bude nastaveno na hodnotu 0. Podobné operace se provádí i nad zbylými poli hlavičky. Tyto operace se liší pouze hodnotou bitového posuvu a maskováním, které se vždy provádí pouze nad aktuálním polem a ještě nepoužitými bity. Pole *idTree* má velikost 10 bitů a je posunuto o 17 bitů vlevo, tak aby následovalo hned za polem *packetType*, za které se vloží pomocí operace logického součtu. Před vložením tohoto pole za pole *packetType* se opět provede maskování. Následuje pole *length*, které má velikost 14 bitů a posune se o 3 bity vlevo a aplikují se na něj stejné operace jako

na předchozí pole. Posledním polem paketu je pole *winning*, které má velikost 3 bity a už se na něj neaplikuje operace bitového posuvu. Po uložení celé hlavičky paketu do proměnné *fullHeader* se provádí rozdělení hlavičky na jednotlivé části o velikosti 8 bitů, které jsou postupně vkládány na po sobě následující místa do paměti.

Příklad generování hlavičky je uveden na obr. 6.2. Na tomto obrázku jsou znázorněny bitové posuny pro jednotlivá pole hlavičky hlasovacího paketu, následované vytvořením hlavičky hlasovacího paketu a jejím rozdělením do jednotlivých oblastí paměti. Pro tento příklad je typ paketu nastaven na hodnotu 1, ID zpětnovazebního stromu na hodnotu 99, délka paketu je 8 bytů a pole *winning* je nastaveno na hodnotu 0.



Obr. 6.2: Ukázka generování hlavičky

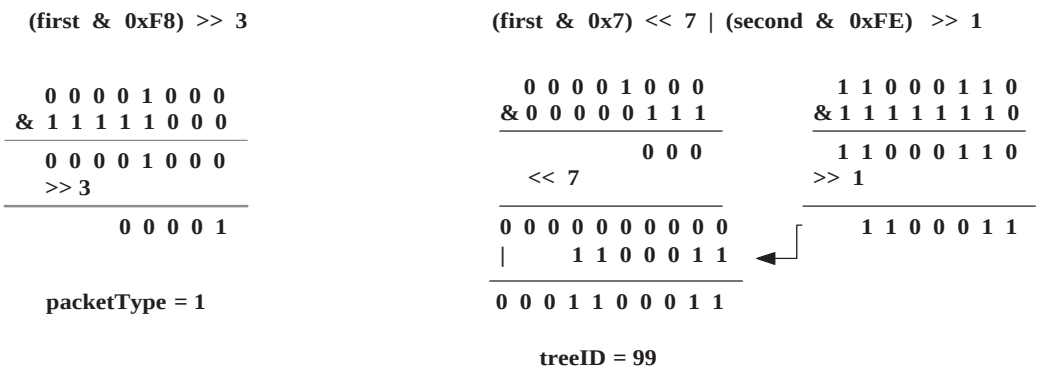
Další důležitou metodou je metoda *parse()*. Tato metoda se používá na přijímací straně, v případě hlasovací služby se jedná o příjímání na straně FT, a slouží k vyčlenění jednotlivých polí z hlavičky paketu. Jedná se tedy o inverzní metodu k metodě *generate()*. Kód metody *parse()* je zobrazen na obr. 6.3. Ve vstupní proměnné *resul-*

```
public int parse(byte[] resultArray, int startPos, int length)
        throws PacketParseException {
    if(length<4){
        throw new PacketParseException("Header is short");
    }
    int first = (int)resultArray[startPos];
    int second = (int)resultArray[startPos + 1];
    int third = (int)resultArray[startPos + 2];
    int fourth = (int)resultArray[startPos + 3];

    this.setType((first & 0xF8)>> 3);
    this.setTreeID((first & 0x7) << 7 | ((second & 0xFE) >>1));
    this.setLength((second & 0x1) << 13 | (third & 0xFF) << 5 | ((fourth & 0xF8) >> 3));
    this.setWinning(fourth & 0x7);

    return 4;
}
```

Obr. 6.3: Metoda *parse()*

tArray, která je datového typu byte, se této metodě předává celá hlavička, která je rozdělená do jednotlivých oblastí paměti. V proměnné *startPos* je uložena informace o tom, na kterém místě v paměti začína hlavička hlasovacího paketu a v proměnné *length* se této metodě předává délka hlavičky paketu v bytech. Na začátku metody *parse()* se provede přetypování jednotlivých bloků paměti na celočíselný datový typ integer a dojde k předání hodnot uložených v těchto blocích proměnným *first*, *second*, *third* a *fourth*. Proměnná *first* obsahuje celé pole *packetType* a část pole *idTree*, respektive tři bity tohoto pole. Metodě *setType()* se jako vstupní parametr předává typ paketu, který se získá z proměnné *first*. Získání informace o typu paketu z proměnné *first* probíhá tak, že je nad touto proměnnou proveden logický součin s bitovou posloupností 1111 1000, která je v kódu vyjádřena hexadecimálně(*0xF8*). Tímto logickým součinem se vynuluje část patřící poli *idTree* a následným bitovým posunem o 3 bity vpravo dostaneme informaci o typu paketu, které se předává jako vstupní parametr metodě, která provede nastavení typu paketu. Podobně, ale trochu složitěji se získává informace o ID zpětnovazebního stromu. Z proměnné *first* jsou potřeba první tři bity, myšleno první tři bity s nejnižší vahou. Ty se získají tak, že se provede logický součin mezi proměnou *first* a bitovou posloupností 0000 0111, která je v kódu vyjádřena hexadecimálně(*0x7*). Po provedení logického součinu zůstanou ony první tři bity, které se posunou o 7 bitů vlevo na celkovou délku pole idTree, což je 10 bitů. Zbylých 7 bitů se dostane obdobným způsobem z proměnné *second*. Výsledky těchto operací nad proměnnými *first* a *second* se za pomoci logického součtu sloučí v jedno pole a to se předá jako vstupní parametr metodě *setTreeID*, pomocí které se nastaví ID zpětnovazebního stromu.

Pro lepší pochopení je pro vygenerovanou hlavičku na obr. 6.2 uveden příklad vyčlenění polí *packetType* a *idTree* na obr. 6.4. Je třeba dodat, že proměnná *first* představuje *Buffer[0]* a proměnná *second Buffer[1]* z obr. 6.2.



Obr. 6.4: Ukázka vyčlenění polí *packetType* a *idTree*

Třída PacketHeader dále obsahuje jen metody pro nastavení jednotlivých polí hlavičky a metody pro získání nastavených hodnot jednotlivých polí.

6.1.2 Třída VotingPacket

Třída VotingPacket je rozšířením třídy PaketHeader. Což znamená, že tato třída převezme beze změny proměnné a metody třídy PaketHeader. Takže v této třídě se skládá celý hlasovací paket, tedy hlavička plus zátěž. K metodám třídy PaketHeader se přistupuje klíčovým slovem *super*. Například k metodě *equals()* třídy PaketHeader se přistoupí následujícím způsobem *super.equals()* a do kulatých závorek se předá objekt, respektive instance třídy PaketHeader, která je určena k porovnání.

Třída VotingPacket obsahuje metodu *generate()*, která volá pomocí klíčového slova *super* metodu *generate()* ze třídy PaketHeader a předává ji své vstupní parametry a zároveň volá metodu *generateRestOfVotPack()*, která generuje zátěž hlasovacího paketu podle návrhu v kapitole 4. Takže tato metoda provádí generování celého hlasovacího paketu včetně hlavičky. Generování zbytku paketu je provedeno obdobným způsobem, jako ve stejnojmenné metodě třídy PaketHeader, jen s tím rozdílem, že jsou pro generováni využity funkce, které jsou již implementovány v projektu TTP4.

Tato třída rovněž obsahuje metodu *parse()*, která volá stejnojmennou metodu ze třídy PaketHeader a metodu *parseRestOfVotPack()*, která slouží k vyčlenění polí *colorOfButton* a *reserved* z hlasovacího paketu.

6.1.3 Třída SumarizationPacket

V této třídě je provedena realizace sumarizačního paketu hlasovací služby. Třída SumarizationPacket opět rozšiřuje vlastnosti třídy PaketHeader, stejně jako třída VotingPacket. Pro realizaci sumarizačního paketu jsou použity stejné metody jako ve třídě VotingPacket, liší se pouze tím, že sumarizační paket obsahuje podle návrhu v kapitole 4 jiná pole v zátěži paketu. Z toho důvodu se lehce liší i metody *equals()*, *generate()* a *parse()*

6.1.4 Testování Hlasovacího paketu

Pro testování správnosti realizace hlasovacího paketu byl v projektu TTP4 ve složce Test vytvořen balíček s názvem cz.vutbr.feec.voting.packet a v něm třída VotingPacketTest. Tato třída obsahuje dvě stejnojmenné metody *parseTest()*, jedna je bez vstupních parametrů a druhá má za vstupní parametr celé číslo, které ukazuje na místo v paměti, kam se začnou data vkládat nebo odkud se budou číst. V metodě se vstupním parametrem se vytvoří dvě nové instance třídy VotingPacket a to *send*

a *recv*. Do vytvořené instance *send* se do všech polí hlasovacího paketu vloží data a pomocí metody *generate()* se v této instanci vytvoří paket. Všechny informace jsou uloženy v paměti, na místě daném vstupním parametrem metody *parseTest()* a je známa délka paketu, respektive počet bytů v paměti, které hlasovací paket zabírá. Všechny tyto informace jsou vloženy to metody *parse()* instance *recv* a tímto se provede vyčlenění jednotlivých polí hlasovacího paketu. Na závěr je metodě *equals()* instance *send* předán jako parametr instance *recv* a je provedeno porovnání obou instancí, respektive jednotlivých polí hlasovacího paketu.

Metoda *parseTest()* bez vstupních parametrů slouží pouze k volání metody s parametry a předává ji různé ukazatele na místa v paměti, aby bylo ověření realizace hlasovacího paketu ověřeno vícekrát.

Tímto způsobem byla ověřena správnost realizace hlasovacího paketu, respektive metod *generate()* a *parse()* a správnost realizace sumarizačního paketu.

6.2 Práce s pakety během komunikace

Během komunikace se s pakety, respektive s instancemi tříd VotingPacket a SumarizationPacket, pracuje v několika třídách. Po vytvoření instance třídy VotingPacket následujícím konstruktorem: „*VotingPacket vote = new VotingPacket()*“, pomocí kterého se přistupuje k metodám třídy VotingPacket v níže popsané třídě VotingSendAction, kde dochází k naplnění hlasovacího paketu v metodě *fillVotingPacket()*. Této metodě se jako vstupní parametry předávají hodnoty jednotlivých polí hlasovacího paketu. Jednotlivá pole se nastavují pomocí metod k tomu určených ve vytvořené instanci *vote* třídy VotingPacket. Například nastavení tlačítka, které bylo zmáčknuto je provedeno následovně: „*vote.setColorOfButton(colorOfButton)*“. Po naplnění paketu se ze stejné instance třídy zavolá metody *generate()*, které se předá paměťový prostor, do kterého se vloží vygenerovaný paket, a ukazatel na buňku v paměti, kde začíná vygenerovaný hlasovací paket.

Výše popsané akce se dějí na vysílací straně. Na přijímací straně je vytvořena nová instance třídy VotingPacket. Z této instance se jako první zavolá metoda *parse()*, které se předají jako vstupní parametry paměťový prostor, ve kterém je uložen paket, ukazatel na pozici, kde paket začíná a celková délka paketu. Tyto předané parametry byly přeneseny prostřednictvím UDP datagramu. Metoda *parse()* vyčlení jednotlivé informace z paketu a vloží do proměnných, které představují jednotlivá pole paketu.

6.3 Realizace komunikace hlasovací služby

V následující části bude popsána realizace hlasovací služby, respektive jednotlivé třídy, které pro tuto komunikaci slouží nebo se na ni nějakým způsobem podílí. Popis realizace jednotlivých tříd je seřazen chronologicky podle jejich využívání během komunikace hlasovací služby.

6.3.1 Třída WindowReceiver

V balíčku cz.vutbr.feec.gui jsou umístěny třídy definující grafická uživatelská rozhraní. Jednim z těchto rozhraní je i třída WindowReceiver, která slouží ke grafickému znázornění přijímače. Pro hlasovací službu byly do tohoto rozhraní přidána čtyři tlačítka představující čtyři barevná tlačítka dálkového ovladače. Pro každé tlačítko byla nastavena akce, která se provede po jejich zmáčknutí. V tomto případě se po stisku tlačítka zavolá metoda *setCoB()* ze třídy ReceiverState a jako vstupní parametr ji bude předána informace o tom, jaké tlačítko bylo stisknuto. Vstupní parametr tedy může být 0 pro červené, 1 pro zelené, 2 pro žluté a 3 pro modré.

6.3.2 Třída VotingSendAction

V balíčku cz.vutb.feec.ttp.session.receiver byla vytvořena třída VotingSendAction, která dědí vlastnosti třídy PeriodicalSender, což v praxi znamená, že se snaží periodicky odesílat hlasovací pakety prostřednictvím metody *sendPacket*. Perioda odesílání hlasovacích paketů byla nastavena na 2 s. Část kódu metody *sendPacket* je na

```
public void sendPacket() {

    try {
        this.cob = stateR.getCoB();

        if(this.cob != -1){
            setFTAndPort(packet);
       fillVotingPacket(99, 8, 0, stateR.getCoB(), 0);

       int len = 0;
    try{
        len = vote.generate(buf, 0);
    }
    catch (PacketGenerateException e1) {

        e1.printStackTrace();
        }
          packet.setData(buf, 0, len);
            socket.send(packet);
```

Obr. 6.5: Ukázka části kódu metody *sendPacket()*

obr. 6.5. Aby se zabránilo odesílání hlasovacího paketu v době, kdy nebylo zmáčknuto žádné tlačítko, byla zde vytvořena podmínka. Hodnota proměnné *cob* je nastavena na hodnotu -1 jak ve třídě VotingSendAction, tak ve třídě ReceiverState. Takže je-li hodnota této proměnné stále -1, žádný paket se neodešle, pokud se ovšem zmáčkne tlačítko, do této proměnné se uloží některá ze čtyř možných hodnot a tím pádem bude splněna podmínka a hlasovací paket se bude moci odeslat. Po splnění podmínky se nejdříve nastaví IP adresa zvoleného FT a port, na kterém bude FT přijímat hlasovací pakety. Po nastavení IP adresy a cílového portu bude za pomoci metody *fillVotingPacket()* naplněn hlasovací paket, který bude následně vygenerován pomocí metody *generate()* z třídy VotingPacket. Po vygenerování už následuje pouze odeslání hlasovacího paketu a znovu nastavení proměnné cob na hodnotu -1.

6.3.3 Třída RRReceiver

Pro příjem hlasovacích paketů a RR paketů je na straně sumarizačního serveru FT v balíčku cz.vutbr.feec.ttp.session.ft určena třída RRReceiver. Po příjmu hlasovacího paketu se provede vyčlenění informace o typu paketu, podle kterého se s paketem dále nakládá. Pro zpracování hlasovacího paketu je určena metoda *handleVotingPacket()*, ve které se provede vyčlenění informace o barvě tlačítka pomocí metody *parse()* třídy VotingPacket. Tato informace slouží dále jako rozhodovací výraz příkazu *switch*, ve kterém se podle barvy tlačítka provede zvýšení hodnoty proměnné, která hlídá, kolikrát bylo ono tlačítko zmáčknuto. Pro zvýšení hodnoty proměnné o 1 jsou vytvořeny čtyři metody ve třídě FTHostState a jedná se o metody *setRed()*, *setGreen()*, *setYellow()* a *setBlue()*.

6.3.4 Třída SumarizationSendAction

Pro zasílání sumarizačních paketů od FT k FTM je určena třída SumarizationSendAction. Je realizována podobně jako třída VotingSendAction. Opět dědí vlastnosti třídy periodicalSender. Perioda odesílání sumarizačních paketů je nastavena na 5 s. Třída rovněž obsahuje metodu *sendPacket*. Tentokrát je podmínka pro odesílání nastavena tak, že dojde-li k nějakým změnám v počtech došlých hlasovacích paketů, respektive ke změně počtu jednotlivých zaslaných hlasů, dojde k odeslání sumarizačního paketu. Po splnění podmínky se nastaví adresa a cílový port FTM, naplní se sumarizační paket a pomocí metody *generate()* se vytvoří sumarizační paket a následně se odešle. Po jeho odeslání se vynulují všechny proměnné, které hlídají informace o tom, kolikrát byla zmáčknuta jednotlivá tlačítka. Takže podmínka je vlastně taková, že jsou-li všechny proměnné nastaveny na hodnotu 0, tak se paket neodešle, dojde-li k nějakým změnám, paket se odešle.

6.3.5 Třída FTMReceiverAction

Příjem sumarizačních paketů na straně FTM řeší třída FTMReceiverAction. Po příjmu sumarizačního paketu se z něj za pomoci metody *parse()* třídy PaketHeadr vyčlení informace o typu paketu. Po zjištění typu paketu se paket předá ke zpracování metodě *handleSumarizationPacket()*, která provede vyčlenění informace o tom, kolikrát byla zmáčknuta jednotlivá tlačítka pomocí metody *parse()* třídy SumarizationPacket. Tyto informace se dále předají metodám určeným ke z pracování těchto informací. Tyto metody obsahuje třída FTMHostState a jedná se o metody *setRed()*, *setGreen()*, *setYellow()* a *setBlue()*, které provedou přičtení hodnot, které byly vyčleněny ze sumarizačního paketu k proměnným, které jsou k tomu určeny.

6.3.6 Třída WindowResults

K zobrazení výsledků hlasování byla v balíčku určeném pro uživatelské grafické rozhraní vytvořena třída WindowResults. K prezentaci výsledků dochází tak, že v okně, které představuje FTM se zmáčkne tlačítko s nápisem Results a po jeho stisknutí se zobrazí okno Voting Result, kde se po stisku tlačítka s nápisem Present result zavolají metody z třídy FTMHostState, které předají výsledky hlasování, které se následně zobrazí.

6.4 Srovnání realizace hlasovací služby s návrhem

Jak již bylo v úvodu kapitoly řečeno, tato realizace úplně nectí návrh z kapitoly 4. Takto realizovaná hlasovací služba využívá pouze jedné vlastnosti TTP protokolu a tou je sumarizace jenž se provádí na vybraném FT. Další vlastností TTP, kterou je agregace, tato realizace nevyužívá, protože jednotlivé koncové uzly neposílají hlasovací paket nejbližšímu sumarizačnímu serveru, ale pouze na jeden předem daný, což u relací s větším počtem účastníků bude znamenat velký problém s udržením šířky pásma na 5% z celkové šířky pásma, která je určená pro zpětnou vazbu. Aby bylo docíleno agregace, bylo by nutné odesílat hlasovací pakety na nejbližší FT a ne na jeden vybraný, jak tomu je teď. Toho by se ale dosáhlo malou úpravou kódu v metodě VotingSendAction, kde by se místo posílání na jedeno zvolené FT využila funkce z metody RRSendAction, kde je již realizována funkce, která zasílá RR pakety k nejbližšímu FT.

Další rozdíl mezi návrhem a realizací je v tom, že přímo z vybraného FT se sumarizační paket odesílá k FTM a ne do vyšší vrstvy stromu. Pro to, aby byl ctěn návrh z kapitoly 4, by bylo nutné zjistit v jaké vrstvě stromu se FT nachází a jestli se nad ním nachází další. V tom případě by se sumarizační paket poslal FT do vyšší

vrstvy v opačném případě, když by bylo FT v nejvyšší vrstvě stromu, zaslalo by sumarizační paket FTM.

Mezi poslední rozdíly patří to, že se neposílá otázka či nějaký paket, který by obsahoval informace o zahájení hlasování, na základě kterého by proběhlo hlasování. Další rozdíl je ten, že není nijak omezen počet hlasů, které může přijímač zaslat.

6.5 Budoucí rozšíření

Jak již bylo řečeno, realizovaná hlasovací služba pro internetovou televizi nectí návrh z kapitoly 4. Takže budoucí rozšíření by mělo spočívat v tom, že bude stávající realizace doplněna o třídy a metody, které by měly realizovanou hlasovací službu přiblížit jejímu návrhu.

První úprava by měla nastat v metodě VotingSendAction a měla by spočívat v tom, že se místo na jeden zvolený sumarizační server budou hlasovací pakety posílat z přijímače na nejbližší sumarizační server. Tato změna by měla být provedena tak, že se v metodě *setFTAndPort()* nebude explicitně nastavovat sumarizační server, ale zavolá se metoda *getFtAddress()* z třídy ReceiverState, která nastaví IP adresu cíle hlasovacího paketu na adresu nejbližšího sumarizačního serveru.

Další úprava by se týkala třídy SumarizationSendAction. V této třídě by musely přibýt metody pro zjištění, v jaké úrovni se sumarizační server(FT) nachází, kterému sumarizačnímu serveru ve vyšší úrovni stromu bude zasílat sumarizační pakety a jestli se náhodou nenachází v nejvyšší úrovní stromu a neplní tím pádem funkci RFT, který by prováděl finální sumarizaci a výsledky by předával správci sumarizačních serverů(FTM).

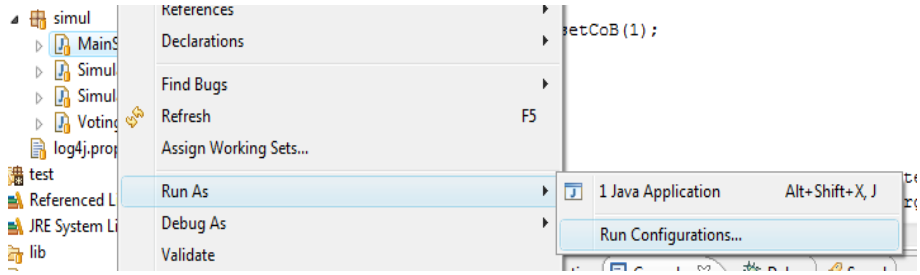
Poslední úprava by se měla týkat zasílání informací o začátku hlasování, kterou by následovala nějaká anketní či soutěžní otázka, nastavení časového limitu pro hlasování a omezení počtu zaslaných hlasů z jednoho koncového uzlu.

6.6 Popis simulace hlasovací služby

Jak již bylo řečeno, realizovaná hlasovací služba je součástí testovací sítě, kterou je nejdříve nutno nastavit. Po jejím nastavení už se může začít se simulací hlasovací služby.

6.6.1 Nastavení testovací sítě

V projektu TTP4 ve složce src se nachází balíček simul a v něm třída MainSimulation. Na tuto třídu je nejdříve nutno kliknout pravým tlačítkem myši a zvolit Run As a dále Run Configuration... přesně podle obr. 6.6.

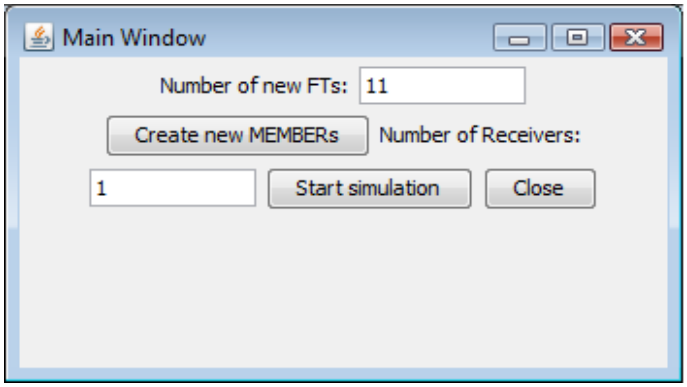


Obr. 6.6: Run Configuration...

Poté se zobrazí okno Run Configurations, kde se zkontroluje, zda je v záložce Arguments vyplněno pole VM arguments následujícím výrazem:
„-javaagent:rewriteagent.jar“, jestli ne, je třeba jej vyplnit.

Po těchto krocích už je možné spustit simulaci. Po spuštění se možná zobrazí nabídka s možnostmi jak spustit simulaci. Zde bude vybrána možnost Java Aplli-cation.

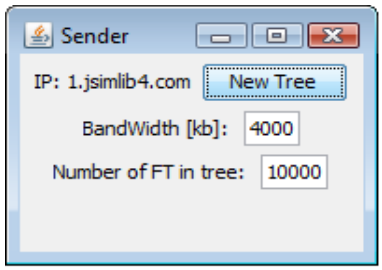
Po spuštění simulace za pomoci třídy MainSimulation v balíčku simul se zobrazí okno Main Window. V tomto okně, které je zobrazeno na obr. 6.7 je možné nastavit počet sumarizačních serveru a počet přijímačů.



Obr. 6.7: Okno Main Window

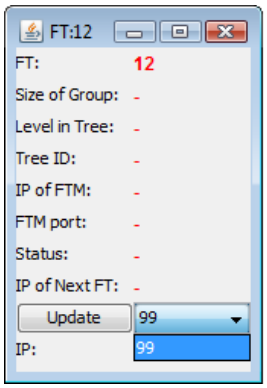
Před vlastním spuštěním simulace se nejdříve musí vytvořit všechny části zpět-novazebního stromu. Toho se docílí stisknutím tlačítka s nápisem „Create new Mem-bers“. Po jeho stisku a stisku tlačítka s nápisem „Start Simulation“ se vytvoří za-daný počet sumarizačních serverů(standardní počet je 11), přijímačů, okno správce sumarizačních serverů(FTM) a okno vysílače(Sender).

Následně se musí vytvořit zpětnovazební strom, ve kterém se budou přenášet hlasovací pakety. To se provede v okně vysílače(obr. 6.8), kde se po stisku tlačítka „New Tree“ vytvoří nový zpětnovazební strom.



Obr. 6.8: Okno Sender

Po vytvoření nového zpětnovazebního stromu je nutné nastavit všechny sumarizační servery. To se provede tak, že se nejdříve v okně, které je zobrazeno na obr. 6.9, nalezne identifikátor zpětnovazebního stromu v roletce, která je umístěna vedle tlačítka s nápisem „Update“. Po jeho nalezení a stisknutí tlačítka „Update“ se automaticky provede nastavení všech parametrů sumarizačního serveru. Stejná akce se bude opakovat ve všech vytvořených sumarizačních serverech.



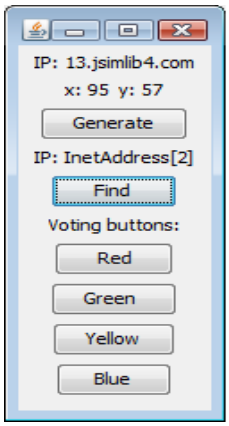
Obr. 6.9: Okno FT

Teď už jen zbývá pomocí tlačítek s nápisy „Generate“ a „Find“ vygenerovat pozici a najít nejbližší sumarizační server u všech vytvořených přijímačích.

6.6.2 Vlastní simulace hlasovací služby

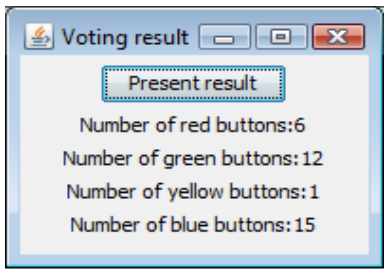
Po úspěšném nastavení už zbývá pouze ověřit funkčnost hlasovací služby. To se provede tak, že se v okně přijímače, které je zobrazeno na obr. 6.10 zmáčkne některé z hlasovacích tlačítek, jejichž názvy odpovídají barvám tlačítek na ovladačích.

Po zvolení různých možností na vytvořených přijímačích už nezbývá nic jiného, než zobrazit výsledky hlasování. To se provede tak, že se v okně správce sumarizačních serverů(FTM) stiskne tlačítko s nápisem „Results“ a po jeho stisku se objeví



Obr. 6.10: Přijímač

okno Voting Result(obr. 6.11). V tomto okně se po stisknutí tlačítka s názvem „Present Result“ zobrazí výsledky hlasování.



Obr. 6.11: Voting Result

7 ZÁVĚR

Cílem této práce byla realizace hlasovací služby pro internetovou televizi pomocí programovacího jazyka Java. Pro realizaci byla využita testovací síť, která je určena pro simulaci vlastností Tree Transmition Protokolu. Součástí zadání bylo realizovat hlasovací paket, vhodně upravit grafické rozhraní přijímačů, realizovat zasílání hlasovacích paketů a zobrazit výsledky hlasování.

V úvodu práce je provedeno srovnání klasického televizního vysílání a internetové televize. Jsou zde také představeny protokoly, které internetová využívá ke své komunikaci a pakety, které k tomu využívá.

Dále tato práce obsahuje návrh komunikace hlasovací služby a paketů určených pro tuto službu.

Pro realizaci hlasovacího paketu byl pod strukturou testovací sítě vytvořen balíček, v kterém je zvlášť realizována hlavička, jejíž vlastnosti přebírá hlasovací paket a sumarizační paket, který byl vytvořen nad rámec zadání. Nad oběma pakety byly provedeny jednotkové testy, aby se ověřila správnost jejich realizace.

Dále byla provedena úprava grafického rozhraní přijímačů, která spočívala v přidání čtyř tlačítek do již realizovaného grafického rozhraní.

Pro akci odesílání paketu po zmáčknutí jednoho ze čtyř tlačítek byla realizována metoda, která odesílá hlasovací pakety na předem zvolený sumarizační server. Na tomto serveru se provádí sumarizace a sumarizační paket se z tohoto serveru posílá správci serverů, který zpracovává výsledky.

Po zpracování výsledků je po stisku tlačítka v grafickém rozhraní správce serverů provedena jejich prezentace.

LITERATURA

[1] Müller J.; Komosný D.; Burget R.; Morávek P.; *Advantage of Hierarchical Aggregation.* International Journal of Computer Science and Network Security, 2008, roc. 2008, c. 8, s. 1-7. ISSN:1738-7906.

[2] Komosný D.; Novotný V.; *Analysis of bandwidth redistribution algorithm for single source multicast*, In Proceedings of the Sixth International Network Conference. Sixth International Network Conference. United Kingdom: University of Plymouth, 2006, s. 45 - 52, ISBN 1-84102-157-1

[3] Burget R.; Komosný D.; Müller J.; *Best Effort Hierarchical Aggregation Tree for IPTV Signaling.* International Journal of Computer Science and Network Security, 2008, roc. 2008, c. 8, s. 1-5. ISSN: 1738-7906.

[4] Peterka J.; *Jak funguje IPTV?* [online]. c2006,[cit. 2010-27-05]. Dostupné z URL: <http://www.lupa.cz/clanky/jak-funguje-iptv/>.

[5] Novotný V.; Komosný D.; *Large-Scale RTCP Feedback Optimization.* Journal of networks, 2008, roc. 2008, c.3, s. 1-10. ISSN 1796-2056

[6] Müller J.; Komosný D.; Burget R.; *Optimizing Feedback Path in Hierarchical Aggregation.* Electronics, 2008, roc. 2008, c. 2, s 1-6. ISSN: 1313-1842.

[7] Burget R.; Komosný D.; *Real-time control protocol and its improvements for Internet Protocol Television.* International Transaction on Computer Science and Engineering, 2006, roc. 2006, c. 31, s. 1-12. ISSN 1738-6438.

[8] Majer M.; *Seriál Síťování v Javě* [online]. c1998, [cit. 2010-27-05]. Dostupné z URL: <http://www.root.cz/serialy/sitovani-v-jave/>.

[9] Grygarek P.; *Síťové API Javy* [online]. c2000, [cit. 2010-27-05]. Dostupné z URL: <http://www.cs.vsb.cz/grygarek/dosys/inprise/net/net.html>.

[10] Komosný D.;Novotný V.; *Tree Structure for Source-Specific Multicast with feedback Aggregation*, in ICN07 - The Sixth International Conference on Networking. Martinique, 2007, ISBN 0-7695-2805-8.

[11] Komosný D.; Müller J.; Burget R.; *Změny ve světě IPTV.* elektrorevue, 2008, roc. 2008, c. 33, s. 1-11. ISSN 1213-1539.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ASM Any Source Multicast

CSRC Contributing Source

FT Feedback Target

FTD Feedback Target Definition

FTI Feedback Target Information

FTM Feedback Target Manager

FTS Feedback Target Specification

IDE Integrated Development Environment

IP Internet Protocol

IPTV Internet Protocol TeleVision

LM LandMark

MPEG Moving Picture Experts Group

SSM Network Type Protocol

RFT Root Feedback Target

RR Receiver Report

RSI Receiver Summarization Information

RTCP Real Time Control Protocol

RTP Real Time Protocol

RTT Round Trip delay Time

SDES Source DEScription

SR Sender Report

SSM Source Specific Multicast

SSM Synchronization Source

TCP/IP Transmission Control Protocol / Internet Protocol

TTP Tree Transmition Protocol

UDP User Datagram Protocol

VoD Video on Demand

SEZNAM PŘÍLOH

A PŘÍLOHA

A.1 Obsah přiloženého DVD

Adresáře:

- Voting service - obsahuje zdrojové kódy projektu TTP a hlasovací služby
- Eclipse IDE - obsahuje předkonfigurované vývojové prostředí

Soubory

- Bakalářská práce - dokument s bakalářskou prací ve formátu pdf