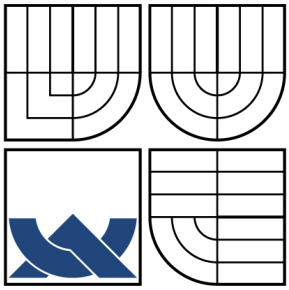
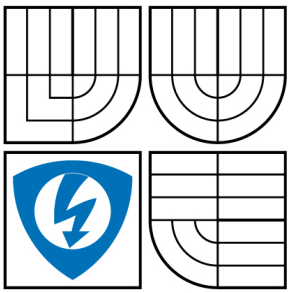


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

SBĚR DAT Z VELKÉHO POČTU POČÍTAČŮ POMOCÍ HIERARCHICKÉ SUMARIZACE

DATA ACQUISITION FROM A GREAT NUMBER OF COMPUTERS USING HIERARCHICAL SUMMARIZATION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. MOJMÍR JELÍNEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. DAN KOMOSNÝ, Ph.D.

SUPERVISOR

BRNO 2008

LICENČNÍ SMLOUVA

POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Bc. Mojmír Jelínek
Bytem: Hrubčice 251, 79821, Hrubčice
Narozen/a (datum a místo): 9.5.1981, Prostějov

(dále jen "autor")

a

2. Vysoké učení technické v Brně

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

(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

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

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

(dále jen VŠKP nebo dílo)

Název VŠKP: Sběr dat z velkého počtu počítačů pomocí hierarchické sumarizace
Vedoucí/školitel VŠKP: Ing. Dan Komosný, Ph.D.
Ústav: Ústav telekomunikací
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

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

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

4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

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

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

Článek 3
Závěrečná ustanovení

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

V Brně dne: .....

....

Nabyvatel Autor

Anotace

Tato práce pojednává o IPTV (Internet Protocol Television) systému televizního vysílání, o možnostech vysílání multimediálního obsahu skrze počítačové sítě a možnosti přenosu signalizace. Rovněž popisuje všechny nezbytné prvky, metody a procesy používané IPTV technologií.

První část se zaměřuje na celkový pohled na IPTV a popis používané terminologie, dále pak na základní informace o klasickém televizním vysílání, znázorňuje a vysvětluje rozdíly oproti IPTV a rovněž popisuje výhody a nevýhody tohoto typu vysílání. Tato část rovněž ukazuje, proč má tato technologie zářivou budoucnost.

Následující části popisují vývoj IPTV od jeho návrhu až po realizaci a je zde znázorněno skutečné začlenění multimediálního vysílání do ADSL (Asymmetric Digital Subscriber Line) prostředí a popsány všechny důležité prvky sítě jako je Head-End, ADSL, DSLAM (Digital Subscriber Line Access Multiplexer) a možnosti přenosu dat. Dále pak stručný popis používaných technik komprese přenášených dat (MPEG-2 a MPEG-4) a možnosti vlastního přenosu jako je broadcast, unicast a multicast. Jistě zajímavým údajem je nezbytná šířka pásma a požadavky na kvalitu přenosu.

Nedílnou součástí je vlastní realizace aplikací v programovacím jazyce C++ pro vysílání a příjem paketů pomocí protokolu UDP (User Datagram Protocol) . Tyto aplikace mají za úkol vytížit hlavní server, kde se provádí měření ztrátovosti paketů a vytížení procesoru. Výstupem je tabulka naměřených hodnot pro dané velikosti paketů a pro dané intervaly mezi jejich vysíláním. Smyslem tohoto měření je nalézt hodnotu maximálního množství počítačových stanic, které je sumarizační server schopen zpracovat.

Posední částí je realizace dvojice aplikací v jazyce JAVA, které mají za úkol zjišťovat informace o koncových uzlech sítě. Oba algoritmy jsou provedeny dvojicí vláken ke zvýšení dynamičnosti a rychlosti získávání daných informací. Klientská strana na jednom z vláken provozuje náhodné generátory, které budou později nahrazeny reálnými algoritmy pro získávání skutečných hodnot.

Klíčová slova

Sumarizace, Hierarchie, IPTV, Monitorování, Tree Transmission Protocol

Abstract

This paper deals with IPTV (Internet Protocol Television) transmission of feedback and is showing options and ways of construction, problems and optimalization of signalization protocol.

In the beginning are described IPTV and technology terms that this technology uses. Here can be found information about classical TV (Television) transmitting and comparing with IPTV technology, the advantages and disadvantaged of IPTV and answers why this solution has future.

Next parts are about history of IPTV and real use over ADSL (Asymmetric Digital Subscriber Line) in present. Here are explained all the necessary units like Head-End, feedback target, root feedback target, ADSL, DSLAM (Digital Subscriber Line Access Multiplexer) and methods of data stream transmission. Also here are described the techniques of video stream compression (MPEG-2 and MPEG-4) and all options of data transmission as broadcast, unicast and multicast. Important part is about transmission speed and needs.

The realization also contains applications, written in the C++ language, for transmitting and receiving packets by UDP (User Datagram Protocol) protocol. The task of these applications is to load the main server, where the measurement of packet loss and CPU (Central Processing Unit) load takes place. The result is a table of measured values for specified packet sizes and for specified time intervals between them. The meaning of this measuring is to find the maximal number of computer nodes, which the feedback target is able to proceed.

Last part is about realization of 2 applications in JAVA language, which get the information about end-nodes. Both algorithms are using 2 threads to increase speed of getting the information. The client has few random generators within one thread, which will be later replaced by special algorithms for getting real values.

Keywords

Summarization, Hierarchical, IPTV, Monitoring, Tree Transmission Protocol

Prohlášení

Prohlašuji, že svou diplomovou práci na téma "Sběr dat z velkého počtu počítačů pomocí hierarchické sumarizace" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

V Brně dne 16.05.2008 ……

Poděkování

Děkuji vedoucímu diplomové práce Ing. Danu Komosnému, Ph.D., za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne 16.05.2008 .....

Seznam použitých zkratek a symbolů

Zkratka	Anglický název	Český název nebo ekvivalent
ADSL	Asymmetric Digital Subscriber Line	
BER	Bit Error Rate	Poměr chybných bitů
CPU	Central Processing Unit	Centrální Procesorová Jednotka
DSL	Digital Subscriber Line	
DSLAM	Digital Subscriber Line Access Multiplexer	
EPG	Electronic Program Guide	Elektronický programový průvodce
FT	Feedback Target	Sumarizační server
HD	High Definition	Vysoké rozlišení
HTTP	Hypertext Transfer Protocol	
IPSTB	Internet Protocol Set Top Box	Přijímač Multimed. Internetového Signálu
IPTV	Internet Protocol Television	Televize přes internetový protocol
MPEG	Moving Picture Experts Group	
PES	Packetized Elementary Stream	
Pu		Celkový počet uzlů
RFT	Root Feedback Target	Hlavní server
RTCP	Real Time Control Protocol	
RTP	Real Time Protocol	
SLA	Service Level Agrrement	
STB	Set Top Box	Přijímač multimediálního signálu
TCP	Transport Control Protocol	
TDBP	Time Diference Between packets	Časový rozdíl mezi pakety
TTP	Tree Transmission Protocol	Stromový Přenosový Protokol
TV	Television	Televize
UDP	User Datagram Protocol	
Ur		Reálný počet aktivních uzlů
VoD	Video on Demand	Video na Požádání
Zp		Požadované zpoždění mezi pakety
Zz		Změřené zpoždění mezi pakety

Obsah

SEZNAM POUŽITÝCH ZKRATEK A SYMBOLŮ 5

ÚVOD 11

1. SYSTÉM IPTV 13

2. MOŽNOSTI A SLUŽBY IPTV 15

3. HISTORIE A VÝVOJ IPTV 16

4. KODEKY POUŽÍVANÉ PRO PŘENOS MULTIMEDIÁLNÍHO OBSAHU 17

4.1. MPEG-2 17
4.2. MPEG-4 17
4.3. PRINCIP MPEG PŘENOSU 18

5. POŽADAVEK NA KVALITU SPOJENÍ A ŠÍŘKU PÁSMA 19

6. MULTICAST A UNICAST 20

7. HIERARCHICKÁ SUMARIZACE 22

8. ŘEŠENÍ PŘENOSU SIGNALIZACE 24

8.1. SIGNALIZACE S VYUŽITÍM TTP PROTOKOLU 24
8.2. MONITOROVÁNÍ POSKYTOVATELEM SLUŽEB 26

9. TESTOVÁNÍ VYTÍŽENÍ A ZTRÁTOVOSTI PAKETŮ HLAVNÍHO SERVERU 27

9.1. TEORETICKÁ PŘÍPRAVA REALIZACE 27
9.1.1. Planetlab 27
9.1.2. Popis protokolu RTCP 28
9.2. VYTVOŘENÝ SOFTWARE PRO ÚČELY MĚŘENÍ 29
9.2.1. Popis skriptů napsaných v jazyce Bash 29
9.3. POPIS PROGRAMŮ NAPSANÝCH V JAZYCE C++ 32
9.4. VÝSLEDKY MĚŘENÍ 34
9.4.1. Testování vytížení procesoru a ztrátovosti paketů 34
9.4.2. Výsledky naměřených hodnot 35
9.4.3. Příklad výpočtu 36
9.4.4. Grafické zpracování výsledků měření 37

10. MONITOROVACÍ SERVER 40

10.1. POPIS KONSTRUKCE PROGRAMU SERVER 40
10.2. POPIS KONSTRUKCE PROGRAMU KLIENT 42
10.3. SHRNUTÍ FUNKCE MONITOROVACÍHO SERVERU 42

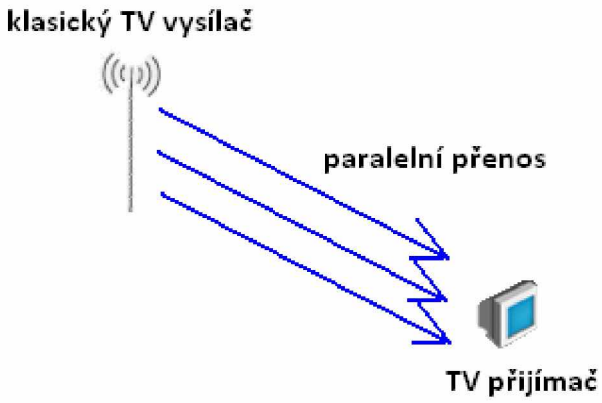
11. ZÁVĚR 43

LITERATURA 44

PŘÍLOHY

Úvod

IPTV je ve své podstatě přenos klasického TV signálu skrze počítačovou síť, pomocí technologie IP protokolu. Klasický analogový signál musí být digitalizován a s použitím speciálních technik je tento odeslán k přijímači. Základním rozdílem mezi klasickým a IPTV vysíláním je množství možných kanálů, které koncoví uživatelé mohou sledovat ve stejný čas. Ovšem IPTV nabízí plnou uživatelskou interaktivitu [13], [10]. Obrázek 1 ukazuje typické všesměrové vysílání (broadcast) klasického TV signálu. Je zřejmé, že touto metodou jsou doručeny všechny vysílané kanály všem potenciálním příjemcům ve stejný čas, který si pomocí svého přijímače vybere ten, který chce sledovat.



Obrázek 1: Klasické TV vysílání

Tato metoda je čistě jednosměrná, takže vysílač nemá informace o počtu aktivních přijímačů, právě sledovaných kanálech a čase jak dlouho je daný kanál sledován. Z tohoto důvodu provádí jednotlivý poskytovatelé náročné průzkumy, které na základě velkého počtu respondentů poskytují nepříliš přesné informace. Ovšem výhodou tohoto typu vysílání je obrovský počet uživatelů, kteří mohou sledovat všechny vysílané kanály.

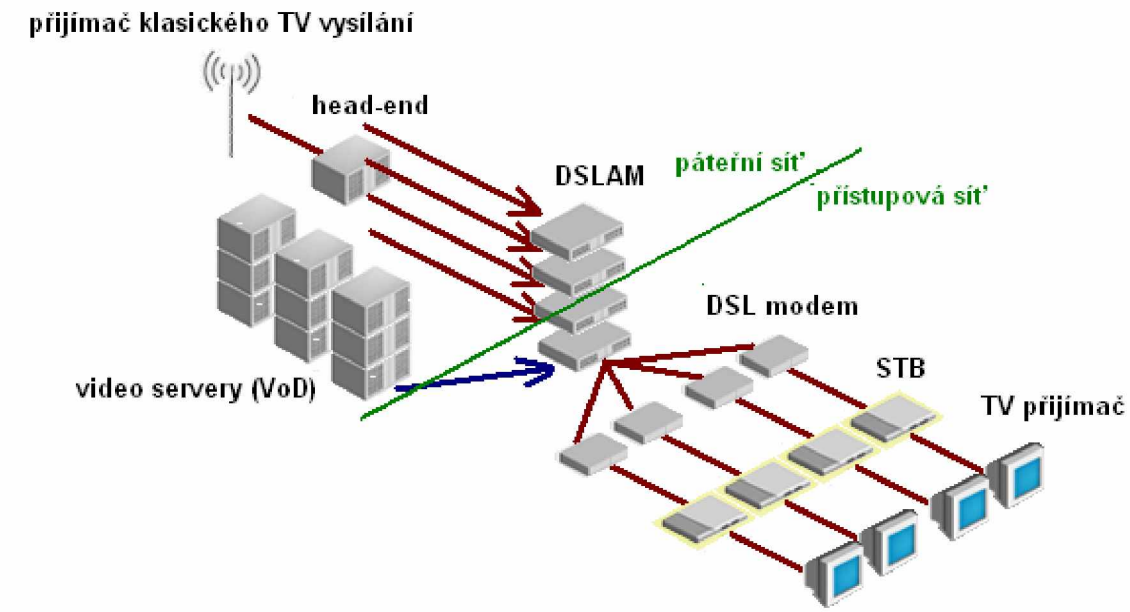
Již zmiňovanou nevýhodou je skutečnost, že každý koncový uživatel dostane stejný multimediální obsah. Tento obsah nemůže být selektivně měněn, např. podle věku koncových uživatelů. V současné době je nedostatek interaktivity nahrazován např. Internetovým připojením, kde si uživatel může koupit kanály (programy), které jsou normálně uzamčené, avšak jsou vysílány všem [13].

Právě IPTV řeší všechny problémy se zpětnou vazbou koncového uživatele, s ohledem na obousměrný způsob komunikace [1],[5],[7]. Poskytovatel rovněž může identifikovat přijímač a sledovat jeho aktivitu jako diváka. Tato zjištění slouží hlavně k lepšímu cílení reklamního vysílání a to nejen podle obsahu, ale i podle času. Např. v čase kolem 17 hodiny bude vysílána reklama zaměřená na diváky nižšího věku, kolem 20 hodiny již na diváky středního věku.

Dalším rozdílem, který je ovšem velkou nevýhodou je schopnost odeslat jen jeden nebo jen málo kanálů k přijímači současně. To je způsobeno malou přenosovou kapacitou Internetového nebo síťového prostředí. Z toho plyne, pokud uživatel má více jak jeden IPTV přijímač, což je běžné, šířka pásma jeho připojení musí vzrůst násobně s množstvím zvolených kanálů. Tato skutečnost je částečně řešena výběrem televizních kanálů v počítačové síti mezi přístupovou a páteřní sítí, protože páteřní síť má dostatečnou kapacitu na přenos velkého množství multimediálních kanálů. Přístupová vrstva pouze přenáší kanály, které si koncový uživatel zvolil. Je zřejmé, že s rostoucím množstvím koncovým uživatelů bude i tato kapacita nedostatečná, z tohoto důvodu rozvoj IPTV závisí na technologickém růstu ADSL, ADSL2+ nebo optickým technologií za účelem zvýšení šířky pásma přístupové sítě [19].

Další nevýhodou je zpoždění mezi přepínáním kanálů. To je způsobeno faktem, že vysílač posílá jen ty kanály, o které je zrovna v této části sítě zájem. V případě, že uživatel změní na jiný, vysílač musí být informován o tomto počinu a začít vysílat další kanál. Na toto zpoždění má vliv i velikost vyrovnávací paměti přijímače uživatele, jejíž velikost záleží na stabilitě šířky poskytovaného pásma a kvalitě sítě. Čím větší je tato paměť, tím déle trvá její naplnění multimediálními daty. Z měření přímo v reálných situacích vyplývá, že tento čas je mezi 1 až 5 vteřinami [10],[19].

1. Systém IPTV



Obrázek 2: Topologie IPTV přenosu pomocí ADSL

Obrázek 2 výše ukazuje topologii IPTV v přístupovém prostředí skrze ADSL. Všechny prvky jsou popsány níže.

Přijímač klasického TV vysílání

Jde o běžný analogový nebo digitální televizní, satelitní či rádiový přijímač.

Head-end

Head-end kóduje signál na digitální video formu MPEG-2 nebo MPEG-4 na jehož výstupu je datový tok typu multicast, kde každý kanál je representován svou unikátní adresou. Tato jednotka omezuje celkovou šířku pásma (všech kanálů) tak, aby ochránila páteřní síť před přetížením. Výstupní část této jednotky přenáší pouze ty kanály, o které mají uživatelé zájem nebo ty, které poskytovatel právě nabízí [18].

Video servery (VoD)

Jde o sestavu počítačových uzlů které representují datové úložiště velkého počtu multimediálních kanálů jako jsou filmy a hudba. Konečný uživatel IPTV si může vybrat v kterémkoliv čase co chce sledovat a jeho volba bude representována přímo do jeho přijímače. Z pohledu uživatele je ovládání stejné jako práce s DVD přehrávačem, je možné film zastavit, přetočit či přehrát znovu [15].

DSLAM a ADSL modem

Tyto jednotky přístupové sítě a jejich místo v topologii záleží na technologii poskytovatele. Tento model zahrnuje způsob připojení pomocí ADSL, ale je možné i připojení pomocí jiných technologií (Ethernet, Optická vklákna) [18].

IPSTB

Internet Protocol Set Top Box je zařízení schopné zpracovat kanál typu multicast, ve formátu MPEG-2 (MPEG-4) a representovat jej do klasického TV signálu. Celé řešení je založeno na vlastních protokolech výrobce. Některé zařízení jsou schopné zpracovat jak klasický TV signál tak IPTV a reprezentovat je společně na obrazovku [19].

TV přijímač

Klasická televize, která je schopna zobrazovat signál přenášený z IPSTB.

Rozdělování kanálů je prováděno na posledním možném místě v topologii, běžně mezi přístupovou a páteřní vrstvou viz Obrázek 2, rozdělení napříč DSLAM zařízeními.

2. Možnosti a služby IPTV

IPTV nabízí pokročilé možnosti a služby jakou jsou televize kdekoliv, celosvětové televizní kanály, osobní kanály, cílená reklama, elektronický programový průvodce, televize na požádání, video na požádání a zobrazení po zaplacení [13], [19].

Televize kdekoliv je možnost sledovat všechny programy na kterémkoliv místě na světě. Obvykle daný poskytovatel nabízí přenos televizních kanálů, typu multicast, skrze jeho vlastní podsíť, ale technologie IPTV ve spojení s vhodným řešením signalizace je schopná nabídnout širokopásmový přenos i na jiné místo v Internetu. Signalizace může být využita k sestavení nejlepší přenosové cesty mezi poskytovatelem a konečným příjemcem.

Cílená reklama je nejvíce zajímavá vlastnost z pohledu poskytovatele. Používá se k doručení mediálního obsahu na specifický přijímač podle adresy koncového uživatele. Používání cílené reklamy umožňuje rychlé a přímé měření efektivnosti reklamních kampaní.

Elektronický programový průvodce (známý jako EPG), je seznam programů, které jsou nebo budou vysílány. V duchu uživatelské interaktivity, uživatel je schopen vyhledat nejlepší programy, nebo si vytvořit seznam jeho oblíbených programů.

Video na požádání (VoD) je služba kde si koncový uživatel může zvolit film či skladbu podle databáze poskytovatele, který bude přehrán jakmile uživatel bude chtít. Film může být zastaven, přetočen, zrychlen… Tato technologie používá k přenosu multimediálních dat technologii typu unicast.

Zobrazení po zaplacení (PPV) je systém pomocí kterého si diváci mohou koupit pořady, které chtějí mít zobrazeny na jejich přijímači. Platí tak za soukromý multimediální příjem. Shodný obsah je zobrazován všem účastníkům, kteří si pořad zaplatili. Ke koupi může sloužit zmíněný seznam programů (EPG), automatický telefonní systém, přímé zavolání do telefonního centra poskytovatele nebo použití sítě Internet. S ohledem na fungující Španělskou IPTV síť, většina objednavatelů pořadů sleduje živé vysílání jejich oblíbeného fotbalového mužstva.

3. Historie a vývoj IPTV

1994 - ABC's World News Now byla prvním televizním pořadem přenášeným přes Internet.

1995 – Termín IPTV se poprvé objevil při založení Percept Software Jidithem Estrinem a Billem Carrioem. Společnost navrhla a vytvořila aplikaci pojmenovanou IP/TV, která byla první aplikací kompatibilní jak s Windows tak s UNIX systémy, schopná přepravit multimediální datový tok, o nízké až vysoké kvalitě, pomocí přenosu typu unicast či multicast.

1998 – Společnost Cisco získala Percept Software a s tím i obchodní značku IP/TV. V témže roce společnost AudioNet spustila první nepřerušovaný živý přenos, následována společností KCTU-LP.

1999 – Kingston Communications, regionální telekomunikační operátor v Anglii, spustil KIT (Kingston Interactive Television), což byla IPTV využívající DSL širokopásmové připojení.

2001 – Tentýž operátor doplnil do své nabídky VoD služby. Kingston byla jedna z prvních společností na světě, která předvedla IPTV a VoD vysílání skrze ADSL. Dříve byla tato možnost přenosu značně omezena, kvůli malé šířce pásma této technologie.

2005 – IPTV zaznamenala skutečně velký rozmach v domácnostech, rychlost tohoto postupu je kromě marketingu také dána rychlostí rozvoje širokopásmových připojení koncových uživatelů. K tomuto roku se odhaduje 200 milionů diváků využívajících IPTV technologii.

2010 – Očekává se navýšení počtu domácích diváků IPTV na 400 milionů.

Je důležité zdůraznit, že historicky zde bylo velmi mnoho variací definicí IPTV, např. jednoduché multimediální toky skrze IP sítě [2],[6],[10],[14].

4. Kodeky používané pro přenos multimediálního obsahu

Video obsah je typicky komprimován pomocí MPEG-2 nebo MPEG-4 kodekem a po té je jako MPEG datový tok odeslán ke svému příjemci pomocí technologie typu unicast nebo multicast. Nejnovější vydaný kodek H.264 (MPEG-4) je navržen jako náhrada za starý MPEG-2 [2] až [6].

4.1. *MPEG-2*

MPEG-2 kodek podporuje široké spektrum aplikací od mobilních až po vysoce kvalitní HD (High Definition) videa. Pro mnoho aplikací je nereálné či příliš nákladné podporovat celý standard, proto MPEG-2 definuje profily a úrovně, které dovolují aplikacím používat pouze některé jeho části. Profily definují seznam schopností jako je kompresní algoritmus apod.. Úroveň definuje seznam kvantitativních možností jako je maximální bitový tok, maximální velikost rámce, atd.

MPEG-2 úrovně			
Název	Počet snímků za vteřinu (Hz)	Max. rozlišení	Max. bit. tok (Mb/s)
Nízký	23.9, 24, 25, 29.9, 30	352x288	4
Hlavní	23.9, 24, 25, 29.9, 30	720x576	15
Vysoký 1	23.9, 24, 25, 29.9, 30, 50, 59.94, 60	1440x1152	60
Vysoký 2	23.9, 24, 25, 29.9, 30, 50, 59.94, 60	1920x1152	80

Tabulka 1: Omezení MPEG-2, spadající pod každou úroveň

4.2. *MPEG-4*

MPEG-4 je soubor metod definujících kompresi zvukového nebo obrazového signálu. Byl poprvé představen v roce 1998, návrh zahrnoval i standardizaci skupin jak zvukového tak obrazového formátu a související technologie.

Většina vlastností obsažená v MPEG-4 je ponechána jednotlivým vývojářům, tak aby se mohli kdykoliv rozhodnout, které z nich použít. Tzn., že v praxi pravděpodobně neexistuje

kompletní implementace celého standardu MPEG-4. Tento standard, stejně jako MPEG-2, zahrnuje profily a úrovně.

4.3. *Princip MPEG přenosu*

Obrazový server generuje PES (Packetized Elementary Stream) paket, který obsahuje max. 65626 B (hlavička má 9 B). Tento PES paket je rozdělen na MPEG datové toky, které jsou tvořeny 188 B dlouhými pakety (min. 4 B hlavička). Datový tok je přenesen skrze IP síť ke koncovému uživateli.

Rámec typu Ethernet, který má maximálně 1514 B může obsahovat 7 MPEG paketů, dlouhých 188 B, datová část má tedy 7 x 188 = 1316 B.

5. Požadavek na kvalitu spojení a šířku pásma

Následující tabulky (Tabulka 2, Tabulka 3) ukazují požadavky na realizaci IPTV přenosu pomocí technologie typu multicast nebo unicast [19].

Zpoždění	< 200ms
Rozptyl zpoždění	< 20ms
Doba přepínání kanálů	< 1,3s
Připraven k přenosu	minuty

Tabulka 2: Požadavky na kvalitu spojení

Parametry v tabulce (Tabulka 2) mohou mít vliv na zpoždění mezi přepínáním jednotlivých kanálů a velikost vyrovnávací paměti, ale nemůže ovlivnit jistotu doručení multimediálního datového toku, která závisí na stabilitě sítě a rychlosti připojení koncového uživatele (Tabulka 3).

Kvalita	pro MPEG-2
Velmi nízká	56 kb/s
Normální	500 kb/s
TV kvalita	3800 kb/s
Vysoká kvalita	5000 – 15000 kb/s

Tabulka 3: Přenosové rychlosti MPEG-2

MPEG-4 kodek má 2x silnější kompresní poměr. Může poskytnout stejnou kvalitu jako MPEG-2 ovšem s přibližně polovičním datovým tokem.

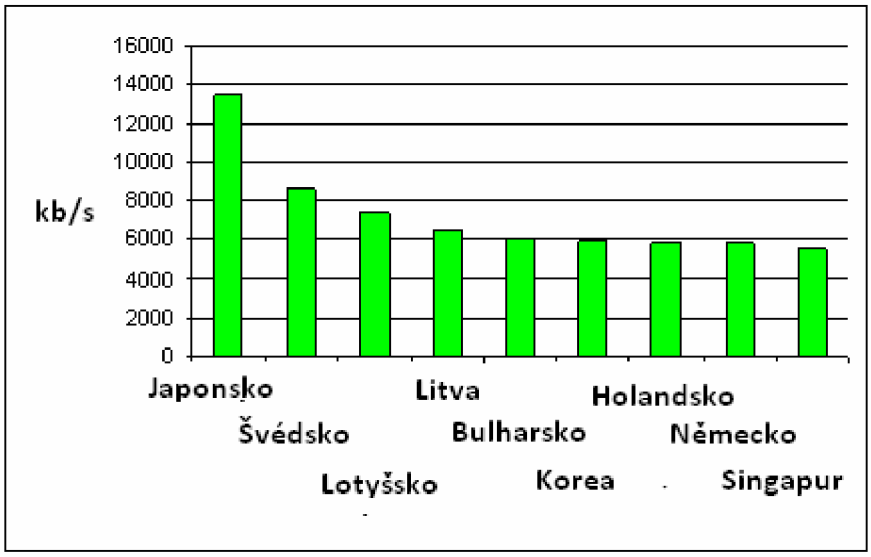
6. Multicast a unicast

Česky tyto termíny můžeme popsat jako paralelní a sériové vysílání. Multicast je přenos jednoho signálu k většímu množství příjemců současně. Každá odeslaná informace obsahuje speciální adresu, pomocí které může velké množství potenciálních přijímačů v síti přijmout stejný signál [12] až [17].

Použití přenosu typu multicast je efektivní, protože ta sama informace je doručena velkému počtu příjemců ve stejný čas. Implementace systému tohoto typu je technologicky složitější než typu unicast, např. je třeba řízení při přidávání a odstraňování členů ze skupin.

Tato metoda přenosu musí být implementována do celé sítě mezi vysílač a přijímač. Celé řešení vyžaduje speciální zařízení jako jsou např. směrovače typu multicast a pro koncové uživatele IPTV širokopásmové připojení.

Tabulka 3 ukazuje, že sledování každého kanálu vyžaduje přibližně 5 Mb/s, pokud tedy jeden uživatel má 2 IPTV přijímače bude potřebovat připojení k poskytovateli o rychlosti 10 Mb/s [18]. S ohledem na nezávislý test rychlostí [16], pouze průměrný japonský divák může přijímat vysoce kvalitní IPTV programy, viz Obrázek 3.



Obrázek 3: Průměrné hodnoty připojení koncových uživatelů

Přenos typu unicast znamená doručení informací pouze k jednomu příjemci v síti. Použití tohoto způsobu přenosu je neefektivní v případě, že mnoho uživatelů přijímá stejný datový tok ve stejný čas, protože každý z nich musí mít vyhrazenu minimální úroveň šířky pásma.

Z toho vyplívá, že s rostoucím počtem koncových uživatelů násobně poroste požadavek na celkovou šířku pásma. Tato skutečnost vede k používání přenosu typu unicast v prostředí IPTV pouze k nenáročné signalizaci, nebo pro speciálně zpoplatněné služby jako např. zmíněné VoD.

7. Hierarchická sumarizace

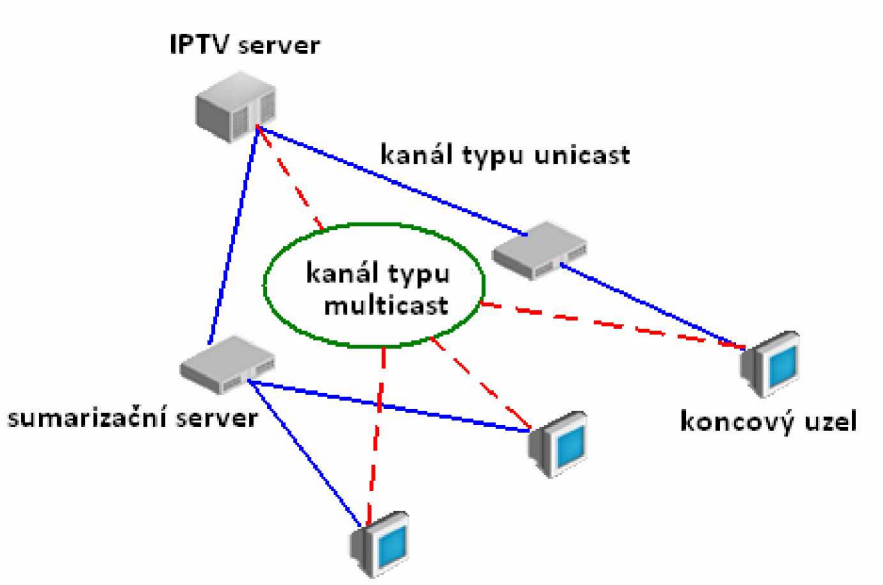
Z důvodu neustále rozvíjejícího se internetu, dochází k vytváření velké nepřehlednosti a zvyšování časové náročnosti při hledání konkrétního dotazu. Z tohoto důvodu jsou v poslední době kladeny požadavky na rychlejší zpracování a sumarizaci obsahu. Byla vytvořena řada technik, z nichž 2 nejdůležitější se nazývají shlukování a sumarizace.

Shlukování je metodou umísťování rozdílných objektů do určitého množství skupin, což se děje na základě podobnosti s obsahem informací v dané skupině. Algoritmus musí být správně navržen, tak aby nalezl a správně umístil daný objekt, což není možné vždy realizovat, proto řešení shlukováním není příliš vhodné.

Sumarizace je prezentace jen důležitých částí daného obsahu ve zjednodušené podobě. Proces zahrnuje 2 podstatné části, vyjmutí a oddělení. Každé vyjmuté části je pomocí váhy udělen koeficient, pomocí něhož i dalších je rozhodnuto do které oblasti daný objekt patří (oddělení).

Výše zmíněný postup je obecným vyjádřením řešení problému sumarizace. V tomto projektu se ovšem pracuje výhradně s čísly, tzn. je nevhodné pomocí vah či skupin stanovovat význam jednotlivých hodnot, je tedy prováděn průměr všech relevantních hodnot systému.

V případě velkého množství uzlů (objektů) by sumarizace jako taková ztrácela na významu, protože by ve výsledku vznikla obrovská řada průměrných hodnot. Proto je třeba vhodným algoritmem vytvořit celý systém jak tyto hodnoty dále hierarchicky sumarizovat. V ideálním případě bude na konci jen jedna informace obsahující sumu všech ostatních beze ztrát.



Obrázek 4: Zjednodušený model sumarizačních IPTV členů

Obrázek 4 ukazuje sumarizaci pomocí signalizačního protokolu Tree Transmission Protocol (TTP). Informace je sumarizována na prvním sumarizačním serveru (Feedback Target, FT) pomocí vhodné hierarchické metody. FT posílá sumarizované informace na jinou vrstvu FT nebo přímo na manažer přenosu signalizace. Server pomocí TTP paketu dynamicky mění celou topologii uspořádání FT na základě získaných informací a musí být konfigurován tak, aby uspořádání bylo úměrné požadavkům sítě nebo poskytovatele.

Tato stromová signalizační technika může být použita pro mnoho aplikací, které pro svoji činnost používají velké množství koncových uzlů, které komunikují pouze s jedním hlavním serverem. Mezilehlé sumarizační servery tedy ve své podstatě fungují jako ochrana hlavního serveru před přetížením.

8. Řešení přenosu signalizace

V současné době některé společnosti nabízí pouze částečné řešení. Informace přenesená ke koncovému uživateli uchovává zařízení DSLAM a v předem definovaných časových intervalech je odesílá na hlavní server ke zpracování. Tento proces je ale značně neefektivní, protože čas odesílání informací od koncového uživatele k hlavnímu serveru je v řádu desítek minut.

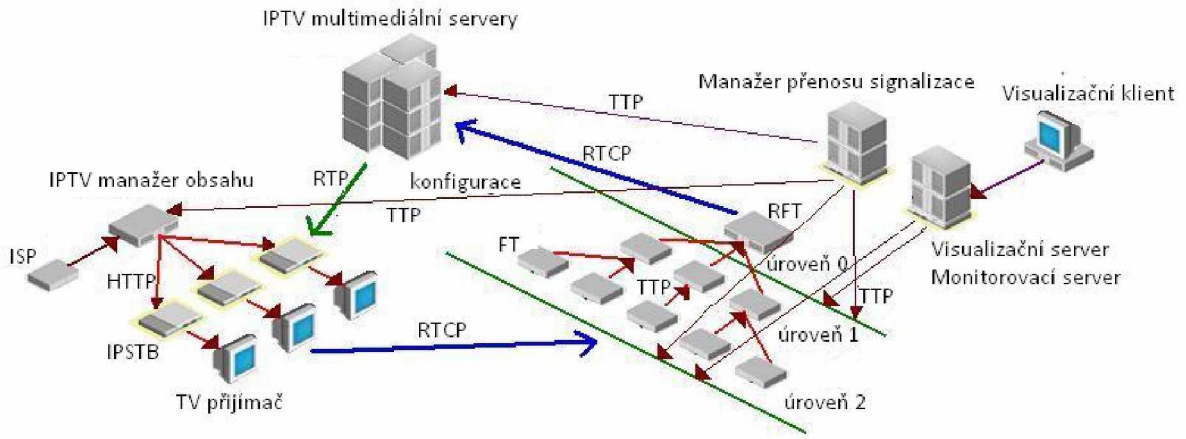
Další možností je použití adresy typu unicast k přenosu signalizace na definovanou adresu hlavního serveru v reálném čase. Toto řešení je vhodné pouze pro malé sítě, protože velké množství komunikujících uzlů může přetížit hlavní server [5],[7]. Nejefektivnějším řešením, které se stále vyvíjí a tato práce o tom pojednává [5], je použití cesty typu unicast s hierarchickou agregací a s dynamickou změnou adres všech cílových sumarizačních serverů v závislosti na fyzické poloze v síti (internet). Část celého řešení této technologie zahrnuje i vizualizaci závislostí, parametrů a uzlů při přepravě IPTV signálu.

8.1. *Signalizace s využitím TTP protokolu*

V síti IPTV je signalizace, směrem k hlavnímu serveru (Root Feedback Target (RTF)) a zpět, přenášena pomocí FT (sumarizačních serverů), které zpracovávají data specifickou metodou. Signalizace je přenášena skrze dynamicky sestavovanou stromovou strukturu, až na hlavní server, ke zpracování. FT systém je rozdělen na několik úrovní. Úroveň 0 obsahuje jen jeden hlavni server. Další úrovně jsou začleněny do stromové topologie jak je možné vidět na následujícím obrázku (Obrázek 5).

Manažer přenosu signalizace, vizualizační server, monitorovací server a sumarizační servery jsou jednotkami, které používají TTP protokol k přenosu signalizačních dat. Tato struktura se nazývá řízená. K řízení stromové struktury, manažer přenosu signalizace potřebuje znát alespoň základní informace o každém koncovém uzlu, jako např. jeho fyzickou polohu v síti. K nalezení konkrétní pozice uzlu je využit systém založený na algoritmech jako je Vivaldi nebo Global Network Positioning. K předání těchto informací je sestrojen algoritmus, jenž je součástí této práce.

Obrázek 5 níže ukazuje všechny jednotky a důležité protokoly signalizačního systému používajícího TTP protokol a hierarchickou agregaci.



Obrázek 5: Struktura IPTV signalizačního systému

ISP

Internet Service Provider (Poskytovatel služeb) řídí celý systém pomocí IPTV manažeru obsahu. Poskytovatel může přistoupit do systému a řídit ho jen skrze tuto jednotku.

IPTV manažer obsahu

Je schopen řídit obsah IPTV přenosu (filmy, reklamy, rádiové stanice atd.)

Manažer přenosu signalizace

Tato jednotka používá TTP protokol k dynamické koordinaci vazeb mezi všemi FT a RFT jednotkou. Rovněž konfiguruje IPTV manažer obsahu.

IPTV multimediální servery

Výstupem této jednotky je datový tok typu multicast, který je přepraven do IPSTB jednotek pomocí RTP protokolu.

RTP a RTCP

Real Time Protocol je použit pro datový tok typu multicast. Real Time Control Protocol je použit, mimo jiné, k přenosu signalizace koncového uživatele.

TTP

Signalizační systém používá TTP protokol k vytvoření, změně a řízení celého systému.

FT a RFT

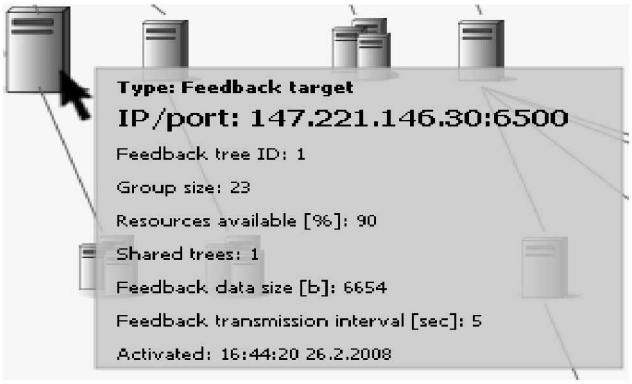
Sumarizační servery (FT) shromažďují, sumarizují a přenáší informace od koncových uživatelů k hlavnímu serveru (RFT).

Monitorovací server

Tento uzel komunikuje s každým uzlem systému (FT, koncový uživatel) a shromažďuje informace o typu jejich spojení, aktuální rychlosti, zpoždění apod..

Vizualizační server

Je schopen v reálném čase zobrazovat polohu každého koncového uzlu systému, každý FT a všechny relevantní informace o nich. Příklad grafického výstupu vizualizace ukazuje následující obrázek (Obrázek 6).



Obrázek 6: Výstup vizualizace

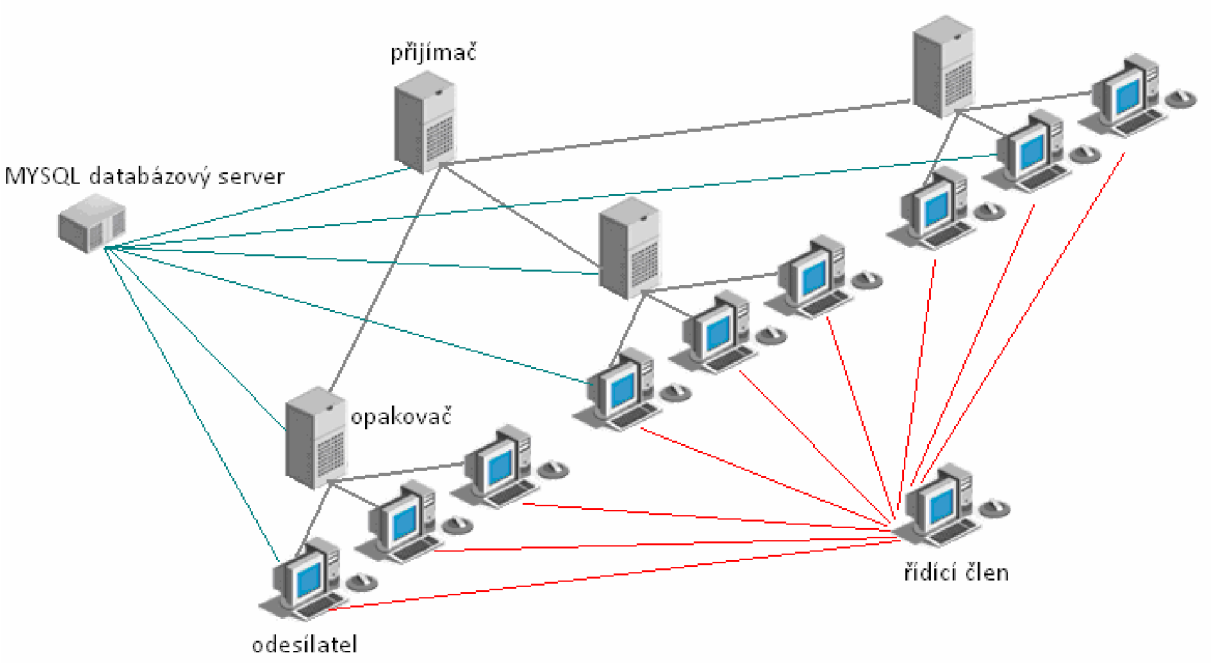
Oba zmíněné servery používají MYSQL databázi ke shromažďování všech informací.

8.2. *Monitorování poskytovatelem služeb*

V současnosti mají poskytovatelé jako součást smluv se svými zákazníky tzv. SLA (Service Level Agrrement) sloužící k porovnávání skutečné kvality poskytovaných služeb se smluvně dohodnutými. SLA je formálně vyjednaná dohoda mezi oběma stranami, obsahující specifikaci poskytovaných služeb, priorit, odpovědností a záruky. Například by měla specifikovat úrovně dostupnosti služby, výkon nebo další součásti jako je účtování či pokuty v případě porušení SLA smlouvy.

9. Testování vytížení a ztrátovosti paketů hlavního serveru

Tato část práce obnáší využití realizovaných programů a skriptů pro měření vytížení procesoru hlavního serveru a ztrátovosti paketů. Celé řešení je rozděleno na několik nezávislých částí viz Obrázek 7.



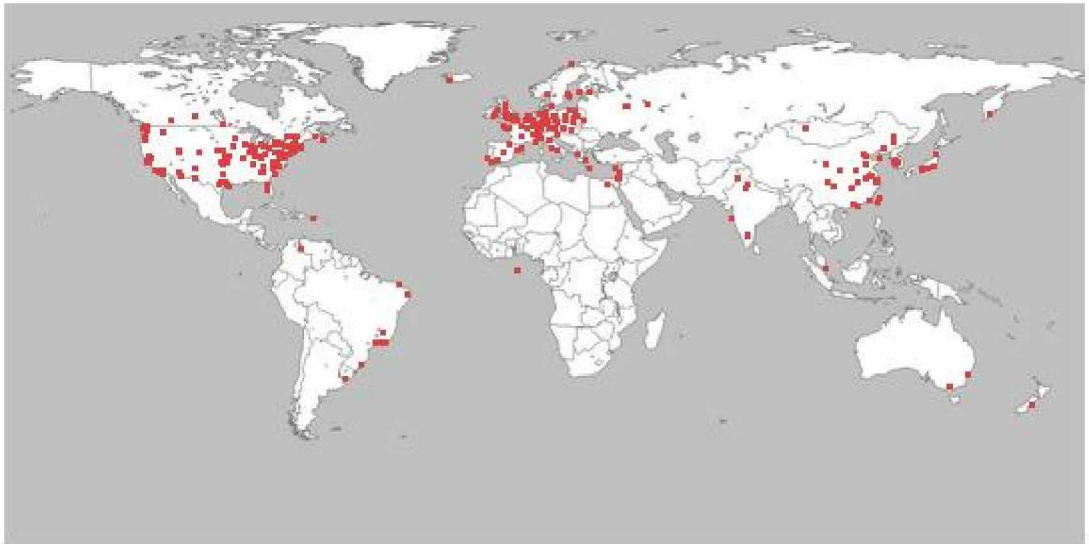
Obrázek 7: Rozmístění jednotlivých programů v experimentální síti

9.1. *Teoretická příprava realizace*

9.1.1. Planetlab

PlanetLab je celosvětová výzkumná síť, která podporuje vývoj nových síťových služeb. Od začátku roku 2003 více než 1000 vývojářů z nejlepších akademických a průmyslových výzkumných laboratoří využilo PlanetLab pro vývoj nových technologií z oblastí poskytování prostoru, síťového mapování, peer-to-peer a databázových systémů. VUT v Brně je členem výzkumného teamu pod záštitou společnosti Cesnet.

Síť aktuálně obsahuje 840 uzlů ve 408 sítích (Obrázek 8). Z důvodů velkého vytížení uzlů ostatními vývojáři je počet aktivních dostupných uzlů přibližně 400.



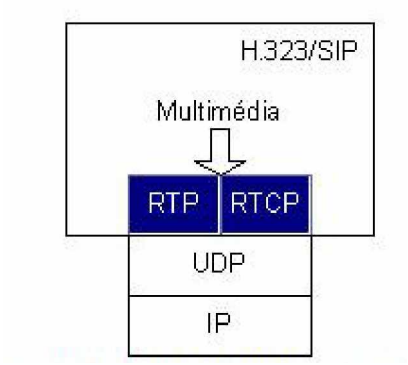
Obrázek 8: Umístění uzlů v síti Planetlab (převzato z „The Trustees of Princeton University“)

9.1.2. Popis protokolu RTCP

Protokol RTCP slouží pro přenos signalizace svázané s přenosem multimédií. Tento protokol také pracuje nad protokolem UDP (jiný protokol transportní vrstvy je však také možný). Signalizace je přenášena ve dvou typech paketů:

- Pakety nesoucí informace o zdroji dat (Sender Report, SR) ve směru od zdroje k přijímači
- Pakety nesoucí informace o příjemci dat (Receiver Report, RR) ve směru od přijímače ke zdroji.

Přenášeny jsou například informace o počtu zaslaných paketů (SR), ztrátovosti (RR), zpoždění (RR) atd. Dále tyto signalizační informace slouží k synchronizaci pomocí časových značek a identifikací účastníků spojení (emailová adresa, uživatelské jméno, atd.). Tato signalizace může být také poskytována dalším zájemcům za účelem vyhodnocování aktuálního stavu sítě (např. monitorovací aplikace).



Obrázek 9: Začlenění protokolu RTP / RTCP do komunikačního procesu

Struktura tohoto protokolu sloužila k vytvoření experimentálního UDP paketu.

9.2. *Vytvořený software pro účely měření*

9.2.1. Popis skriptů napsaných v jazyce Bash

Jak je nastíněno výše, bylo nezbytné vytvořit řadu podpůrných skriptů automatizující procesy v prostředí velkého množství testovacích uzlů, tyto skripty jsou koncipovány pro operační systém Linux Fedora Core 4, jazyk Bash.

Skript „getip“

Struktura zadání příkazu: getip

Stručný popis funkce: Zjištění IP adres serverů PlanetLabu z DNS názvů

```
#!/bin/bash
  processline(){                          // procházení souboru řádek po řádku
  line="$@"
}
FILE="DNS.txt"                            // načítat ze souboru DNS.txt
exec 3<&0
exec 0<$FILE
 while read line
  do
    host $line | cut -d " " -f 4 >> adresy.txt      // export IP adres do souboru adresy.txt
    echo $line
```

```
   done
exec 0<&3
exit 0
```

Skript „aktivni“

Struktura zadání příkazu: aktivni počet_IP_adres název_souboru

Stručný popis funkce: Zjišťování stavu programu „odesílatel“ v PlanetLabu.

```
#!/bin/bash
 processline(){                          // procházení souboru řádek po řádku
 line="$@"
 }
I=0
POCET="999"                              // maximální počet procházených řádků
FILE="adresy.txt                         // soubor s IP adresami
 if [ "$2" == "" ]; then                 // obsluha chybného vstupu
  echo "chybny pocet argumentu"
  exit 1
 else
POCET="$1"                               // pomocná proměnná
FILE="$2"

 if [ ! -f $FILE ]; then                 // ošetření existence souboru
  echo "$FILE : nenexistuje"
  exit 2

elif [ ! -r $FILE ]; then                // ošetření čitelnosti souboru
   echo "$FILE : necitelny"
   exit 3
 fi
fi
exec 3<&0
exec 0<$FILE
 while read line
  do
   if [ $I == $POCET ]; then             // ošetření maximálního počtu čtených řádků
    echo "konec odesilani"
    exit 4
   else
    ssh -f -l cesnet_vutbr1 $line ps -u cesnet_vutbr1 | grep sender  // zobrazí proces sender
    let I=$I+1
    echo "odesilam $I: $line"
   fi
```

```
  done
exec 0<&3
exit 0
```

Ostatní skripty se ve své podstatě liší jen odeslaným příkazem.

Skript „delete“

Struktura zadání příkazu: delete počet_IP_adres název_souboru

Stručný popis funkce: Smazání programu „odesílatel“ ze sítě Planetlab

```
ssh -f -l cesnet_vutbr1 $line rm ./sender
```

Skript „kill“

Struktura zadání příkazu: kill počet_IP_adres název_souboru

Stručný popis funkce: Ukončení procesu „odesílatel“ v Planetlabu

```
ssh -f -l cesnet_vutbr1 $line killall -r sender
```

Skript „odeslat“

Struktura zadání příkazu: odeslat počet_IP_adres název_souboru

Stručný popis funkce: Odeslání programu „odesílatel“ do sítě Planetlab

```
if  ping -c 1 -w 5 "$line" &>/dev/null ; then        // test dostupnosti uzlu
  scp ./sender cesnet_vutbr1@$line:./sender
fi
```

Skript „spustit“

Struktura zadání příkazu: spustit počet_IP_adres název_souboru

Stručný popis funkce: Spouštění procesu „odesílatel“ v síti Planetlab

```
ssh -f -l cesnet_vutbr1 $line ./sender
```

Skript „testerip“

Struktura zadání příkazu: testerip počet_IP_adres název_souboru

Stručný popis funkce: Otestuje zda je stanice dostupná, pokud ano zapíše její IP do soboru „online“, pokud ne zapíše do souboru „death“

```
if ! ping -c 1 -w 3 "$line" &>/dev/null ; then
  let DEATH=$DEATH+1
  echo "$line je death, celkem OFFLINE/ONLINE $DEATH/$ONLINE "
  echo "$line" >> death
else
  let ONLINE=$ONLINE+1
  echo "$line je online, celkem OFFLINE/ONLINE $DEATH/$ONLINE"
  echo "$line" >> online
fi
```

Skript „vypis“

Struktura zadání příkazu: vypis počet_IP_adres název_souboru

Stručný popis funkce: Vypíše seznam aktivních procesů v síti Planetlab

```
ssh -f -l cesnet_vutbr1 $line ls
```

Dále byla realizována celá řada dalších skriptů automatizujících spouštění lokálních programů. Vzhledem k jejich triviálnosti a podobnosti s předchozími uvedenými skripty zde nebudou podrobně rozepsány.

9.3. *Popis programů napsaných v jazyce C++*

K vlastní realizaci programů pro posílání UDP paketů podle protokolu RTCP (z důvodu dynamické velikosti datové části zde nebude uvedena struktura tohoto paketu), ale i TCP paketů byl použit programovací jazyk C++. Řešení obsahuje 4 nezávislé programy (Obrázek 7). Pro odesílání a přijímání paketů byly využity sokety.

Z hlediska komplexnosti těchto programů zde nebude uveden jejich celý zdrojový kód, ale jen podstatné úseky.

ŘÍDÍCÍ ČLEN

Program na portu 9931 odešle TCP paket na všechny IP adresy uvedené v souboru adresy.txt. Jednotlivé IP adresy jsou zadávány pomocí skriptu napsaném v jazyce Bash.

```
// otevře TCP soket
   sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_IP);
   if ( sock < 0 )                                    // chybová obsluha
         perror("Error creating socket"), exit(0);

// definice adresy, IP adresa, PORT 9931, IP adresa je zapsána v proměnné „adreska“
   saddr.sin_family = AF_INET;                        // rodina adres IP
   saddr.sin_port = htons(PORT);                      // PORT 9931
   saddr.sin_addr.s_addr = inet_addr(adreska);        // IP adresa zapsaná v prom. „adreska”

// příprava konečné podoby odesílaného paketu
   header_mc_t *hdr = (header_mc_t*)buffer;           // definice proměnné buffer
   hdr->pkt_size = htonl(size);                       // zapíše požadovanou velikost paketu
   hdr->pkt_delay = htonl(delay);                     // zapíše zpoždění mezi pakety
   hdr->pkt_count = htonl(count);                     // zapíše počet paketů

// odeslání paketu podle zadaných parametrů
   status = sendto(sock, buffer, 40, 0, (struct sockaddr *)&saddr, socklen);
```

ODESÍLATEL

Tento program pracuje podobně jako program Řídící člen, alespoň co se týče odesílání paketů. Jediným rozdílem v tomto směru je, že odesílá UDP pakety na jednu již nastavenou IP adresu. Zároveň čeká na portu 9931 na příchozí řídící TCP paket, který obsahuje informace, podle kterých je vytvořen odesílaný UDP paket. Tento program je nastaven, tak aby běžel jako služba na pozadí systému.

```
// otevře UDP soket
   if ((sUC=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP))==-1)
      diep("socket");
// nastavení adresy příjemce, v tomto případě adresa školního serveru
   sockaddr_in serversock;
   serversock.sin_family = AF_INET;
   serversock.sin_port = htons(PORT);
   serversock.sin_addr.s_addr = inet_addr("195.113.161.83");
```

```
// čekání na příchozí paket podle nastavených parametrů
status = recvfrom(sock, buffer, MAXBUFSIZE, 0, (struct sockaddr *)&saddr, &slen);
```

OPAKOVAČ

Program v podstatě totožný s programem Odesílatel. Program čeká na UDP pakety, po obdržení určitého počtu, či po stanovené době provede sumarizaci dat, a odešle již sumarizovaný paket na svůj nadřazený server. V současné době program neprovádí sumarizaci v pravém slova smyslu, ale generuje náhodné hodnoty.

PŘIJÍMAČ

Program vycházející z částí předchozích zmíněných programů. Čeká na nastaveném portu 9931 na příchozí paket, který po svém přijmutí dále zpracuje nebo zahodí podle nastavení.

Všechny tyto programy pracují v režimu unicast, ale jsou připraveny pro zavedení vysílání typu multicast. Pro testování posílání paketů skrze síť internet není možné použít vysílání tohoto typu bez úpravy přenosového prostředí.

9.4. *Výsledky měření*

9.4.1. Testování vytížení procesoru a ztrátovosti paketů

Zmíněné programy byly vyvinuty pro odesílání testovacích paketů s náhodnými daty, tak aby bylo možné co nejefektivněji měřit vytížení serveru a jejich ztrátovost při přijímání velkého množství paketů z velkého množství uzlů.

Program „Řídící člen“ odešle TCP paket do sítě nastavených IP adres PlanetLabu. Program běžící v síti PlanetLab „Odesílatel“ po obdržení tohoto TCP paketu, přečte jeho datovou část s informacemi o podobě odesílaného UDP paketu. Dále je zahájeno odesílání UDP paketů ze všech aktivních „Odesílatelů“ na jeden konečný server s programem „Přijímač“, který pakety přijímá a zahazuje. A právě zde je prováděno měření vytížení CPU a ztrátovosti paketů.

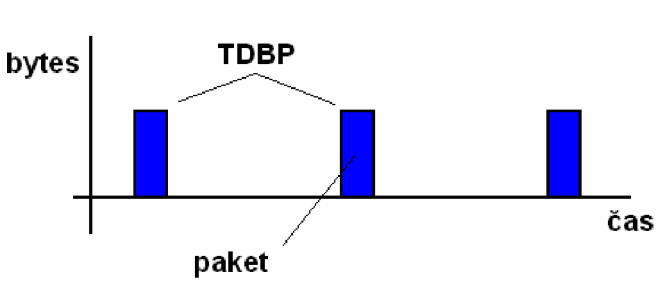
9.4.2. Výsledky naměřených hodnot

Velikost paketů (byte)	TDBP (ms)	Aktivních uzlů	Testovacích paketů		Vytížení CPU (%)	Ztracených paketů	Ztracené pakety (%)	Doba Měření (min)
60	100	394	10000		57,49%	518052	12,918953%	16,67
890	100	394	10000		59,12%	637819	16,065970%	16,67
60	250	383	4000		32,95%	16579	1,076558%	16,67
890	250	383	4000		31,06%	64679	4,199935%	16,67
60	500	379	2000		23,47%	3962	0,517232%	16,67
890	500	379	2000		22,78%	11385	1,501979%	16,67
60	1 000	376	1000		12,57%	1327	0,347382%	16,67
890	1 000	376	1000		12,68%	6296	1,639583%	16,67
60	1 250	376	800		10,27%	1620	0,532895%	16,67
890	1 250	376	800		11,25%	3236	1,072944%	16,67
60	1 500	376	600		8,75%	1754	0,771328%	15,00
890	1 500	376	600		8,64%	2382	1,050265%	15,00
60	2 000	376	500		6,44%	1058	0,559788%	16,67
890	2 000	376	500		6,46%	2286	1,206332%	16,67
60	2 500	376	400		5,41%	626	0,412929%	16,67
890	2 500	376	400		5,23%	1225	0,805921%	16,67
60	5 000	351	200		2,60%	323	0,454930%	16,67
890	5 000	351	200		2,61%	350	0,492958%	16,67
60	10 000	412	100		1,53%	210	0,525000%	16,67
890	10 000	412	100		1,56%	463	1,118357%	16,67
60	20 000	412	50		1,76%	183	0,892683%	16,67
890	20 000	412	50		0,87%	129	0,637037%	16,67

Tabulka 4: Naměřené hodnoty

Z předchozí tabulky (Tabulka 4) můžeme vyčíst parametry řídících paketů i výsledky testování. Sloupec „testovacích paketů“ je množství odeslaných paketů z „Odesílatelů“ na „Přijímač“, jejich počet určuje statistickou přesnost měření, stejně jako množství ztracených paketů. Doba měření je doba, po kterou „Přijímač“ přijímal pakety a je násobkem množství paketů a zpoždění mezi nimi.

TDBP z anglického Time Diference Between Packets znamená zpoždění mezi jednotlivými pakety (Obrázek 10).



Obrázek 10: Grafické znázornění významu TDBP

Zanedbáme-li vlastnosti některých segmentů sítě, můžeme podle vzorce 1 aproximovat výsledek skutečného vytížení serveru, při násobně větším zatížení. Z grafu 1 je patrné, že čím menší je rozdíl mezi Zp a Zz, tím menší bude možná chyba.

$$Pu = \frac{Zp}{Zz} * Ur$$

Vzorec 1: Přepočet velikosti zpoždění mezi pakety na počet stanic

Pu ..... celkový počet uzlů

Zp ..... požadované zpoždění mezi pakety

Zz ..... změřené zpoždění mezi pakety

Ur..... reálný počet aktivních uzlů

9.4.3. Příklad výpočtu

Z tabulky lze dále vyčíst, že pro hodnoty 0,5 sec., 379 uzlů, 60 bytový paket, odpovídá vytížení procesoru 23,47% a ztrátovost 0,52%.

Dosadíme tyto hodnoty do vzorce 1. Předpokládejme, že chceme aproximovat výsledek pro 5 sec. zpoždění mezi pakety.

$$Pu = \frac{5{,}0}{0{,}5} * 379 = 3790 \text{ uzlů}$$

Výpočet 1: Aproximovaný výsledek

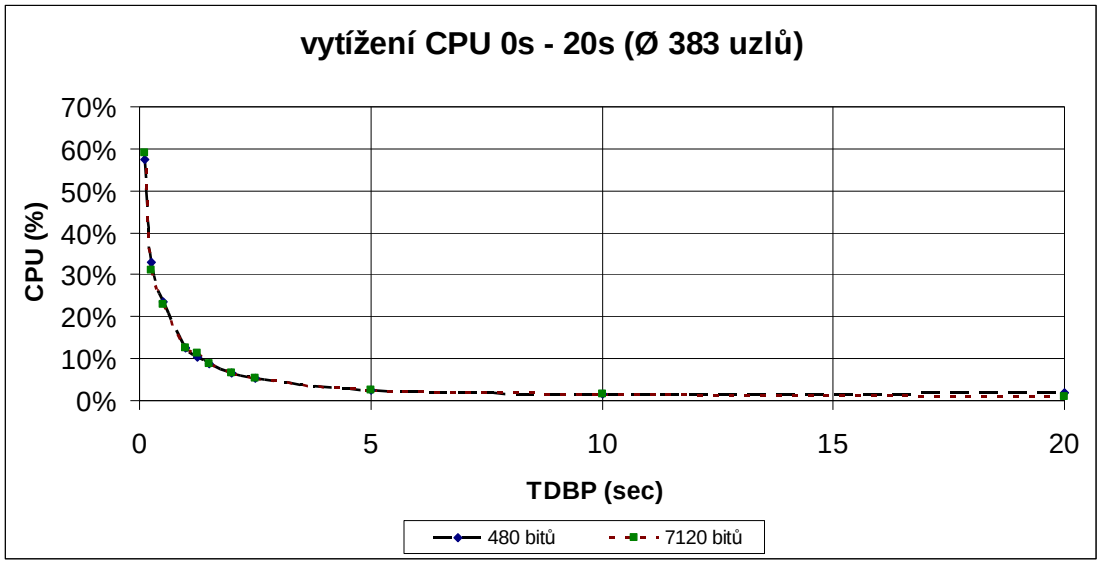
Je třeba brát v úvahu, že vytížení CPU není lineárně závislé na ztrátovosti paketů (Graf 2).

Pro vytížení procesoru byla předběžně stanovena hranice 10%. Pro pakety s velikostí 60 a 890 bytů, pro zpoždění mezi pakety 5sec (dopočítáno podle vzorce 1), pak této hranice dosáhne 1504 aktivních uzlů.

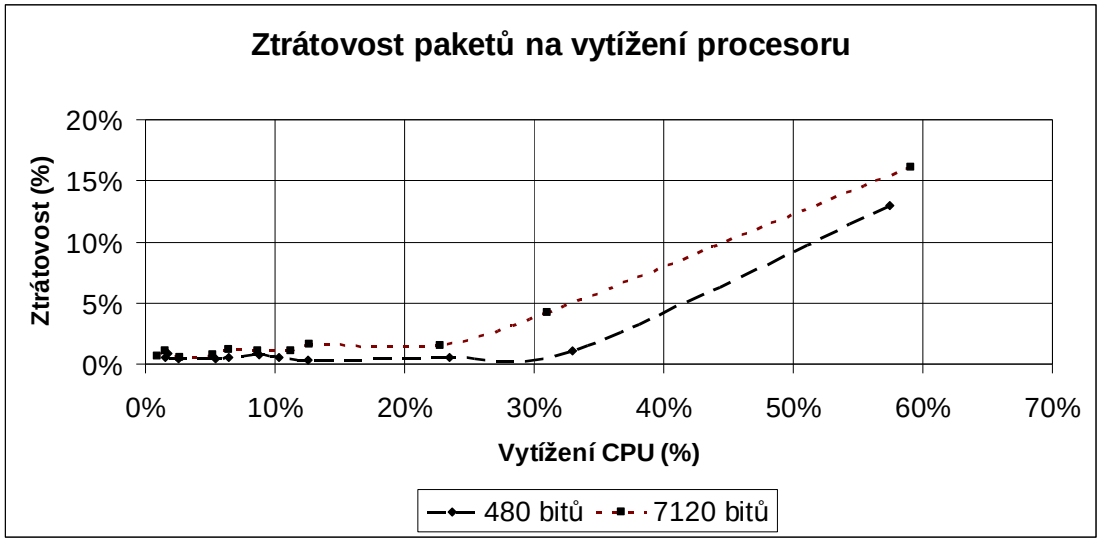
Pokud uvažujeme zpoždění 10sec, pak této hranice dosáhne 3008 uzlů. 20sec pak 6016 uzlů.

9.4.4. Grafické zpracování výsledků měření

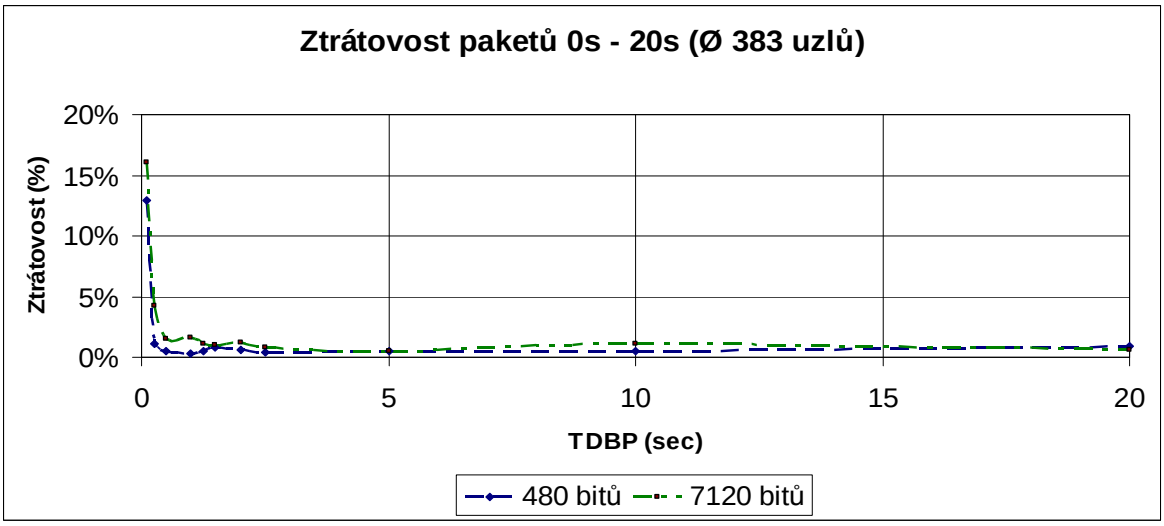
Pro lepší přehled jsou zde uvedeny jednotlivé závislosti a případně jejich zaměření na relevantní hodnoty měření.



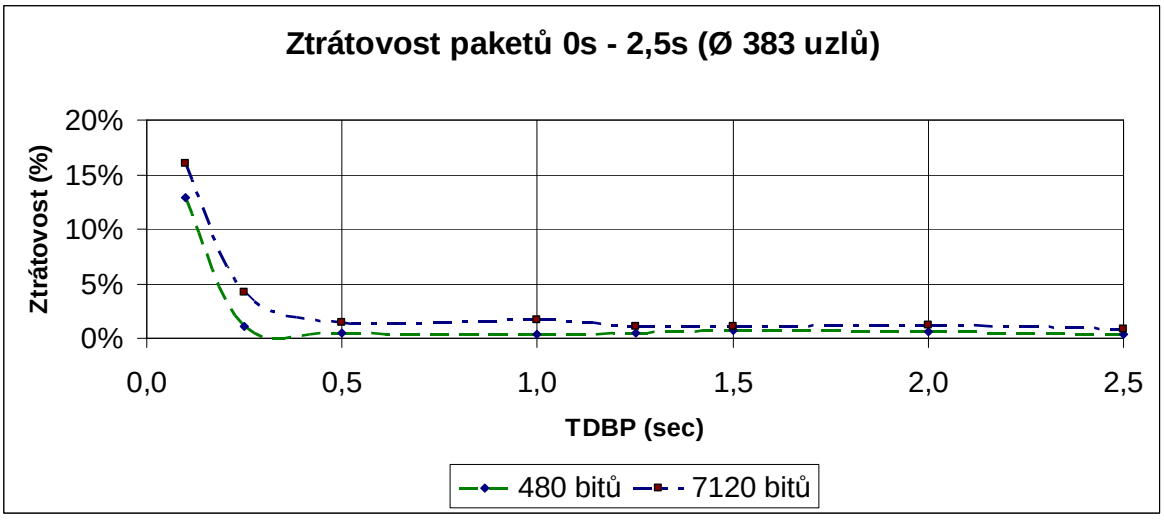
Graf 1: Vytížení CPU 0s – 20 s



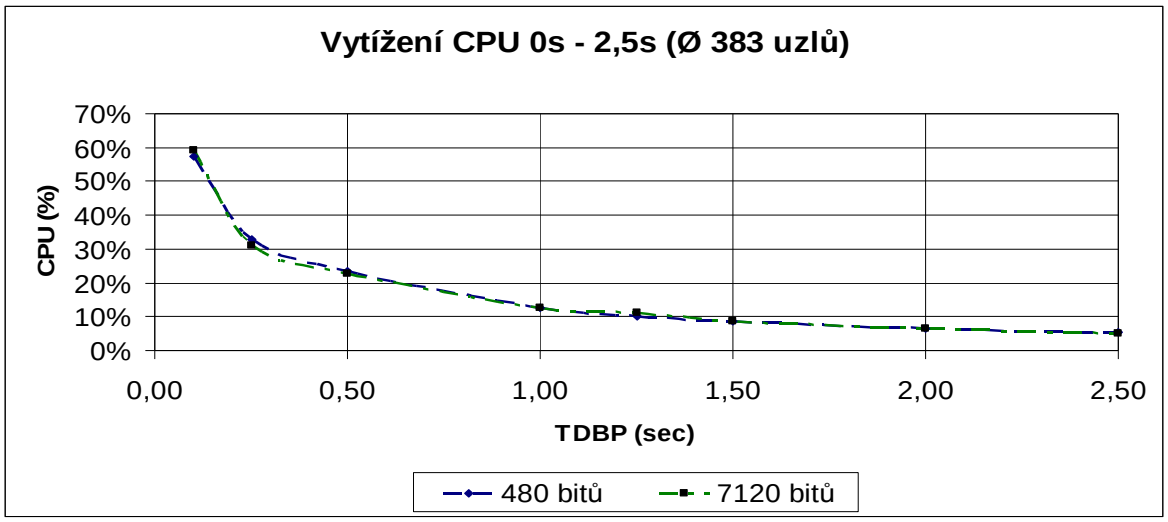
Graf 2: Ztrátovost v závislosti na vytížení procesoru



Graf 3: Ztrátovost paketů 0s-20s



Graf 4: Ztrátovost paketů 0s – 2,5s



Graf 5: Vytížení CPU 0s – 2,5s

Grafické i číselné výsledky ukazují, jaký má vliv počet aktivních komunikujících uzlů na výkon testovaného serveru. Z těchto výsledků lze jednoduše odvodit, kolik počítačových stanic je schopen jeden sumarizační server obsloužit, tak aby nedošlo k jeho přetížení. Výsledkem je představa o konečné podobě hierarchického stromu.

10. Monitorovací server

Vytvoření toho serveru bylo realizováno jazykem JAVA. Řešení zahrnuje 2 nezávislé programy z nichž jeden nese nazván SERVER, který je spojen s databází adres koncových uživatelů (klientů), kteří jsou pomocí TCP paketů oslovovány žádostmi o informace. Tyto žádosti mohou být měněny v závislosti na aktuálních potřebách. KLIENT pracuje ve své podstatě jako zrcadlo, tedy čeká na paket s žádostí, analyzuje jej a odpoví na něj. Zároveň neustále v pravidelných intervalech generuje náhodné hodnoty, které jsou vysílány zpět na SERVER.

10.1. *Popis konstrukce programu SERVER*

Uvnitř zdrojového kódu programu jsou realizována 2 vlákna. První vlákno odesílá paket se žádostí (Tabulka 5) na seznam adres uložených v databázi MYSQL. Struktura kódu pro odeslání informace je následující.

0	8	16	24 31
attributes_id	attributes_ip_id	attributes_ip_master_id	attributes_port
atributes_type	attributes_feedback_tree	attributes_resources_available	attributes_shared_trees
attributes_feedback_data_size	attributes_feedback_trans_interval	attributes_feedback_activated	attributes_packet_loss
attributes_delay	attributes_jitter	attributes_state	

Tabulka 5: Struktura TCP paketu se žádostí, oblast dat

```
// definice proměnných

iptvAddress address = new iptvAddress();
AddressManager mng = new AddressManager();

// do proměnné list je načten obsah databáze, tabulky address

List<iptvAddress> list = mng.getAllAddress();

// následující cyklus projde všechny IP adresy v databázi

while (j < list.size()) {

 // přepis IP adresy do proměnné address

 address = list.get(j);  j++;
```

```
 // odeslání paketu na zvolenou IP adresu a port, a tisk informace na obr.

 System.out.println("Adresa IP: " + address.getAddress_ip());
 Socket skt = new Socket(address.getAddress_ip(), port);
 PrintWriter out = new PrintWriter(skt.getOutputStream(), true);

 System.out.println("Odeslano: " + data);

 out.print(data);

 // zavření socketu

 out.close();
 skt.close();
}
```

Paket se žádostí se může ptát na následující druhy informací, které se dají libovolně aktivovat nebo deaktivovat (Tabulka 6). Co dané informace znamenají je zřejmé z jejich názvu.

attributes_id
attributes_ip_id
attributes_ip_master_id
attributes_port
atributes_type
attributes_feedback_tree
attributes_resources_available
attributes_shared_trees
attributes_feedback_data_size
attributes_feedback_trans_interval
attributes_feedback_activated
attributes_packet_loss
attributes_delay
attributes_jitter
attributes_state

Tabulka 6: Možné druhy informací k zaslání

Další vlákno je zkonstruováno pro příjem informací od jednotlivých klientů a jejich zápis do MYSQL databáze do tabulky attributes. Data, která jsou zapisována odpovídají struktuře uvedené v předchozí tabulce (Tabulka 6). Algoritmus tohoto vlákna je uveden v příloze.

10.2. *Popis konstrukce programu KLIENT*

I KLIENT používá pro svou činnost 2 vlákna, ovšem pro příjem paketu s žádostí o informace a následné odeslání paketu s odpovědí (Tabulka 7) je využito pouze jedno vlákno. Druhé generuje v pravidelných intervalech (500 ms) náhodné hodnoty, které jsou vhodnou metodou předávány do prvního vlákna k odeslání. V budoucnu bude algoritmus upraven tak, aby v tomto vlákně byly načítány reálné hodnoty k čemuž je program plně připraven.

0	8	16	24	31
attributes_id		attributes_ip_id		
attributes_port		atributes_type		
attributes_feedback_tree	attributes_resources_available	attributes_shared_trees	attributes_feedback_trans_interval	
attributes_feedback_data_size		attributes_feedback_activated	attributes_state	
attributes_packet_loss		attributes_jitter		
attributes_delay				

Tabulka 7: Struktura TCP paketu s odpovědí, oblast dat

10.3. *Shrnutí funkce monitorovacího serveru*

Oba programy fungují obdobně jako známý příkaz „ping“ používaný všemi rozšířenými operačními systémy. Server testuje zda je daná cílová stanice (KLIENT) dostupná a pokud ano vyžádá si o ní informace. Komunikace v obou směrech probíhá pomocí protokolu TCP, tzn. jde o spolehlivý přenos informací. Z předchozí kapitoly vyplývá, že pro velký počet souběžně komunikujících uživatelů, nebude server schopný přijímat v reálném čase odpovědi a dojde k jeho zahlcení. V případě jeho nasazení na skutečné rozsáhlou síť řekněme tisíců koncových členů je třeba použít hierarchickou sumarizaci za účelem ochrany tohoto serveru před přetížením. Můžeme tedy říci, že i v tomto případě hierarchická sumarizace je klíčem k budoucímu rozvoji a implementaci tohoto projektu do většího celku.

11. Závěr

IPTV přenos typu multicast je bezpochyby budoucností internetového přenosu. Stále více poskytovatelů nabízí VoIP (hlas přes IP sítě) služby společně s internetovým připojením. Nejčastější nabídkou bývá takzvaný „triple" který znamená internet, telefon, televize. Poslední a nejtěžší krok je aplikace IPTV systému do stávající struktury poskytovatele, který vyžaduje skutečné vysokorychlostní připojení koncových uživatelů. K přepravě jednoho kanálu typu multicast ve vysokém rozlišení (HD) je zapotřebí až 15 Mb/s. Pro 2 televizní kanály, VoIP a Internet, šířka pásma musí být okolo 35 Mb/s.

Přeprava vlastních multimediálních dat byla již v minulosti vyřešena a v současné době pracuje uspokojivě. Ovšem zatím neřešená je zpětná signalizace jednotlivých uživatelů směrem k hlavnímu serveru. Toto řešení obzvláště pro rozlehlé sítě nabízí aplikace systému hierarchické sumarizace, který ochrání hlavní server před přetížením v případě, že obrovské množství koncových uživatelů bude posílat signalizační pakety (jejich vlastní interaktivita). Tato technologie ovšem naráží na jisté obtíže, zejména s organizací stromu sumarizačních serverů (FT), předáváním paketů skrze FT a s jejich „tvarováním".

Měření vytížení procesoru a ztrátovosti přijatých paketů mělo za úkol experimentálně zjistit jaké množství komunikujících klientů může jeden sumarizační server pojmout. Z důvodů relativně malého množství reálných uzlů, použitých k testování, nebylo možné stanovit maximální množství těchto uzlů. Proto byl sestaven vzorec, pomocí kterého je možné aproximovat skutečný počet uzlů, tj. 3790, který odpovídá vytížení procesoru 24% při ztrátovosti 0,52%.

Monitorovací server je soustava 2 programů, které slouží ke zjišťování informací o klientech v síti. Serverový program má k dispozici přístup k databázi IP adres koncových uživatelů (diváků IPTV) a postupně je oslovuje specifickou žádostí o zaslání jejich informací. Po přijmutí odpovědi provádí zápis do téže databáze, kde se předpokládá jejich použití pro vizualizaci a archivaci. Klientský program po přijmutí žádosti, odešle náhodně vygenerované hodnoty zpět na adresu odkud žádost přišla. Pro budoucí rozvoj je třeba i zde počítat s implementací hierarchické sumarizace.

Literatura

[1] NAVRÁTIL, J. PlanetLab - model budoucího Internetu. Zpravodaj ÚVT MU. ISSN 1212-0901, 2006, roč. XVI, č. 5, s. 1-5.

[2] A. Lugmayr, S. Niiranen and S. Kalli, Digital Interactive TV and Metadata, Tampere University of Technology, Finland, 2004. ISBN: 0-387-20843-7

[3] S. Morris and A. Smith-Chaigneau, Interactive TV Standards: A Guide to MHP, OCAP, and JavaTV, Focal Press, 2005.

[4] Robert W. Crandall , Does Video Delivered Over A Telephone Network Require A Cable Franchise?, AEI-Brookings Joint Center for Regulatory Studies, 2007

[5] J. Chesterfield, E. M. Schooler, An Extensible RTCP Control Framework for Large Multimedia Distributions, 2003

[6] J. Ott, J. Chesterfield, E. Schooler, RTCP Extensions for Single-Source Multicast Sessions with Unicast Feedback, University of Cambridge, 2008

[7] V. Novotny, D. Komosny, Optimization of Large-Scale RTCP Feedback Reporting in ICWMC 2007. ICWMC 2007 - The Third International Conference on Wireless and Mobile Communications, Guadeloupe, 2007, ISBN: 0-7695-2796-5

[8] D. Komosny, V. Novotny, 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.

[9] NEMETH, E., SNYDER, G., HEIN T. Linux - Kompletní příručka administrátora. Computer Press, 2004. 880 s. ISBN: 80-722-6919-4.

[10] N. Anderson, An introduction to IPTV, 2006, [online], Available at: <http://arstechnica.com/guides/other/iptv.ars>

[11] L. Harte, Athos, Inc., Intorduction to IP Television (IPTV), 2005, ISBN 193281335-7

[12] T. Macaulay, Securing Converged IP Networks, Auerbach, 2006, ISBN 0849375800

[13] M. Castro, P. Druschel, A. Kermarrec, A. Rowstron “A large-scale and decentralized application-level multicast infrastructure”, IEEE Journal on Selected Areas in Communications, vol. 20, no. 8, 2002

[14] S. Ratnasamy, M. Handley, R. Karp, S. Shenker “Topologically - Aware Overlay construction and Server Selection”, 2002.

[15] Randa El-Marakby, David Hutchison "A Scalability Scheme for the Real-time Control Protocol". Proceedings of the IFIP TC-6 Eigth International Conference on High Performance Networking, Kluwer, B.V, ISBN:0-412-84660-8, 1998

[16] S. Bhattacharyya, "An Overview of Source-Specific Multicast", RFC3569, July 2003.

[17] A. Yarali and A. Cherry, "Internet Protocol Television (IPTV)", Proceeding of IEEE TENCON05, Melbourne, Australia, 2005.

[18] E. Choi, W. Lee and J. Kim, Mobility Driven Vertical Handover for Mobile IPTV Traffic in Hybrid IEEE 802.11e/16e Network, Digital Media Research Lab., LG Electronics, Seoul, Korea, 2007, ISBN: 978-0-387-74158-1

[19] P. Goudarzi, M. Adeli, M. M. Azadfar and F. Ayatollahi, Dynamic Total Cost of Ownership Optimization for IPTV Service Provider, Islamic Azad University of Dezful, Dezful, 2008, ISBN: 978-0-387-77215-8

Seznam příloh

Příloha 1: Algoritmus vlákna pro příjem paketů na straně serveru
Příloha 2: Algoritmus vlákna pro příjem a odeslání paketů na straně klienta

Příloha 1: Algoritmus vlákna pro příjem paketů na straně serveru

```
// vlákno čeká na přijetí žádosti o spojení a vytvoří soket skt

Socket skt = srvr.accept();

// načte přijatý paket do paměti

BufferedReader in = new BufferedReader(new InputStreamReader(skt.getInputStream()));

while (!in.ready()) {}

// definice proměnných

String data = in.readLine();
StringBuffer datatisk= new StringBuffer();

//  rozdělí přijatý string na jednotlivé části podle přijatých kódu a uloží je do pole

pole[0] = data.substring(0, data.indexOf("#a"));
pole[1] = data.substring(data.indexOf("#a") + 2, data.indexOf("#b"));
pole[2] = data.substring(data.indexOf("#b") + 2, data.indexOf("#c"));
pole[3] = data.substring(data.indexOf("#c") + 2, data.indexOf("#d"));
pole[4] = data.substring(data.indexOf("#d") + 2, data.indexOf("#e"));
pole[5] = data.substring(data.indexOf("#e") + 2, data.indexOf("#f"));
pole[6] = data.substring(data.indexOf("#f") + 2, data.indexOf("#g"));
pole[7] = data.substring(data.indexOf("#g") + 2, data.indexOf("#h"));
pole[8] = data.substring(data.indexOf("#h") + 2, data.indexOf("#i"));
pole[9] = data.substring(data.indexOf("#i") + 2, data.indexOf("#j"));
pole[10] = data.substring(data.indexOf("#j") + 2, data.indexOf("#k"));
pole[11] = data.substring(data.indexOf("#k") + 2, data.indexOf("#l"));
pole[12] = data.substring(data.indexOf("#l") + 2, data.indexOf("#m"));
pole[13] = data.substring(data.indexOf("#m") + 2, data.indexOf("#n"));
pole[14] = data.substring(data.indexOf("#n") + 2, data.indexOf("#o"));

// definice proměnných

iptvAttributes attribute = new iptvAttributes();
AttributesManager mng = new AttributesManager();
iptvAddress address = new iptvAddress();
```

```
StringBuffer adresa_odesilatele= new StringBuffer();
adresa_odesilatele.append(skt.getInetAddress().toString());

// načte adresu odesílatele z přijatého paketu

address = new AddressManager().getIDByIP(adresa_odesilatele.substring(1));

// uložení přijatých hodnot do pole

attribute.setAttributes_id(Integer.valueOf(pole[0]));
attribute.setAttributes_ip_id(Integer.valueOf(address.getAddress_id()));
attribute.setAttributes_ip_master_id(Integer.valueOf(pole[2]));
attribute.setAttributes_port(Integer.valueOf(pole[3]));
attribute.setAtributes_type(pole[4]);
attribute.setAttributes_feedback_tree(Integer.valueOf(pole[5]));
attribute.setAttributes_resources_available(Integer.valueOf(pole[6]));
attribute.setAttributes_shared_trees(Integer.valueOf(pole[7]));
attribute.setAttributes_feedback_data_size(Integer.valueOf(pole[8]));
attribute.setAttributes_feedback_trans_interval(Integer.valueOf(pole[9]));
attribute.setAttributes_inserted(String.valueOf(pole[10]));
attribute.setAttributes_packet_loss(Integer.valueOf(pole[11]));
attribute.setAttributes_delay(Integer.valueOf(pole[12]));
attribute.setAttributes_jitter(Integer.valueOf(pole[13]));
attribute.setAttributes_state(Boolean.valueOf(pole[14]));

// uložení hodnot do databáze

mng.saveAttributes(attribute);

// spojení pole do jednoho stringu a výpis na obrazovku

for (int d = 0; d < pole.length; d++) {
  datatisk.append(pole[d] + " / ");
}

System.out.println("Prijato konec: " + datatisk);

// zavření otevřených socketů

in.close();
}
srvr.close();
```

Příloha 2: Algoritmus vlákna pro příjem a odeslání paketů na straně klienta

```
// otevření soketu pro příjem

ServerSocket srvr = new ServerSocket(port);
while (isRunning) {Socket skt = srvr.accept();

// načtení přijatého paketu do paměti

BufferedReader in = new BufferedReader(new InputStreamReader(skt.getInputStream()));
while (!in.ready()) {}

// výpis přijatých dat na obrazovku

String s = in.readLine();
System.out.println("prijata odpoved: " + s);

// předání přijatého sřetězce pomocí speciální metody do druhého vlákna

sm2.zapis2(s);
in.close();

// čekání vlákna na další cyklus, čas volen tak aby druhé vlákno mělo dostatek času na
vygenerování nových hodnot

Thread.sleep(1100);

// načte hodnoty z druhého vlákna do prvního

String data = sm2.value();

// otevře druhý soket pro odeslání informací na server, odešle je a vypíše na obrazovku

try {
Socket skt2 = new Socket(skt.getInetAddress(), skt.getLocalPort());
PrintWriter out = new PrintWriter(skt2.getOutputStream(), true);
out.print(data);
System.out.println("odeslano: " + data);

// zavře otevřené sokety

out.close();
skt.close();
```