VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

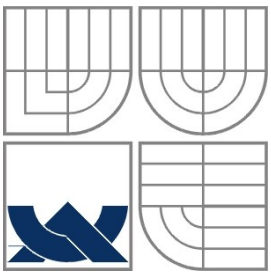
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE A ZPRACOVÁNÍ MEDIÁLNÍCH STREAMŮ

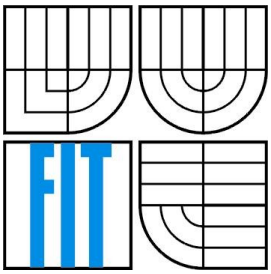
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE ONDŘEJ NOVÝ
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE A ZPRACOVÁNÍ MEDIÁLNÍCH STREAMŮ

DETECTION AND PROCESSING OF MEDIA STREAMS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ONDŘEJ NOVÝ

VEDOUCÍ PRÁCE
SUPERVISOR

DOC. DR. ING. PAVEL ZEMČÍK

BRNO 2009

Abstrakt

Práce se zabývá možnostmi detekce a samotnou analýzou multimediálních dat přenášených v různých formách po počítačových sítích, satelitu či pozemně. Jsou v ní uvedeny některé existující komerční produkty a dále navrhnuto a zrealizováno OpenSource řešení. To je otestováno na výkonnost a jsou uvedeny různé možnosti rozšíření.

Abstract

This work discuss technics of detection and processing of multimedia stream transferred in various formats over computer networks, satellites or terrestrial. It describes some of existing commercial products and it designs OpenSource variant. It's tested for performance. At the end is discussion about extension.

Klíčová slova

audio, video, multimédia, monitorování, MPEG-TS, MPEG-2, MPEG-4, IPTV

Keywords

audio, video, multimedia, monitoring, MPEG-TS, MPEG-2, MPEG-4, IPTV

Citace

Ondřej Nový: Detekce a zpracování mediálních streamů, bakalářská práce, Brno, FIT VUT v Brně, 2009

Detekce a zpracování mediálních streamů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Doc. Dr. Ing. Pavla Zemčíka. Další informace mi poskytla studijní literatura.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Ondřej Nový
19. 5. 2009

Poděkování

Rád bych poděkoval vedoucímu této práce, Doc. Dr. Ing. Pavlu Zemčíkovi za poskytnutou pomoc při vypracování a také společnosti SMART Comp. a.s. za poskytnuté testovací prostředí.

Obsah

1 Úvod

S rozvojem informačních technologií a trendem digitalizace veškerého obsahu jsou čím dál častější případy přenášení a zaznamenávání obrazových a zvukových dat v digitální formě. Multimediální informace je na jedné straně zdigitalizována, zkomprimována, přenesena třeba i na druhou stranu planety, tam rozbalena a poté zobrazena. Využíváním těchto technologií bylo možné zvýšit kvalitu přenášených multimédií, spolehlivost, ale také umožnit snadné ukládání a to vše by nebylo možné bez počítačů. Záznamy můžeme kopírovat bez ztráty původní kvality a také objem pro uložení je proti analogové formě znatelně menší.

Digitální přenos multimédií je dnes široce využíván u satelitního, pozemního a kabelového vysílání, ale také v IP sítích, ať již po Internetu nebo ve formě IPTV poskytovatelů. Nejčastěji se v tomto kontextu setkáme s pojmem DVB, což je zkratka tří anglických slov Digital Video Broadcasting, čili digitální video vysílání. Často bývá také doplněna pomlčkou a písmenem, které určuje konkrétní typ přenosu. Písmeno T – terrestrial pro pozemní, S – satellite pro satelitní a C – cabel pro kabelový. V případě satelitu, kde přechod z analogového na digitální vysílaní proběhl nejdříve, bylo možné zvýšit počet vysílaných kanálů až o řád. Poslední analogové satelitní vysílání nad Evropou má být vypnuto na konci roku 2012. Pozemní digitální vysílání, jehož přechod je v České republice aktuálně realizován, umožňuje vysílat čtyři až pět programů v jednom multiplexu (jedna frekvence) a kromě toho nabízí vyšší kvalitu obrazu hlavně v lokalitách, kde analogové vysílání mělo slabší příjem a to díky využití kvalitní modulace a podpory oprav poškozených paketů. Přechod kabelových operátorů na digitální vysílání byl motivován hlavně nedostatkem frekvencí pro vysílání a rostoucím počtem televizní kanálů, které chtěli distribuovat. Přenos multimédií po Internetu byl pouze důsledkem stále se zvyšujících rychlostí připojení do této celosvětové sítě. Mezi jeden z nejrozšířenějších portálů v této kategorii patří YouTube, nabízející prohlížení multimediálního obsahu zdarma online. Kvalita obrazu a zvuku je sice znatelně menší než u dříve jmenovaných, ale postupem času bude stoupat, tak jak dále budou stoupat rychlosti vysokorychlostního připojení až domů. Mezi největší Internetový multimediální projekt v České republice patří NACEVI – Národní Centrum Videa, zrealizovaný na žádost ministerstva informatiky společností Visual Connection, nyní KIT digital. Servery rozmístěné u několika největších českých ISP šíří živý přenos všech čtyř programů České televize, ale také obsah, který v televizi nenajdete, jako např. první hokejovou ligu. Ve špičce obsluhuje až 20 tisíc uživatelů a datový průtok přesahuje 10 Gb za sekundu přes IPv4 a desítky megabitů přes protokol IPv6. Pro srovnaní v centrálním uzlu NIX, který se stará o datové přenosy po Internetu v rámci České republiky, je ve špičce datový tok 84 Gb za sekundu. Detaily projektu jsou uvedeny na jeho webových stránkách [1]. Nejnovějším trendem v přenosu multimediálního obsahu jsou IPTV provideři, kteří po sdílených datových sítích přenášejí většinou

televizní a rádio obsah. Největším průkopníkem této technologie v České republice je společnost Telefónica O_2 nabízející služby 'digitální kabelové televize' plošně na svých ADSL připojeních. Bohužel díky velké datové náročnosti multimediálního přenosu jsou její produkty dostupné pouze ve velkých městech a zákazníkům blízko centrálním bodům DSLAM.

S rozvojem uvedených technologií vzniká také potřeba mít možnost různé multimediální streamy sledovat, zkoumat a monitorovat, čímž se zabývá právě tato bakalářská práce. Mojí motivací k realizaci bylo podpořit OpenSource komunitu vytvořením nástroje, který v současnosti existuje pouze v komerčních variantách. Cílem není vytvořit produkt, který by se těm komerčním mohl rovnat, ale spíše k nim nabídnout otevřenou variantu pro začínající projekty, ať už se jedná o novou společnost, která chce poskytovat IPTV služby nebo studentské rádio.

V následující kapitole jsou popsány metody komprimování mediálních dat a jejich přenos přes síť. Také jsou zde uvedeny možnosti minimalizování snížení kvality přehrávání v případě ztráty části dat a varianty detekce validnosti přijatých dat. V kapitole 12 je provedena analýza problému a navrženo řešení, které je v kapitole 14 popsáno a zhodnoceno. Závěrečná kapitola celou práci shrnuje a uzavíra.

2 Komprese a přenos multimédií

Tato kapitola pojednává o možnosti komprimování multimediálních dat, jejich přenos a případnou opravu v případě poškození. Je zde popsán nejpoužívanější mediální kontejner MPEG-TS a uvedeny příklady některých jiných. Existuje velké množství jiných kontejnerů, těmi se ovšem tato práce nebude zabývat. V další části této kapitoly jsou uvedeny již existující řešení pro detekování a analýzu multimediálních proudů dat, včetně popsání jejich vlastností. Ke konci kapitoly jsou probrány možnosti šifrování a ochrany před kopírováním.

S rozvojem rychlostí datových síti se čím dál častěji setkáváme s živým přenosem multimédií, čili takových, které není nutné nejdříve stáhnout, ale je možné je přímo prohlížet. Tento přenos se nazývá streaming a kromě různých internetových portálů nabízejících multimediální obsah se také rozšířil ve formě IPTV poskytovatelů. Data jsou přednačtena do mezipaměti, ze které jsou poté přehrávána a během toho se dále přijímají další data. Obecně platí, že rychlost média po kterém jsou data přenášena by měla být alespoň tak velká jako je průměrný datový tok multimediálních dat. Pokud šířka pásma není dostatečná, dojde k podtečení mezipaměti a obraz je poškozen, nebo pozastaven do té doby, než se mezipaměť zase naplní.

Data je možné po síti přenášet ve dvou formách, unicast (např. Youtube, Google Video) nebo multicast (IPTV poskytovatelé, školy, interní kamerové systémy, ...). První z jmenovaných je využíván hlavně po Internetu, kde je lokální obsah nabízen ke shlédnutí různým uživatelům. Nevýhodou je, že s rostoucím počtem uživatelů stoupá datový tok směrem od poskytovatele streamů ke klientům, ale zase může každý dostávat jiné data. Většinou je přenášen spolehlivým protokolem TCP/IP a tím je obraz vždy v pořádku i v případě ztráty paketů, které se v takové situaci zašlou znova.

Multicast bývá využíván v uzavřených sítích, kde poskytovatel má kompletní kontrolu nad síťovou architekturou a konfigurací síťových prvků. Z vysílaného místa je stream šířen pouze jednou, ale může ním být distribuován pouze obsah, který má být pro všechny klienty stejný – typicky tedy živé vysílání. Multicast je většinou šířen protokolem UDP, který nepodporuje přeposlání dat v případě ztráty, v takovém případě tedy dojde k poškození obrazu či zvuku. Existují ale jiné technologie, které umí tento problém řešit a jsou detailněji popsány v kapitole 5. Detailní popis výhod a nevýhod poskytuje kniha [2]. Multicast bývá často šířen přes paketově přepínané sítě, např. Ethernet. Protože tyto sítě jsou koncipovaný jako sdílené, čili jsou přes ně šířeny různé data, je nutno řešit prioritu doručování. Pokud po stejném datovém médiu bude přenášen hlas, video a internet, tak je nutno určit pořadí v jakém budou pakety odbavovány. V uvedeném případě bude nejdříve obsloužen hlas, který je nejvíce citlivý na zpoždění a ztrátovost, poté živé multimediální streamy

a až pokud zbude kapacita v šířce pásma, tak se obslouží zbylý datový provoz. Uplatnění těchto politik na přepínačích a směrovačích se nazývá QoS – Quality of Service. Většina kvalitních zařízení vyšší kategorie dokáže dle TOS hlavičky IP paketu upřednostňovat určitý typ provozu. Díky této technologii je možné zaručit spolehlivost služby, pokud by nebyly QoS politiky uplatňovány, tak v případě velkého zatížení linky třeba Internetovým stahováním by došlo k poškození hlasových či video dat.

Protože nekomprimované audio a video data jsou velmi náročná na šířku pásma, velice často se provádí ztrátová komprese. Pro zvuk se nejčastěji využívá kodek MPEG-1 Layer III, pro obraz ve standardním rozlišení se nejvíce prosadil kodek MPEG-2 (hlavně co se týká šíření dat po DVB-T a DVB-S či DVD) a pro obraz ve vysokém rozlišení MPEG-4. Tyto kodeky využívají nedokonalosti lidského sluchu a vidu, kde informaci kterou nedokážeme rozeznat, jako např. jemné barevné přechody, odstraní a tím sníží datový objem. Další metodou ušetření šířky pásma je nepřenášení vždy celého snímku. V případě takřka statického obrazu, kde se mění jenom jeho část se přenáší pouze její výřez. Součástí této komprese jsou také kontrolní součty, na základě kterých můžeme ověřovat validnost těchto dat.

MPEG-2 kodek přenáší multimediální data po paketech třech různých typů. První z nich, I-FRAME je pouze komprimovaná verze původního obrazu rozděleného na bloky 8x8 pixelů. Další z typů je P-FRAME, který ukládá pouze rozdíly mezi předchozím snímkem a tím aktuálním. Poslední je B-FRAME, který dokáže využívat k přenosu nových dat odkaz na předchozí i následující snímek. Detailní popis kodeku je možné najít v literatuře [3].

Samotné komprimované data je nutné v nějakém formátu uložit, či přenášet. Tomuto formátu se říká kontejner. Pro živý přenos je nejčastěji využíván MPEG-TS, pro uložení MPEG-PS, AVI či proprietární formát WMV od společnosti Microsoft. Většina kontejnerů obsahuje sekvenční čísla vnitřních paketů dat, podle kterých je možné kontrolovat ztrátu některých paketů. Příklad využití MPEG-TS je multicast přenos dat po IP sítích, DVB-T, DVB-S a DVB-C vysílání, MPEG-PS je využit při ukládání multimediálních dat ve formátu DVD a kontejner AVI pro distribuci menších filmů v jakékoliv formě.

2.1 FEC a retransmission

FEC neboli Forward Error Correction je technologie pro dopřednou opravu chyb, přesněji o přidávání kontrolních součtů datovým paketům kde v případě ztráty některých z nich je možné provést částečnou nebo úplnou obnovu. V určitém místě sítě, logicky vzato co nejblíže ke zdroji, se tedy vezme multimediální stream a přidá se k němu další kontrolní součty, což způsobí zvětšení množství přenášených dat, řádově o 10–20 %. Pokud se během přenosu některý paket ztratí tak pomocí jiných

a kontrolního součtu je dopočítán. Tuto operaci typicky dělá přijímací zařízení (STB) či softwarový přehrávač. Tento typ ochrany dat je využíván hlavně při bezdrátovém přenosu, jako např. digitální terestriální vysílání DVB-T, či satelitní DVB-S2.

Nejnovější technologie od společnosti Cisco Systems umožňují jít ještě dál. Všechna přenášená data jsou v přijímaném bodě sítě na krátký čas ukládána do rychlé paměti. Pokud se během přenosu později ztratí, klient si o ně požádá pomocí unicast paketu a jsou mu doručena opět přes unicast paket. Výhodou dočasného ukládání dat většinou na headendu je možnost využít technologii rapid channel switching, od stejné společnosti. Při klasickém přepnutí kanálu musí set-top box vyčkat než je přijat klíčový snímek, to jest ten, který překreslí celou plochu obrazovky, a poté teprve může být přehráván nový kanál. Uvedená technologie si o dotyčný klíčový snímek požádá pomocí unicast paketu, ten je na požádání vygenerován a odeslán zpět. Tím může dojít k přepnutí kanálu v řádu desítek milisekund, tak jak je většina lidí zvyklá u analogové televize. Tuto technologii je možné použít pouze tam, kde existuje zpětný kanál od uživatele po kterém si může požádat o vygenerování klíčového snímku. U DVB-T a S je to prakticky neřešitelný problém, v případě DVB-C to možné je. Nejčastěji se ale tato nová technologie využije u IPTV poskytovatelů, kteří zpětný kanál k zákazníkovi mít musí. Detailní popis je možné najít na webu společnosti Cisco [4].

2.2 Kontejner MPEG-TS

MPEG-TS je nejrozšířenější kontejner pro přenos multimediálních dat po síti ve streamované podobě. Mezi jeho využití patří přenos DVB a ATSC jak v pozemní, kabelové tak satelitní formě a také přenos dat po IP sítích. Je definován jako součást MPEG-2 standardu části první ISO/IEC 13818-1. Stará se o skladbu video a audio dat, stejně tak titulků a též o jejich vzájemnou synchronizaci. Je navržen pro použití na nespolehlivých sítích a z tohoto důvodu také podporuje opravu chyb.

Data jsou kontejnerem přenášena v paketech, kde každý z nich začíná hodnotou 0x47, třemi jednobitovými parametry (chyba transportního streamu, začátek PES nebo PSI informace, priorita) a následujících 13 bitů tvoří PID – identifikace přidruženosti ke konkretnímu toku mediálních dat jednoho typu. Význam jednotlivých PID je zaznamenán v PMT tabulce, která je šířena jako zvláštní PID stejným datovým kanálem. Další položkou v paketu jsou 4 bitové parametry (použité šifrování, přítomnost tabulky přizpůsobení, přítomnost dat), a 4-bitový čítač. Podle něj se pakety po přijmutí řadí do správného pořadí a je též možné detekovat ztrátu některého z nich. Zatím popsané části se říká 4 bajtový TS prefix. Za touto hlavičkou následuje volitelná tabulka přizpůsobení a poté samotné data, nazývané elementární stream. Tabulka přizpůsobení není pro realizaci této práce podstatná a proto se ní dále nebude zabývat. Detailnější popis je možné naleznout v literatuře [5].

Kromě multimediálních dat jsou uvnitř MPEG-TS kontejneru šířeny také dvě tabulky souhrnně označené jako PSI – Program Specific Information.

•PAT – Program Association Table – tabulka obsahující seznam všech programů dostupných v tomto toku dat. Každý program má svou vlastní PMT tabulku.

•PMT – Program Map Table – tabulka obsahující informace o každém konkrétním programu a hlavně čísla PID, ze kterých se skládá. U každého z nich také určuje o jakým typ se jedná a případně identifikaci jazyka u audio stop a titulků.

Čítač paketů je možné přirovnat k TCP/IP sekvenčním číslům kde každý odeslány paket má identifikaci o jednu vetší než ten předchozí a v případě dosáhnutí 4-bitové kapacity přeteče a začne se čítat od nuly. Přijímací strana skládá pakety do fronty podle jejich čítače a po určité době je ve správném pořadí zpracuje a to i za cenu, že některý z nich chybí. To se poté projeví rozpadnutím obrazu, často označovaným jako kostičkování, byť nikdo z nás nevlastní 3D televizi a kostičky tedy asi nemá, případně dojde k degradaci zvuku. Na obrázku 2.1 je příklad poškozeného záběru z krátkého filmu Elephants Dream. Při detekování ztraceného paketu, ještě předtím než se začnou ostatní zpracovávat, je za určitých okolností možné tento paket obnovit pomocí technologií popsaných v předchozí kapitole.

Obr. 2.1: Ukázka poškozených fragmentů obrazu MPEG-2 komprese

2.3 Komerční řešení dohledu IPTV

Existuje několik zahraničních, ale i českých společností, které se zabývají monitorováním multimediálních streamů v různých formách. Některé z těchto řešení obsahují hardwarové sondy, jiné jsou čistě softwarové, kde každá z těchto variant má své výhody, nevýhody i opodstatnění. Hardwarové sondy mají větší výkon, ale dokáží většinou provádět méně činností, z důvodu jejich složitosti a pořizovací cena bývá též větší. Softwarové sondy jsou naopak propracovanější, dokáží provádět hlubší analýzu ovšem za cenu menšího množství zpracovatelného toku dat.

Norská společnost BRIDGE Technologies se zabývá vývojem hardwarových monitorovacích sond. Nabízí širokou škálu produktů od malých sond (viz třeba [6]) pro analyzování pár multimediálních proudů dat umístěných u zákazníků až po sondu připojenou k headendu, která může monitorovat všechny přijímané streamy zároveň v reálném čase. Moduly systému jsou spravovány jedním softwarovým serverem, který se také stará o zobrazování výstupů monitoringu. Systém dokáže kromě klasické analýzy dat také generovat náhledy každého kanálu a ty zobrazit na sumární stránce ve formě mozaiky. Tu je možné prohlížet přes velké zobrazovací zařízení, např. dataprojektor na dohledovém středisku. Operátor pouze letmým pohledem ověří, že je kanál plně funkční. V obraze se také zobrazuje ukazatel intenzity zvuku, podobný tomu používaným v nahrávacích studiích. Detailnější popis je možné najít na webu společnosti [7].

Švédská společnost Agama Technologies, na rozdíl od předchozího řešení, více využívá softwarovou variantu monitorování multimédií. Také nabízí širokou škálu produktů, navíc ale dokáže provádět monitorování přímo za pomocí set-top boxu u zákazníka, který při zapnuté televizi analyzuje zobrazované data a předává je na řídící server. Tím je možné vybrat určitý vzorek zákazníků umístěných v různých lokalitách sítě a kontrolovat validnost přijímaných multimediálních dat přímo u nich doma. Poté pouze stačí statisticky vyhodnocovat výsledky z různých míst a případně informovat dohledový systém o překročení pevně daných limitů. Vyhodnocení takových výsledků technikem je zjednodušeno o agregovaný pohled na síť, kde každá lokalita má procentuálně určenu chybovost konkrétně šířených streamů. Toto řešení je nevýhodné pro malé IPTV poskytovatele a to protože je nutné míst dostatek zákazníků kteří se dívají na různé programy v každé lokalitě. Může např. nastat stav, kdy se na určitý program dívá statisticky bezvýznamný vzorek diváků a pak není možno z těchto dat vyhodnotit zda došlo k nějakému problému a nebo je to pouze chyba na kabeláži u zákazníka doma. Otázka ovšem zní, zda poskytovateli vadí, že k výpadku došlo, když se na program takřka nikdo nedívá. Všechny dohledové informace a to včetně těch z velkých sond jsou agregovaný na server, který obsahuje webové rozhraní s globálním pohledem na systém. Produkt této společnosti na rozdíl od předchozího neumí živé zobrazení mozaiky všech monitorovaných streamů, dokáže ale zobrazit jeden z nich tak, jak kdyby byl spuštěn na konkrétně vybrané sondě. Tím

si může operátor vizuálně ověřit, že ve vzdáleně lokalitě je již technikem právě opravený problém opravdu vyřešen. Popis produktů od této společnosti je možné najít na jejich webu [8].

Poslední z této kategorie dohledových řešení dodává česká společnost Alef 0. Jedná se o produkt nangu.TV, kompletní middleware pro IPTV poskytovatelé, který také umožňuje dohledovat multimediální streamy. Informace o tomto produktu je možné naleznout na webu [9]. Společnost ho získala akvizicí americké firmy ITonis. Je postaven kompletně softwarově s využitím skupiny knihoven ffmpeg. Jedná se spíše o minimalistické řešení, které provádí pouze kontrolu validnost kontejneru a že datový průtok je větší než pevně daná hodnota. Výhodou tohoto jednoduchého přístupu je dobrý výkon při větším množství sledovaných multimediálních streamů, protože nedochází k dekódování dat komprimovaných v elementárním streamu.

Společnost Cisco Systems nahlíží na monitoring mírně jinak. Akvizicí společnosti Scientific Atlanta získala do svého portfolia produkt DCM – Digital Content Managment, což je hardwarový multimediální konvertor. Dokáže přijímat streamy ze satelitu či pozemního vysílání přes DVB-S či DVB-T a poté je šířit po IP, či v jiné formě dále. Tento produkt již musí provádět demuxing – rozklad kontejneru na elementární streamy, tak proč právě při tomto procesu neprovádět kontrolu validnosti? Odpovědí na tuto otázku je produkt ROSA od stejné společnosti. Pokud tedy již poskytovatel vlastní tento headend a nechce investovat do dalšího hardwaru, který by prováděl monitoring jeho multimediálních streamů, může využít právě toto řešení. Na rozdíl od všech předchozích softwarů, ROSA dokáže rekonfigurovat síťové prvky v případě výpadku tak, aby se začal využívat záložní headend. Tím je možné nikoliv o problému jenom zavčasu vědět, ale částečně ho vyřešit tak, aby o něm nevěděli zákazníci. Protože ale popsané řešení provádí pouze rozklad kontejneru, nedokáže odhalit chyby uvnitř něj např. při poškození elementárního streamu, stejně jako předchozí produkt. Detailnější informace je možné najít na webu společnosti Cisco [10].

V následující tabulce je provedeno jednoduché srovnání vlastností uvedených komerčních řešení. Ač tato tabulka určitě neobsahuje všechny existující produkty pro monitorování multimediálních streamů, pokusil jsem se zde shrnout nejvýznamnější z nich.

Produkt	**Hardwarová analýza**	**Kontrola elementárních streamů**	**Rekonfigurace sítě**	**Mozaikový pohled**
BridgeTech	Ano	Ano	Ne	Ano
Agama	Ne	Ano	Ne	Ne
Nangu.tv	Ne	Ne	Ne	Ne
ROSA	Ano	Ne	Ano	Ne

2.4 Šifrování

Za určitých okolností je třeba přenášená multimediální data chránit. Jedním z příkladů je šíření video obsahu přes satelit, kde televizní společnost má licenci na distribuované filmy pouze pro určité území, u nás je to např. Česká televize. Ta zákazníkům, kteří prokážou trvalý pobyt na území České republiky, prodá dešifrovací kartu pomocí které mohou její programy sledovat. Detailnější popis konkrétního mechanizmu pro zabezpečení dat je popsán dále. Dalším příkladem pro nutnost šifrování je prodej prémiových kanálu, opět třeba přes satelit. Společnost distribuující tento vetšinou měsíčně placený kanál vysílá data zašifrovaně a pro jejich rozkódování poskytuje svým zákazníkům dešifrovací karty. V obou uvedených případech šlo o přenos vzduchem, čili o sdílené médium kde nedokážeme určit komu obsah doručíme a komu ne. Nutnost obsah šifrovat ale může být i v případě přenosu po nesdíleném médiu, kde možnost odposlechu může být menší a kde dokážeme zaručit že každý koncový uživatel má přístup pouze k obsahu, který má zaplacen. Distributor obsahu totiž může požadovat jeho zabezpečení tak, aby nemohl být v digitální kvalitě zaznamenáván či distribuován. Sice zde existuje možnost analogového nahrávání, v nejhorším případě videokamerou, která nahrává televizní obrazovku, ale kvalita takového záznamu je většinou citelně horší než původní video obsah. Dokonce již dnes existují technologie water-marking, neviditelného vodotisku do obrazu, který je možné kdykoliv z nelegálně okopírovaného záznamu přečíst a může obsahovat např. informace o zákazníkovi, kterému byl obsah poskytnut. Jedním z dodavatelů této technologie je společnost Verimatrix.

Mezi nejrozšířenější dodávané řešení patří produkt společnosti Conax. Již od roku 1994 dodává špičkovou technologii pro zaručení conditional-access, čili přístupu k obsahu pouze určité skupině zákazníků. V případě satelitního přenosu se v Evropě nejvíce využívá systém Cryptoworks, který je speciálně vyvinut pro DVB přenos společností Philips. Funguje tak, že zákazníkům jsou dodány karty, které v sobě obsahují privátní šifrovací klíč. Poté je určitou formou, např. zase satelitem šířen dočasný klíč, který je pro všechny uživatele stejný a kterým je aktuálně šifrován multimediální přenos. Aby mohl být tento dočasný klíč bezpečně doručen k uživateli, je vysílán stále dokola, ale pokaždé zašifrován veřejným klíčem jiného uživatele. Až se konkrétní kartě povede paket rozšifrovat, uloží si dočasný klíč do své paměti a může začít dešifrovat multimediální data. Za určitých okolností je přidána ještě jedna vrstva klíčů, které se mění ještě častěji, např. každých 5 minut, čímž je zaručena ještě větší bezpečnost celého řešení. Tento na první pohled složitý systém má své opodstatnění. Jsou zde totiž dva naprosto protichůdné faktory – prvním je co největší odolnost proti napadení, čili teoreticky co nejdelší klíč, který není možné odhadnout ani hrubou silou a druhým je co nejnižší cena, čili levný hardware. Pokud by byl použit dlouhý klíč pro asymetrické šifrování tak by sice data byla velice dobře zabezpečena, ale výkon potřebný pro dešifrování by byl velký.

V případě symetrického šifrování s krátkým klíčem se vystavujeme riziku prolomení tohoto klíče hrubou silou. Pokud ale budeme asymetricky šifrovat malé množství dat, ve kterých budeme zasílat kratší symetrický klíč, dosáhneme dobrého poměru bezpečnosti a ceny. Na tomto, či velice podobném přístupu pracuje většina dnes dostupných systému pro šifrování živých multimediálních datových proudů.

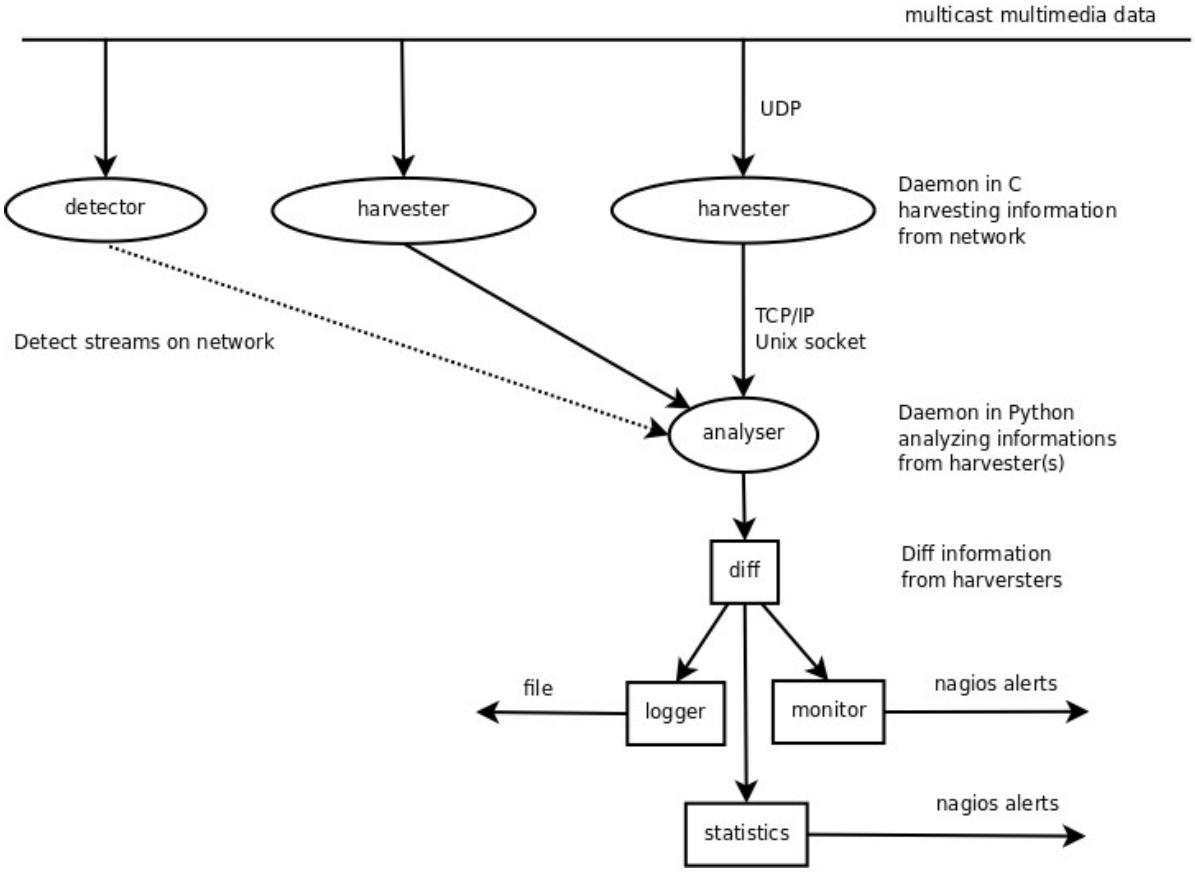
Takto popsáné řešení funguje dobře v případě přenosu bez zpětného kanálu, čili tam, kde zákazníkovo zařízení, set-top box, nemůže samo požádat o klíč. V případě IP sítí je situace o poznání jednodušší. Po zapnutí klienta si ten požádá unicast paketem šifrovací server o aktuálně platný dočasný klíč. Protistrana provede autentizaci uživatele a v případě, že má na program nárok, mu odešle správný klíč a postará se o jeho aktualizaci předtím než vyprší. Výhodou je, že klienti nedostávají pro ně nesmyslná data, obsahující zašifrovaný klíč pro ostatní uživatele. Tento systém je využit např. u společnosti Verimatrix, která se specializuje právě na IPTV poskytovatele. Detaily je možné zjistit na webu společnosti [11]. S rozvojem IP přenosu multimediálních dat i společnosti původně se specializující pouze na satelitní či pozemní přenos vyvinuli šifrovací produkty i pro tuto platformu, jak je uvedeno např. na webu společnosti CONAX [12].

Všechna zde uvedená řešení při zašifrování samotného multimediálního streamu zachovávají validní kontejner, čili upravují pouze data uložená kodekem komprese videa a audia. Pokud to kontejner umožňuje, jsou tato data označeny v hlavičce, jako je tomu např. u MPEG-TS. Výhodou tohoto řešení je možnost kontrolovat kompletnost dat i přesto, že je není možné dešifrovat, čili v situaci kdy k nim není k dispozici klíč. Za zmínku též stojí, že většina uvedených řešení umožňuje nastavit úroveň šifrování, přesněji množství paketů které mají být kryptovány v čase. Toho se využívá pro ušetření výkonu na zdrojové, ale hlavně cílové straně, kde je třeba šetřit na hardwaru. Obraz se stává nepoužitelným již v případě nastavení poměrně malých procent, zhruba dvaceti.

3 Analýza a návrh řešení

Aby byl projekt využitelný pro široké spektrum projektů, vyšel jsem z nutnosti použít jednu z dostupných multimediálních knihoven, která se postará o rozklad na elementární streamy a jejich dekódování. Protože výsledná práce je uvolněna pod otevřenou licencí, nebylo možné využít žádnou dostupnou komerční knihovnu. Z otevřených variant se nabízely tři možnosti: GStreamer, xine-lib a ffmpeg. Po detailnějším prozkoumání se jako nejlepší jevil Gstreamer. Jedná se o mladý projekt s rychlým vývojem a přijatelným výkonem podporující široké spektrum multimediálních kodeků a protokolů. Je napsán v jazyce C s využitím knihovny GLib 2.0, která nabízí objektově orientované programování v jazyce, který to přímo neumožňuje.[1].

3.1 Struktura systému



Obr. 3.1: Struktura systému

1 Tato volba se později ukázala jako chybná, což je detailně popsáno v kapitole 4.1

V další fázi analýzy byl projekt rozdělen do více částí, kde je možné každou navrhnout a zrealizovat nezávisle na jiné. Výslednou strukturu systému ukazuje obr. 4.2. Detailní popis každého modulu je rozveden v následujících kapitolách.

3.2 Předpokládané vlastnosti modulů systému

3.2.1 Modul harvester

Cílem tohoto modulu je získávat informace o sledovaných mediálních streamech a tato data předávat dále pro zpracování následujícímu modulu. Neměl by vyžadovat takřka žádnou konfiguraci, veškeré potřebné údaje získá od modulu analyser. Měl by zvládat zkoumání více proudů zároveň a jeho výkon je velice klíčový pro praktické využití.

3.2.2 Modul analyser

Tento modul musí řídit celou aplikaci jako celek, konfigurovat harvester a předávat data k analýze svým komponentám. Měl by být schopen běžet na jiném serveru než modul předchozí z důvodu škálovatelnosti a případné redundance celého řešení. Již nezpracovává tolik dat a proto nemusí být tolik kladen důraz na jeho výkon.

3.2.3 Modul detector

Detector slouží ke sledování změn na síti, přesněji změn v konkrétních skupinách vysílaných streamů. Měl by detekovat přidání či odebrání nového multicast proudu dat a dokázat informovat analyser, aby požádal harvester o případné sledování tohoto streamu.

Při detailnější analýze tohoto modulu se ukázalo, že knihovna ffmpeg není použitelná pro realizaci tohoto modulu. Zmíněna knihovna nedokáže z množství dat vybrat ty užitečné a oddělit je od zbytku. Řešení tohoto modulu je pravděpodobně nutné udělat kompletně vlastním softwarem, který v datovém toku hledá určité vzorky paketů, konkretního formátu přenosu multimediálních dat, např. hlavičku MPEG-TS. Pokud takový proud najde, předá informaci modulu analyser, který se pokusí tento stream dekódovat a informovat o výsledku a případném obsahu. Jako optimální výkonová varianta toho modulu se jeví využití hardwarové akcelerace. Existují různé komerční produkty pro hardwarové analyzování síťových toků dat, např. SCE od společnosti Cisco. Ty by se nastavily na vzorek paketu, který hledáme a v případě nalezení by pouze předaly zprávu.

V rámci těchto návrhů je v další kapitole popsána praktická realizace a detailně rozebrána problematika řešení.

4 Popis realizace

V této kapitole je popsán způsob realizace celého programu a problémy které při vývoji nastaly. Je zde také vysvětlen způsob komunikace jednotlivých modulů, jsou testovány různé výkonnostní parametry a navrženy možné použití systému v praxi.

4.1 Výměna knihovny gstreamer za ffmpeg

Výběr knihovny gstreamer se po započetí realizace projektu ukázal jako chybný. Byť dokumentace popisuje velice příjemné chybové rozhraní, které je nutné pro detekování poškozených mediálních proudů, realita byla horší. Rozhraní sice funguje dobře, ale knihovna bohužel neposílá takřka žádná chybová hlášení. Toto bylo konzultováno přímo s vývojáři projektu, kteří i přes mou ochotu určité zprávy doplnit, odmítli odesílat více informací koncové aplikaci z důvodu výkonu. Tím se ovšem stala tato knihovna nepoužitelná, protože jakákoliv ne-konzistence v mediálních streamech by nemohla být odhalena. V případě, kdy knihovna gstreamer detekuje chybu v zpracovávaných datech, informuje o této skutečnosti pomocí standardního chybového výstupu, který ovšem nedokáže aplikace postavená nad touto knihovnou jednoduchým způsobem přijímat. Pomocí rozhraní pro předávání chyb koncové aplikaci jsou posílány pouze kritické informace, např. nemožnost dále stream přehrávat, nebo že není k dispozici kodek pro dekomprimování elementárního streamu.

Po druhotné analýze byla vybrána jiná, vhodnější knihovna. Poučen z předchozích nezdarů jsem vše nejdříve konzultoval přímo s vývojáři ffmpeg, kteří mi potvrdili použitelnost jejich knihovny pro tento projekt. Obsahuje sice více strohé rozhraní pro chybové hlášky ve formě volání uživatelské funkce, zato ale počtem nešetří. Byl tedy proveden hlubší průzkum této knihovny a nakonec byla i použita. Protože ale obsahuje opravdu velké množství různých chybových hlášek, bylo nutné navrhnout sofistikovanější komponentu statistics, která pro analýzu zpráv využívá regulární výrazy.

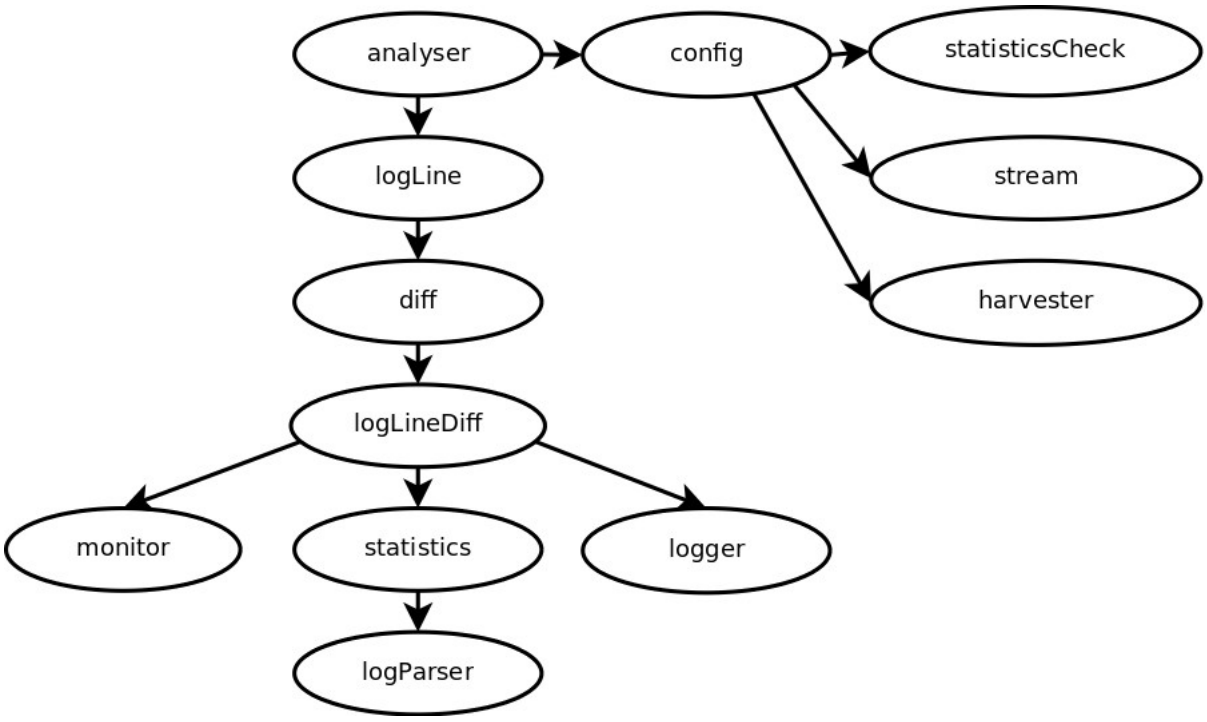
4.2 Harvester

Modul harvester je zrealizován v jazyce C s využitím multimediální knihovny ffmpeg, což by měla být z výkonnostního hlediska nejlepší varianta. Jedná se o vícevláknovou samostatnou aplikaci bez složité konfigurace. Program čeká na příchozí TCP/IP spojení na daném portu od modulu analyser. Ten protokolem popsaným v kapitole 4.4 zažádá o sledování určitého mediálního streamu a přijímá o něm informace, které dále zpracovává.

Harvester se spouští příkazem ./harvester s jedním parametrem, kterým se určuje TCP/IP port pro příchozí spojení. K tomuto portu se může připojit teoreticky neomezené množství instancí modulu analyser, jediný limit je výkon serveru, který je testován v kapitole 4.5.1. Soket pro příchozí spojení je otevřen na 0.0.0.0, čili přijímá spojení z jakéhokoliv vstupního zařízení. V případě nevhodnosti tohoto řešení je možné nastavit firewall.

4.3 Analyser

Modul analyser je zrealizován v programovacím jazyce Python. Je jedno-vláknový a dokáže zpracovávat pouze jeden multimediální stream. Pokud je nutné jich monitorovat více, spustí se více instancí stejného programu. Jako jeho jediný parametr při spuštění se zadává jméno konfiguračního souboru ve formátu XML. Celý tento proces je možné zjednodušit inicializačním skriptem, který spustí jeden proces pro každý konfigurační soubor v daném adresáři. Výhoda tohoto řešení se projeví při nutnosti překonfigurovat jeden z monitorovaných streamu. V případě vícevláknové aplikace by bylo nutné celou aplikaci vypnout a znovu zapnout s novou konfigurací, případně složitě programovat opětovného načtení konfigurace např. signálem. Takto pouze stačí vypnout a znovu spustit jeden z procesů s upraveným konfiguračním souborem. Výsledná struktura modulu analyser je na obr. 4.1.



Obr. 4.1: Výsledná struktura modulu analyser

4.3.1 Konfigurace modulu analyser

První element *name* určuje vlastní pojmenování monitorovaného streamu, které je možné zasílat např. dohledu. Následuje element *uri* obsahující URL adresu požadovaného multimediálního toku dat. Tento řetězec je na straně harvesteru předáván přímo knihovně ffmpeg, čili seznam podporovaných formátu přenosu je možné vyčíst z její dokumentace. Nejčastěji používaný formát přenosu bude UDP a poté URL vypadá následovně: udp://<adresa multicastové skupiny>:<port>/, např. tedy udp://239.1.1.1:11111/. Podporován je ale i protokol HTTP či RTP. Další z elementů je *harvester* se třemi atributy. ID určuje interní číslování, které je možné stejně jako jméno streamu odesílat callback, kterému se ale budeme věnovat později. Dalšími atributy jsou *host* a *port* určující doménové jméno či IP modulu harvester a portu na kterém čeká na příchozí spojení. Počet tohoto elementu a tím i harvesterů není omezen. Při využití více harvesterů se analyser připojí ke každému z nich a data porovnává komponentou diff. Následuje element *stream* určující každý konkrétní elementární stream dat, který chceme monitorovat. Atribut *type* určuje zda se jedná o video, audio nebo titulky (subtitle). Další atribut PID je číslo elementárního streamu, *lang* jazyk ve kterém je elementární stream šířen, případně 'unknown' pro obraz. Poslední atribut tohoto elementu *maxLeak* určuje počet sekund, po které nemusí harvester přijímat žádné data z tohoto streamu než zavolá callback. Poté již následuje konfigurace každé z komponent.

Element *monitor* jako první obsahuje atribut *callback,* což je příkaz který má být spuštěn v případě, že komponenta monitor detekuje problém se streamem. Dle typu problému se využije jeden z formátovacích řetězců v atributech. Každý tento řetězec může obsahovat proměnné, které obsahují detailní informace a popis problému.

- *formatStream* – přidání či odebrání elementárního streamu, obsahuje tyto proměnné:
  - pid – PID elementárního streamu ve kterém došlo k problému
  - error – textový popis chyby která nastala
  - type – požadovaný typ dle konfiguračního souboru
  - shouldType – reálně přijatý typ streamu
  - lang – požadovaný jazyk dle konfiguračního souboru
  - shouldLang – reálně přijatý jazyk streamu
- formatLeak – formátovací řetězec použitý při detekování výpadku streamu, čili že po dobu delší než je uvedenou v atributu maxLeak nepřišel ani jeden datový paket jednoho z elementárních streamů
  - harvester – ID harvesteru, který tuto chybu detekoval
  - pid – PID elementárního streamu ve kterém došlo k výpadku

Element *logger* konfiguruje stejnojmennou komponentu. První atribut *stdout* určuje zda mají být všechny zprávy přijaté od harvesterů vypisovány na standardní výstup. Této funkcionality je

nejlépe možné využít při prvotní konfiguraci systém a pro testovaní funkčnosti dílčích částí. Další volitelný atribut *file* odpovídá názvu souboru do kterého mají být zprávy zaznamenávány. Teto element obsahuje ještě dva formátovací atributy:

- formatStream – přidání či odebrání elementárního streamu
  - harvesters – čárkou oddělený seznam harvesterů od kterých tato zpráva přišla
  - pid, type, lang – mají stejný význam jako u komponenty monitor
- formatMessage – obecná zpráva přijatá od harvesteru
  - harvesters – stejný význam jako u formatStream
  - module – název modulu ffmpeg, který tuto zprávu zaregistroval
  - message – samotná textová zpráva

Příklad formátovacího řetězce: pid=%(pid)s error=%(error)s type=%(type)

4.3.2 Komponenta diff

Tato komponenta tvoří frontu přijatých zpráv od všech harvesterů a jednou za určený interval provede vyhledávání shodných zpráv, které jsou dále předány dalším komponentám. Tím je např. možné detekovat, že se určitá chyba projevila pouze na jednom z harvesterů. Informaci o tom který z nich to byl je předáván v proměnné externím aplikacím přes callback.

Interně komponenta řadí všechny přijaté zprávy do pole, ve kterém poté hledá duplicity. Zprávy starší než zadaný interval jsou předány dál, i když přišly pouze z jednoho z harvesterů. V takovém případě je ale tato zpráva opožděna. Délku tohoto maximálního zpoždění je možné upravit v konfiguračním souboru. Úspěšně jsem celý projekt testoval s hodnotou 3 sekundy, které se zdá pro většinu případů dostačující. Větší čas by byl potřeba pouze v případě velkého zpoždění na síti při komunikaci mezi modulem harvester a analyser.

4.3.3 Komponenta logger

Komponenta sloužící k ukládání či vypisování všech přijatých zpráv. Její nejčastější použití bude při spouštění nové konfigurace pro ověření správné funkčnosti. Pro každý typ zprávy má vlastní formátovací řetězec oddělený od toho, který se používá v komponentách monitor a statistics. Zprávy dokáže ukládat do zadaného souboru a případně je také zároveň vypisovat na standardní výstup.

Ukázka ladícího výstupu:

```
harvesters=1  pid=101 type=VIDEO lang=unknown
harvesters=1  pid=111 type=AUDIO lang=unknown
harvesters=1  pid=101 module=mpeg2video message=00 motion_type at 0 13
...
```

Ukázka použitých formátovacích řetězců:

```
harvesters=%(harvesters)s pid=%(pid)s type=%(type)s lang=%(lang)s'
harvesters=%(harvesters)s pid=%(pid)s module=%(module)s message=%(message)s
```

4.3.4 Komponenta monitor

Komponenta monitor je jedna ze dvou klíčových. Stará se o ověřování, že PID, typ a jazyk elementárních streamů jsou v pořádku a také že od každé z nich jsou stále přijímána data. Pokud jedna z kontrol selže je dle konfiguračního souboru spuštěn uživatelem definovaný callback s jedním parametrem. Za ten je dosazen výsledek formátovacího řetězce podle konkrétní chyby. Tím je možné externí aplikaci předat dostatek informací o tom co problém přesně způsobilo.

4.3.5 Komponenta statistics

Statistics je druhá klíčová komponenta. Každou přijatou zprávu od harvesteru projde přes sadu regulárních výrazů, kde každý má vlastní ID a je možné v XML formátu definovat své vlastní. V případě shody je podle této identifikace vytvářena statistika zpráv. Pokud dojde k překročení přípustných mezí uvedených v konfiguračním souboru je zavolán uživatelem definovaný callback opět s jedním parametrem ve formě výsledku formátovacího řetězce. Tento na první pohled složitý systém má své opodstatnění. Při příjmu signálu přes satelit, či po DVB-T dochází k nahodilým chybám. To může být neřešitelný stav, např. při špatném počasí. Díky statistické analýze můžeme tyto falešné alarmy eliminovat a získávat informace až ve chvíli, kdy se jedná opravdu o závažný problém.

Příklad chybové hlášky knihovny ffmpeg a regulárního výrazu který ji zpracuje:

```
mpeg2video concealing 360 DC, 360 AC, 360 MV errors
mpeg2video concealing [0-9]+ DC, [0-9]+ AC, [0-9]+ MV errors
```

4.4 Protokol komunikace modulů harvester a analyser

Aby byla zaručena spolehlivá komunikace mezi modulem harvester a analyser, byl použit protokol TCP/IP. Harvester čeká na příchozí spojení na daném portu, analyser se k němu připojí a poté odešle URI požadovaného streamu ke zkoumání a za ním znak '\n', čili symbol konce řádku. Harvester

provede otevření tohoto multimediální toku dat a začne odesílat informace. Pro ukončení stačí když jedna ze stran uzavře TCP/IP spojení. Všechny zprávy následující po prvním řádku jsou ve formátu:
command pid param1 param2 ...
Každý řádek zprávy je zakončen znakem '\n'. Počet *param* může být různý, případně nemusí být žádné. Pokud se jedná o zprávu, které se netýká konkrétního PID je v místě *pid* napsána 0. Klíčové slovo *command* může nabývat následujících hodnot:

•*STREAM* – Informuje o nalezení nového elementárního streamu, v prvním parametru zasílá jeho typ (*VIDEO*, *AUDIO*, *SUBTITLE*) a ve druhém jazyk. Pro typ *VIDEO* je zasílán vždy jazyk *UNKNOWN*.

•*DATA* – Tato zpráva je zaslána jednou za sekundu pro každý elementární stream pro který přišel alespoň jeden validní paket. Neobsahuje žádné parametry.

•*ERROR* – Informuje o mezních stavech, např. nedostupnosti kodeku pro dekódování mediálních dat. V parametru je uveden textový popis problému.

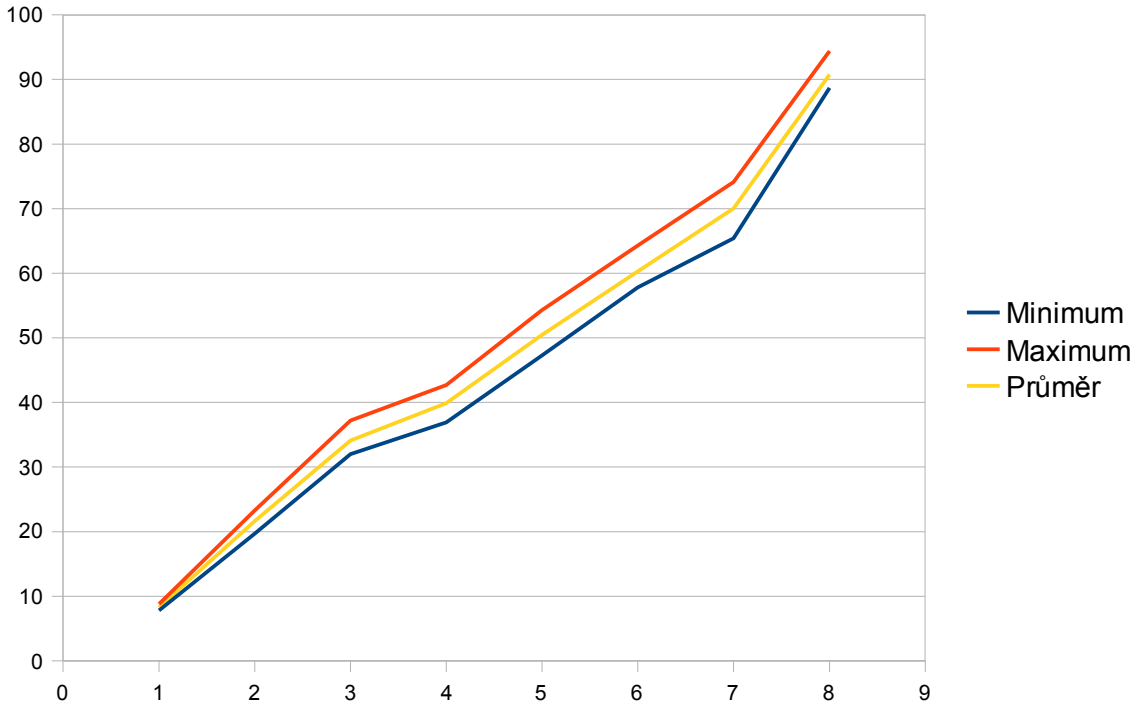
4.5 Parametry systému

4.5.1 Výkon

Výkon je velice důležitý parametr při vývoji aplikací, které musejí zpracovávat data v reálném čase. Pokud se systém provádějící monitorování a analyzování přetíží, výsledky jsou zkreslené a tím nepoužitelné. V této kapitole jsou ověřeny různé parametry projektu a nalezeny horní hranice, kdy je možné ještě spolehlivě aplikaci využívat. Veškeré testy byly provedeny na sestavě Sun Fire X2100 M2, Dual-Core AMD Opteron(tm) 2,6 Ghz, 4 GB DDR2 RAM 667 MHz, 1x SATA II HDD250 GB a operačním systému GNU/Linux. V případě testování pouze na jednom procesoru bylo druhé jádro softwarově vypnuto.

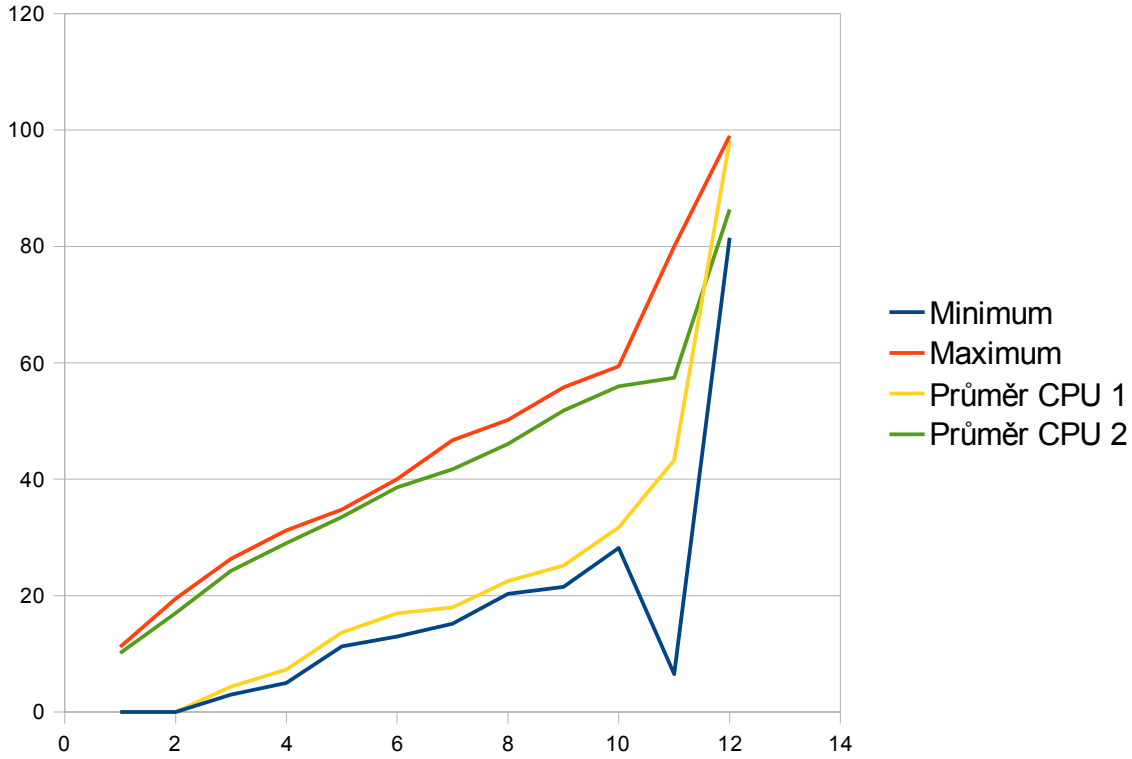
Prvním z důležitých parametrů systému je množství streamů, které je možné v jednom čase analyzovat. K měření zatížení výkonu byla využita aplikace *top*. Ta zobrazuje krátkodobý průměr procentuálního zatížení procesoru v čase, čili poměr mezi výpočtem na procesoru a režimu idle, čili takovém kdy procesor nic nedělá. Aby byla co nejméně vyloučena chyba měření, bylo provedeno přečtení vzorku celkem 10x pro každé množství multimediálních streamů a to v intervalu jedné minuty. Jako vzorek dat byl využit satelitní příjem veřejnoprávní České televize, kanálu ČT1, kontejner MPEG TS, video kodek MPEG-2, audio kodek MPEG-1 Layer III, celkový průměrný datový tok 6 Mb za sekundu. Výsledek měření je zobrazen na grafu 4.1. Z něho vyplývá, že závislost zatížení procesoru a počtu monitorovaných multimediálních streamů je lineární. Většinu procesorového času spotřeboval modulu harvester, což je pochopitelné. Měření bylo ukončeno u osmi

multimediálních streamů, kde zatížení procesoru bylo takřka 100%, ale hlavně již modul harvester hlásil chyby při dekódování, což u sedmi nedělal. Tyto chyby, byť reálně na živém streamu neexistovaly, vznikly díky přetížení systému a nestíháním dekódovat data.



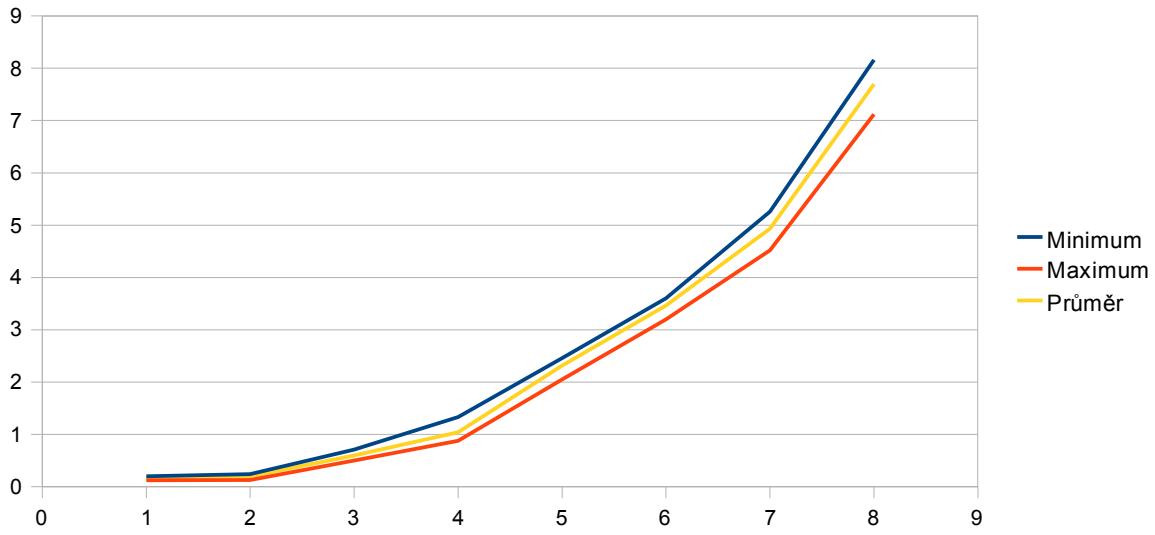
Graf 4.1: Závislost zatížení procesoru na počtu multimediálních streamů

V následujícím testu výkonu je otestována možnost paralelního zpracování na více procesorech či dvou jader jednoho procesoru na stejném serveru. Metodiky měření byla použita stejná jako v předchozím odstavci, pouze měření zatížení každého jádra bylo prováděno zvlášť. Zdrojový stream byl využit opět z veřejnoprávní České televize. Graf 4.2 ukazuje závislost procentuálního zatížení procesoru a počtu multimediálních streamů. Systém byl přetížen při 11 streamech současně, což je i viditelné na minimální hodnotě zatížení procesoru – zde software prostě zazmatkoval. Přidáním jednoho jádra procesoru, čili teoreticky dvojnásobného výkonu bylo získáno pouze 50 % streamů navíc. Zdá se tedy, že procesorový výkon nebude jediným faktorem ovlivňující získatelný výkon.



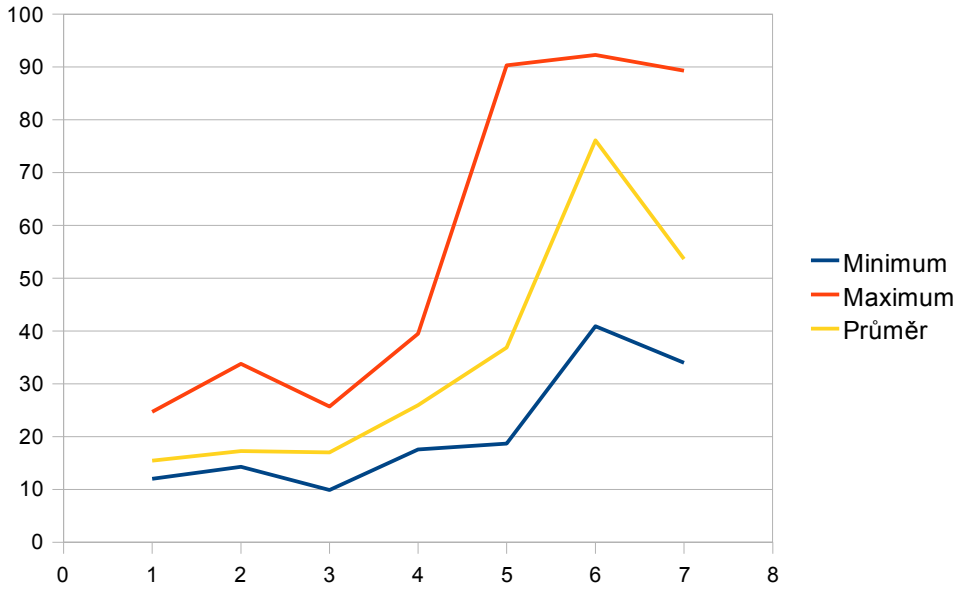
Graf 4.2: Závislost zatížení dvou jader procesoru na počtu multimediálních streamů

Protože měření popsané metodikou výše počítá pouze průměrný procesorový čas, nezískáme tím žádné informace o čekání na vstupně-výstupní operace a práci s pamětí. V operačním systému Linux existuje ukazatel *load average*, který ukazuje zatížení včetně čekání na dokončení ostatních operací po sběrnicích. V tomto případě bylo aktivní pouze jedno procesorové jádro. Graf 4.3 ukazuje závislost mezi jeho hodnotou a počtem multimediálních streamů. Závislost již není lineární jako u předchozích grafů, nýbrž exponenciální. Protože data byla přijímány gigabitovou síťovou kartou připojenou na sběrnici PCI-E, nemělo by toto býti úzkým hrdlem. Zdá se tedy, že problémem prostupnosti je rychlost paměti, což i plyne z principu funkčnosti modulu harvester. Ten všechny přijaté data dekomprimuje do paměti a poté odstraní, což je zbytečně neefektivní. Bohužel ale knihovna ffmpeg nedokáže provést pouze kontrolu validnosti aby výstup někam ukládala.



Graf 4.3: Závislost hodnoty load na počtu multimediálních streamů

Všechna předchozí měření přijímala naprosto validní data, čímž nejvíce procesorového času spotřeboval modul harvester. Co když ale dojde k výpadku či poškození? Jak se změní výkon systému? Následující test analyzuje opět multimediální stream České televizi, tentokrát ale pouze jeden. Postupně je softwarově simulováno ztracení datových paketů a jejich množství stoupá. Graf 4.4 ukazuje závislost procentuálního zatížení procesoru na množství ztracených paketů za sekundu. Tím, že se paket ztratí, knihovna ffmpeg vygeneruje skupinu zpráv o chybném multimediálním streamu, které jsou přeposlány modulu analyser a ten se pokusí je zpracovat. Jak ale graf ukazuje, výkon tohoto modulu není příliš velký, už ztracením 7 paketů do sekundy začíná být systém přetížen, tedy přesněji maximální zatížení je 100%. To je způsobeno relativně složitým algoritmem porovnávání zpráv v modulu analyser. Pokud by ale ztrátovost paketů nebyla trvalá, tak dojde pouze k opožděnému zpracování a systém bude dále funkční. V praxi je ztráta či poškození paketů spíše nahodilá, než trvalá, takže i tento výkon může být uspokojivý. Pokud by totiž nějaký stream vykazoval takovouto chybovost, tak je stejně nepoužitelný pro sledování.



Graf 4.4: Závislost zatížení procesoru na počtu ztracených paketů

4.5.2 Škálovatelnost

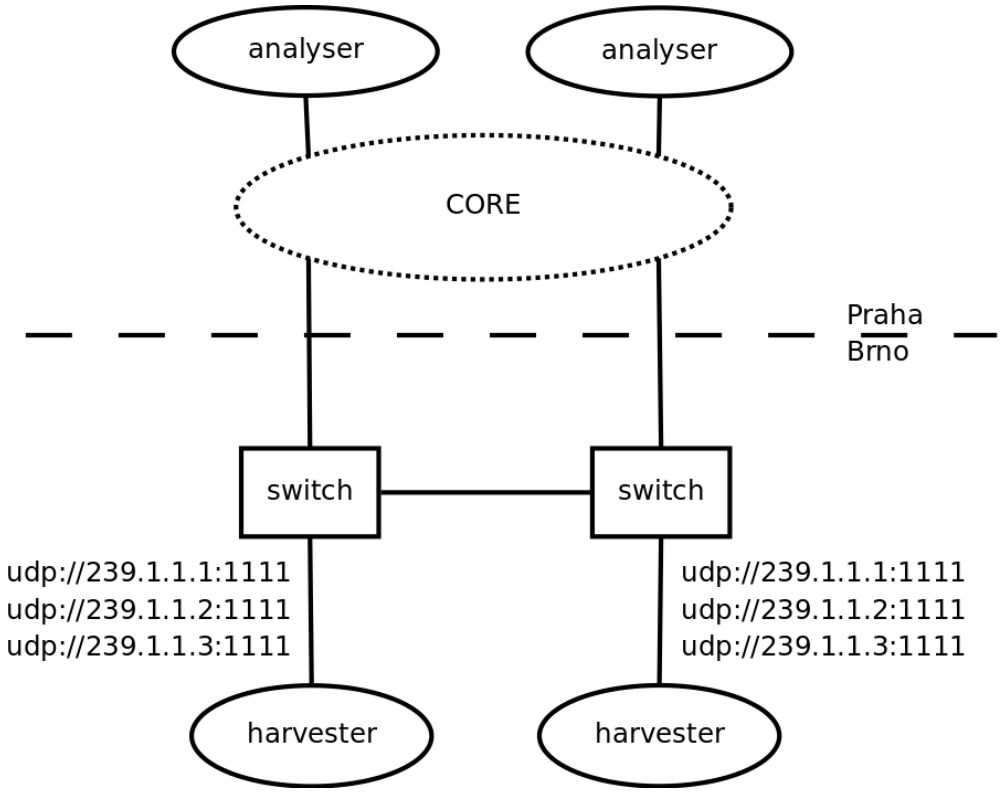
Díky jednotné konfiguraci umístěné v modulu analyser je možné umístit harvester na teoreticky neomezené množství serverů a tím rozložit zátěž mezi ně. Analyser poté provádí řízení po síti. Každý server musí monitorovat vždy alespoň jeden celý stream, není tedy možné distribuovat zpracování jednoho datově velmi náročného multimediálního toku dat. Pokud je potřeba takový stream analyzovat je možné ho přijímat pomocí aplikace VLC a dále znova šířit jako několik streamů. Tuto aplikaci je možné spustit na dalším serveru tak aby nebyl ovlivněn výkon. Optimální je umístit distribuované servery modulu harvester do stejné lokality abychom vyloučili chybu přenosové sítě. O teoretickém výkonu samotné aplikaci pojednává kapitola 4.5.1.

4.5.3 Redundance

Pro dosáhnutí High Availability, čili vysoké dostupnosti celé aplikace, je možné v síti spustit více analyserů, kteří paralelně provádí stejnou kontrolu dat přijímaných z harvesterů. Poté předávají informace dohledovému systému. Pokud jeden z procesů analyser selže, stále jsou dodávány data z druhého.

Optimální je mít redundantní i harvestery umístěním více serverů do stejné lokality a na obou monitorovat stejné multimediální streamy. Tím se vyloučí chyba např. síťové karty, či přepínače v dotyčné lokalitě. Kompletně redundantní řešení je zakresleno na obr. 4.2. Oba dva procesy

harvester provádí monitorování stejných toků dat a předávají informace buďto každý do jednoho z modulů analyser, případně je možné data předávat do obou dvou do kříže. Protože ale harvester neobsahuje mezivláknovou komunikaci bude v případě požadavku na sledování dvou stejných multimediálních streamů provádět zkoumaní dvakrát. Tím musí být výkon dvojnásobný, případně je nutné je rozdělit na více serverů. Na obrázku je též zakreslena redundantní trasa mezí dvěma body sítě Brnem a Prahou. Tím by měl být vyloučen možný výpadek spojení mezi oběma moduly systému a dosáhnout tím High Availability.



Obr. 4.2: Redundantní zapojení projektu

4.6 Praktická aplikace

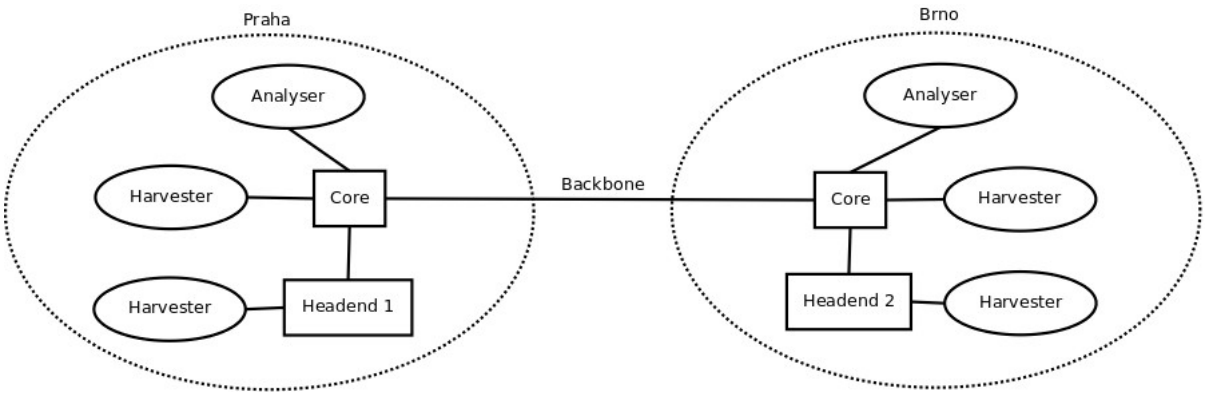
4.6.1 Dohledový systém IPTV

Jedno z možných využití aplikace je dohledování šíření multimediálních streamů po síti poskytovatele IPTV řešení. Na obr. 4.3 je zakreslen návrh možné realizace. První harvester je umístěn co nejblíže zdroji signálu, který může být přijímán např. satelitem, kde vzniká určitá trvalá chybovost. Umístěním do lokalit dvou redundantní zdrojů signálu můžeme porovnávat výsledky a tím ověřit,

že se jedná o chybu již na zdroji při příjmu signálu. V případě chybovosti pouze na jednom z procesu harvester je zřejmé, že je jeden z přijímačů vadný.

Umístěním dalších harvesterů na přístupové síti ověřujeme, že se data po cestě nepoškodila. Optimální umístění je co nejblíže zákazníkovi, čímž provádíme kontrolu celé přenosové sítě. V případě detekování problému může být informace předána do dohledovacího řešení poskytovatele, např. OpenSource systému Nagios, který nabízí široké možnosti napojení svých testů a dále se již postará o předání informace technikům, kteří problém vyřeší.

Protože množství datových streamů, které by chtěl poskytovatel monitorovat, může být větší než dokáže existující, či cenově dostupný, server zvládnout, je možné harvester rozdělit na více serverů umístěných ve stejné lokalitě. Dnes běžně prodávané video služby nabízejí i 100 televizních programů. Dle výkonnostních testů provedených v kapitole 19 by bylo v tomto případě nutné pořídit i 12 serverů pouze pro běh modulů harvester, což může být nemalá investice a to ne jenom na pořizovacích nákladech hardwaru. V tomto směru se zdá, že kombinace softwarové analýzy bez hardwarové akcelerace a kompletní dekomprimace elementárních streamů není příliš efektivní. Poskytovatel ale stále může provádět analyzování pouze pro něj nejdůležitějších televizních stanic – tedy těch s největší sledovaností. Celé toto řešení je možné rozšířit o redundantní analyser a tím zajistit High Availability celého systému, jak je popsáno v kapitole 4.5.3.

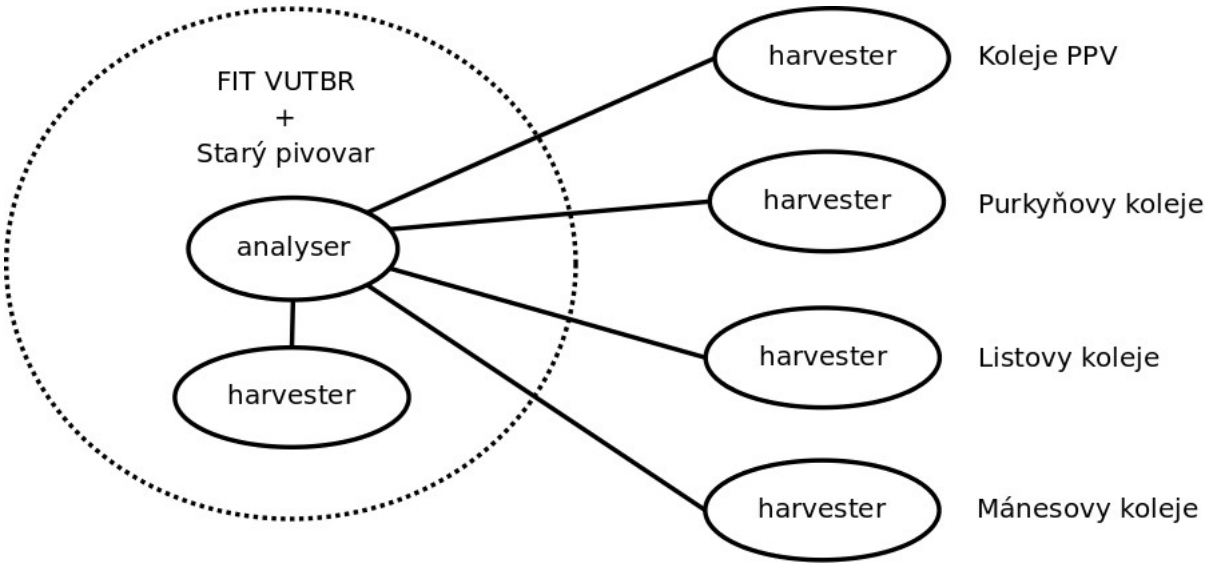


Obr. 4.3: Dohledový systém IPTV

4.6.2 Dohledový systém FIT

Čím dál větší množství studentů sleduje přednášky z pohodlí své koleje a tím pádem i výpadek šíření těchto multimediálních streamů je čím dál více bolestivý. Návrh řešení je zakreslen na obr. 4.4. Do lokální serverovny FIT by se umístil první harvester a na každou kolej další. V jednom centrálním bodě by běžel analyser, který by zpracovával poskytnutá data. Bohužel ale ne všechny přednášky mají povoleno šíření a v noci stream není distribuován vůbec. Muselo by se tedy provést

doprogramování modulu pro napojení na video serveru FIT. Ten by porovnával informace s databází povolených streamů a v případě nešíření potlačil informaci o výpadku.



Obr. 4.4: Dohledový systém FIT

Díky použití multimediální knihovny jsou možnosti využití opravdu široké, zde jsou rozepsány pouze návrhy dvou z nich. Projekt by měl být teoreticky použitelný kdykoliv, kdy je cílem detekovat a analyzovat multimediální streamy, čili pro ISP, televizní a rádio společnosti, apod.

5 Závěr

Cílem této bakalářské práce bylo zrealizovat software pro detekování a analýzu multimediálních proudů dat přenášených po počítačové síti, což bylo splněno. Před započtením realizace bakalářské práce jsem prostudoval literaturu a poznatky z ní sepsal do kapitoly 4, provedl analýzu problému a navrhl jeho řešení v kapitole 12.Výsledná implementace je v kapitole následující, je otestována výkonnost a předvedena dvě možná praktická využití projektu. Program dokáže zpracovávat televizní kanály dostupné přes pozemní digitální vysílání a byl otestován na stabilitu, kdy běžel na testovacím hardwaru v plném zatížení po dobu 7 dnů bez výpadku. Jsou zde uvedeny různé varianty rozšíření, kde některé z nich bych rád v budoucnu sám realizoval a díky otevřenému zdrojovému kódu předpokládám, že jiné vytvoří někdo z komunity.

Během analýzy se ukázalo, že modul detector je složitější než na první pohled vypadá a právě jeho realizace je jedno z možných pokračování této bakalářské práce. Bohužel není možné použít existující knihovny a musí být ještě výkonnější než modul harvester. Množství zpracovávaných dat může být znatelně větší a proto by bylo vhodné navrhnout hardwarovou akceleraci.

Dalším z možných rozšíření tohoto projektu je podpora hardwarové akcelerace dekódování multimediálních dat. Nyní se kompletní analýza provádí softwarově, což může být dosti neefektivní při nutnosti monitorovat opravdu velké množství streamů – řádově stovky megabitů za sekundu. Na trhu jsou k dispozici akcelerační čipy pro dekódování MPEG-2 či MPEG-4 videa a audia. Ovšem realizace kompletně vlastního hardwaru by byla značně náročná z důvodu nutnosti provádět vysokorychlostní přenosy mezi desítkami akceleračních čipů a řídícím serverem a kompletního návrhu hardwaru. Cenová dostupnost takového řešení by také nebyla nejlepší. Protože ale většina dnes dostupných grafických karet podporuje hardwarovou akceleraci dekódování videa, nabízí se možnost využít je. Použitá knihovna ffmpeg v současnosti podporuje VDPAU, což je technologie společnosti NVIDIA pro akcelerování výpočetně náročných operací.

Poslední z návrhů rozšíření je možnost detekovat statický obraz. I přesto, že dekódování streamu proběhne bez chyby, nemusí totiž výsledný obraz nic obsahovat. Pokud dojde k problému již při získávání zdrojových dat tak detekcí chyb při dekódování není možné tuto skutečnosti odhalit. Tento stav je možné řešit detekováním statického obrazu, čili takového, který se v čase takřka či vůbec nemění.

Závěrem bych rád doplnil, že realizace projektu byla zábavná a získal jsem mnoho nových zkušeností hlavně z kategorie dekódování multimediálních dat a jejich přenosu po síti. Díky společnosti SMART Comp. a.s. jsem mohl vyzkoušet i praktické využití tohoto projektu.

Literatura

[1] O *projektu*, Visual Connection, Praha, Česká republika, 2006, [cit. 2009-04-10], <http://www.nacevi.cz/projekt.htm>

[2] Miller, C. Kenneth: Multicast Networking & Applications, Londýn, Anglie, 1998, Addison-Wesley Professional, ISBN 02-013-0979-3

[3] John Watkinson: *The MPEG Handbook*, Focal Press, St. Louis, Missouri, USA, 2001 ISBN 02-405-1656-7

[4] *Cisco Visual Quality Experience, Systems*, San Jose, Kalifornie, USA, 2008, [cit. 2009-04-10] <http://www.cisco.com/en/US/prod/collateral/video/ps7191/ps7127/product_data_sheet0900aecd806c0bfb.html>

[5] Daniel Cardens o kolektiv Wikipedia: *MPEG-TS*, [cit. 2009-04-10] <http://en.wikipedia.org/wiki/MPEG-TS>

[6] *VB12 RUGGEDIZED BROADCAST IP-PROBE*, BRIDGE Technologies, Oslo, Norsko, 2008, [cit. 2009-04-10], <http://www.bridgetech.no/product_vb12.php>

[7] *VB280 CONTENT EXTRACTOR*, BRIDGE Technologies, Oslo, Norsko, 2008, [cit. 2009-04-10], <http://www.bridgetech.no/product_vb280.php>

[8] *Agama Analyzer*, Agama Technologies, Linköping, Švédsko, [cit. 2009-04-10], <http://www.agama.tv/analyzer.html>

[9] *Products*, Alnair, Praha, Česká republika, 2008, [cit. 2009-04-10], <http://nangu.tv/content/products>

[10] *ROSA Network and Video Service Management*, San Jose, Kalifornie, USA, 2008, [cit. 2009-04-10], <http://www.cisco.com/en/US/products/ps9128/>

[11] *Verimatrix Content Security Manager™*, Verimatrix, San Diego, Kalifornie, USA, 2007, [cit. 2009-04-10] <http://www.verimatrix.com/products/contentsecuritymanager.php>

[12] *Conax CAS7 DVB over IP*, Conax, Oslo, Norsko, 2008, [cit. 2009-04-10], <http://www.conax.com/en/products/dvboverip/>

Seznam příloh