TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Multimediální nástroje pro on-line a off-line training (Red Fox s.r.o.)

Multimedia Tools for On-line and Off-line Training (Red Fox s.r.o.)

Diplomová práce

Autor: **Bc. Luboš REMPLÍK**
Vedoucí práce: Ing. Igor KOPETSCHKE
Konzultant: Ing. Vojtěch WRNATA

V Liberci 21. 5. 2010

Zadání

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum 21. 5. 2010

Podpis

Poděkování

Tímto děkuji vedoucímu diplomové práce Ing. Igoru Kopetschkemu a konzultantovi Ing. Vojtěchu Wrnatovi za vedení, cenné rady a podnětné připomínky při tvorbě této práce.

Abstrakt

Společnost Red Fox s.r.o. vyvíjí e-learningový portál pro potřebu Škoda Auto a.s.. Jeho součástí se má stát rozhraní pro multimediální on-line (přenos live audio a video streamů) a off-line (záznam audio a video materiálů a následné přehrávání u klienta) training. Cílem práce je analyzovat současný stav aplikace, navrhnout technické řešení v závislosti na technologiích používaných ve Škoda Auto a.s. a následně toto řešení implementovat do existujícího portálu. Portál je postaven na J2EE, aplikačním serveru WebSphere a DB Oracle.

Klíčová slova: Multimediální proudy v počítačové síti, Java Media Framework (JMF), Competence Management Program (CMP).

Abstract

Red Fox s.r.o. organization is developing e-learning portal for Škoda Auto a. s. needs. The portal is going to have interface for on-line (the transmission of live audio and video streams) and off-line (record audio and video materials, and playback at the client's machine) trainning. The aim is to analyze the current state of the application, propose technical solutions based on technologies used in the Škoda Auto a.s. and then implement the solution into an existing portal. The portal is built on J2EE, WebSphere application server and Oracle DB.

Keywords: Multimedia streams in network, Java Media Framework (JMF), Competence Management Program (CMP).

Obsah

Slovník pojmů

B2B – Business to business je označení pro obchodní vztahy mezi obchodními společnostmi pro jejich potřeby, které neobsluhují konečné spotřebitele v masovém měřítku.

CMP, CMS – Competence Management Program (System) poskytuje nástroje pro organizaci, které vytvářejí prostředí, ve kterém se zaměstnanci vzdělávají, kvalifikují, motivují a pravidelně účastní aktivit ke zlepšování výkonnosti za účelem splnění cílů organizace.

LMS – Learning Management System je on-line nástroj, který umožňuje přístup k výukovým zdrojům.

J2EE – Java 2 Enterprise Edition je součást platformy Java určená pro vývoj a provoz podnikových aplikací a informačních systémů.

JMF – Java Media Framework je multiplatformní framework s podporu standardních multimediálních formátů

RTP – Real Time Protocol je protokol zajišťující podporu pro koncové multimediální přenosy v reálném čase.

ACL – Access Control List je způsob řízení oprávnění uživatelů pomocí seznamu operací povolených nad objektem.

ARO – Access Requested Object souvisí s ACL a je to objekt vyžadující přístup, obvykle uživatel.

ACO – Access Controlled Object souvisí s ACL a je to chráněný objekt.

Úvod

Internet je nedílnou součástí dnešního světa. Internetové aplikace pomáhají společnostem a organizacím dělat jejich práci efektivně. Kvalita, stabilita a efektivita samotné aplikace je velmi důležitá. I když se internetové přípojky stále vyvíjejí a v dnešní době téměř vymizely dial-up připojení, rychlost většiny linek je na syrový přenos multimediálních dat stále dost nízká. Navrhnout a realizovat část aplikace, která efektivně pracuje s multimediálními daty v internetové síti je proto jedna část kterou se práce zabývá.

Aplikace CMP do které má být vyvinut modul pro práci s multimediálními daty, je vyvíjena společností Red Fox s.r.o. pro portál společnosti Škoda Auto a.s.. Analýza současného stavu e-learningového portálu je jedna z dalších částí této práce.

Aplikační vrstva portálu Škoda Auto a.s. je postavena na platformě Java 2 Enterprise Edition (J2EE) a serveru WebSphere. Datová vrstva je reprezentována databázovým serverem Oracle. Práce by měla zhodnotit klady a zápory těchto technologií, vyhodnotit následující použití dalších Java balíčků (frameworku).

Jedním z cílů práce je navrhnout a vzorově implementovat nástroj pro multimediální on-line a off-line streaming s ohledem na technologie použité ve Škoda Auto a.s..

1 Analýza současného stavu e-learningového portálu ve Škoda Auto a.s.

Současný systém CMP – Competence Management Program je on-line aplikace dostupná na portálu Škoda Auto a.s., která slouží k administraci vzdělávacích akcí a k podpoře vzdělávání obchodní a servisní sítě.

Aplikace je typu business to business (B2B), jak je uvedeno v [6], B2B je označení pro obchodní vztahy mezi obchodními společnostmi, pro jejich potřeby, které neobsluhují konečné spotřebitele v masovém měřítku. CMP je tedy určeno pro obchodní partnery prodávající produkty Škoda Auto a.s..

1.1 Competence Management Program

Competence Management Program nebo také Competency Management Systém (CMS) je nástupcem Learning Management Systemu (LMS). LMS je typický on-line nástroj, který umožňuje přístup k výukovým zdrojům. Competency Management systémy mají tendenci mít větší, komplexnější přístup a zahrnují nástroje jako je řízení kvalifikace, skills-gap analýza (analýza nedostatku vědomostí), hodnocení úspěšnosti, jakož i schopnosti dalších analýz a profilování. CMS má sklon zaměřit se více na vytváření prostředí pro trvale udržitelnou kvalifikaci.

Competency Management systémy jsou založeny na vzdělávání dospělých a profesní analýzy úkolů, která identifikuje obchodní procesy ve společnosti a jejich rozdělení na úkoly. Tyto úkoly jsou to, co jednotlivec musí znát, aby dělal svou práci.

Konkrétní CMP systém společnosti Škoda Auto a.s. se snaží obsáhnout všechny výše zmíněné vlastnosti CMS systému. Jako jeden z mnoha nástrojů pro komplexnost systému bude použit i modul pro záznam, vysílání a přehrávání multimediálních dat vytvořený v rámci diplomové práce.

1.2 Požadavky na software klienta

Jelikož je CMP on-line aplikace, jeden ze samozřejmých požadavků je připojení k Internetu. Detailní fotografie a video tutoriály jsou náročné na datový tok, proto je doporučeno používat systém na lince 256kBit/s a rychlejší.

Java Runtime Environment (JRE) musí být nainstalovaná na daném operačním systému a prohlížeč musí podporovat applety.

Applet je programová komponenta běžící v rámci jiného softwaru (webového prohlížeče), typicky orientována na plnění konkrétní funkce v daném kontextu. Applet je spouštěn na straně klienta, typicky v prostředí webového prohlížeče.

Systém je vyvíjen pro OS Microsoft Windows 2000 a novější a prohlížeč Internet Explorer 6.0 a novější.

1.3 Autentizace a Role v CMP

Systém je schopen řízení přístupu – určovat role uživateli. Skupina uživatelů, či jednotlivý uživatel má určeno do jaké sekce, či podsekce má přístup.

Jde o řízení pomocí Access Control List (ACL), kde jsou určeny dva pohledy: objekty které vyžadují přístup (ARO), nejčastěji uživatelé, a objekty které jsou pod dohledem, ke kterým je vyžadován přístup (ACO). Princip spočívá v udělení operací (čtení, editace, mazání) uživatelům (ARO), které můžou být vykonány nad kontrolovanými objekty (ACO).

Existuje několik druhů uživatelů (objektů) v systému. Základní dva druhy jsou uvedeny níže.

Koordinátor vzdělávání – uživatel s rolí koordinátora vzdělávání má, kromě práv účastníka, možnost administrovat všechny přiřazené účastníky své skupiny. Tzn.. přihlašovat je na vzdělávací akce, schvalovat jejich přihlášky, žádat o omluvy ze vzdělávacích akcí, přesouvat je na jiné termíny a sledovat výsledky testů účastníků tříděné podle různých kritérií.

Účastník – pracovník obchodní sítě Škoda, který má přístupová práva na Škoda Portál a do aplikace CMP. Všichni účastníci mají v rámci aplikace možnost přihlašovat se na vzdělávací akce na termíny i mezi zájemce, absolvovat on-line testy a pracovat s výukovými manuály a programy.

1.4 Části a ovládání CMP

Competence Management Program portálu Škoda Auto a.s. je rozdělen do přehledných kategorií a podkategorií. Mezi základní části aplikace patří administrativa, testování, vzdělávání, správa vzděláván a sys admin.

Do části administrativa spadají osobní karta, ve které jsou uvedeny osobní údaje účastníka, přiřazené profese, které korespondují s jeho profesní specializací a funkčním zařazením, seznam testů a e-learningových kurzů, které účastník absolvoval, seznam vzdělávacích akcí a kvalifikačních cest, na které je přihlášen, které absolvoval, ze kterých se omluvil nebo neomluvil, přehled vzdělávacích akcí, seznam vzdělávacích akcí, které korespondují s profesí účastníka a na které se může přihlásit a přehled kvalifikačních cest.

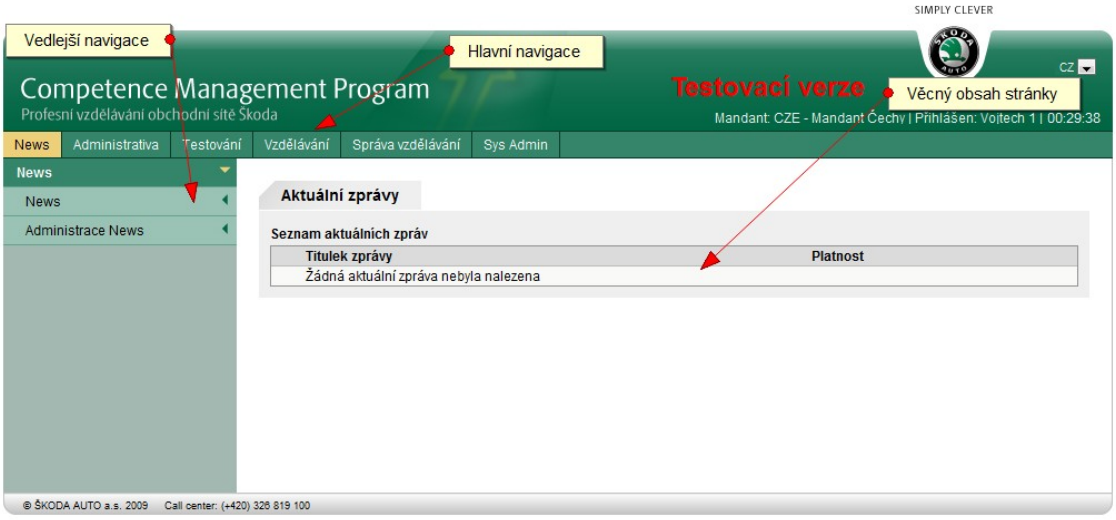
V části testování účastník nalezne testy rozdělené na cvičné testy, testy ke splnění kvalifikačních předpokladů a jednorázové testy.

V části vzdělávání jsou zobrazeny produktové manuály – launch & training guide, které slouží k zobrazení souhrnných informací o vozech jednotlivých modelových řad značky Škoda Auto a.s.. Dále zde mohou být zobrazeny i podrobné informace o jednotlivých modelech a jejich konkurenci v daných segmentech.

V části správa vzdělávání jsou zobrazeny informace o vybraných aplikacích používaných ve Škoda Auto a.s. v podobě interaktivních manuálů.

V části sys admin jsou k dispozici vybrané pojmy používané v automobilovém průmyslu, případně další informace technického charakteru.

Ovládání aplikace uvedený na obrázku č. 1 je intuitivní. Hlavní navigace se nachází horizontálně v hlavičce stránky pod logem CMP. Druhá navigace, která zobrazuje podkategorie hlavní navigace se nachází v levé dolní části stránky. Obsah (testy, tutoriály, a další moduly) jsou v pravé dolní části stránky.



Obr. 1: Competence Management Program – ovládání aplikace

Rozložení stránky vychází z testů uživatelské přístupnosti provedené společností Red Fox s.r.o..

1.5 Současný systém přehrávání multimédií

Nyní je použito pro přehrání videa Adobe Flash a s ním spjaté Flash video – tento formát je v současnosti jedním z nejpoužívanějších.

Jak je uvedeno v [4], technicky se vlastně jedná o variantu MPEG4 formátu H.264. Původně byl tento formát určen pro nahrazení zastaralého MPEG2 na DVD discích. Jeho výhodou byla podpora vysokých rozlišení a výborné kompresní vlastnosti, při zachování vysoké kvality obrazu. Konkrétně se s ním počítalo při přechodu filmového průmyslu na Blu-ray a HD-DVD disky. Dnes je H.264, vedle kodeků MPEG2 a VC-1, používán u některých titulů s vysokým rozlišením.

Flash video je variantou dosti osekanou. K účelům streamování není potřeba tak vysoké rozlišení a ani tak vysoký bitrate jako u vysokokapacitních disků. Daleko větší důraz je kladen na vysokou kompresi při nízkých datových tocích a možnost přehrávání na co nejširším spektru operačních systémů. Flash Video je možné přehrávat ve slušné kvalitě a při běžném rozlišení 320x240 i na linkách pomalejších, než je 1Mbit.

1.6 Flash Streaming a požadavky na práci s multimediálními daty

Jeden z požadavků na systém, kvůli němuž nelze použít současný Flash video systém, je on-line přenos streamu. U Flash videa je obtížné přehrát video, které nemá fyzicky k dispozici.

Výhodou Flashe je bezpochyby fakt, že přes 90% počítačů podporuje přehrávání Flash aplikací, stejně tak Flash podporuje mnoho mobilních telefonů na platformě Symbian.

Z pohledu streamingu se Flash většinou používá na přehrávání hotových videí (video on-demand) díky poměrně jednoduché implementaci do prohlížeče a kompatibilitě napříč operačními systémy – to je hlavní důvod proč použít Flash na rozdíl od ostatních služeb (Windows Media Playeru, Real Time Player). Zdroj [3] uvádí, že před nástupem Silverlightu bylo složité vkládat do webových stánek přehrávače na bázi Windows Media Player, zejména na Linuxu a MacOS. Tento fakt zcela řeší použití Silverlightu.

Flash Video nepodporuje přímou komunikaci mezi dvěma flash playery, tzn. nelze vysílat například z web kamery, či video soubor z klientského počítače.

Existuje řešení streamingu videí v reálném čase a to pomocí Flash Media Serveru, který kromě bohatých funkcí pro interaktivní streaming, umožňuje efektivní enkódování videa spolu s robustní ochranu obsahu a správy práv.

Nutnost pořízení serveru pro streaming může odradit potenciální uživatele.

1.7 Vývoj pro CMP

Competence Management Program (CMP) je vyvíjeno skupinou programátorů společnosti RedFox s.r.o.. Při implementaci nového modulu do systému jsou na výběr dva způsoby – vytvořit samostatný kurz a potom pomocí správce kurzů importovat komprimovaný soubor zip, nebo naprogramovat část aplikace (modul) do současného systému vyvíjeného v Javě.

Kurz je možno vytvořit pouze v rámci sekce vzdělávání. Koordinátor má oprávnění nahrát soubor s kurzem v sekci správce vzdělávání, interní kurzy. Při vytváření kurzu je nutné specifikovat základní údaje jako jsou jazyk, název, popis, typ kurzu, status, určení vedoucího projektu a nastavení parametrů okna. V rámci vytvoření kurzu je nahrán komprimovaný soubor zip, který obsahuje HTML soubory, obrázky a další části kurzu. Kurz je vytvořen pomocí HTML nebo je použita Flash aplikace.

Systém CMP je vyvíjen v Javě, proto pokud potřebujeme vyvinout modul pro aplikaci, je logické jej naprogramovat. Tímto způsobem můžeme implementovat modul kdekoli v aplikaci, ne jenom v sekci vzdělávání. Žádoucí je psát s dostatečnou abstrakcí a řádně vést dokumentaci.

Modul vyvíjený v rámci diplomové práce může být implementován obojím způsobem. Pokud Škoda Auto a.s. nebude chtít z jakéhokoli důvodu zasahovat do již hotového CMP portálu, je zcela jednoduché spustit hotový modul jako HMTL kurz. Využije se k tomu tagu *applet* jeho parametrů pro import zdrojového kódu a JMF knihoven potřebných k jeho funkčnosti. Ukázka HTML kódu je níže:

```
<applet name="StreamApplet"
        code="Stream.class"
        width="640"
        height="480"
        archive="include/jmf.jar, include/mediaplayer.jar,
include/multiplayer.jar"></applet>
```

Name a *code* jsou dle W3C vyžadované parametry. V případě JMF aplikace je potřeba importovat knihovny pomocí parametru *archive*. *Width* a *height* parametry jsou zde pro zobrazení bez posuvníků.

Existuje modernější *object* tag, který je používán pro použití objektů jako jsou obrázky, audio, video, ale také Java applety a Flash. *Object* má tendenci nahradit specifičtější tagy *img* a *applet*. Ačkoli ještě není plně podporován, respektive ne všemi webovými prohlížeči, je v některých případech vhodné *object* tag použít.

Rozhodnutí zda pro Java applet použít *applet*, či *object* tag závisí na několika předpokladech. Za prvé musí být bráno v potaz jestli Java applet bude běžet na Internetu, či Intranetu a za druhé v jakých webových prohlížečích bude spouštěn. Pokud bude spouštěn z Internetu, je z důvodu velké škály webových prohlížečů použitých na internetu, lepší použít starší tag *applet*. V případě Intranetu a použití webového prohlížeče Internet Explorer, je doporučeno použít tag *object*.

Příklad použití tagu *object*:

```
<OBJECT
  classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
  width="640" height="480">
  <PARAM name="code" value="Stream.class">
</OBJECT>
```

Atribut c*lassid* definuje, kterou verzi Java pluginu použít. Uvedená hodnota určuje Internetu Exploreru použití poslední nainstalované verze Java pluginu.

2 Návrh rozhraní pro multimediální on-line a off-line training.

Současný systém je koncipován systémem nezávislých softwarových balíčků, což je dobré pro vývoj a hlavně při aplikování vývojové verze do aktuálního e-learningového portálu.

Předpokladem pro snadnější integrace nové aplikace je návrh vhodného rozhraní, které by mělo ošetřovat akce jako je příjem streamu, vysílání streamu, přehrání videa, atd. Jak z definice rozhraní vyplývá, bude tím zaručena implementace všech potřebných metod.

Základní rozhraní je uvedeno níže

```
import java.awt.Container;
import java.net.URL;
public interface StreamInterface {
      public void play(URL url, Container parent);
      public void transmitStream(String [] args);
}
```

Pro lepší abstrakci a následnou implementaci je v aplikaci použit koncept factory tříd.

2.1 Softwarový koncept factory třídy

Návrhový vzor factory poskytuje způsob, jak zapouzdřit skupiny jednotlivých tříd, které mají společné téma. Při běžném použití klientský software vytvoří konkrétní implementaci abstraktní factory třídy a pak využívá generické rozhraní k vytvoření konkrétních objektů, které jsou součástí tématu. Programátor neví (nebo ho nezajímá), jak jsou realizovány konkrétní metody tříd, které implementují factory třídu, protože používá pouze generické rozhraní. Tento vzor odděluje údaje o provedení souboru objektů od jejich obecného užívání.

Factory třídy se běžně používají v toolkinech a frameworcích, kde daný framework potřebuje vytvořit objekt typu, který může deklarován v podtřídě dané aplikace, nebo kde třída jedné větve často vyžaduje vytvoření objektů z větve druhé

Factory metody jsou použity ve vývoji aplikací řízené testováním, umožňují nám vložit třídy pod testovací verzi softwaru a naopak.

V rámci vývoje modulu pro CMP je použita Factory třída *StreamClassFactory*, která rozhoduje zda je vrácena instance třídy pro off-line streaming nebo instance třídy pro on-line streaming. Následuje ukázka factory třídy, která je použita ve vytvořeném modulu CMP systému.

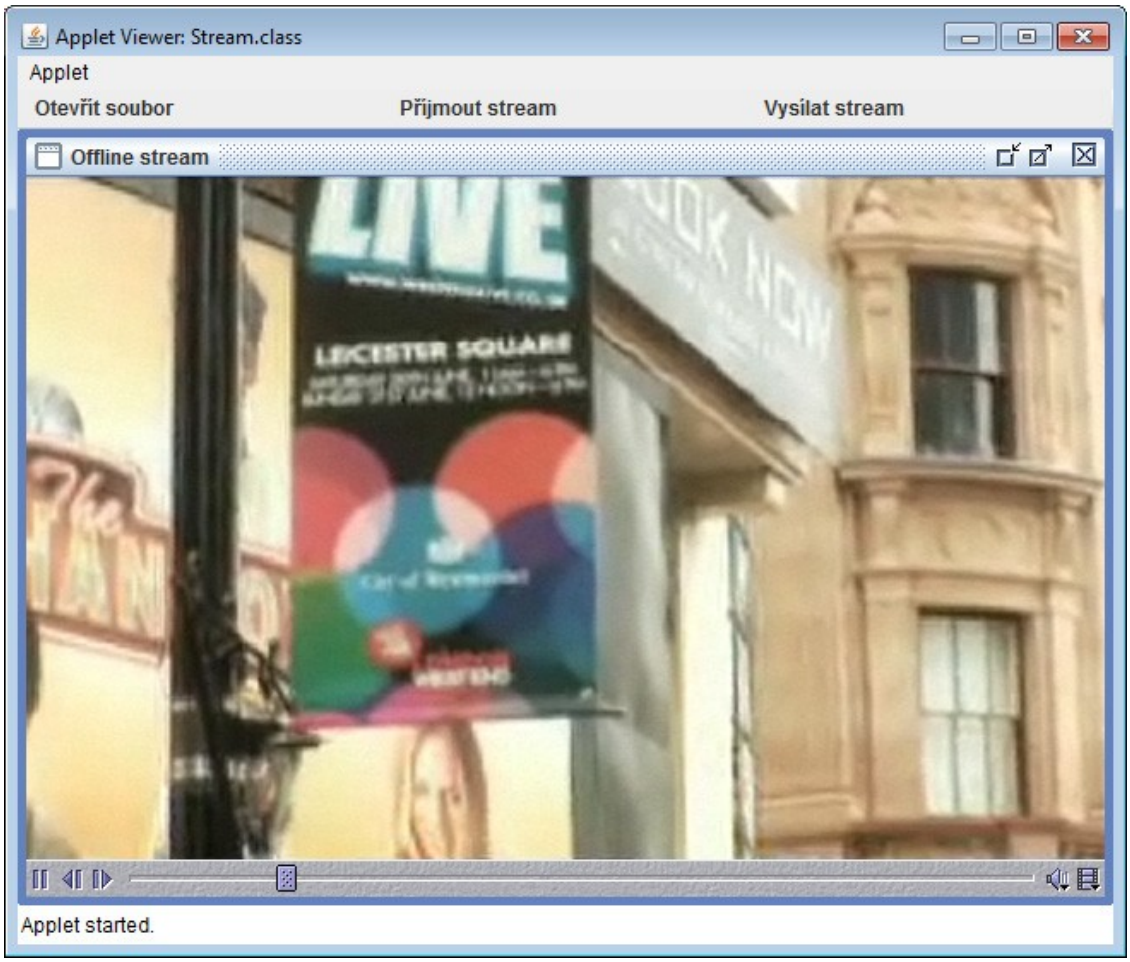
```
public class StreamClassFactory {
    public static final int OFF_LINE_STREAM = 1;
    public static final int ON_LINE_STREAM = 2;

    public static StreamInterface getInstance(int type) {
        if (type == OFF_LINE_STREAM) {
            return new OfflineStream() ;
        } else {
            return new OnlineStream();
        }
    }
}
```

Metoda *getInstance* musí být statická, protože ze třídy *StreamClassFactory* nidky nevytváříme instanci třídy. Není potřeba vytvářet objekt, tato medota, jak je výše uvedeno, slouží pouze k určení použití *OnlineStream*, či *OfflineStream* třídy a jejich metod.

2.2 Vytvoření uživatelského rozhraní a jeho interakce

I když to není podstatou práce rád bych zde zmínil jak je vytvořeno uživatelské rozhraní (obrázek č. 2) a jak aplikace reaguje na akce uživatele.



Obr. 2: Přehrání souboru - ukázka instance playeru

Stream applet má 4 základní metody, který musí mít každý applet, a to init, start, stop, destroy. Ve funkci init je volána metoda *createGUI()*, která vytvoří uživatelské rozhraní ze swing component. Tato metoda nedělá nic jiného než přidává základní prvky menu. Například vytvoření položky „Otevřít soubor“ obstarává následující část kódu

```
menuItem = new JMenuItem(ACTION_RECEIVE_STREAM);
menuItem.addActionListener(this);
menuBar.add(menuItem);
```

MenuItem a *menuBar* jsou swingovské komponenty, *addActionListener* je metoda přidávající položku mezi komponenty, které mohou vyvolat přerušení uživatelem. Pokud bude tlačítko stisknuto, zavolá se metoda *actionPerformed*, která je implementovaná z rozhraní *ActionListener*. V této metodě vyhodnotíme, které z tlačítek bylo stisknuto. Jedna z podmínek vypadá takto:

```
if (ACTION_OPEN_FILE.contentEquals(e.getActionCommand())) {}
```

Jak je vidět, je zde použito konstanty *ACTION_OPEN_FILE*. Ve většině případů je lepší používat konstanty z důvodu případné budoucí změny popisku tlačítka – nemusím měnit popisek v celé aplikaci, ale pouze v části deklarace proměnných. Po splnění podmínky je již konán funkční kód, v tomto případě kód pro výpis souborů a následného přehrání multimediálního souboru.

Třída *StreamApplet* také implementuje rozhraní *ListSelectionListener*, které nás nutí implementovat metodu *valueChanged*, která se volá při změně prvku komponenty *Jlist*. V této komponentě je jak v případě otevření souboru, tak v případě vysílání streamu, seznam souborů, kde jsou uloženy soubory dostupné na serveru a s kterými je možno pracovat k přehrání nebo k vysílání.

2.3 Off-line přehrání multimediálních dat

Pro off-line přehrávání multimediálních dat, tzn. audia, videa, které je uloženo na serveru, nikoli na lokálním disku klienta je použita metoda *play* třídy *OfflineStream*, která přehraje multimediální soubor s předaný v parametru.

První parametr metody musí být ve tvaru URL, který specifikuje RFC2396. Tvar URL je *<schéma>://<autorita><cesta>?<dotaz>#<fragment>*. Druhý parametr je *container* prvku na který se přidá přehrávač v *JMFramu*.

Ze zadané proměnné *url* je možné vytvořit *Player*, což je objekt, který umožnuje vykreslení, nikoli ovládání média.

```
Player player = null;
try {
      player = Manager.createPlayer(url);
} catch (NoPlayerException e) {
      System.out.println("Error: " + e);
} catch (IOException e) {
      e.printStackTrace();
}
```

Vytovřený *player* je umístěn *na JMFrame*, kterému předáme objekt playeru a titulek. *JMFrame* se postará o vykreslení základních ovládacích prvků (play, stop, pause), prvků pro maximalizaci, či minimalizaci okna přehrávače a vypíše titulek do záhlaví okna.

Pokud je vytvořený player, vytvoříme objekt zmíněného *JMFramu*, nastavíme mu potřebné vlastnosti, jako maximalizaci okna, a přidáme ho na container předaný v parametru.

```
if (player != null) {
    JMFrame jmframe = new JMFrame(player, "Offline stream");
    try {
        jmframe.setMaximum(true);
    } catch (PropertyVetoException e) {
        e.printStackTrace();
    }
    parent.add(jmframe);
}
```

Jak je vidět, vytvoření přehrávače bylo díky hotovým knihovnám Java Media Frameworku celkem snadné.

2.4 Příjem streamu

Je realizován v třídě *OnlineStream*, která má stejně jako třída *OfflineStream* metodu *play* pro přehrávání videa, či audia. Realizace metody je podobná, ale podoba medoty, vytvářející instanci třídy *StreamClassFactory*, programátora nezajímá.

Na akci stisku tlačítka „Příjmout stream“ je volaná statická metoda *getInstance* třídy *StreamClassFactory*, která jako parametr má druh streamu (on-line, off-line) a díky tomu vrátí instanci správné třídy *OnlineStream*. Z vytvořené instace třídy již voláme metodu *play*, které předáme *url* vysílaného streamu jako první parametr a *container* na který bude vykreslen přehrávač jako parametr druhý.

```
si = StreamClassFactory.getInstance(
    StreamClassFactory.ON_LINE_STREAM
);
si.play(url, this);
```

Na rozdíl od metody *play* třídy *OfflineStream*, musí být pro příjem vysílaného streamu předán místo *URL* objektu řetězec ve tvaru *<protokol>://<adresa>:<port>/<typ>*, kde jako protokol je v aplikaci použit Real time transport protocol (RTP), adresa, port na kterém je medium vysíláno a typ vysílaného média. Z předaného řetězce se vytvoří *MediaLocator*, což je objekt specifikující lokaci vysílaného média. *MediaLocator* je podobný *URL*. *URL* může být získáno *z MediaLocatoru* a naopak.

```
MediaLocator loc = null;
try {
      loc = new MediaLocator("rtp://147.230.155.23:2222/video");
} catch (Exception e1) {
      e1.printStackTrace();
}
```

Z vytvořeného objektu *MediaLocatoru* je již možné, stejně jako při vytváření playeru v *OfflineStreamu*, pomocí metody *createPlayer* třídy *Manager* vytvořit objekt playeru.

```
Player player = null;
try {
      player = Manager.createPlayer(loc);
} catch (NoPlayerException e2) {
      System.out.println("Error: " + e2);
} catch (IOException e2) {
      e2.printStackTrace();
}
```

Vytvoření objektu *JMFrame*, nastavení potřebných vlastností a vykreslení přehrávače na *contentPane* appletu

```
if (player != null) {
	JMFrame jmframe = new JMFrame(player, "title");
	try {
		jmframe.setMaximum(true);
	} catch (PropertyVetoException e) {
		e.printStackTrace();
	}
	panel.add(jmframe);
}
```

Příjem streamu je realizovaný. Možnosti vylepšení je možno vidět v následujících kapitolách.

2.5 Vysílání streamu

Je zřejmé, že vysílání streamu je možné pouze u typu on-line streamingu. Ale protože třída *OfflineStream* implementuje rozhranní *StreamInterface*, musí implementovat metodu *transmitStream*, ta ovšem zůstane prázdná. Není zde žádný ovládací prvek k vysílání off-line streamu, proto se není potřeba obávat k chybě v programu.

Při stisknutí tlačítka „Vysílat stream“ je volána funkce *actionPerformed* třídy *Stream*, kde po vyhodnocení události stisku tlačítka je vytvořená url souboru pro vysílání a pokud je uspěšně vytvořena, volá se statická metoda *getInstance*, která vrátí instanci třídy *OnlineStream*. Na vytvořeném objektu se zavolá metoda *transmitStream*, kde jako první parametr je vytvořená url, pak IP adresa a port, na které se stream bude vysílat. Je zde také předán čtvrtý parametr komponenta, který se předá prvku *contentPane,* na který bude vykresleno ovládání vysílání.

```
try {
	url = new URL("http://ref.nti.tul.cz:9080/cmp/courses/155/files/
soubor1.mpg");
} catch (MalformedURLException e1) {
	e1.printStackTrace();
}
if(url != null) {
	si = StreamClassFactory.getInstance(
		StreamClassFactory.ON_LINE_STREAM
	);
	si.transmitStream(
		url, "147.230.155.23", "2222", getContentPane()
	);
}
```

Pro vysílání streamu je použita třída *VideoTransmit*, která při její konstrukci zpracová adresu vysílaného souboru (videa) a vysílá stream na požadované adrese a portu.

```
VideoTransmit vt = new VideoTransmit(
	new MediaLocator(url),
	ipAddress,
	port
);
```

Vytvořená instance třídy *VideoTransmit* umožňuje základní metody pro vysílání, jako je *start* a *stop*.

```
String result = vt.start();
```

Pří startu vysílání je vytvořen *procesor* pro konkrétní media lokátor, ten je rozdělen na jednotlivé stopy. Pokud je nalezena video stopa, tak je transformována do formátu JPEG/RTP a připravována k vysílání.

Pokud je *procesor* úspěšně vytvořen, třída *createTransmittter* se pokusí vytvořit media lokátor pro RTP data kontejner a vytvořit relaci pro vysílání procesoru na specifické IP adrese a portu. Výsledný media lokátor je ve tvaru *rtp://<ip_adresa>:<port>/video* (například *rtp://147.230.155.33:2222/video*).

Pokud je start neúspěšný, je vrácený výsledek s chybami.

```
if (result != null) {
    System.err.println("Error : " + result);
    System.exit(0);
}
```

Stream je vysílán a je potřeba ošetřit čekání na akci uživatele pro ukončení vysílání. Pro tento druh operace se jeví jako vhodná komponenta dialogové okno s informací o přerušení vysílání. V aplikaci je tedy použita swing componenta *JoptionPane* a její metoda *showMessageDialog*. Jako parametry se předává rodičovská komponenta, zpráva, titulek okna a typ.

```
JoptionPane.showMessageDialog(
      getContentPane(),
      "Chcete přerušit vysílání?",
      "Stop",
      3
);
```

Jakmile dojde k interakci uživatele – příkazu k ukončení vysílání, je volána medota *stop*, která nejdříve zastaví a ukončí práci s *Processorem* a poté ukončí *RTP Transmiter*.

```
vt.stop();
```

2.5.1 Třída VideoTransmit

Třída *VideoTransmit* je jednoduchý obal, který je naprogramován pro zpracování video vstupu z jakéhokoli zdroje a vysílání videa do cílového počítače nebo sítě ve formátu JPEG.

JMF API je použito ke čtení zdroje a jeho převedení na data tvořená z JPEG paketů. RTP API implementace v JMF umí přenášet video pomocí protokolu RTP.

Třída *VideoTransmit* má tři parametry v konstruktoru - zdroj, cílovou IP adresu a číslo cílového portu.

Zdrojem může být:

- lokální soubor určený cestou, např.: "*file:/C:/media/speech.mov*",
- zdroj v síťi "*http://mediacentral.com/speech.avi*",

- nebo zdroj kamery a jiných externích zařízení, specifikované ve Windows jako "*vfw://0*" a v OS Solaris jako "*sunvideo://0/1/JPEG*"

  Zadaná IP adresa je adresa cílového počítače (příjemce). Pokud je žádoucí aby všechny počítače v podsíti přijímaly vysílaný signál, pak lze použít číslo 255 jako poslední číslo IP adresy.

Jako číslo portu může být použito jakékoli číslo, které nevyužívá žádná jiná služba v cílovém počítači. Číslo portu musí být sudé.

2.6 Důležité třídy JMF použité v aplikaci

V aplikaci je použito několik tříd, u kterých je vhodné popsat funkcionalitu, vhodnost použití a některé jejich metody.

2.6.1 Manager

Manager je třída obstarávající přístup k systémově závislým zdrojům jako jsou *Playery, DataSource, Processory, DataSingy*, systémové *TimeBase* a nástroje *DataSources* pro klonování a slučování.

Manager je schopný vytvořit *Player* z *URL*, *MediaLocatoru*, či *DataSource*.

Vytvoření *Playeru* vyžaduje:

- Získat připojený *DataSource pro* daný protokol
- Získat *Player* pro content-type určený *DataSource*
- Připojit *DataSource* na *Player* použitím *setSource* metody

2.6.2 Player

Player je *MediaHandler* pro zobrazování a řízení časově závislých multimediálních dat. *Player* rozšiřuje *Controller* rozhraní a poskytuje metody pro získání AWT komponent, ovládací prvky pro ovládání multimédií a další prvky pro řízení multimediálních dat.

2.6.3 MediaLocator

MediaLocator popisuje umístění multimediálního obsahu. *MediaLocator* je hodně podobný *URL*. *URL* může být získáno z *MediaLocatoru*, a *MediaLocator* muže být vytvořen z *URL*. Na rozdíl od *URL*, *MediaLocator* může být instancí bez instalace *URLStreamHandleru* v systému.

Abstraktní třída *URLStreamHandler* je základní super třídou pro ovládání stream protokolů. Umí vytvořit připojení pro jednotlivé typy protokolů jako jsou http, ftp a další.

2.6.4 Processor

Rozhraní *Procesoru* definuje modul pro zpracování a řízení časově závislých multimediálních dat. *Processor* rozšiřuje rozhraní *Playeru*. Na rozdíl od *Playeru*, který zpracovává data jako černá skříňka a vykresluje data pouze pro daný cíl, *Processor* poskytuje rozhraní, které umožňuje kontrolu nad multimediálními daty, jejich zpracování a vysílání datového streamu.

Zpracování *Processorem* je rozděleno do tří etap:

- Demultiplexing – multimediální proud je nejprve rozložen do několika stop, které mohou být zpracovány individuálně
- Data transcoding – každá stopa může být transformována z jednoho formátu na druhý
- Multiplexing - rozdělené stopy mohou být seskupeny do multimediálního proudu určitého datového typu

Oba procesy kódování a multiplexování jsou programovatelné

2.6.5 Data source

DataSource poskytuje abstrakci pro protokoly multimediálních proudů tím, že je poskytuje jako datový zdroj. Řídí životní cyklus multimediálního proudu a poskytuje jednoduchý komunikační protokol.

DataSource je schopná poskytnout operace, které nejsou součástí definice třídy. Například pro multimediální datový zdroj poskytuje operace pro možnost přístupu na jakoukoli časovou pozici media. Z uvedeného příkladu je zřejmé, že některé operace jsou závislé na typu média.

Je-li potřeba zjistit objekty a druhy operací nad objekty, lze použít metodu *getControls* nebo *getControl* pro daný objekt.

2.6.6 Data Sink

DataSink je základní rozhraní pro objekty, které pracují s obsahem medii určené *DataSourcem a* poskytuje medium určitému cíli. Cíl je specifikován *MediaLocatorem*. Příkladem *DataSinku* je objekt, který ukládá medium do souboru.

DataSink je vytvořen v souladu s factory mechanismem k vytvoření *Playeru*.

2.7 Možnosti vylepšení vytvořeného modulu

V případě větších video/audio konferencí by bylo vhodné umožnit příjem streamu více stanicemi, nebo-li broadcasting. Současný systém umožňuje přijímat pouze tolik signálů, kolik je vysíláno. V tomto případě by se musela řešit optimalizace datového přenosu v síti.

Vysílání streamu je provedeno z fyzicky existujícího souboru. Jedním z předpokládaných vylepšení bude možnost vysílat stream z externího zařízení, jako je mikrofón, či web kamera.

Jakmile je audio/video odvysílané, opožděný účastník není schopen přehrát konferenci zpětně. Ukládání vysílaného streamu na klientskou, či v lepším případě na serverovou stanici může být jedno z dalších vylepšení. Uživatel by byl schopen přehrávat konferenci se zpožděním, nebo dokonce po dokončení konference přehrát záznam znovu.

Současný systém vyžaduje zadání IP adresy a port vysílaného streamu. Pro zjednodušení práce navrhuji nadefinovat adresy a porty, na kterých bude prováděn streaming, nebo v lepším případě listovat všechny uživatele, či uživatele s daným oprávněním a jejich IP adresy. Seznam uživatelů by měl indikátor jestli daný stream právě vysílá.

2.7.1 Návrh appletu umožňující snímání

Applet umožňující snímání z externího zdroje, například z web kamery, může naprogramován a implementován následujícím postupem:

- vytvoření vstupního *DataSource* ze zdroje web kamery pomocí *MediaLocatoru*,
- konstrukce *Processoru* z *DataSource*, ve formátu specifikující video,
- získání výstupní *DataSource* z *Processoru*

- vytvoření *MediaLocatoru* pro ukládání médií do souboru a následné ukládání do objektu *DataSink.*

Stream videa je snímán z web kamery a uložen do souboru pro určitý časový interval. Tento postup je opakován dokud odesílatel neukončí akci.

2.8 Funkční schéma implementovaného modulu

Pro lepší představu je funkčnost přehrávání, vysílání a přijímání multimediálních dat znázorněna na obrázku č. 3.

Obr. 3: Schéma funkčnosti modulu

Horní dva klientské počítače představují použití třídy *OnlineStream* a její metody pro příjem a vysílání streamu. První klientský počítač snímá video signál připojené video kamery, zpracovává ho a upravuje do podoby vhodné pro vysílání. Následně vysílá video signál přes rtp protokol na adresu cílového klienta. Druhý počítač naslouchá na portu 2222, zpracovává vysílaný signál, vytváří přehrávač a spouští video.

Spodní část obrázku představuje použití třídy *OfflineStream,* která je schopná přehrát multimediální soubory uložené na serverovým počítači. Klientský počítač použije metodu play pro výběr souboru ze servrového počítače, vytvoření přehrávače a přehrání audio/video souboru.

3 Návrh přenosu a archivování audio a video streamů

E-learningový portál Škoda Auto a.s. je spuštěn na aplikačním serveru WebSphere a databázi Oracle. Většina aplikačních serverů je vyvíjena v programovacím jazyku Java, proto volba platformy Java Enterprise Edition (J2EE) je přirozenou volbou pro vývoj aplikace.

V předchozích letech studia Informačních technologií na Fakultě mechatroniky, informatiky a mezioborových studií, Technické univerzity v Liberci jsem se seznámil a dobu pracoval s Java Media Frameworkem (JMF) a tudíž pro aplikaci, která má umožnit práci se streamovaným audiem, či videem jsem zvolil zmíněný balíček. Porovnání, shrnutí výhod a nevýhod dalších technologií je uvedeno v dalších kapitolách.

3.1 Aplikační server a IBM WebSphere

Aplikační server tvoří vrstvu mezi operačním systémem a aplikacemi. Podobně jako operační systém poskytuje základní funkce programům (například pro přístup k souborovému systému, nebo ke správě procesů), poskytuje aplikační server často používané funkce enterprise aplikacím. Vytváří další vrstvu abstrakce, aby bylo psaní aplikací jednodušší. Příkladem takových funkcí mohou být podpora transakčního zpracování požadavků, persistence objektů do databáze, výměna zpráv mezi aplikacemi a další. Nabízí se samozřejmně otázka, co to vlastně je enterprise aplikce a jak se liší od běžné aplikace. Není to nic složitého, vlastně se jedná o běžnou aplikaci, na kterou jsou kladeny určité nároky co se týče spolehlivosti, dostupnosti, robustnosti, výkonnosti. [5]

IBM WebSphere Aplikační server (WAS) je určen pro řízení, provoz a integraci podnikových aplikací mezi více výpočetních stanic (architektura SOA) pomocí Java technologie. To zahrnuje jak run-time komponenty, tak nástroje pro vývoj aplikací, které běží na WAS.

Mezi další významné aplikační servery patří JBoss, JOnAS, Apache Geronimo, Apache Tomcat jako Open Source severy a BEA WebLogic, Sun Java System Application Server, Oracle AS jako komerční servery.

3.2 Oracle

Oracle je systém řízení báze dat, moderní multiplatformní databázový systém s velice pokročilými možnostmi zpracování dat, vysokým výkonem a snadnou škálovatelností.

Aktuální verzí je Oracle Database 11g. Tento systém podporuje nejen standardní relační dotazovací jazyk SQL podle normy SQL92, ale také proprietární firemní rozšíření Oracle (např. pro hierarchické dotazy), imperativní programovací jazyk PL/SQL rozšiřující možnosti vlastního SQL (v tomto jazyce je možné tvořit uložené procedury, uživatelské funkce, programové balíky a triggery), dále podporuje objektové databáze a databáze uložené v hierarchickém modelu dat (XML databáze, jazyk XSQL). Dále též obsahuje širokou paletu nástrojů pro podporu snadného nasazení na gridových sítích (písmeno g v označení verze je zkratkou "Growing to Grid"). Grid Computing podporovala i verze 10g (zde písmeno g značí pouze slovo Grid). [7]

3.3 Java 2 Platform, Enterprise Edition

Java 2 Platform, Enterprise Edition (J2EE) je široce používáná platforma pro programování na serverových stanicích v programovacím jazyce Java. Liší se od Java 2 Standard Edition (J2SE) v tom, že přidává knihovny, které poskytují funkce pro nasazení chybám tolerantního, distribuovaného, multi-vrstvého softwaru, založeného především na modulárních komponentách, běžícího na aplikačním serveru.

3.4 Java Applety

Applet je zvláštní typ programu Java, který umožňuje prohlížeči s podporou Javy stáhnout a spustit aplikaci z Internetu. Applet je typicky vložený do webové stránky a běží v rámci prohlížeče. Applet musí být podtřídou *java.applet.Applet* třídy, která poskytuje standardní rozhraní mezi appletem a prostředím prohlížeče.

Swing poskytuje zvláštní podtřídu *Applet* třídy s názvem *javax.swing.JApplet, JApplet* třída by měla být použita pro všechny typy appletů, které používají Swing komponenty k výstavbě jejich grafického uživatelského rozhraní (GUI).

3.4.1 Bezpečnostní omezení appletů

Java applety jsou všeobecně svým postojem k bezpečnosti nepříjemné pro většinu programátorů Javy. Schopnost appletu spuštění programu prohlížečem bez nutnosti stáhnutí a instalace na desktop je velice příjemná, ale přináší to rizika – applet jako nástroj pro tvůrce škodlivého softwaru. Naštěstí Java vývojáři vzali v potaz rizika a vytvořili bezpečnostní model, který chrání systém proti neoprávněným útokům. Omezení spočívá například v nemožnosti práce appletu s lokálními soubory a je realizováno pomocí tzv. sandboxu, nebo-li pískoviště.

Bezpečnostní omezení jsou různá pro jednotlivé prohlížeče. Některé prohlížeče mají velice přísný bezpečnostní model, který však lze zmírnit nastavením prohlížeče. Výchozí nastavení prohlížeče bývá s největší bezpečností, proto by měl programátor počítat vždy s maximálními restrikcemi. Internet Explorer verze 3 a vyšší umožňuje digitálně podepsaným třídám mít menší restrikce než ostatní třídy.

Přístup k souborům je jedna z nejvíce riskantních operací. Pokud by někdo byl schopen modifikovat soubory na systému, když běží applet, mohli by systém napadnout virem, nebo zničit důležitá data. Z tohoto důvodu není appletu povolen přístup k lokálním souborům a to ani ke čtení.

3.5 Multimediální proudy a Java Media Framework (JMF)

3.5.1 Multimediální proudy v počítačové síti

Úkoly multimediálních přenosů v síti jsou následující:

- konverze analogových dat na data digitální ve zdrojovém uzlu a jejich následná komprese;
- přenos toku multimediálních dat s potřebnou kvalitou služby (QoS), umožňující cílovému uzlu korektní příjem těchto dat;
- příjem, dekomprese a interpretace datového toku do multimediálního výstupu v cílovém uzlu.

3.5.2 Kompresní formáty

Komprese je nedílnou součástí multimediálních přenosů, protože soubory jsou velice objemné a ani širokopásmové přípojky nejsou většinou schopné je přenášet v „syrovém“ stavu. Kompresní algoritmus musí být odolný proti možným ztrátám některých paketů z datového proudu a nesmí být závislý na příjmu předchozího ani následného paketu. Součástí algoritmu by mělo být také určení povolené tolerance k možným ztrátám dat. Metody používající kompresní a dekompresní algoritmy se nazývají kodeky (codec, compressing – decompressing).

Výše uvedené kroky se snadněji provádějí u přenosů na vyžádání, než živých přenosů, protože u multimediálních přenosů probíhajících v reálném čase musí v reálném čase proběhnout komprese, vlastní přenos a na straně klienta i dekomprese.

Nejznámějšími kompresními formáty jsou:

- JPEG (Joint Photographic Experts Group) zahrnuje několik typů komprese digitálních barevných a černobílých obrazů (soubory s příponou .jpg). JPEG používá kompresní techniku s prostorovou redundancí DCT (Discrete Cosine Transformation), proto se hodí pro statická zobrazení.
- MPEG (Moving Picture Experts Group) zahrnuje skupinu mezinárodních norem kompresních technik pro digitální video a digitální audio určené pro specifické účely. MPEG video soubory (mají příponu .mpg) jsou na straně klienta interpretovány přehrávači (MPEG player nebo MPEG viewer), které jsou i volně šiřitelné.

3.5.3 Přenos multimediálních dat

U přenosů v reálném čase je nezbytně nutné zajistit rovnoměrnost toku paketů, což znamená, že časové zpoždění mezi jednotlivými pakety má být konstantní. Jestliže se časové odstupy mezi jednotlivými přijímanými pakety výrazně mění, aplikace v reálném čase může mít nekvalitní výstup nebo dokonce může dojít k jejímu selhání.

Na straně příjemce lze vhodně využít vyrovnávací paměti, kdy se mírně oddálí čas přehrávání paketů od času příjmu paketů, a to na základě časového odstupu jednotlivých paketů.

Připouští se přitom určitá tolerance pokud se jedná o ztráty paketů, vytvářejících datový proud. Multimediální data jsou značně redundantní z hlediska lidského vnímání zvuku a obrazu. Pokud tedy během přenosu proudu paketů dojde k přijatelným ztrátám, interpretovaná informace na přijímací straně se může uživateli zdát méně kvalitní, nicméně akceptovatelná. Pokud ale dojde k porušení časového sledu paketů, obraz se „roztrhá“ a zvuk ztrácí zcela srozumitelnost.

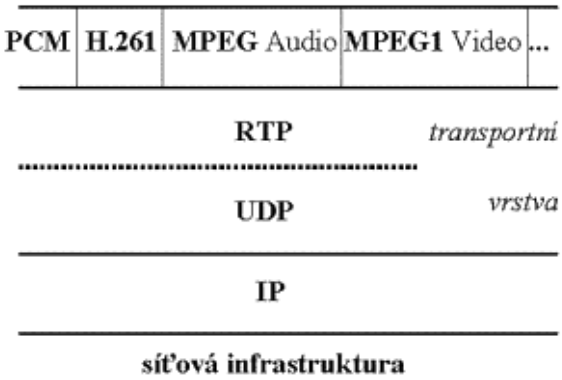
3.5.4 Transportní protokoly

TCP (Transmission Control Protocol), klasický spolehlivý transportní protokol architektury TCP/IP zajišťující doručení paketů prostřednictvím mechanismů potvrzování a opětovného vysílání, není vhodný pro přenosy v reálném čase.

Nejnevhodnější vlastností TCP v tomto případě je jeho zabudované řízení chyb: opětovný přenos ztraceného nebo chybného segmentu. Při přenosu v reálném čase je potřeba naopak ztracené nebo poškozené pakety ignorovat: jejich opětovný přenos je zbytečný, protože paket už je nepoužitelný. Vzhledem k dnešní nadbytečnosti audio a video signálů (i s kompresí) lze jednoduše ztracené pakety ignorovat.

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

Protokolová architektura pro aplikace v reálném čase s RTP je naznačena na obrázku č. 4.



Obr. 4: Protokolová architektura

3.5.5 Real-time Transport Protocol: RTP

Přenosový protokol v reálném čase (RTP, Real-time Transport Protocol) je protokol zajišťující podporu pro koncové multimediální přenosy v reálném čase. Nezaručuje doručení dat ani správné pořadí jednotlivých paketů, ale definuje jejich pořadová čísla, podle kterých mohou multimediální aplikace rozpoznat chybějící pakety. Zakládá se na synchronizaci časového přenosu a zjištění ztráty nebo nesprávného pořadí dat. RTP nejčastěji používá protokol UDP (čísla portů 5004, 5005, 6970), ale může využít i jiné protokoly.

RTP protokol byl navržen jak pro individuální tak skupinové přenosy, pro jednosměrný i obousměrný přenos. Je tedy použitelný pro aplikace videokonference i pro IP telefonii.

K multimediálnímu obsahu RTP připojuje záhlaví, které obsahuje pořadové číslo paketu pro zjištění ztrát nebo duplicity paketů a označení typu obsahu, tj. informaci o formátu multimediálního souboru, který tvoří obsah paketu (např. JPEG, G.722, H.261). Kódování obsahu se může změnit, pokud se má přizpůsobit rozdílům v šířce pásma. Dále RTP pakety obsahují indikaci začátku a konce rámce, identifikaci zdroje a synchronizaci pro detekci různého kolísání zpoždění v rámci daného toku a pro potřebnou kompenzaci tohoto kolísání při vlastním přehrávání obrazů a zvuků.

Formát RTP datagramu je jednoduchý a obecný, takže vyhovuje všem aplikacím pro přenos dat v reálném čase.

3.5.6 Komprese RTP

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

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

3.5.7 Jiné protokoly pro multimediální data

Multimédia na Internetu podporují i jiné, následující protokoly:

Real-time Transfer Protocol Audio Video Profiles (RTP/AVP, RFC 3551) – pro specifikaci kódování médií a dalších parametrů;

Real-Time Streaming Protocol (RTSP, RFC 2326) – protokol určený pro vzdálené řízení přehrávání médií;

Session Initiation Protocol (SIP) – protokol pro navázání relací mezi procesy, které běží na počítačích připojených k IP síti, pro telefonii, video, chat, hry apod.

Session Description Protocol (SDP, RFC 2327 a 3266) – protokol určený k popisu charakteristik (parametrů) multimediální relace (např. typ média);

Presence and Instant Messaging for SIP – protokol pro předávání zpráv v reálném čase na základě zjištění on-line přítomnosti (připojení k síti) komunikačních partnerů;

3.5.8 Možnosti Java Media Frameworku

Nabízí multiplatformní framework pro zobrazování dat reálného času. Java Media Player je postaven pro podporu standardních multimediálních formátů, včetně MPEG-1, MPEG-2, QuickTime, AVI, WAV, AU a MIDI.

Použitím JMF můžeme synchronizovat a poskytovat data z různých zdrojů.

Existující media přehrávače pro PC jsou hodně programově závislé pro výpočetně citlivé úlohy jako dekódování a přehrávání. JMF poskytuje abstrakci, která programátorovi schovává detaily implementace. Například jakákoli instance přehrávače může jednoduše vyžadovat schopnosti daného operačního systému pomocí intuitivních metod.

3.5.9 Výhody a nevýhody Javy (Java Media Frameworku)

Největší výhodou Javy je její přenositelnost. Pokud napíšete aplikaci v Javě, spustíte ji potom na jakékoliv platformě, pro kterou bylo vytvořeno JVM (Java Virtual Machine). Díky tomu odpadají problémy s portováním pro jinou platformu.

Vývoj aplikací je poměrně rychlý, protože Java už v základu obsahuje spoustu hotových řešení, které můžete při vývoji využít. A pokud neobsahuje v základu, existuje mnoho dalších knihoven, který lze použít. Jednou z nich je právě JMF. Tato abstrakce je velice výhodná, jak již je viděno například u vytvoření přehrávače - nemusíme řešit jakým způsobem operační systémy zpracovávají videa, vytvoříme přehrávač pomocí dané metody a o zbytek se postará JMF.

Jistou nevýhodou je podpora pouze několika nejpoužívanějších kodeků. Pro registraci externích kodeků lze použít nástroj JMF registry.

Nevýhodami mohou být pomalejší startování aplikace a větší nároky na operační paměť. Dřívěji byla rychlost náběhu aplikace až 10x pomalejší oproti konkurenčním programovacím jazykům. Dnes je tento rozdíl rapidně menší a s každou novou verzí Javy se rychlost vyrovnává s konkurencí.

3.5.10 Výhody a nevýhody JMF oproti Flash videím

Mezi výhody použití Java Media Frameworku (JMF) patří:

- JMF je schopno přenášet on-line i off-line stream
- Implementace jako Applet, tzn. není nutný zásah do sytému

Jako nevýhody jsou:

- Klient musí mít nainstalováno JMF na svém počítači, nebo musí být k appletu přibalena jako externí knihovna, což znamená donucení klienta stahovat několik kilobajtů dat navíc.
- Klient musí mít nainstalovaný JRE a pluginu pro prohlížeč

Závěr

Problematika multimediálních proudů v prostředí sítí je velice rozsáhlé a významné téma. Výběr správného kompresního formátu a protokolu pro přenos sítí umožňuje snížit hardwarové nároky na server a redukovat zatížení sítě.

Vzhledem k rychlostem a spolehlivosti dnešních sítí je díky tomu, že není potřeba potvrzování či posílání paketů znovu, a díky spolehlivosti s ohledem na pořadí a časového zpoždění paketů, Real-time Transport Protocol (RTP) skvělé řešení pro přenos dat v reálném čase.

Použití Java platformy, obzvláště J2EE edice s aplikačním serverem WebSphere, a knihovny JMF se ukázalo programově velice vhodné a rychlé řešení s možností multiplatformního použití, které umožňuje stavět rozsáhlé, ale přesto velice stabilní projekty. CMP aplikace společnosti Škoda Auto a.s. je zdárným příkladem rozsáhlého a stabilního projektu, do niž je relativně snadné implementovat další moduly postavené na stejné platformě.

Přestože JMF nepodporuje všechny audio/video formáty, vzhledem k univerzálnosti bych ho doporučil pro většinu projektů zabývajících se přenosem multimediálních proudů v síti.

I přes jisté bezpečnostní omezení appletu a nutnosti elektronického podpisu je Java applet vhodnou alternativou k současnému systému Flash videí. Velkou výhodou je možnost přímého vysílání streamu a možnost rozšíření na broadcasting.

Ve finální předkládané verzi appletu není řešeno vysílání streamu formou broadcastingu, ale je prezentováno pouze v teoretické rovině. Důvodem byla konkrétní poptávka ze strany Škoda Auto a.s. Kvůli nedořešeným bezpečnostním aspektům a současné existující konfiguraci firewallů v intranetu zadavatele.

Reference

[1] SULLIVAN, Sean C.; BROWN, Deanna; WINZELER, Loren. *Programming With the Java Media Framework*. Indianapolis : John Wiley & Sons, 1998. 384s.

[2] MACK, Steve. *Streaming Media Bible*. [s.l.] : Wiley, 2002. 850s.

[3] *Sun Microsystems* [online]. 2001 [cit. 2010-05-20]. Java Media Framework API. Dostupné z WWW: <http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/apidocs>.

[4] ČECH, Nikola. *Emag* [online]. 2007 [cit. 2010-05-20]. K čemu je Flash Video?. Dostupné z WWW: <http://www.emag.cz/k-cemu-je-flash-video>.

[5] VEČEŘA, Martin. Root.cz [online]. 18. 2. 2008 [cit. 2010-05-20]. *JBoss: Aplikační server*. Dostupné z WWW: <http://www.root.cz/clanky/jboss-aplikacni-server>.

[6] B2B. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 13. 9. 2005, 10:02, last modified on 26. 1. 2010, 21:49 [cit. 2010-05-20]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/B2B>.

[7] Oracle. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 7. 1. 2005, last modified on 28. 4. 2010, 09:58 [cit. 2010-05-20]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Oracle>.

Příloha A

CD příloha obsahuje všechny zdrojové kódy praktických příkladů a elektronickou formu diplomové práce ve formátu *.odt a *.pdf. Také je přiložen instalační soubor softwaru Acrobat Reader pro čtení PDF formátů.