UNIVERSIDADE SALVADOR – UNIFACS
MESTRADO ACADÊMICO EM SISTEMAS E COMPUTAÇÃO

FÁBIO SOBREIRA CORIOLANO FILHO

ESTRATÉGIAS PARA TRANSMISSÃO E RECEPÇÃO DE *STREAMING* DE VÍDEOS EM DISPOSITIVOS MÓVEIS

Salvador
2010

FÁBIO SOBREIRA CORIOLANO FILHO

ESTRATÉGIAS PARA TRANSMISSÃO E RECEPÇÃO DE *STREAMING* DE VÍDEOS EM DISPOSITIVOS MÓVEIS

Dissertação apresentada ao mestrado acadêmico em Sistemas e Computação da Universidade Salvador - UNIFACS, como requisito parcial obrigatório para obtenção do grau de Mestre em Sistemas e Computação

Orientador: Prof. Dr. Thomas de Araujo Buck

Salvador
2010

Ficha Catalográfica
(Elaborada pelo Sistema de Bibliotecas da Universidade Salvador - UNIFACS)

Coriolano Filho, Fábio Sobreira
Estratégias para transmissão e recepção de streaming de vídeos em dispositivos móveis / Fábio Sobreira Coriolano Filho. – Salvador, 2010.
96 p.: il.

Dissertação apresentada ao Curso de Mestrado Acadêmico em Sistemas e Computação da Universidade Salvador – UNIFACS, como requisito parcial para a obtenção do grau de Mestre.
Orientador: Prof. Dr. Thomas de Araújo Buck.

1 Sistemas de computação. 2. Interfaces – programação. I. Buck, Thomas de Araújo, orient. II. Universidade Salvador – Unifacs. III. Título.

CDD: 004.62

FÁBIO SOBREIRA CORIOLANO FILHO

ESTRATÉGIAS PARA TRANSMISSÃO E RECEPÇÃO DE *STREAMING* DE VÍDEOS EM DISPOSITIVOS MÓVEIS

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre em Sistemas e Computação, Universidade Salvador - UNIFACS, pela seguinte banca examinadora:

Thomas de Araújo Buck (Orientador) _____
Doutor em Informática pela Universität Tübingen, Alemanha
Universidade Salvador - UNIFACS

Antonio Lopes Apolinário Júnior_____
Doutor em Engenharia de Sistemas e computação pela Universidade Federal do Rio de Janeiro - UFRJ, Brasil
Universidade Federal da Bahia - UFBA

Jorge Alberto Prado de Campos _____
Ph.D. in Spatial Information Science, University of Maine, 2004, EUA
Universidade Salvador - UNIFACS

Salvador, 06 de outubro de 2010

AGRADECIMENTOS

Aos meus pais, Fábio e Maria Antônia, e ao meu irmão, Diego, pelo apoio, pela confiança e paciência.

À minha esposa, Luciana. Muito obrigado por estar ao meu lado em todos os momentos. Você foi peça fundamental para a conclusão dessa etapa de minha vida.

À minha avó, tios e primos, que sempre me incentivaram.

Aos amigos do mestrado, em especial a Carlos Alberto e George. Muito obrigado pela amizade e pela ajuda na construção desse projeto.

A todos os professores e colaboradores da instituição, pelo trabalho sério que exercem.

Ao meu orientador professor Thomas Buck, por ter me aceitado com orientando e por tornar possível a concretização desse sonho.

"Nem tudo que se enfrenta pode ser modificado,
mas nada pode ser modificado até que seja
enfrentado"
Albert Einstein

RESUMO

Streaming de áudio e vídeo é uma tecnologia que permite dispor um fluxo de dados de um servidor, através do disparo contínuo de pacotes audiovisuais, captados por um *player* dedicado ou *plugin*, em um tempo suficiente para serem carregados trechos contínuos de vídeo na tela, sem interrupções. A idéia é que o servidor e o *player* trabalhem juntos, fazendo com que os dados sejam transmitidos em tempo real. O presente trabalho propõe técnicas para transmissão, recepção, persistência e busca de *streaming* de vídeo em plataformas móveis, utilizando software livre e padrões existentes no mercado. Foram feitas comparações dessas técnicas e apontadas vantagens e desvantagens de cada uma delas e dos *players* existentes. Foram também utilizados padrões para especificações 3GP e protocolos especializados, os quais definem o controle de acesso e a distribuição do conteúdo multimídia.

Palavras-chaves: *Streaming.* J2ME. MMAPI. 3GP. RTSP.

ABSTRACT

Streaming audio and video is a technology that can provide a data stream from a server, through the continuous burst of media packets, captured by a stand-alone *player* or plugin in a sufficient time to load continuous portions of the video screen, without interruption . The idea is that the server and *player* work together, causing the data to be transmitted in real time. This paper proposes techniques for transmission, reception, persistence and search *streaming* video on *mobile* platforms, using open source software and standards. These techniques were compared and pointed out advantages and disadvantages of each and the *players* on the market. There were also used standard specifications for 3GP and specialized protocols, which define access control and distribution of multimedia content.

Keywords: *Streaming.* J2ME. MMAPI. 3GP. RTSP.

LISTA DE FIGURAS

Figura 1 - Transmissão de um *streaming* de vídeo 15
Figura 2 - Transmissão via *broadcast* 16
Figura 3 - Transmissão via *Multicast* 17
Figura 4 - Transmissão via *unicast*. 18
Figura 5 - Arquitetura básica de um *streaming* (Introduction to How Streaming Video and Audio Work) 22
Figura 6 - Arquitetura do Helix Server (Helix Server) 29
Figura 7 - Pacote RTP 33
Figura 8 - Arquitetura de um *WebService* 39
Figura 9- Interface inicial do *YouTube Mobile* 42
Figura 10 - Tipos de pesquisa do *YouTube Mobile* 43
Figura 11 - Conta de um usuário logado 44
Figura 12 - Aplicação do *YouTube Mobile* em um celular Nokia s60 45
Figura 13 - Classes do perfil MIDP 48
Figura 14 - Arquitetura da MMAPI 49
Figura 15 - Estrutura de um *RecordStore* 50
Figura 16 - Utilização do *Floggy* em uma aplicação (Floggy – J2ME persistence framework) 51
Figura 17 - Arquitetura do ambiente cliente-servidor utilizando *WebService* e J2ME 53
Figura 18 - Visão integrada do caso de uso 55
Figura 19 - Diagrama de classe da arquitetura de distribuição *dataAccess* 61
Figura 20 - Arquitetura do servidor 62
Figura 21 - Envio dos dados do servidor WEB para o DM 64
Figura 22 - Lista de vídeo pesquisada pelo servidor 65
Figura 23 - *JDragon Player Mobile* 66
Figura 24 - Transmissão das informações na arquitetura do sistema 68
Figura 25 - Lista de vídeo pesquisada pelo servidor e exibida com LWUIT 70
Figura 26 - Lista de vídeo pesquisada pelo servidor e exibida pela Web 73

LISTA DE QUADROS

Quadro 1 - Principais características dos tipos de *streaming*.....19
Quadro 3 - Discrição do caso de uso Consultar vídeos no Servidor 55
Quadro 4 - Discrição do caso de uso Consultar vídeos no RMS.....56
Quadro 5 - Discrição do caso de uso tocar vídeo no player 57
Quadro 6 - Discrição do caso de uso pausar vídeo no Player.....57
Quadro 7 - Discrição do caso de uso parar vídeo no Servidor 58
Quadro 8 - Discrição do caso de uso Persistir vídeo no RMS.....58
Quadro 9 - Comparativo das quatro formas de transmissão e recepção de conteúdo multimídia para DM 76

SUMÁRIO

1 INTRODUÇÃO 6
1.1 MOTIVAÇÃO 6
1.2 OBJETIVOS 9
1.3 JUSTIFICATIVA 9
1.4 CONTEXTO 10
1.5 CONTRIBUIÇÕES 11
1.6 ORGANIZAÇÃO DO TRABALHO 12
2 TRANSMISSÃO DE *STREAMING* DE VÍDEO 13
2.1 STREAMING 13
2.1.1 O que é *Streaming* *14*
2.1.2 Tipos de *Streaming* *18*
2.1.3 Vantagens do uso de *streaming* *19*
2.1.4 Arquitetura de um *Streaming* *20*
2.2 STREAMING NO MUNDO MOBILE 22
2.3 FORMATOS DE VÍDEO 23
2.4 GERAÇÕES DA TELEFONIA MÓVEL 25
2.5 SERVIDORES DE *STREAMING* *28*
2.6.1 RTP 32
2.6.2 RTCP 34
2.6.3 RTSP 34
2.6.4 UDP x TCP 36
2.7 WEBSERVICE 37
2.8 *PLAYERS* EXISTENTES NO MERCADO 39
2.8.1 Real *Player* *39*
2.8.2 *YouTube Mobile* *41*
3 ARQUITETURAS DE TRANSMISSÃO E RECEPÇÃO DE STREAMING DE VÍDEO PARA DM 46
3.2 DOCUMENTAÇÃO DO SISTEMA 53
3.2.1 Ambiente 53
3.2.2 Diagrama de Caso de Uso 54
3.3 JDRAGON *PLAYER* MOBILE (ESTUDO DE CASO 1) 58
3.3.1 Introdução 59
3.3.2 Recursos 59
3.3.3 Arquitetura 60
3.3.4 Desenvolvimento 62
3.3.5 Limitações 64
3.3.6 Projeto de Interfaces 64
3.4 JDRAGON MMAPI *PLAYER* MOBILE (ESTUDO DE CASO 2) 65
3.4.1 Introdução 65
3.4.2 Recursos 66
3.4.3 Arquitetura 67
3.4.4 Desenvolvimento 68
3.4.5 Limitações 68
3.4.6 Projeto de Interfaces 69
3.5 JDRAGON REAL*PLAYER* MOBILE (ESTUDO DE CASO 3) 69
3.5.1 Introdução 70
3.5.2 Recursos 70
3.5.3 Arquitetura 70

3.5.4 Desenvolvimento **71**
3.5.5 Limitações **71**
3.5.6 Projeto de Interfaces **71**
3.6 RESULTADOS PARCIALMENTE OBTIDOS 72
4 CONCLUSÃO **76**
REFERÊNCIAS **79**

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Já se foi a época em que o celular servia apenas para fazer ligações. Há não muito tempo, os celulares eram enormes, pesados, com três tipos de bateria: fina (tempo de duração muito curto), grossa e pesada (que durava um pouco mais do que a fina) e uma outra com função *vibracall*. Não era possível sequer mandar mensagens de texto por ele. Hoje a realidade é outra. Os aparelhos possuem outras funcionalidades, como jogos, aplicativos, câmera fotográfica, armazenamento de arquivos multimídia, entre outros tantos recursos (Nokia N95 User Guide; Sony Ericsson Java ME Platform Docs & Tools).

As aplicações móveis estão crescendo muito nos últimos anos e as taxas de *downloads* de aplicativos e jogos são altíssimas. Com aparecimento da TV digital e da tecnologia 3G – geração de sistemas móveis de telecomunicações, o celular se tornou uma máquina de conexão com a internet banda larga com fortes indícios de se tornar a forma mais popular de navegação, pois o usuário poderá acessar a internet de qualquer lugar a qualquer momento (RODRIGUES, 2006; YEO; YEUNG, 1997).

Com o aumento da capacidade de armazenamento, processamento e poder de banda dos DM, os aplicativos estão com um maior número de imagens, vídeos e sons, razão por que as operadoras estão investindo cada vez mais em serviços multimídia, como toques, papéis de parede, vídeos, e outros (RIES et al, 2005).

Hoje em dia, uma das maiores taxas de *downloads* feitas na internet é a de vídeo, por isso o empenho de muitos em sua transmissão nos DM (ZHIHUA; HUANG; BENSHUN, 2008). Porém, o interesse dos usuários é assistir os vídeos em tempo real e é nessa questão que se utiliza a transmissão com *streaming*.

Streaming de áudio e vídeo é uma tecnologia que permite dispor um fluxo de dados de um servidor, através do disparo contínuo de pacotes audiovisuais que são captados por um *player* dedicado ou *plugin* que funciona como navegador *Web*, em um tempo suficiente para que se carregue trechos contínuos de vídeo na tela, sem que hajam interrupções (TANENBAUM, 2003; ZHIHUA; HUANG; BENSHUN, 2008). A idéia é que o servidor e o *player* trabalhem juntos, fazendo com que os dados sejam transmitidos em tempo real.

O crescimento dessa tecnologia é tão amplo, que várias empresas já estão utilizando. A *Cyber University* fornece aulas pela internet e também as disponibiliza para celulares através de *streaming* de vídeo. A referida universidade japonesa funciona desde abril de 2008, com aprovação do governo para emitir certificados, e tem 1,8 mil estudantes (YEO; YEUNG, 1997). Igualmente, a rede de videolocadoras *Blockbuster* está em negociação com as principais fabricantes de telefones celulares sobre uma possível parceria para facilitar consumidores a assistirem vídeo através de DM (YEO; YEUNG, 1997).

Segundo a *Insight Research*, *streaming* de vídeo e áudio irá gerar receitas de US$ 70 bilhões nos próximos seis anos, todavia, se os custos do *streaming* caírem mais do que o esperado, se os consumidores aceitarem o IPTV – protocolo de transmissão de sinais televisivos via *streaming*- e a 3G decolar mais rápido que as expectativas, o crescimento da indústria pode alcançar níveis ainda mais prodigiosos (VLAN – VideoLan Doc).

Existem algumas formas para exibir e transmitir vídeo em DM: umas mais eficientes que outras, em razão da arquitetura de transmissão e recepção sob a qual o projeto foi desenvolvido.

No caso de (GOYAL, 2006), é apresentada a API opcional do J2ME, MMAPI, e, por fim, desenvolvido um estudo de caso de um sistema móvel para uso de *Blog*, o qual faz a captura de áudio, vídeo e imagem do DM e envia via HTTP para um servidor Web desenvolvido em Java.

Já em (CUNHA, 2007; LIZHONG; WANG; CHENMING, 2008; SUMIT e outros 2003), é abordada uma visão geral das principais normas e padrões para serviços multimídia em DM e confirma a importância do protocolo RTSP para exibição de vídeo em DM.

Em Vazquez e Vicent (2007) é apresentado um sistema para transmissão de *streaming* de áudio utilizando dois servidores: o primeiro, um servidor PHP, que serve para fazer conversão dos dados, e o segundo, um servidor RTSP, cuja importância é fazer a transmissão do áudio para os DM. Essa aplicação é que mais se aproxima do tema proposto, contudo o foco do estudo mencionado é o *streaming* de áudio.

O intuito desse projeto é apresentar a melhor forma para desenvolver a transmissão e recepção de *streaming* de vídeo em DM, utilizando software livre e padrões existentes no mercado. No decorrer desse trabalho serão apresentadas as tecnologias empregadas para construção de ambientes e aplicações existentes no mercado.

A motivação desse projeto vem do grupo de pesquisa Grupo de Pesquisas em Aplicações Multimídia Avançadas (GAMA) que tem como objetivo a realização de atividades de pesquisa e desenvolvimento na área de multimídia. Tendo em vista que o foco principal do grupo são projetos de processamento de vídeo e que já existe uma gama de trabalhos Web e *Desktop*, percebeu-se uma carência na área de processamento de vídeo em DM, que é um tema atual e bastante discutido na área de computação.

1.2 OBJETIVOS

O presente trabalho tem como objetivos principais: (i) comparar técnicas de implementação para recepção e transmissão de conteúdo multimídia em DM, destacando as vantagens de se utilizar *streaming* de vídeo; (ii) apresentar os principais componentes para tal recepção e transmissão; (iii) contribuir com pesquisas científicas na área de *streaming* de vídeo para DM.

Podem ser ressaltados dois objetivos secundários: (i) criação e configuração de um servidor de vídeo; (ii) desenvolvimento de três estudos de caso sobre *streaming* de vídeo em DM, utilizando tecnologias distintas, mas com as mesmas funcionalidades.

1.3 JUSTIFICATIVA

O *download* que mais utiliza banda na internet são os *streamings* de vídeo, por exemplo, sites como o *YouTube* (CUNHA, 2007). Com o avanço tecnológico na área de DM, os usuários querem, cada vez mais, acessar vídeos dos seus celulares em tempo real. Deste modo, os *players mobile* deverão utilizar tecnologias que acessem *streaming* de vídeo e este conteúdo tem que estar armazenado em servidores especializados.

Existem duas formas para os usuários assistirem vídeos em DM. A primeira seria fazendo *download* dos vídeos pela Web, armazenando-os nos computadores, para depois adicioná-los em DM, utilizando softwares que convertem a mídia para o formato aceitado pelo celular (CUNHA, 2007). A segunda forma seria utilizando sites especializados como o *YouTube Mobile*, entretanto somente celulares que têm acesso à tecnologia Flash poderão acessá-lo. Além do mais, o conteúdo é transmitido via HTTP, o que causa uma lentidão no sistema, pois o DM é obrigado a gravar a mídia à medida

que o vídeo é exibido (ZHIHUA; HUANG; BENSHUN, 2008). Existem, todavia, técnicas de transmissão de vídeo em tempo real, com o protocolo RTSP, assim, o DM pode fazer *download* direto do servidor e exibi-lo em tempo real.

A construção desse projeto busca apresentar a melhor prática para o desenvolvimento de *streaming* de vídeo em DM, demonstrando técnicas de implementação para acessar, pesquisar e persistir vídeos em celulares através de um servidor Web.

1.4 CONTEXTO

Os sistemas de transmissão de *streaming* de conteúdo multimídia devem prever os seguintes componentes básicos na sua arquitetura (AUSTERBERRY, 2004):

a) Protocolos para transmissão e estrutura de distribuição dos dados;
b) O formato da mídia que será transmitida;
c) O software que permite que os usuários reproduzam os arquivos multimídia – *Player*;
d) Servidores para a distribuição do conteúdo para os clientes, utilizando um protocolo definido.

Os protocolos utilizados para distribuição são o UDP e TCP, que realizam a distribuição entre um servidor de *streaming* e um *player*. O UDP realiza esse papel com mais qualidade, pois a sua arquitetura prioriza a distribuição dos conteúdos através de fluxos contínuos de dados.

O contexto deste projeto está relacionado com a configuração, apresentação e desenvolvimento da melhor prática para executar *streaming* de vídeo em DM utilizando protocolos existentes no mercado.

1.5 CONTRIBUIÇÕES

Além da habitual contribuição decorrente da revisão literária no campo de *streaming* de vídeo para DM, o trabalho apresenta outras quatro contribuições para criação, configuração, desenvolvimento e a comparação do melhor ambiente para transmitir e recepcionar *streaming* na plataforma móvel, quais sejam:

a) Proposição e análise de diferentes metáforas de visualização e transmissão de *streaming* de vídeo em DM.
b) Criação de uma arquitetura para transmissão do conteúdo multimídia utilizando padrões existentes no mercado. A idéia dessa arquitetura consiste em disponibilizar o conteúdo multimídia por meio de consultas que serão realizadas pelo cliente móvel.
c) Desenvolvimento de três formas diferentes de transmitir *streaming* de vídeo para DM e compará-las com implementações existentes, destacando a indicação da forma mais vantajosa (rápida, p.ex.) de recepção e transmissão de streaming de vídeo para DM, utilizando: HTTP Connection (J2ME), RTSP com MMAPI e RTSP utilizando o real player.
d) Uso do software livre para desenvolvimento do trabalho. A linguagem de programação, ferramentas e tecnologias utilizadas possui código aberto ou, ao menos, gratuito.

É válido ressaltar que este trabalho resultou na publicação de dois artigos científicos em congressos na área de computação, quais sejam (CORIOLANO; BUCK, 2009).

1.6 ORGANIZAÇÃO DO TRABALHO

A dissertação está dividida em quatro capítulos. O segundo capítulo apresenta conceitos e técnicas para utilização de *streaming* de vídeos, abrangendo conceitos de *streaming*, formatos de vídeo, criação e configuração de servidores de *streaming*, protocolos de transmissão e de recepção de dados.

O terceiro capítulo demonstra a implementação, documentação, tecnologias utilizadas e a comparação da melhor forma de utilizar *streaming* de vídeo em DM.

O último capítulo apresenta algumas considerações finais sobre o trabalho, seguidas pelas referências bibliográficas.

2 TRANSMISSÃO DE *STREAMING* DE VÍDEO

Nesse capitulo serão abordados conceitos sobre streaming de vídeo, que abrangerá: tipos de streaming, vantagens do uso de streaming e arquitetura de um streaming.

Ainda nesse capitulo será feito uma analogia do streaming de vídeo no mundo *mobile*, cujo objetivo é demonstrar a utilização do streaming de vídeo em DM.

Para que um vídeo toque em um player é necessário padronizar um formato de vídeo, com isso ainda nesse capitulo serão abordados os principais formatos de vídeo dando uma maior ênfase ao 3GP.

Para transmissão de um streaming de vídeo é necessária a utilização de um servidor de vídeo especializado, então também serão mencionados os principais servidores de streaming utilizados no mercado e os protocolos empregados para tal transmissão.

2.1 STREAMING

Há algum tempo atrás assistir um vídeo através da internet só era possível através do *download* prévio de todo o vídeo na máquina do cliente, para assim poder ser tocado no *player*. Esse processo é prático para arquivos multimídia de pequena dimensão, mas para grandes vídeos implicará em alguns transtornos, tais como: espera desagradável para o *download* da mídia e interrupções inesperadas, o que causaria o reinicio de todo processo.

Para evitar esses contratempos, pode-se utilizar a tecnologia de *streaming* para transmitir e acessar conteúdo multimídia em uma rede de dados através de fluxos contínuos de dados. Em outras palavras, isso significa que o cliente pode visualizar os

arquivos de áudio e vídeo, em tempo real, através de softwares específicos sem que os dados tenham sido completamente descarregados.

2.1.1 O que é *Streaming*

Streaming é uma tecnologia de distribuição de dados multimídia, em uma rede, através de pacotes codificados (THIRD GENERETION PARTNERSHIP PROJECT; ZHIHUA; HUANG; BENSHUN, 2008). O cliente vai visualizando o conteúdo dos dados ao ritmo em que eles vão chegando, com uma espera somente da criação de um *buffer* e sincronização da mídia (THIRD GENERETION PARTNERSHIP PROJECT; ZHIHUA; HUANG; BENSHUN, 2008). Diferente de um *download* convencional, onde os arquivos ficam salvos na máquina do cliente, no *streaming* real não é feito cópia do arquivo solicitado na máquina do requerente, mantendo a integridade do conteúdo, já que o streaming não permite download do arquivo multimídia, mas nada impede que o usuário utilize softwares especializados para tal tipo de operação (CUNHA, 2007).

Na figura 1 pode ser visualizada a transmissão de um *streaming* de vídeo do servidor ao cliente, na qual demonstra: uma aplicação servidora, onde estariam persistidos os vídeos; a aplicação receptora, que seria os DM; e a internet que seria o meio de comunicação entre o servidor e o cliente. A idéia é que o servidor envia a streaming de vídeo ao cliente, através da internet, e o DM armazena a informação por meio de um buffer e apresenta a mídia no player.

A vantagem do streaming é quem, semelhante a uma transmissão de TV, o mesmo conteúdo pode ser transmitido para vários clientes no mesmo espaço de tempo. Diferente do download convencional, que por utilizar o protocolo HTTP fica inviável fazer

esse tipo de transmissão, já que o servidor teria que ter certeza que a mídia foi recebida, por completo, para todos receptores.

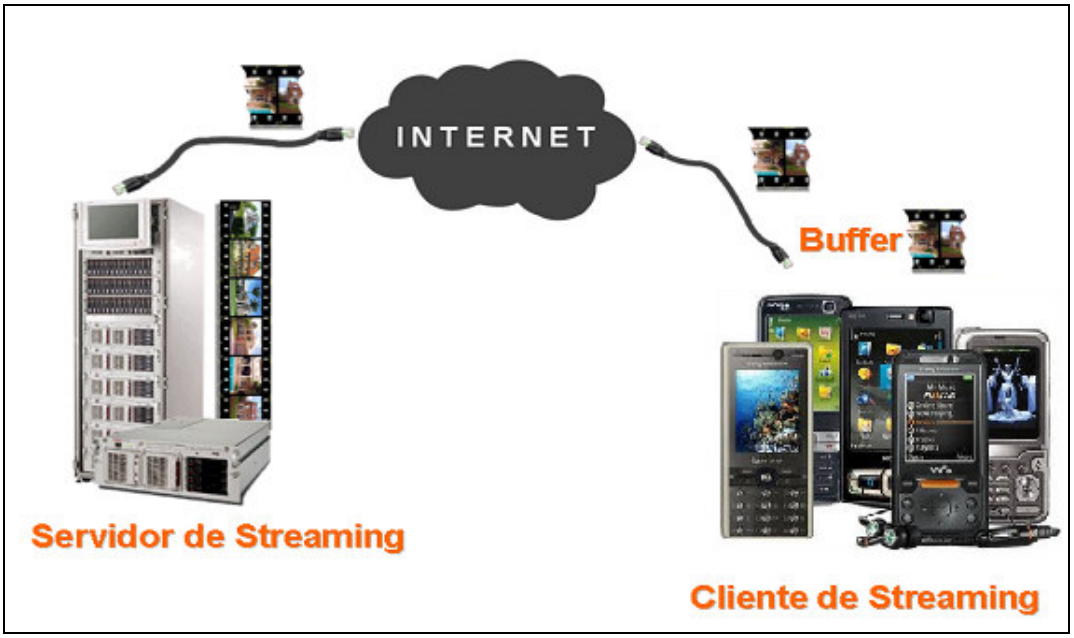

Figura 1 - Transmissão de um *streaming* de vídeo

Para transmissão de um *streaming* de áudio ou vídeo é necessário a utilização de uma estrutura bem definida, cujos componentes são: protocolos para transmissão e distribuição de dados; formatos de arquivos específicos; *player* para reproduzir a mídia transmitida; servidor para distribuição do conteúdo para os usuários.

O protocolo mais utilizado para distribuição de arquivos de *streaming* é o UDP, pois realiza a distribuição entre o servidor e o *player* através de fluxos de dados e ignora os pacotes perdidos, com isso permite uma experiência em tempo real. Com o protocolo UDP pode ser feito transmissão de dados da seguinte forma (CUNHA,2007; SUMIT et al, 2003):

- **Broadcast**: é um processo de transmissão em que a informação é enviada pelo servidor e recebida, ao mesmo tempo, por todos os receptores. Nesta abordagem, o cliente não tem controle no fluxo da transmissão, pois a

informação está sendo enviada em tempo real, assim não são permitidas ações como: parar, continuar ou reiniciar. Esse tipo de transmissão é utilizado em rádio, telecomunicações e em alguns casos na informática. Na figura 2 pode ser visualizada a representação de uma transmissão via *broadcast,* onde possui um servidor central enviando a mesma informação para todos os clientes.

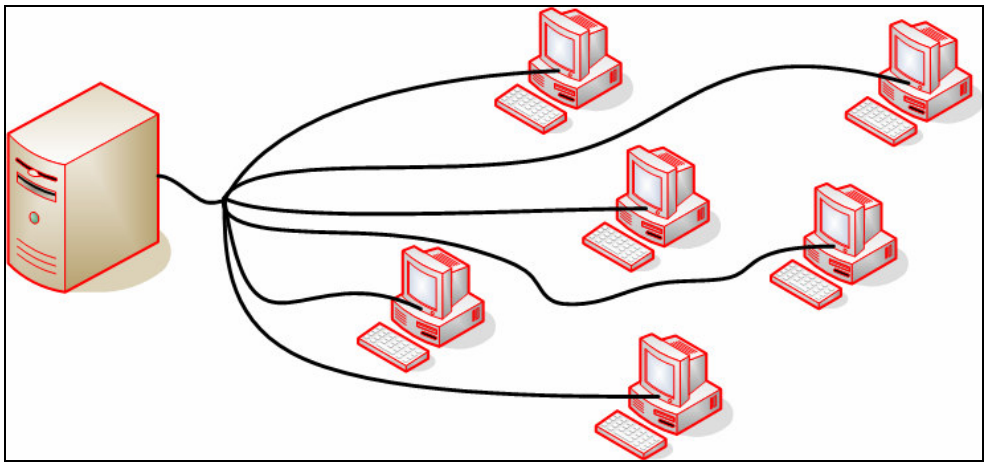

Figura 2 - Transmissão via *broadcast*

- **Multicast**: processo onde a comunicação é feita de um emissor para um conjunto selecionado de receptores. Durante uma transmissão *Multicast*, o transmissor envia os pacotes de dados somente uma vez, ficando a cargo dos receptores captarem esta transmissão e reproduzi-la. Esta técnica diminui consideravelmente o tráfego em diversas situações, como por exemplo, uma partida de futebol assistida por várias pessoas simultaneamente e propagada por um servidor. Na figura 3 pode ser visualizada a representa de uma transmissão via *multicast,* onde a informação é enviada somente aos clientes selecionados.

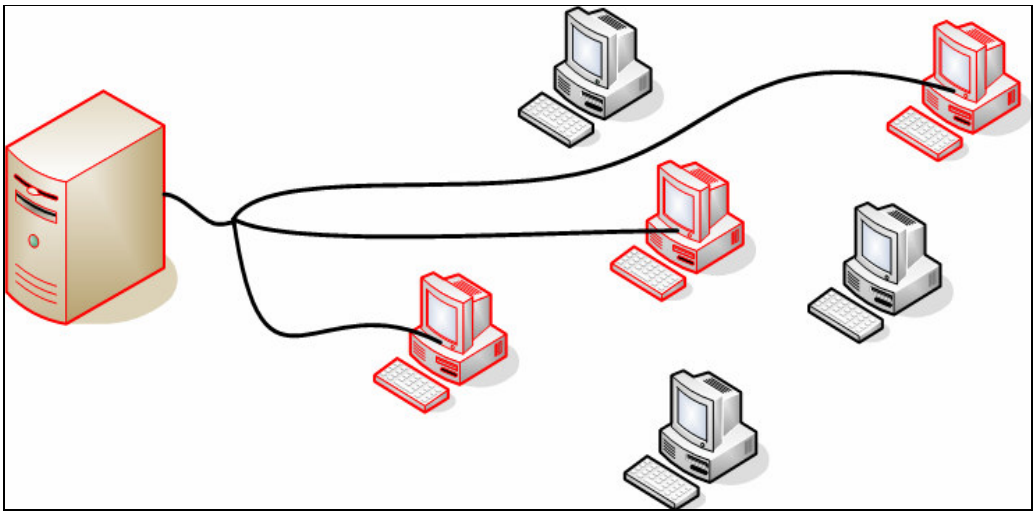

Figura 3 - Transmissão via *Multicast*

- **Unicast**: nesse tipo de transmissão é estabelecido um canal dedicado de comunicação entre o emissor e o receptor, isso é, os dados são enviados na medida em que o receptor solicita. Um servidor de *streaming* pode se comunicar com vários canais *unicast* e sua limitação é a banda disponível. Na figura 4 pode ser visualizada a representação de uma transmissão via *unicast,* onde o servidor envia a informação para o cliente que fez a solicitação.

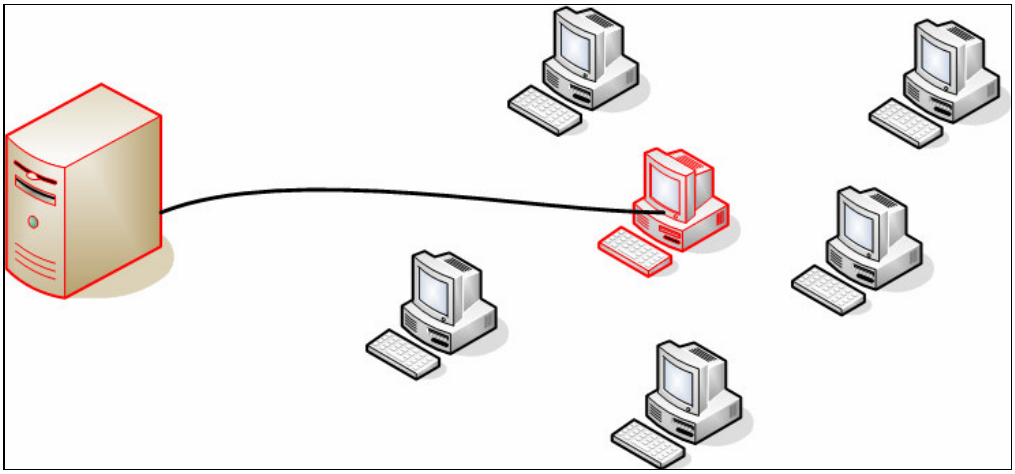

Figura 4 - Transmissão via *unicast.*

2.1.2 Tipos de *Streaming*

Os *streamings* de vídeo são divididos em três formas de transmissão (THIRD GENERETION PARTNERSHIP PROJECT; ZHIHUA; HUANG; BENSHUN, 2008; CUNHA, 2007):

- ***Streaming* ao vivo**: nesse caso a técnica utilizada é a transmissão *broadcast*, pois todos os usuários estão assistindo a mesma apresentação ao mesmo tempo. Essa seria uma distribuição de *streaming* ao vivo, no qual pode ser comparada à transmissão de um canal de TV. Caso um usuário acesse o vídeo em um determinado momento esse vai assistir o arquivo multimídia no ponto em que a transmissão está sendo feita. Esse tipo de transmissão é utilizado, por exemplo, para aulas tele-presenciais, pois a aula é transmitida em tempo real para todos os alunos.
- ***Streaming* sob demanda**: esse tipo de transmissão é utilizado em sites como o *YouTube*. O usuário vai assistir o conteúdo no momento que ele quiser e esse arquivo sempre será executado a partir do seu inicio.

- ***Streaming* ao vivo simulado**: essa transmissão é semelhante ao *streaming* ao vivo, contudo o conteúdo já está persistido no servidor.

Nesse trabalho será abordado a streaming ao vivo, também conhecido como streaming real, no Quadro 1 são apresentadas as principais características dos tipos de *streaming*.

Quadro 1 - Principais características dos tipos de *streaming*

***Streaming* Ao Vivo**	***Streaming* Simulado**	***Streaming* sob Demanda**
Acesso à apresentação apenas se ela estiver em progresso.	Acesso à apresentação apenas se ela estiver em progresso.	Acesso à apresentação a qualquer momento
Não existe arquivo	Existe arquivo	Apresentação armazenada em arquivo
Começa da parte que está sendo transmitida no momento	Começa da parte que está sendo transmitida no momento	Sempre começa desde o inicio da apresentação

2.1.3 Vantagens do uso de *streaming*

Ainda existem alguns desafios para que um sistema de *streaming* de vídeo funcione com perfeição tanto para plataforma móvel como Web. Um problema no uso dos *streamings* de vídeo é a perda de pacotes no envio do conteúdo multimídia. Uma transmissão de dados sem perda de informação seria utilizando o protocolo HTTP, com a tradicional forma de *download*, mas o usuário não visualizaria o conteúdo antes que os dados fossem totalmente descarregados. Entretanto, existe o *download progressivo,* conhecido como *pseudo-streaming* ou *http streaming*, que consiste na utilização de um servidor Web tradicional, cujo objetivo é disponibilizar conteúdo multimídia e permitir a reprodução dos conteúdos antes que tenham sido completamente descarregados. Esse tipo de transmissão tem algumas restrições, que são:

a) Não é possível a utilização de *broadcast* nos eventos em tempo real;
b) Não aceita controle na largura de banda, logo é provável que aconteça um estouro no buffer, causando paradas na reprodução dos conteúdos enviados aos clientes;
c) Não tolera controle de fluxo do conteúdo, com isso não é possível avançar de uma cena a outra adiante sem que o *download* do conteúdo tenha sido feito até o ponto do dado desejado;
d) O conteúdo é totalmente transferido para o cliente, assim é requerido espaço em disco para inserção dos dados.

Já o *streaming* real oferece vantagens sobre os métodos citados anteriormente, como:

a) Não é necessário esperar o *download* completo do arquivo. Os dados são visualizados através de fluxos;
b) Os fluxos de dados não são gravados no disco, estes são tratados e exibidos para o cliente e em seguida descartados;
c) É possível transmitir eventos ao vivo em tempo real ao redor de todo mundo;
d) Suporta interatividade no uso de streaming sob demanda, permitindo que o conteúdo seja exibido de uma maneira não linear. O usuário pode dar saltos de um tempo a outro da mídia.

2.1.4 Arquitetura de um *Streaming*

Na arquitetura de um *streaming* há quatro componentes básicos (ZHIHUA; HUANG; BENSHUN, 2008):

a) Codificar o arquivo multimídia

b) Adicionar no servidor de dados

c) Distribuir a informação

d) Exibir no *player* do cliente

Codificar o arquivo multimídia implica em transformá-lo em um padrão de vídeo especializado. Esses arquivos codificados são adicionados em um servidor de dados, que possui um software especial que pode transmiti-lo em tempo real, sendo assim é criado um canal de comunicação entre o *player* e o servidor. A arquitetura da transmissão e recepção de dados vai depender bastante do tipo de codificação que o vídeo esteja utilizando, pois dependendo do tipo e/ou tamanho de um vídeo, deverá ser estudada a melhor forma de enviá-lo ou recebê-lo. A figura 5 ilustra a arquitetura básica de um *streaming*.

Para transmissão dos dados entre cliente e servidor podem ser utilizados os protocolos HTTP (TCP) e RTSP (UDP). No primeiro caso são enviados repetitivamente os pacotes de dados para o cliente até que sejam entregues com sucesso, com isso causando uma certa lentidão no processo. Já no segundo caso, a distribuição entre servidor de *streaming* e o *player* é feita com mais qualidade, que é alcançada graças a arquitetura que prioriza a distribuição em fluxos contínuos, pois o UDP continua a enviar os dados mesmo se ocorrer alguma perda no caminho, o que permite uma experiência em tempo real, que é o fator principal de uma transmissão via *streaming* (MADEIROS; PETROLI, 2008).

Figura 5 - Arquitetura básica de um *streaming* (Introduction to How Streaming Video and Audio Work)

2.2 STREAMING NO MUNDO MOBILE

As aplicações móveis estão sendo muito procuradas nos últimos anos, devido ao número de assinantes móveis no mundo, que é estimado em cerca de 3,2 bilhões, e segundo estudos realizados pela China *Mobile* - maior empresa de telefonia móvel do mundo, esse número pode crescer em pelo menos um bilhão nos próximos anos (RODRIGUES, 2006).

Os DM são mais predominantes do que carros e cartões de crédito, que são estimados em 800 milhões e 1,4 bilhões, respectivamente. Os seres humanos estão tão dependentes dos DM, que segundo estudo realizado pela China *Mobile*, mais da metade dos usuários móveis dormem a menos de um metro de distância dos seus dispositivos,

logo pode ser confirmado o motivo de tamanha procura dos clientes em aplicativos móveis.

As taxas de *download* e *upload* de vídeos na internet são altíssimas, em um estudo feito por Ryan Junne (gerente de produto do *YouTube*) o *YouTube* recebe cerca de 20 horas de *uploads* de vídeo por minuto, por isso o interesse de muitos no que diz respeito sobre *streaming* de vídeo para DM (VAZQUEZ; VICENT, 2007).

A computação móvel permite aos usuários acessarem sistemas através de redes de computadores, que proporcionam aos clientes muitas necessidades do mercado através de serviços móveis. Uma dessas necessidades são os sistemas multimídia, no qual estão enquadrados os *streamings* de vídeo (TANENBAUM, 2003; VAZQUEZ; VICENT, 2007).

Um desafio para *streaming* em DM é obter uma boa transmissão e recepção entre o servidor e o cliente móvel e também um formato de vídeo que seja transmitido com uma boa qualidade.

Para transmissão dos dados multimídia podem ser utilizados os protocolos RTSP e HTTP, para o qual a MMAPI tem suporte. Já em relação ao formato de vídeo o padrão mais reconhecido pelos DM atualmente é o 3GP (BOUAZIZI; MISK; CURCIO, 2007; YEO; YEUNG, 1997).

2.3 FORMATOS DE VÍDEO

Com a popularização da banda larga e dos sistemas de *players* multimídia, a circulação de vídeos teve um aumento considerável. Contudo, junto com esse aumento, também foram crescendo a quantidade de formatos de vídeo, no qual, às vezes, torna a experiência de um *download* de um vídeo decepcionante, pois o usuário não sabe qual

codec (complemento que comprime ou descomprime um vídeo) deve ter em sua máquina para poder tocar o vídeo.

Uma vantagem dos *streamings* de vídeo é que o usuário que está assistindo o vídeo não precisa se preocupar com o formato, pois o próprio *player* já está configurado no ambiente *mobile* ou Web para recepção de tal mídia.

O formato de vídeo utilizado nos DM vai depender da implementação feita no player, porém a maioria dos DM suporta o padrão MPEG-4 e o 3GP (MPEG-4 parte 12), por isso é mais viável utilizar um desses formatos (GOYAL, 2006; WANG; BOUAZIZI; MISK; CURCIO, 2007). Em caso de implementações MMAPI, o programador tem um leque de opções para formatos de vídeos, mas o DM pode não suportar o formato escolhido, por isso é interessante testar se o celular aceita o formato para depois adicioná-lo ao *Player*. Para verificar os formatos compatíveis com o DM pode ser utilizado a API opcional do J2ME, que é a MMAPI, a qual tem o método *getSupportedContentTypes(),* cujo objetivo é retornar uma lista de tipos suportados (GOYAL, 2006).

O *MPEG-4* foi definido pelo MPEG, e absorve muita das funcionalidades do MPEG-1 e MPEG-2, que também foram especificados pela MPEG (MONTEZ; WILLRICH; PISTORI, 2001; RIES, et al 2005). Vários pesquisadores de todo o mundo contribuíram para o MPEG-4, que foi finalizado e introduzido no mercado em 1998. Esse padrão está dividido em várias partes e muitas empresas dizem aderir ao padrão, mas não deixam claro a qual nível de partes de compatíbilidade elas se referem (MONTEZ; WILLRICH; PISTORI, 2001).

A vantagem do MPEG-4 em relação ao MPEG-2 é o conceito de objetos em seu sistema de transmissão, pois esse padrão proporciona o envio de partes diferentes do conteúdo, como áudio e vídeo, dessa forma os objetos podem ser enviados

separadamente e reagrupados por um decodificador, com isso permite-se que cada quadro de vídeo seja codificada da melhor maneira possível (HELIX SERVER; RIES et al 2005; TANENBAUM, 2003; WINDSON; VIANA; ANDRADE, 2006).

Já o 3GP se distingue em duas formas, o 3GPP, que é utilizado em redes GSM e o 3GPP2, que é empregado em CDMA (TANENBAUM, 2003). Ambos são derivados do MPEG-4, por isso são bem semelhantes, porém o 3GPP2 possui uma melhor qualidade de vídeo, pois esse acrescenta a opção de utilizar QCELP, essa tecnologia permite que o conteúdo seja entregue de forma incremental através de redes TCP padrão sem fio, proporcionando uma experiência mais imediata na transmissão final para o usuário. O padrão 3GP foi projetado especialmente para o uso em DM e para distribuição de conteúdo multimídia em redes sem fio de alta velocidade (THIRD GENERETION PARTNERSHIP PROJECT; BOUAZIZI; MISK; CURCIO, 2007; YEO; EUNG,1997).

Arquivos e *streams* 3GP podem conter diferentes tipos de mídia, como áudio, vídeo e texto. Isso se deve ao fato desta especificação suportar vários formatos de mídia, entre os quais: MPEG-4, AMR, JPEG, GIF, H.263 (SUN Java Wireless Toolkit for CLDC). Caso o arquivo 3GP seja codificado utilizando o *codec* H.263 para vídeo, e AMR para áudio, vai tocar no *player* do DM sem problema e a mesma regra para os outros formatos (RODRIGUES, 2006; BOUAZIZI; MISK; CURCIO, 2007; SUN JAVA WIRELESS TOOLKIT FOR CLDC; SUMIT et al, 2003).

2.4 GERAÇÕES DA TELEFONIA MÓVEL

As conexões móveis estão crescendo vertiginosamente, causando uma revolução no mercado das telecomunicações, pois estão disponíveis em praticamente todos os lugares das áreas urbanas as rurais. Logo, cidadãos poderão trabalhar a longa distância

ou manter contatos com amigos mantendo uma conexão contínua. Essa evolução constante tornou-se possível graças ao conceito de comunicação celular (SILVA, 2009; SUMIT e outros, 2003).

A telefonia móvel passou por quatro gerações distintas e com tecnologias diferentes. A primeira foi analógica, a segunda digital, terceira e a quarta também digital, mas com poder de transferência de dados (internet, multimídia, correio eletrônico e outros). Essas quatro gerações são divididas em (VLAN – VIDEOLAN DOC; MOBILE MEDIA API SUN; GOYAL, 2006; GUBERT; BAZZOTI, 2007; MELO, 2008):

- **Primeira geração**: essa geração foi apelidada 1G (*First Generation*) e utilizava a modulação analógica de sinais em uma onda portadora de rádio frequência. Neste tipo de rede, um circuito de voz é alocado permanentemente enquanto dura a chamada. Trata-se de um serviço orientado a conexão. Na geração 1G cada território implantou seu próprio sistema celular, sendo assim os celulares com tecnologias diferentes não podiam se comunicar.
- **Segunda geração:** na década de 80, surgiram os sinais digitais e com a demanda de novos usuários móveis, o sistema de telefonia analógico foi extinto. Novos sistemas com novas tecnologias foram criados, mas a segunda geração, também conhecida como 2G(*Second Generation*), continuou sem padrão. Assim surgiram vários sistemas de telefonia móvel: GSM - Sistema global de comunicação móvel, na Europa; o TDMA ou D-AMPS e o CDMA, nos EUA; e o PDC - Celular digital pessoal, no Japão. Em 1992 o padrão GSM foi lançado na Europa e com isso, muitos países adotaram esse padrão.

A grande diferença da tecnologia 2G para 1G são: a possibilidade de uso de técnicas de codificação digital de voz, maior segurança na transmissão de informações, utilização de códigos corretores de erro, melhoria da eficiência espectral, rejeição à interferência, imunidade ao ambiente de propagação e sensibilidade.

- **Terceira geração**: essa tecnologia foi desenvolvida por um esforço conjunto de várias organizações mundiais com a finalidade de definir um sistema celular global de terceira geração, o UMTS. Esse padrão define três domínios: o equipamento de usuário, a rede de acesso sem fio e a rede de núcleo (Third Generetion Partnership Project). As grandes vantagens do uso desse padrão são: utilização de banda larga nos DM, que podem fornecer taxas de até 10 *Megabits* por segundo e o suporte a um maior número de clientes de voz e dados a um custo menor que a tecnologia 2G.
- **Quarta geração**: A desejada rede 4G (*4th generation mobile networks*) começou a ser utilizada no Japão, com o sucesso foi implantado em alguns países europeus e também está em fase de adaptação nos Estados Unidos (Third Generetion Partnership Project). Essa tecnologia é bastante superior a 3G, pois a velocidade de transmissão é superior a 40 *Megabits* comparada a geração anterior (Germany Embraces 4G). A idéia dessa geração é aumentar o acesso de aplicações multimídia em DM, tornando a transmissão de streaming, *download*s de música, jogos online e outros mais velozes e acessíveis. Um problema do 4G é que sua banda é compartilhada entre os celulares de uma determinada área de cobertura, isso porque uma torre de transmissão compartilha a banda entre os

celulares da região de alcance. Como as torres são muito mais caras que um ponto de acesso wireless, a adoção dessa nova tecnologia torna-se mais difícil. Como a Copa do Mundo de 2014 será no Brasil existe uma boa probabilidade que essa tecnologia seja implantada em algumas cidades brasileiras, lembrando que existem cidades que não tem acesso as redes 3G.

2.5 SERVIDORES DE *STREAMING*

Para transmitir vídeos via *streaming* é necessário o uso de servidores especiais e bem configurados, cuja função é armazenar e controlar o transporte da mídia até o cliente. Existem no mercado vários servidores de mídia e cabe ao desenvolvedor escolher o que se encaixa melhor com seu problema. O interesse por esses tipos de servidores foram aparecendo devido ao fato de servidores Web não terem capacidade de transmitir áudio e vídeo em tempo real.

Podem ser citados vários servidores de *streaming*, entretanto os mais utilizados são:

- **Helix Server**: é uma solução comercial da *RealNetworks*. Esse servidor executa vários formatos diferentes, logo possui um diferencial entre as soluções da Apple e Microsoft, que suportam uma quantidade menor de formatos (Helix Server). Pode ser instalado nas plataformas Windows, Linux e Solaris (TANENBAUM, 2003
- **Helix DNA Server**: a *RealNetworks* disponibilizou uma versão de código aberto, porém com algumas limitações comparada a sua versão comercial. Entre as limitações podem ser citadas a que o servidor não suporta os

formatos WMV, MOV, QT e também não suporta *streaming* simulado (Helix Server). A figura 6 ilustra a arquitetura do Helix Server.

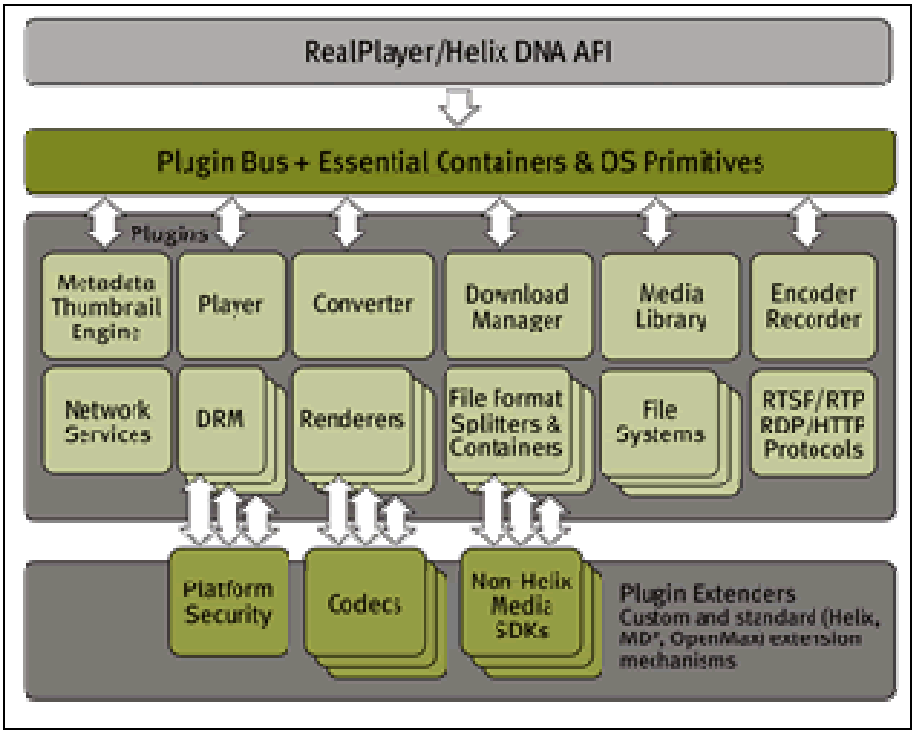

Figura 6 - Arquitetura do Helix Server (Helix Server)

- **QuickTime *Streaming* Server**: versão comercial da *Apple*, esse servidor executa somente no Mac OS e suporta os formatos *MOV*, MPEG *e 3GP*. Nesse servidor pode ser feito *streaming* das seguintes formas: ao vivo, ao vivo simulado e sob demanda. Uma vantagem desse servidor é a possibilidade de replicar o *stream* para um ou mais servidores, diminuindo o consumo de banda dos servidores e aumentando a capacidade de transmissão de conteúdo no sistema (Quick Time Streaming Server). O Quadro 2 ilustra as características do QuickTime *Streaming* Server.

Quadro 2 - Características do QuickTime *Streaming Server* (Quick Time Streaming Server)

Formatos suportados	**QuickTime Movie (.MOV)**
	MPEG-4 (.MP4)
	3GPP (.3GP)
Protocolos de Streaming	**Real Time Streaming Protocol (RTSP)**
	Real Time Transport Protocol (RTP)
	Session Description Protocol (SDP)
	3GPP Release 5 (TS 26.234 v5.6)
Performance	
Streaming ao vivo	**20 kbps (AAC audio)**
	64 kbps (MPEG-4 video & AAC audio)
	300 kbps (MPEG-4 video & AAC audio)
Streaming sob demanda	**20 kbps (AAC Audio)**
	64 kbps (MPEG-4 video & AAC audio)
	300 kbps (MPEG-4 video & AAC audio)
Sistema Requerido	**Xserve, Power Mac G5, G4 or G3, iMac, eMac or Mac mini computer; 128MB of RAM, at least 256MB of RAM for high-demand servers running multiple services; built-in USB; 4GB of available disk space.**
Plataformas	**Mac OS X Server Admin**
	Web-based Admin
	UNIX command line
Transmissão	**Multicast**
	Unicast

- **Darwin *Streaming* Server**: como o *RealNetworks* a Apple disponibilizou uma versão de código aberto para a comunidade, com a vantagem de poder executar o serviço nas plataformas *Windows*, *Linux* e *Solaris* (Quick Time Streaming Server). Também foi aumentada sua capacidade quanto aos formatos de vídeo. Em relação aos protocolos de transporte, esse servidor funciona da mesma forma da versão comercial e também pode fazer transmissão tanto *multicast* como *unicast*.
- **Windows Media Server**: o servidor da *Microsoft* só é executado em ambiente Windows e além de transmitir conteúdo através dos protocolos

HTTP e RTSP, também possui um conjunto de protocolos proprietários. O protocolo mais conhecido e utilizado é o MMS (*Multimedia Server Protocol*) (ROY; WEE, 2003).

- **VideoLan**: VLC media *player* (inicialmente VideoLAN Client) é um *player* multimídia que possui portabilidade para vários formatos de áudio e vídeo (MPEG, DivX / Xvid, Ogg, e outros), bem como DVDs, VCDs e vários protocolos de *streaming*. No entanto, nos últimos anos tornou-se também um servidor extremamente poderoso para transmitir vídeo ao vivo e sob demanda em diversos formatos (VLAN – VideoLan Doc).
- **DarkIce**: é um servidor de código aberto que pode ser executado em múltiplas plataformas. Tem como especialidade transmissão de áudio em vários formatos (DarkIce).
- **Andromeda**: esse servidor tem como especialidade transmitir MP3 para os clientes. Caso o usuário tenha um site, basta adicionar o Andromeda em uma pasta do servidor para oferecer serviços de *streaming*. Esse tem versões desenvolvidas em ASP e PHP (MONTEZ; WILLRICH; PISTORI, 2001).

2.6 PROTOCOLOS DE TRANSMISSÃO

Transmitir dados multimídia através da internet requer o uso de protocolos para empacotamento e transporte do conteúdo na rede. Os mais conhecidos são: HTTP, FTP, RTP, RTCP e RTSP.

O HTTP e FTP utilizam o protocolo TCP, que é um protocolo da camada de transporte desenvolvido para comunicação confiável de dados em baixa largura de banda e em redes com alta taxa de erros.

Já o RTP, RTCP e RTSP são baseados no protocolo UDP, sendo esse não muito confiável, pois não garante que cada pacote irá alcançar o seu destino e nem que o pacote vai chegar na ordem em que foi enviado (MADEIROS; PETROLI, 2008; SUMIT e outros, 2003).

Quando um *streaming* de áudio ou vídeo é enviado a um cliente em tempo real, o cliente pode começar a acessar o conteúdo sem que o mesmo tenha sido baixado completamente. Sendo assim, pode acontecer perda de pacotes na entrega dos dados para que o conteúdo seja transmitido em tempo real e o protocolo utilizado para essa transmissão é o UDP (CUNHA, 2007; MADEIROS; PETROLI, 2008; SILVA 2009; VAZQUEZ; VICENT, 2007; ZHIHUA; HUANG; BENSHUN, 2008).

Os protocolos envolvidos com *streaming* de áudio e vídeo são RTSP, RTP e RTCP. O RTSP é responsável pelo comando de um *streaming* de vídeo, sendo conhecido como o "controle-remoto" da *Internet*. O RTP é responsável pelo transporte do conteúdo multimídia e o RTCP tem como principal função fornecer um *feedback* da qualidade dos serviços oferecidos pelo RTP (ZHIHUA; HUANG; BENSHUN, 2008; CUNHA,2007; MADEIROS; PETROLI, 2008; SUMIT e outros 2003).

2.6.1 RTP

RTP provê uma entrega de serviços em rede fim-a-fim para a transmissão de dados em tempo real. RTP é independente de rede e de protocolo de transporte, embora seja freqüentemente usado com o UDP, suportando assim transmissão *unicast* e *multicast* (ZHIHUA; HUANG; BENSHUN, 2008; CUNHA, 2007; MADEIROS; PETROLI, 2008; SUMIT e outros, 2003).

Em um serviço de rede *unicast*, cópias separadas de dados são enviadas da origem para cada destino. E em serviço *multicast*, os dados são enviados da origem

apenas uma vez e a rede é responsável pela transmissão dos dados para os múltiplos destinos. *Multicast* é mais eficiente para muitas aplicações multimídia, como videoconferências. O protocolo padrão da Internet (IP) suporta perfeitamente o *multicast*.

Quanto à QoS, é importante ressaltar que o RTP não possui nenhum mecanismo que a garanta. Contudo, através do RTCP é possível obter informações de serviços QoS, tais como : número de relatório de pacotes, comprimento de pacotes, identificação de fonte especifica ao receptor, e outras as informações necessárias para uma implementação de mecanismos que suporte QoS (EVANGELISTA; GUARDIEIRO, 2008).

No cabeçalho do RTP podem ser encontradas informações importantes sobre o conteúdo transmitido, como: versão do protocolo; codificação de áudio; tempo e número de sequência de cada pacote e outros (MADEIROS; PETROLI, 2008; SUMIT e outros 2003). Na figura 7 pode ser visualizado o pacote RTP, que logo mais abaixo são explicadas suas principais funcionalidades.

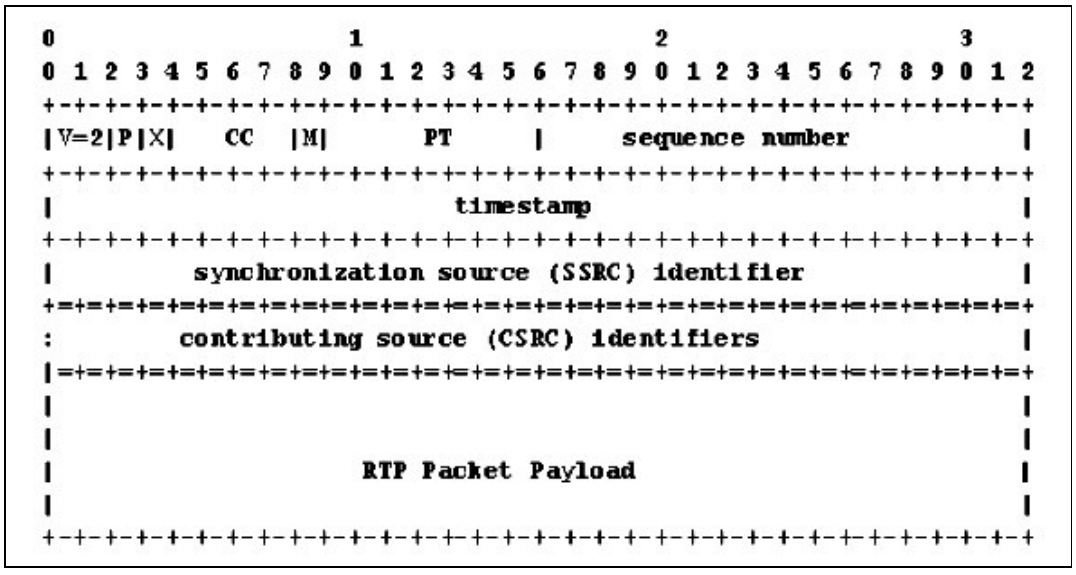

Figura 7 - Pacote RTP

a) *Versão* (V) – Os primeiros dois bits indicam a versão do protocolo utilizada.

b) *Payload type* (PT) – São usados sete bits para identificar o tipo do *payload*, ou seja, a codificação utilizada na mídia.

c) *Sequence number* – São usados dezesseis bits para identificar cada pacote de forma sequencial.

d) *Timestamp* – São usados trinta e dois bits para informar o tempo em que foi exibido o primeiro byte dos dados trafegados.

e) *Syncronization Source* (SSRC) - São usados trinta e dois bits para identificar a fonte da seqüência dos pacotes.

f) *RTP Packet Payload* – Dados da mídia que está sendo transmitida.

2.6.2 RTCP

Real-Time Control Protocol (RTCP) é utilizado em conjunto com o protocolo RTP para fornecer *feedback* sobre a qualidade do serviço. Em uma sessão RTP, pacotes RTCP são transmitidos entre os participantes daquela sessão, contendo relatórios estatísticos que podem ser úteis para a aplicação. Tais estatísticas incluem: número de pacotes enviados; número de pacotes perdidos; e variações nos atrasos durante a transmissão (CUNHA, 2007; MADEIROS; PETROLI, 2008; SUMIT e outros 2003). A especificação do RTP não indica o que a aplicação deve fazer com tais informações; esta tarefa fica sob a responsabilidade do desenvolvedor.

Para o protocolo RTCP desempenhar suas funções existem cinco tipos de pacotes (CUNHA, 2007; MADEIROS; PETROLI, 2008; SUMIT e outros 2003):

a) *Sender report* - SR, provê estatísticas de transmissão e recepção de participantes que enviam mídia.

b) *Receiver report* - RR, fornece estatísticas de recepção de participantes que recebem dados multimídia.

c) *Source Description* - SDES, possui informação sobre a fonte.

d) BYE – Indica o fim de uma participação.

e) APP – Funções específicas de aplicações.

2.6.3 RTSP

O RTSP é um protocolo de aplicação utilizado para controle da entrega de dados em tempo real: ele pode ser comparado a um controle remoto de um *DVD player* para servidores multimídia (CUNHA, 2007; MADEIROS; PETROLI, 2008; SUMIT e outros 2003). Com isso, o usuário pode controlar o que deseja assistir com comandos como: pausa, avanço, retorno e interrupção de uma apresentação.

O RTSP funciona através de troca de mensagens, transmitindo requisições e respostas entre cliente e servidor. Como não existe um conceito de conexão RTSP, o servidor identifica cada sessão através de um código único, isso porque o servidor precisa guardar o estado de cada cliente. Um exemplo seria quando um cliente recebe uma resposta do servidor de mídia e invoca o *Player* multimídia, em seguida o reprodutor e o servidor trocam uma série de mensagens RTSP entre si, como: *Play, Pause, Stop* e outros (ZHIHUA; HUANG; BENSHUN, 2008; CUNHA, 2007; MADEIROS; PETROLI, 2008; SUMIT e outros 2003).

Para realizar o transporte dos dados multimídia o RTSP precisa se relacionar com outros protocolos. E essas mensagens podem ser transportadas tanto via UDP como TCP. Na grande maioria das vezes esse transporte é feito através do protocolo RTSP, que criará uma conexão UDP para realizar o tráfego das informações. Para controlar o sincronismo da mídia e informação de pacotes perdidos será utilizado o RTCP, que criará duas conexões UDP, cada um em um sentido com o objetivo de controlar as

informações entre o cliente e o servidor (ZHIHUA; HUANG; BENSHUN, 2008; CUNHA, 2007).

O protocolo RTSP não realiza as seguintes tarefas:

a) Não define métodos de compressão de áudio e vídeo;
b) Não define como o sinal é encapsulado em pacotes para transmissão em uma rede. Esse encapsulamento pode ser realizado pelo protocolo RTP ou protocolo proprietário;
c) Não restringe como a mídia será transportada. Pode ser via UDP ou TCP;
d) Não restringe como um reprodutor de mídia armazena o sinal de vídeo em um *buffer*. O vídeo pode ser exibido da seguinte maneira: assim que chegar ao cliente; com o atraso de alguns segundos; pode ser transferido totalmente antes de ser exibido. (ZHIHUA; HUANG; BENSHUN, 2008; CUNHA, 2007).

2.6.4 UDP x TCP

O TCP é um protocolo da camada de transporte orientado à conexão que fornece um serviço confiável de transferência de dados fim-a-fim, implementando mecanismos de recuperação de dados perdidos, danificados ou recebidos fora de ordem. A versatilidade e robustez deste protocolo tornou-o adequado a redes globais, já que este verifica se os dados são enviados pela rede de forma correta, na sequência apropriada e sem erros.

O UDP é um protocolo de transporte pertencente à família de protocolos TPC/IP, no qual fornece um serviço mínimo de transporte em redes que usam o protocolo IP, permitindo que as aplicações tenham acesso direto aos serviços da camada de rede (ZHIHUA; HUANG; BENSHUN, 2008; CUNHA, 2007). UDP é tradicionalmente usado em redes TCP, mas ao contrário do TCP, não garante o envio dos dados com confiabilidade, isso é, alguns pacotes podem não chegar ao receptor.

Enquanto o TCP é utilizado principalmente para a comunicação entre um servidor e um único cliente, o UDP é usado para o pacote de difusão ou *multicast*, no qual os dados são enviados para todos os clientes na rede, que são os casos do *streaming* de áudio ou vídeo.

Tradicionalmente, utiliza-se o processo de *download* para descarregar arquivos no computador do cliente, mas esses dados devem ser totalmente descarregados para assim o receptor ter acesso ao conteúdo: essa técnica é utilizada através de conexões HTTP.

Já as conexões UDP trabalham com fluxos de dados em tempo real, que é a tecnologia de *streaming*, embora tenha perda de alguns pacotes na transmissão. Esse protocolo é ideal para uso de *streaming* de áudio e vídeo, já que o conteúdo é transmitido para todos os usuários que fizeram a requisição e caso algum pacote não seja entregue o conteúdo continua sendo transmitido.

Como já foi mencionado anteriormente, ainda existe o *download* progressivo, também conhecido como HTTP *streaming.* Apesar de ser uma solução interessante, esse tipo de *download* tem algumas limitações comparadas ao *streaming* real, que são: paradas inesperadas, impossibilidade na mudança de faixa, somente disponibiliza conteúdos previamente gravados, requer espaço em disco para armazenamento do conteúdo transmitido.

2.7 WEBSERVICE

Hoje em dia a tendência no desenvolvimento de sistemas são as aplicações voltadas para Web. Portanto, novos tipos de ferramentas e padrões estão surgindo e está aparecendo a necessidade de integrá-las. Essa complexidade é causada pela falta de um padrão para comunicação entre aplicações desenvolvidas em plataformas diferentes, contudo essa interligação pode ser atingida através da utilização dos *WebServices* (TOPLEY, 2002; OLIVEIRA; SOARES, 2005; YEO; YEUNG,1997).

Muitas empresas adotaram os *WebServices* com o intuito de melhorar a capacidade de colaboração entre empresas e processos, que são: redução da complexidade de soluções distribuídas; diminuição nos custos de desenvolvimento; agilidade na integração de sistemas e também na redução do consumo de banda (TOPLEY, 2002; OLIVEIRA; SOARES, 2005;YEO; YEUNG,1997).

WebServices são tecnologias utilizadas para integrar sistemas por meio da Internet, independentes de plataforma ou linguagem de programação. Isso é, são aplicações definidas e descritas usando a XML (*Extensive Markup Language*), que é uma linguagem de marcação projetada para a criação de documentos *Web*, contribuindo para a interoperabilidade entre os diferentes dados na *Web*, por isso é a linguagem que representa os *WebServices* (TOPLEY, 2002; OLIVEIRA; SOARES, 2005; YEO; YEUNG,1997).

Geralmente, é utilizado o protocolo HTTP para efetuar a comunicação entre o cliente e o *WebService*, mas também podem ser utilizados outros protocolos como SMTP e FTP.

Para troca de informações entre o cliente e o serviço Web é necessário a utilização do SOAP, que é um protocolo especializado em troca de conteúdo entre sistemas Web, o qual foi desenvolvido para ser utilizado em acessos a objetos remotos, através de mensagens XML, que contém as informações para serem transitadas entre os sistemas (TOPLEY, 2002; OLIVEIRA; SOARES, 2005; YEO; YEUNG,1997).

O WebService exibe sua interface para os usuários utilizando um documento XML conhecido como WSDL. Para definir uma maneira padrão de publicar informações e anunciar os serviços de um *WebService*, foi criado o UDDI, que funciona basicamente como um diretório de *WebService*, ou seja, trata-se de um cadastro global de serviços *Web*. Na figura 8 pode ser visualizado a arquitetura de um *WebService.*

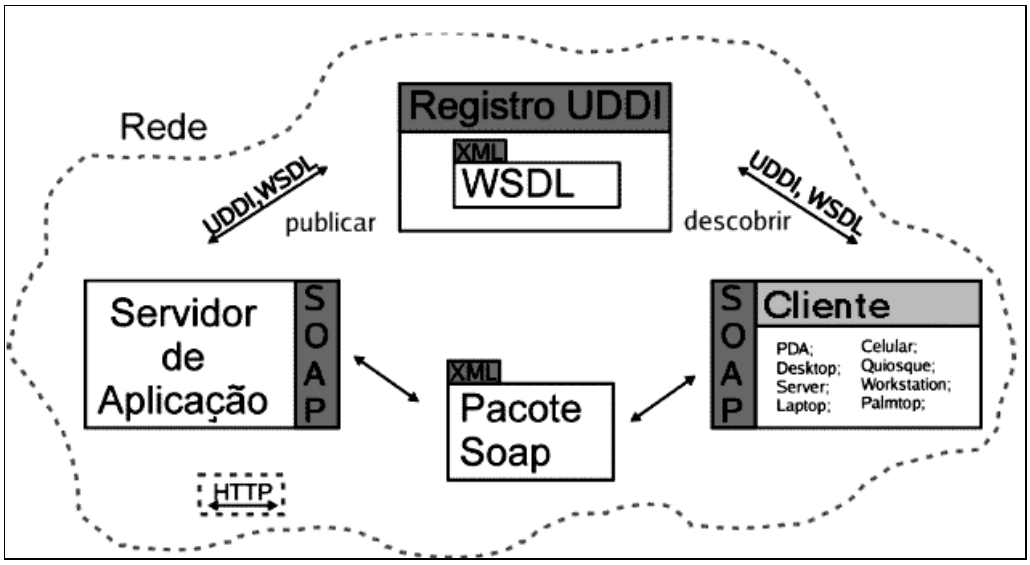

Figura 8 - Arquitetura de um *WebService*

No decorrer desse estudo foram desenvolvidas algumas formas para transmitir streaming de áudio e vídeo para DM, uma delas foi a transmissão do conteúdo multimídia via *WebService*, cuja idéia era transmitir o vídeo frame a frame e o DM fazer a recepção, porém o resultado não foi o esperado, já que a transmissão ficou lenta, isso devido aos arquivos XML trafegados que possuíam muitas informações desnecessárias.

Sendo assim, essa tecnologia ficou sendo utilizada apenas para busca de conteúdo multimídia, já que a resposta do serviço Web ao cliente seria apenas uma URL e o nome do vídeo.

2.8 *PLAYERS* EXISTENTES NO MERCADO

Existem algumas abordagens de transmissão e busca de informações multimídia em DM, desde pesquisas acadêmicas a soluções com fins lucrativos.

2.8.1 Real *Player*

Há mais ou menos quinze anos atrás o *RealPlayer* foi lançado para plataforma Windows e suas primeiras versões foram muito discutidas, causando debates sobre a

incompatibilidade com arquivos de áudio e vídeo mais utilizados na época. O *Player* se desenvolveu e as listas de formatos cresceram paralelamente, tornando o *RealPlayer* como um dos *players* mais completos no que se diz respeito a *streaming* de arquivos multimídia (Real Networks).

Com esse crescimento, hoje, o usuário, em seu celular, pode assistir *streaming* de vídeos, através de conexões wireless ou pela internet, arquivos que ficam persistidos em servidores Web ou até mesmo aulas tele presenciais.

O *RealPlayer Mobile* é um *player* de arquivos multimídia universal, no qual suporta a reprodução de formatos populares como: RealMedia, Windows Media, H264, MPEG-4, 3GP, AVI, MP3, AAC e outros (Real Networks).

O requisito mínimo para o *RealPlayer Mobile* é que o DM tenha o Symbian OS. O Real*Player* já vem de fábrica nos celulares da Nokia série 9200, 60 e nos Nokia 7650 e 3650.

O destaque desse *player* está na reprodução de arquivos via *streaming*, o qual pode ser tocado vídeos em tempo real, além disso, o programa permite conectar-se a estações de rádio online e canais multimídia cujo conteúdo são bastante amplos, sendo assim os DM não teriam capacidade para armazená-los.

O *RealPlayer*, quando aberto, possui algumas opções iniciais:

1. Vídeo Clipes: exibe todos os vídeos salvos na galeria de vídeo do DM;
2. Links *Streaming*: exibe todos os links RTSP utilizados, caso o usuário queira adicionar um novo link é necessário selecionar *opções > Novo Link* e entrar com um endereço RTSP válido;
3. *Download* Vídeos: exibe todos os vídeos que foram feitos através de *download* pela internet.
4. Reproduções recentes: exibe as últimas 6 reproduções mais recentes.

Entre as principais vantagens desse *Player* podem ser citados:

- É de uso gratuito;
- Reproduz arquivos de formatos populares: RealAudio, RealVideo, MP3 e 3GPP;
- Acessa rádios online;
- É compatível com o protocolo RTSP.

Já as desvantagens:

- O *player* possui poucas funções avançadas.

2.8.2 *YouTube Móbile*

O *YouTube* foi fundado por Chad Hurley e Steve Chen, esse site de vídeo foi criado em uma garagem de San Francisco (California, EUA), em Fevereiro de 2005. Inicialmente o sistema foi criado no intuito de compartilhar vídeos entre amigos, já que os arquivos digitais eram muito grandes para serem enviados via e-mail (MELO, 2008). Em outubro de 2006 a invenção foi comprada pela Google por U$$ 1,65 bilhões.

Segundo a *comScore*, no mês de janeiro de 2009 o número de pessoas que acessaram o *YouTube* nos EUA, para assistir vídeos, ultrapassou 100 milhões, isso representa 80% dos internautas americanos. Esse sistema ocupa o topo do ranking de vídeos online nos EUA, com 6,4 bilhões de clipes assistidos no primeiro mês desse ano.

Conforme o crescimento da empresa, do número de vídeos e acesso no site, o *YouTube* também investiu em uma versão para DM, no qual segundo o blog oficial da comunidade do ano 2009 ao 2010 houve um crescimento de 1700% em número de acesso na versão, e ainda um crescimento de 400%, por dia, no número de vídeos publicados pelo celular.

A versão *mobile* pode ser acessada através de site http://m.*YouTube*.com/, sua interface inicial possui uma lista com quatro vídeos, no qual é exibido através de frames dos arquivos de vídeos postados pelos clientes do sistema e todos no formato *Adobe Flash Vídeo*. Nessa interface também é possível fazer pesquisa pelo nome do vídeo. A interface principal pode ser visualizada na figura 9.

Figura 9- Interface inicial do *YouTube Mobile*

Os vídeos podem ser pesquisados e listados por quantidade de visibilidade, categoria e período. Todo vídeo adicionado no sistema é cadastrado em uma determinada categoria, no qual pode ser citada: esporte, educação, filmes, humor e outras categorias. A figura 10 ilustra os tipos de pesquisas que podem ser feitas pelo *YouTube Mobile.*

Figura 10 - Tipos de pesquisa do *YouTube Mobile*

Existe ainda a opção minha conta, no qual o usuário deve fazer um cadastro para assim acessar o sistema. Na figura 11 pode ser visualizada a conta de um usuário logado. Depois de logado, o usuário, tem as opções:

a) Meus Vídeos: exibe todos os vídeos adicionados pelo usuário logado.
b) Meus favoritos: exibe os vídeos favoritos do usuário logado.
c) Minhas listas de reprodução: exibe a lista de vídeo visualizada pelo usuário logado.
d) Minhas inscrições: exibe todas as inscrições do usuário logado.
e) Atividade do amigo: exibe todas as atividades dos amigos do usuário logado.
f) Caixa de entrada: exibe vídeos enviados pelos amigos do usuário logado.

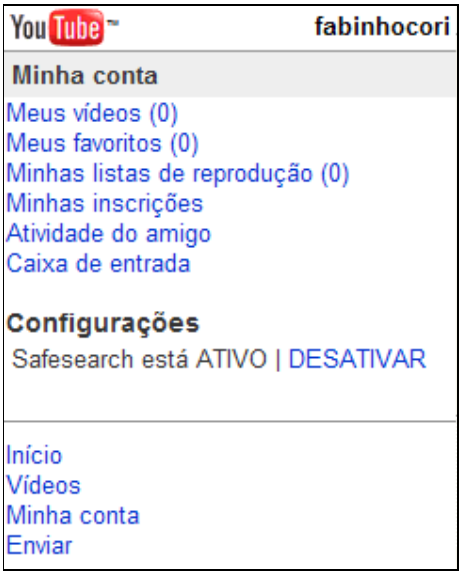

Figura 11 - Conta de um usuário logado

A figura 12 ilustra a aplicação do *YouTube Mobile* em um celular *Nokia s60,* que apresenta o menu inicial, onde o usuário pode buscar vídeo (*Search*), visualizar os vídeo mais populares (*Popular Videos*) e acessar o perfil da conta (*My Account*). Essa aplicação é gratuita e foi lançada pelo *YouTube* para os sistemas operacionais *Symbian* e *Windows Mobile.* O *download* dessa aplicação pode ser feito através do site http://www.m.*YouTube*.com. De acordo com o *YouTube*, os DM´s carregarão os vídeos do site do *YouTube Mobile* de forma mais rápida, pois foi melhorada a qualidade do *stream* e *buffering* da aplicação comparado ao portal para DM.

Figura 12 - Aplicação do *YouTube Mobile* em um celular Nokia s60

3 ARQUITETURAS DE TRANSMISSÃO E RECEPÇÃO DE STREAMING DE VÍDEO PARA DM

Nesse capítulo serão abordadas as tecnologias utilizadas para o desenvolvimento dos projetos, uma modelagem básica de um sistema móvel para transmissão e recepção de *streaming* de vídeo, três formas diferentes de desenvolver o mesmo problema seguindo a modelagem apresentada e por fim a comparação da melhor prática para transmissão e recepção de streaming de vídeo para DM.

Nas tecnologias serão apresentadas as linguagens de programação, ferramentas e API´s opcionais utilizadas no projeto. Inicialmente será feita uma breve introdução sobre a família Java, focando na API J2ME e API´s opcionais da mesma, depois o foco será nas ferramentas e protocolos utilizados na transmissão e recepção da mídia.

Na modelagem será feita uma breve documentação do que foi desenvolvido (funcionalidades, cenário e requisitos do sistema), como também o diagrama de caso de uso sobre as principais funcionalidades implementadas.

O primeiro estudo de caso foi desenvolvido no intuito de somar com projetos multimídia feito pelo grupo GAMA. A idéia era a criação de um *player mobile* para acessar um único servidor, que distribui o mesmo conteúdo para diferentes plataformas como: Web e *mobile*. O protocolo de transmissão foi o HTTP.

No segundo, o objetivo foi buscar um maior desempenho na transmissão dos dados entre cliente-servidor, foi utilizado a API opcional do J2ME, que é o MMAPI, mas o protocolo de transmissão foi o RTSP, assim foi desenvolvido um novo servidor e adicionado uma mídia específica no mesmo.

No terceiro, utilizou-se também o protocolo RTSP, entretanto foi desenvolvido um sistema Web para DM, no intuito de fazer busca dos vídeos do servidor e exibir os arquivos multimídia. Nos DM, ainda, foi utilizada a versão *mobile* do *Real Player*,

tecnologia que se encontra em uma quantidade significativa dos celulares que possui suporte ao protocolo RTSP.

Por fim foi feito um quadro comparativo entre as formas de transmissão apresentadas nesse trabalho e conclusões do melhor meio de transmissão e recepção de dados em DM.

3.1 TECNOLOGIA UTILIZADA

Nessa seção serão apresentadas as tecnologias utilizadas nos projetos, na qual será mencionada a linguagem J2ME e suas API's opcionais.

Para desenvolvimento em DM foi escolhido o J2ME pelos seguintes motivos: por ser uma versão reduzida do J2SE; por ser uma linguagem de fácil domínio, portátil, segura e cerca de dois milhões de programadores fazem uso dela; por ter sido criada especificamente para desenvolver aplicativos em DM com as vantagens da plataforma Java.

A plataforma J2ME é dividia em configurações e perfis e como o projeto será focado para dispositivos CLDC (celulares), foi dado enfoque ao perfil MIDP, que é dividido em interface de alto nível e baixo nível (EVANGELISTA; GUARDIEIRO, 2008).

Na interface de alto nível, existem as classes *Display* e *Displayable*; deriva dessa interface a classe *Screen*, que possui os seguintes componentes: *TextBox*, *List*, *Alert*, *Form*.

Já a interface de baixo nível, é utilizada para o desenvolvimento de aplicativos gráficos, como jogos e menus com animação, e nela é utilizada a classe *canvas*. Na figura 13 podem ser visualizadas as classes do perfil MIDP.

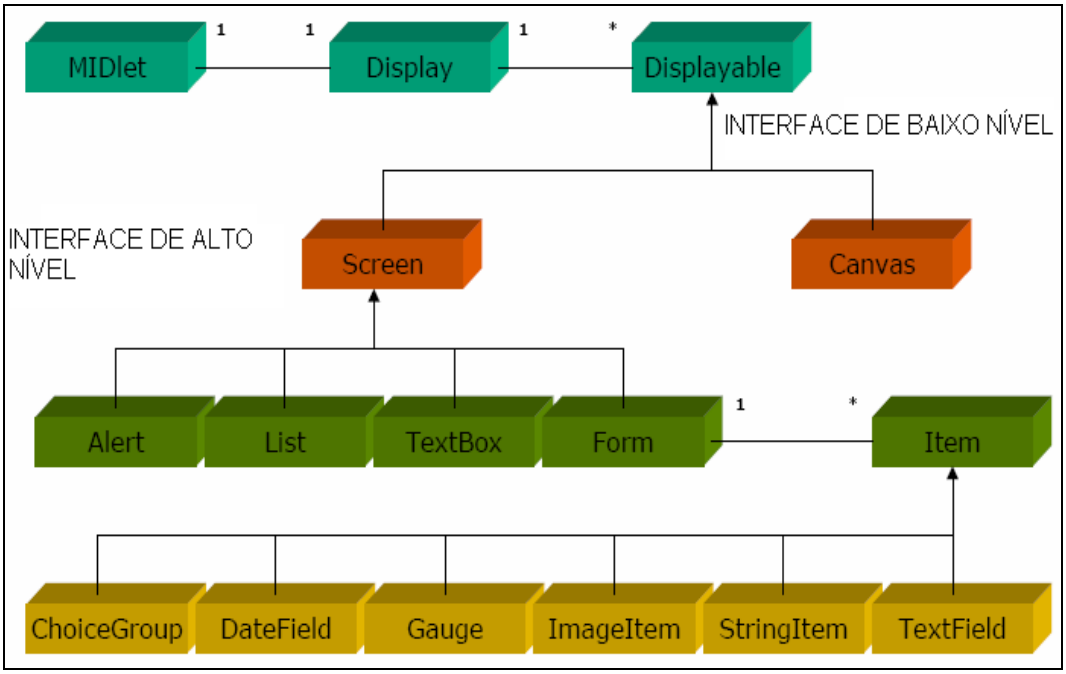

Figura 13 - Classes do perfil MIDP

No desenvolvimento dos projetos foram utilizadas API's opcionais do J2ME, as quais auxiliaram para: conexão dos DM com o servidor de vídeo; *download* e persistência dos vídeos nos aparelhos móveis; recepção do *streaming* de vídeo no *player*; layouts mais profissionais.

Para recepção do streaming de vídeo no player móvel foi utilizada a MMAPI, que é uma API opcional da plataforma J2ME. Ela estende opções de áudio, vídeo e outros recursos multimídia existentes nos aparelhos móveis (GOYAL, 2006). Com esse pacote os desenvolvedores *Java mobile* conseguem acesso a recursos multimídia nos dispositivos que suportam essa tecnologia.

O *MMAPI* trabalha de maneira bem similar ao *JMF*, API Java utilizada para aplicações multimídia, cuja função é reproduzir áudio e vídeo independente de protocolo (Mobile Media API SUN). A idéia do MMAPI é reproduzir arquivos multimídia sem especificar o protocolo de transmissão ou formato de arquivo trafegado.

As aplicações móveis que utilizam essa API podem receber dados multimídia por diferentes plataformas, por exemplo: através do arquivo JAR da própria MIDlet, na rede por intermédio de protocolos HTTP e ou RTSP, e até mesmo por meio de uma captura de vídeo feito pelo próprio aparelho (GOYAL, 2006) Mobile Media API SUN).

Para transmissão de um vídeo no MMAPI, o conteúdo multimídia passa por quatro elementos: *DataSource, Player, Manager* e *Control*. Para manter a portabilidade nas plataformas de dados, o MMAPI utiliza dois elementos: *Player* e o *DataSource*. O primeiro é uma interface que lida com o processamento de dados multimídia e o segundo é responsável por capturar as informações multimídia, seja através de um sistema de arquivos, de um protocolo padrão, ou de alguma estrutura proprietária, tudo isso tem o objetivo de manter independência no protocolo. Desta forma, toda a complexidade de acesso à mídia fica encapsulada no *DataSource*. O *Player* está definido no pacote *javax.microedition.media*, e o *DataSource* é definido em *javax.microedition.media.protocol* (GOYAL, 2006); Mobile Media API SUN. Na figura 14 pode ser visualizada a arquitetura desses elementos.

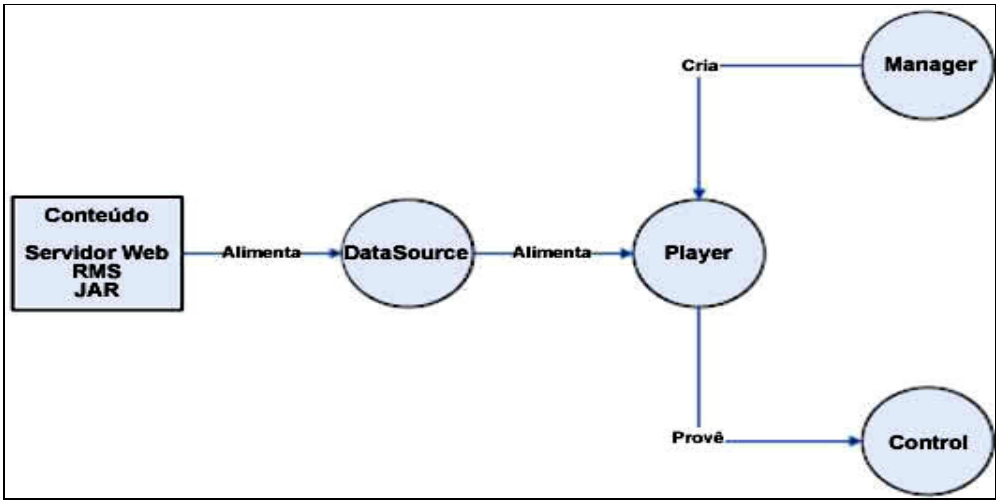

Figura 14 - Arquitetura da MMAPI

Para persistência dos vídeos nos aparelhos móveis foi utilizado *Record Management System* e *Floggy*.

O *RecordStore* é um banco de dados orientado para registros. Pode ser comparado com uma série de registros em uma tabela, e cada um com seu identificador exclusivo. Nesse repositório de dados é onde são armazenados os vídeos exibidos no *Player* do DM, caso os usuários desejem persistir o *streaming* (ZHIHUA; HUANG; BENSHUN, 2008; TOPLEY, 2002), Mobile Media API SUN. Na figura 15 pode ser visualizada a estrutura de um RMS, no qual o campo id representa a chave única e o vetor de Byte representa o conteúdo que será adicionado no registro.

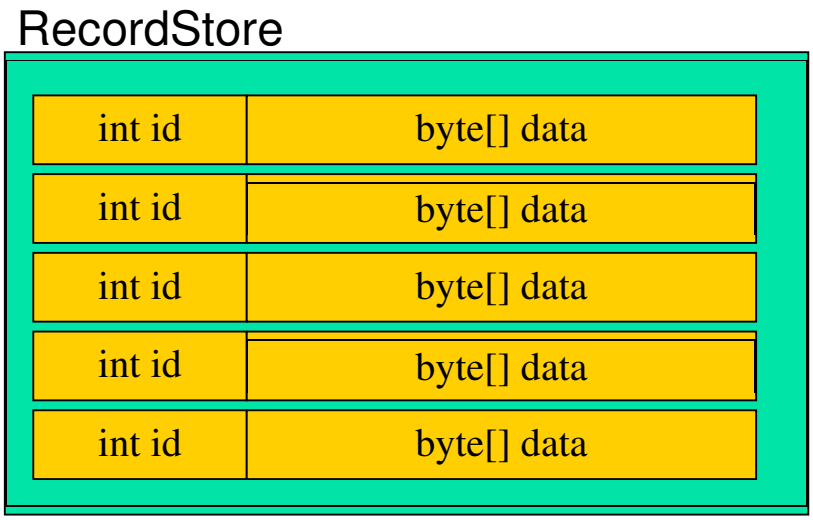

Figura 15 - Estrutura de um *RecordStore*

Para auxiliar no armazenamento dos dados foi utilizado *Floggy*, que é um *framework* de persistência *free* para J2ME/MIDP, desenvolvido por brasileiros. Seu principal objetivo é abstrair os detalhes da persistência de dados para o desenvolvedor, reduzindo os esforços de desenvolvimento e manutenção. O *Floggy* foi desenvolvido para agilizar o desenvolvimento de aplicações MIDP, pois estas aplicações podem fazer uso das facilidades oferecidas por este framework de persistência. Desta forma o desenvolvedor pode focar seus esforços na solução do problema principal a ser tratado pela aplicação, delegando toda a persistência dos dados ao framework.

Na figura 16 é apresentada a utilização do *Floggy* em uma aplicação. No início do processo os arquivos fontes são compilados (javac) e transformados em arquivos de classes (bytecode). Estas classes são analisadas pelo compilador do *Floggy* (floggy-compiler), que gera e adiciona o código de persistência às classes consideradas persistentes. Posteriormente, as classes passam pelo processo de pré-verificação (preverify) e são empacotadas (jar) em um arquivo Java (jar) junto com o módulo Framework. Este arquivo é instalado e executado nos dispositivos MIDP. Durante a execução da aplicação, as operações de persistência são gerenciadas pelo módulo *Floggy*, que utiliza os Record Stores para armazenar os objetos (LUGON; ROSSATO, 2005).

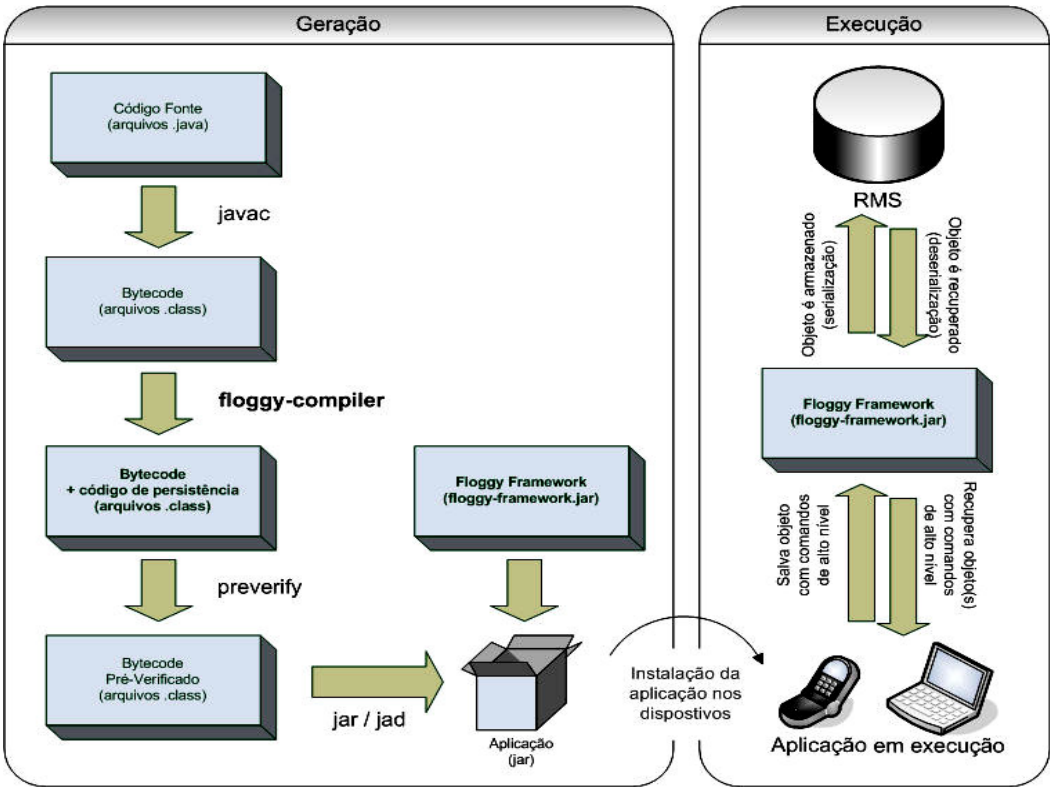

Figura 16 - Utilização do *Floggy* em uma aplicação (Floggy – J2ME persistence framework)

No intuito de obter interfaces gráficas mais atraentes e profissionais foi utilizada a biblioteca LWUIT, a qual possui código aberto, componentes visuais ricos, maior flexibilidade e facilidade do desenvolvimento em J2ME (LWUIT: Home).

Entre as principais funcionalidades que o LWUIT disponibiliza, estão (LWUIT: Home):

a) A utilização de vários layouts para a organização interna dos componentes gráficos;
b) A possibilidade da utilização de recursos em 3D;
c) Hierarquia simples de classes de componentes;
d) Garante a portabilidade entre diversos dispositivos;
e) Efeitos de transição em 2D e 3D entre as diferentes telas do aplicativo;
f) Suporte a trocas de temas similares ao CSS para Web;
g) Suporte a tecnologia *Touch Screen* sem a necessidade de mudanças no código
h) Acesso ao código fonte, o que garante uma maior portabilidade das aplicações.

Para acesso ao servidor Web foi utilizado *WebService, que* é independente de plataforma, assim esse tipo de serviço pode ser consumido em Java e até mesmo em J2ME, isso graças à interoperabilidade dos serviços que ficam disponíveis a diferentes aplicações.

A API Java para DM possui total suporte à tecnologia de *WebService* e seu uso é recomendado, pois os serviços criados podem ser consumidos em outras plataformas, possibilitando a integração de aplicações *Web*, *Desktop* e *Mobile* (MADEIROS; PETROLI, 2008; TOPLEY, 2002; OLIVEIRA; SOARES, 2005).

A comunicação com *WebService* é feito através de *XML* e do protocolo SOAP. O J2ME não possui uma classe específica para tratar estas implementações, por isso a necessidade da utilização de duas *API´S* específicas, que são: KSOAP e KXML (EVANGELISTA; GUARDIEIRO, 2008; OLIVEIRA; SOARES, 2005). A primeira é responsável por oferecer suporte ao protocolo SOAP e a segunda é responsável por dar

suporte à linguagem XML. A figura 17 ilustra a arquitetura do ambiente cliente-servidor utilizando *WebService* e J2ME.

Na camada do serviço WEB, o *Axis* é utilizado para construção de serviços WEB no padrão SOAP. O Apache *Axis* é um framework de código aberto, baseado na linguagem Java e no padrão XML, utilizado para construção de Web services no padrão SOAP. Através do *Axis,* os desenvolvedores podem criar aplicações distribuídas. Além da versão para Java, existe uma implementação baseada na linguagem C++.

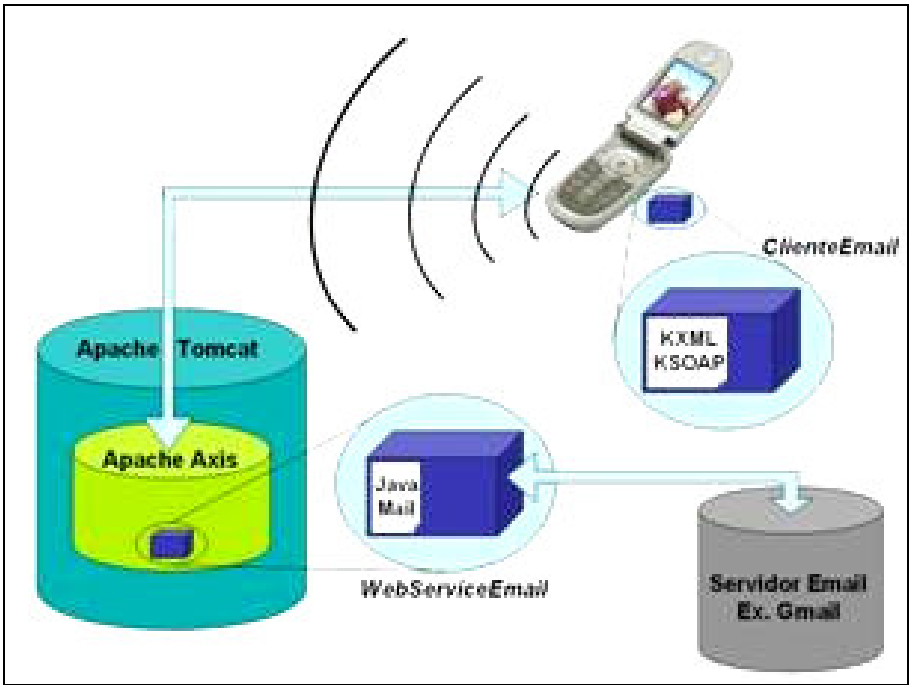

Figura 17 - Arquitetura do ambiente cliente-servidor utilizando *WebService* e J2ME

3.2 DOCUMENTAÇÃO DO SISTEMA

Nessa seção será apresentada a documentação do projeto desenvolvido, a qual está dividida em: ambiente, que tem como objetivo apresentar o que é e qual o motivo do

desenvolvimento do projeto; diagrama de caso de uso, cuja finalidade é apresentar as principais funcionalidades do sistema.

3.2.1 Ambiente

O sistema consiste em um *player* para DM, cujo intuito é acessar um servidor Web e executar o conteúdo multimídia utilizando a tecnologia de *streaming*.

Com a utilização dessa aplicação, o cliente poderá assistir *streaming* de vídeo através de um servidor Web por intermédio de DM. Esse conteúdo multimídia pode ser acessado por meio de uma pesquisa, desenvolvida com WebService, através do nome do vídeo persistido no servidor. Os vídeos tocados pelo *Player* podem ser persistidos no DM, através de RMS, e esses dados podem ser exibidos nos DM, como também podem ser excluídos pelo usuário.

3.2.2 Diagrama de Caso de Uso

Os casos de usos referem-se aos serviços, tarefas ou funções que podem ser utilizados de alguma maneira pelos usuários do sistema, como emitir um relatório, fazer uma pesquisa, apagar um registro. Os casos de uso são utilizados para expressar e documentar os comportamentos para as funções do sistema.

A figura 18 ilustra a visão integrada do caso de uso e os Quadros 3 à 8 são apresentadas as descrições textuais dos casos de uso.

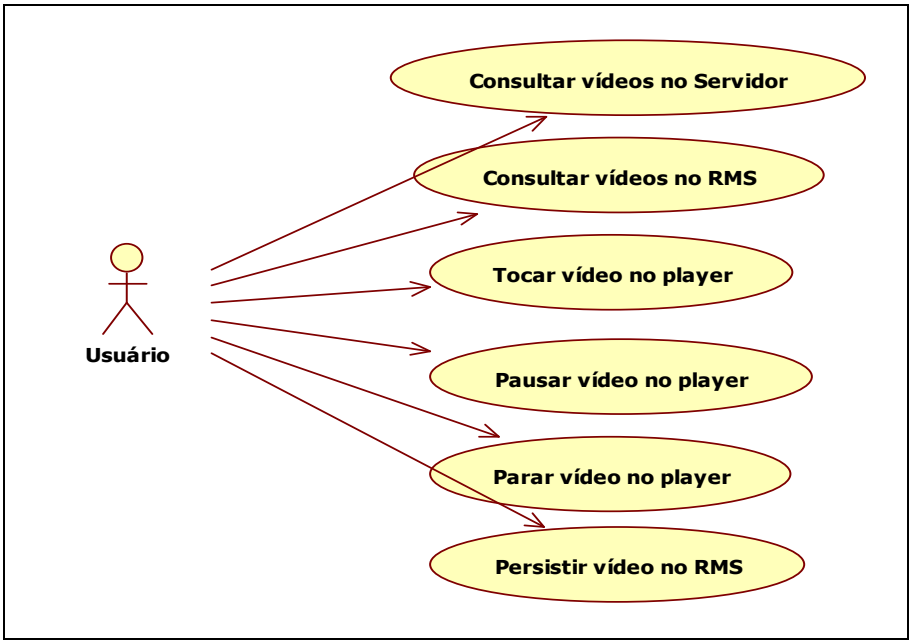

Figura 18 - Visão integrada do caso de uso

Quadro 2 - Discrição do caso de uso Consultar vídeos no Servidor

Nome do Caso de Uso	Consultar vídeos no Servidor
Caso de uso Geral	
Ator Principal	Usuário
Atores Secundários	
Resumo	Este caso de uso descreve as etapas percorridas por um usuário para consultar os vídeos de um servidor para aplicação móvel.
Pré-Condições	Servidor Web deve estar conectado
Pós-Condições	
Ações do ator	Ações do sistema
1. Selecionar a opção busca de vídeo no Servidor	
	2. Entrar com o nome do vídeo a ser pesquisado.

3. Aceitar a conexão com o servidor Web	
	4. O servidor Web vai receber o nome do arquivo.
	5. Processa a informação e retorna a lista de vídeo com nome pesquisado
6. O usuário escolhe o vídeo a ser tocado	
	7. É feito uma nova conexão com o servidor para ser feito o *streaming* do vídeo.
	8. Deverá ser chamada a tela de listagem de vídeo
Restrições/Validações	
	1. Se o usuário não aceitar a conexão com o servidor a pesquisa não é feita
	2. Se houver problema na conexão com o servidor a *streaming* ou a pesquisa é cancelada.

Quadro 3 - Discrição do caso de uso Consultar vídeos no RMS

Nome do Caso de Uso	Consultar vídeos no RMS
Caso de uso Geral	
Ator Principal	Usuário
Atores Secundários	
Resumo	Este caso de uso descreve as etapas percorridas por um usuário para consultar os vídeos do repositório de dados do DM
Pré-Condições	
Pós-Condições	
Ações do ator	Ações do sistema
1. Selecionar a opção busca de vídeo no Servidor	
	2. Entrar com o nome do vídeo a ser pesquisado.

3. Processa a informação e o RMS retorna a lista de vídeo com nome pesquisado	
4. O usuário escolhe o vídeo a ser tocado	
	5. Deverá ser chamada a tela de listagem de vídeo
Restrições/Validações	

Quadro 4 - Discrição do caso de uso tocar vídeo no player

Nome do Caso de Uso	Tocar vídeo no *Player*
Caso de uso Geral	Consultar vídeo no servidor ou Consular vídeo no RMS
Ator Principal	Usuário
Atores Secundários	
Resumo	Este caso de uso descreve as etapas percorridas por um usuário tocar o vídeo no *player* da aplicação móvel.
Pré-Condições	Ter pesquisado um vídeo no servidor ou RMS
Pós-Condições	
Ações do ator	Ações do sistema
1. Selecionar um vídeo pesquisado	
	2. Escolher a opção play para tocar o vídeo.
	3. O vídeo será tocado *player*
Restrições/Validações	
	1. O sistema só vai para tela do *player* caso tenha escolhido um vídeo

Quadro 5 - Discrição do caso de uso pausar vídeo no Player

Nome do Caso de Uso	Pausar vídeo no *Player*
Caso de uso Geral	Tocar vídeo no *Player*
Ator Principal	Usuário
Atores Secundários	
Resumo	Este caso de uso descreve as etapas percorridas por um usuário para pausar os vídeos no *player*.
Pré-Condições	Vídeo deve estar sendo tocado no *player*
Pós-Condições	
Ações do ator	Ações do sistema
1. Escolher a opção pause no *player*	
	2. O vídeo é pausado
Restrições/Validações	

	1. O botão pause só é habilitado, caso o vídeo esteja tocando no *player*.

Quadro 6 - Discrição do caso de uso parar vídeo no Servidor

Nome do Caso de Uso	Parar vídeo no *Player*
Caso de uso Geral	Tocar vídeo no *Player*
Ator Principal	Usuário
Atores Secundários	
Resumo	Este caso de uso descreve as etapas percorridas por um usuário para parar de tocar um vídeo no *player*.
Pré-Condições	Vídeo deve estar sendo tocado no *player*
Pós-Condições	
Ações do ator	Ações do sistema
1. Escolher a opção *stop* no *player*	
	2. O vídeo é parado
	3. É chamada a tela principal da aplicação
Restrições/Validações	
	1. O botão *stop* só é habilitado, caso o vídeo esteja tocando no *player*.

Quadro 7 - Discrição do caso de uso Persistir vídeo no RMS

Nome do Caso de Uso	Persistir vídeo no RMS
Caso de uso Geral	Tocar vídeo no *Player*
Ator Principal	Usuário
Atores Secundários	
Resumo	Este caso de uso descreve as etapas percorridas por um usuário para parar persistir um trecho de um vídeo no repositório de dados de um aplicativo móvel.
Pré-Condições	Vídeo deve estar sendo tocado no *player*
Pós-Condições	
Ações do ator	Ações do sistema

1. Escolher a opção *rec* no *player*	
	2. O trecho de vídeo é gravado até o usuário selecionar *rec* novamente
Restrições/Validações	
	1. O botão *rec* só é habilitado, caso o vídeo esteja tocando no *player*.

3.3 JDRAGON *PLAYER MOBILE* (ESTUDO DE CASO 1)

Nessa seção é feita uma breve introdução sobre a criação do *player*, logo depois serão explicados os recursos e tecnologia utilizada para o desenvolvimento do aplicativo e, por fim, será apresentada a arquitetura e limitações do software.

3.3.1 Introdução

Para desenvolvimento do primeiro caso de uso foi desenvolvido um *Player Mobile* próprio, isso é sem utilização de API's multimídia opcionais, tudo isso pensando em uma maior integração com a arquitetura do servidor e logo, uma melhor portabilidade nas aplicações.

Foi implementado um módulo de pesquisa de vídeo, que pode ser feito através de RMS ou do servidor Web. Pelo servidor, a pesquisa é feita através de JSP e WebService, que recebe do DM o nome do vídeo como parâmetro. No RMS, é feita através de algoritmos de busca, pois o repositório de dados se encontra local no dispositivo. Ambos retornam uma lista de vídeo do nome pesquisado, ou, caso o cliente não passe nenhum nome, o sistema retorna todos os vídeos da base.

Para armazenamento e visualização nos DM, inicialmente, é adicionada a *stream* na memória do aparelho, formando um buffer de armazenamento. Esse *buffer* de armazenamento tem por objetivo controlar as imagens na memória do dispositivo e nunca permitir seu estouro, para que, à medida em que é feito o *download* do vídeo, o mesmo seja exibido no display do aparelho. Caso aconteça o estouro de memória, o vídeo é persistido no repositório de dados do dispositivo o RMS.

No *Player Mobile*, há a opção “pause”, cujo objetivo é parar o vídeo e a opção de gravação, a qual é feita através de RMS, repositório de dados dos DM (por ser limitado, só pode ser feita uma gravação de poucos minutos).

3.3.2 Recursos

Como recursos foram utilizados tecnologias open source ou *free* software.

Como linguagem de programação foi utilizada Java tanto na programação da camada servidora como receptora. No caso da segunda, como é uma aplicação móvel foi empregado a plataforma Java para DM, que é o J2ME. Para implementação dos códigos foram utilizados as IDE's *eclipse* e *NetBeans*.

A aplicação servidora visa controlar o acesso, realizar consultas e transmitir os dados multimídia para os clientes móveis de um base de dados relacional, que neste caso é o *PostGreSQL* e é onde está armazenado todo conteúdo multimídia.

O conteúdo multimídia segue o padrão MPEG-7. Esse padrão oferece uma vasta gama de ferramentas de descrição padronizadas, criando anotações em recursos audiovisuais, ou seja, descrevendo padrões de texturas, cor, áudio, objetos em movimento, operações de câmera, dentre outros. Observa-se que este padrão descreve como os recursos de mídia devem ser representados (SANTOS; SANTOS; TAVARES, 2007 ; YEO,YEUNG,1997).

Para acesso à base de dados foi utilizado a tecnologia JSP e WebService. A primeira será responsável pelo acesso do conteúdo multimídia por meio de um browser. A segunda, por seu turno, permitirá que os usuários acessem as informações dos conteúdos através dos DM.

Na implementação do *player* móvel foi utilizada a API de alto e baixo nível da tecnologia J2ME. No caso da primeira os componentes visuais foram utilizados para

transição de formulários (telas) na aplicação. Já a segunda foi empregada para criação do layout do *player* e dos botões da aplicação, como *pause, stop, rec e play*.

Para persistência do conteúdo multimídia nos DM´s foi utilizado o RMS, que é considerado o banco de dados para J2ME.

3.3.3 Arquitetura

Todo conteúdo multimídia está armazenado em uma base de dados relacional, utilizando o SGBD *PostgreSQL*, no qual está utilizando a especificação do padrão MPEG-7 e é responsável por definir regras para a criação de *metadados* sobre o conteúdo multimídia. Estas informações serão utilizadas na busca do conteúdo.

Para o acesso ao conteúdo, foi desenvolvida uma API em Java chamada *DataAccess*, que utiliza o padrão DAO. Ela é responsável por toda comunicação entre a aplicação e a base de dados do MPEG-7.

A busca do conteúdo é realizada através da Web com implementações baseadas na tecnologia *WebService*. A figura 19 ilustra o diagrama de classe do *DataAccess*.

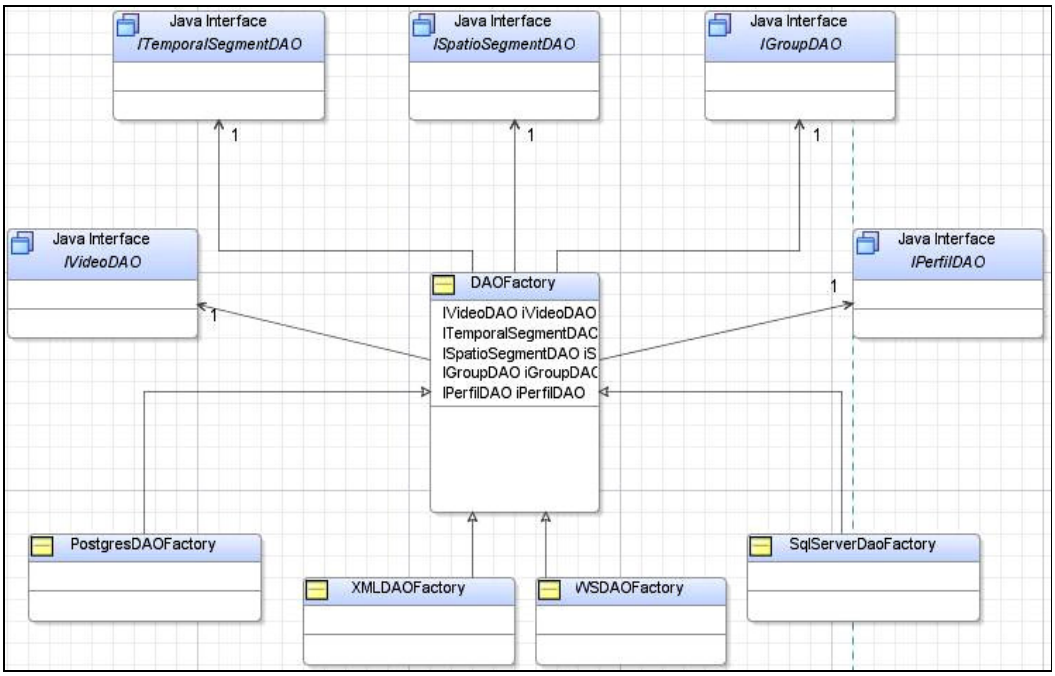

Figura 19 - Diagrama de classe da arquitetura de distribuição *dataAccess*

Esta arquitetura é composta por um servidor central e/ou vários servidores de replicação, os quais são sincronizados automaticamente para garantirem a unicidade do conteúdo. Todavia, é função do servidor central realizar as consultas dos usuários e o controle de acesso, o log e gerenciamento dos outros servidores existentes. Na figura 20 pode ser visualizada a arquitetura do servidor.

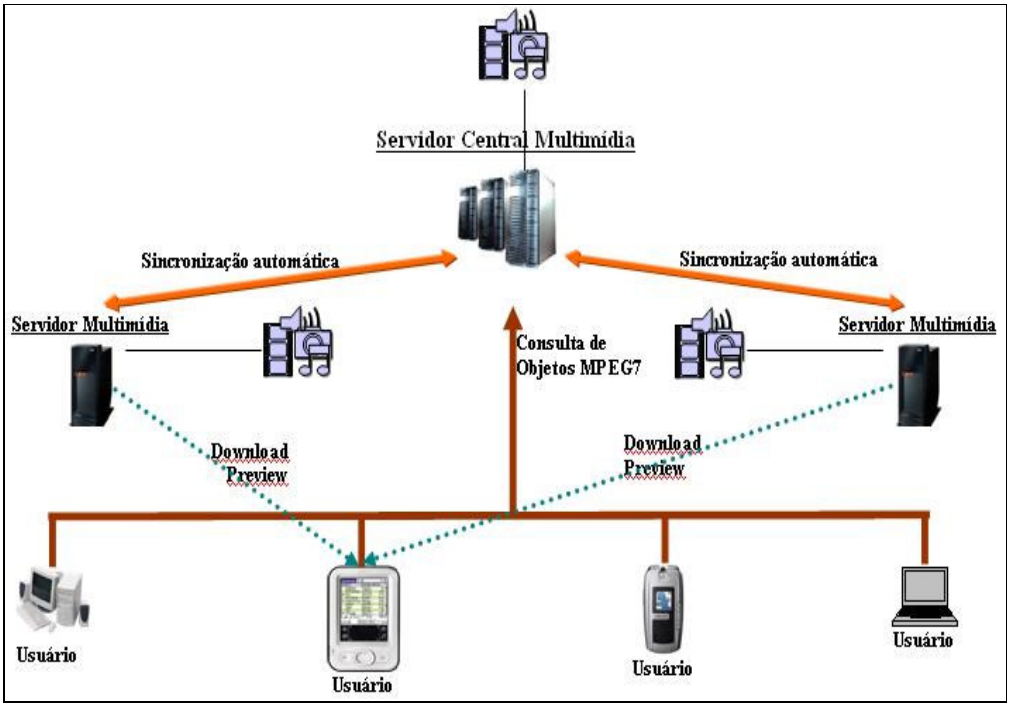

Figura 20 - Arquitetura do servidor

Quando um usuário solicitar a exibição de determinado conteúdo multimídia ao servidor, será feito o *download* da mídia de forma particionada entre os servidores que a possua, tornando o processo mais rápido e eficiente. O número de servidores participantes do *download* poderá ser ajustado pelo usuário na medida de sua necessidade.

Uma vantagem dessa proposta é a arquitetura versátil do sistema, pois os dados do mesmo servidor podem ser acessados tanto em plataforma Web, como móvel.

3.3.4 Desenvolvimento

No desenvolvimento para DM foi utilizado a plataforma J2ME. E em virtude da arquitetura da aplicação, para esse primeiro estudo de caso, foi mais conveniente desenvolver um *Player Mobile* próprio.

A implementação do *Player* móvel consiste, basicamente, em fazer o *download* de vários frames e organizá-los de forma seqüencial no DM. Para que isso funcione, foi necessário utilizar a classe *HttpConnection* e o *InputStream*; este, para fazer a conexão com o WebService, e, aquele, para receber os frames enviados.

Para armazenamento e visualização nos DM, inicialmente, são adicionados frames na memória do aparelho, formando dois buffers: o primeiro para armazenamento de imagens, o qual possui como objetivo controlá-las na memória do dispositivo e nunca permitir seu estouro, para que, à medida em que é feito o *download* do vídeo, o mesmo seja exibido no display do aparelho; O segundo para armazenamento do áudio e segue o mesmo procedimento do buffer de imagem. Caso aconteça o estouro de memória, o vídeo e a imagem são persistidos no repositório de dados do dispositivo - RMS. Na figura 21 pode ser visualizado o envio de dados do servidor para o DM.

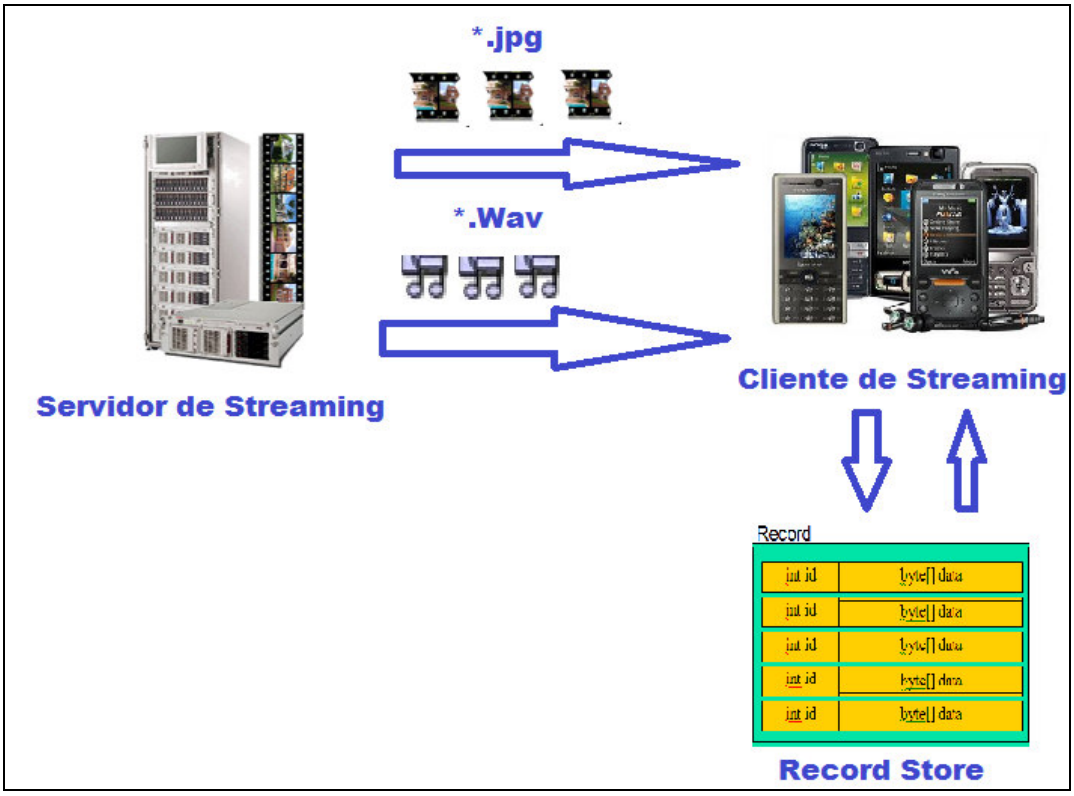

Figura 21 - Envio dos dados do servidor WEB para o DM

No *Player Mobile*, há a opção “pause”, cujo objetivo é carregar todo o vídeo para então assisti-lo, e a opção de gravação, a qual é feita através de RMS, repositório de dados dos DM.

3.3.5 Limitações

Uma limitação desse *player* é o sincronismo do áudio com o vídeo, já que foi desenvolvido um *player* próprio, sem API´s adicionais.

Esse problema acontece quando é feito o *buffer* do áudio e do vídeo para um determinado segmento de tempo e esses dados são exibidos no DM, entretanto o player desenvolvido apresentou falhas e em certos momentos o vídeo foi transmitido mais rápido do que o áudio.

3.3.6 Projeto de Interfaces

Na figura 22 pode ser visualizada a lista de vídeos enviada pelo servidor, a qual foi desenvolvida com API de alto nível do J2ME. Para listagem dos vídeos foi utilizado o *choiceGroup*, através de uma lista exclusiva, que só permite selecionar um item por vez. Para transição de tela foi utilizado o componente *command*, que é menu da aplicação.

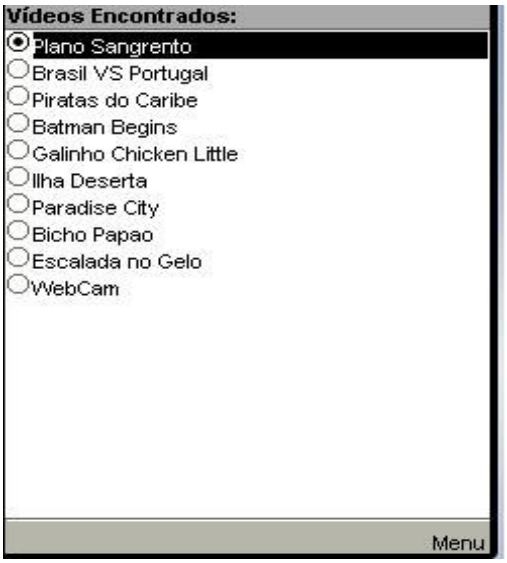

Figura 22 - Lista de vídeo pesquisada pelo servidor

Na figura 23 é ilustrado o player desenvolvido para essa arquitetura que foi denominado de JDragon *Player Mobile*, com as opções de *play, pause, stop e record*, no qual foi desenvolvido utilizando API de baixo nível (*canvas*). Para exibir o vídeo foi adicionado um componente de imagem, na qual é mudado na medida em que o vídeo está sendo tocado e para as opções foram adicionados botões que são acionados através do método *KeyPressed* do *canvas*.

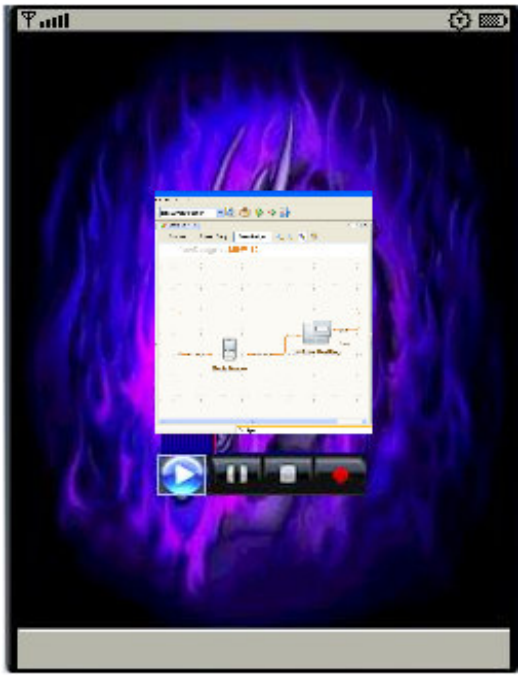

Figura 23 - *JDragon Player Mobile*

3.4 JDRAGON MMAPI *PLAYER MOBILE* (ESTUDO DE CASO 2)

Para corrigir o problema do sincronismo do áudio e vídeo foi criado um novo *Player Mobile*, nesse também foi utilizada a plataforma J2ME, mas foi empregado o pacote adicional multimídia, MMAPI, cuja principal função é reprodução de áudio e vídeo em DM e também um novo protocolo de transmissão, que é o RSTP.

3.4.1 Introdução

As informações multimídias são enviadas para seus receptores através do protocolo RTSP, que é baseado no UDP, suportada pela tecnologia J2ME. Uma grande vantagem no uso desse protocolo é a transmissão de conteúdo em tempo real. Um exemplo seria transmissão de uma aula tele presencial, onde os alunos poderiam assisti-la nos seus DM's ao vivo.

Desenvolver um *Player* de vídeo utilizando MMAPI não é muito diferente de como se desenvolve na API JMF, porque a MMAPI permite o acesso a vários protocolos e formatos diferentes. Basta adicionar um serviço RTSP válido e criar uma *Thread* para

executar o *Player*, uma vez que acessar o *Web Service* em uma aplicação J2ME exige processamento paralelo.

Essa arquitetura continua com as mesmas funcionalidades da apresentada na sessão anterior, contudo com formas de acesso diferente, já que foram utilizadas API's opcionais para correções dos bugs da versão anterior.

3.4.2 Recursos

Continuaram sendo utilizadas as tecnologias open source ou *free* software e a linguagem J2ME, contudo foi utilizando a MMAPI, que facilita o acesso a mídia e não tem problemas com o sincronismo do áudio e vídeo, já que essa API possui um Player próprio.

A aplicação servidora foi mudada, pois agora o protocolo de envio é o RTSP, que será responsável por fazer o *streaming* dos dados multimídias com o DM, por isso foi utilizado o DSS, que é um servidor para *streaming* de vídeo de código aberto.

O banco de dados utilizado foi o MySQL. Isso porque é o banco de dados nativo do DSS.

Para acesso à base de dados foi utilizado a tecnologia *Java* junto com *WebService*. Uma vantagem desse padrão é que pode ser feito chamada do serviço de qualquer aplicação, não importando a linguagem de desenvolvimento.

Na aplicação móvel foram empregados os mesmos recursos do estudo de caso anterior, porém com as API's: MMAPI, para recepção do conteúdo multimídia vinda do servidor; LWUIT*,* para facilitar a criação da interface de comunicação com o usuário; *FLOGGY,* que é um *framework* que facilita a persistência de dados no RMS*;* KSOAP *e* KXML*,* para acesso ao *WebService*.

3.4.3 Arquitetura

Para esse estudo de caso a arquitetura foi a grande diferença, pois na aplicação móvel foi utilizado o MMAPI e na servidora o DSS, que possui suporte ao protocolo RTSP.

Todo conteúdo multimídia está armazenado em disco e gerenciado pelo DSS e as URL's estão persistidas no MySQL, com isso a pesquisa dos vídeos é feita através de *WebService* ao MySQL, que retorna todos os links pesquisados pelo usuário, para serem visualizados no *Player*. A figura 24 ilustra a busca dos dados no MySQL e a exibição do vídeo no DM.

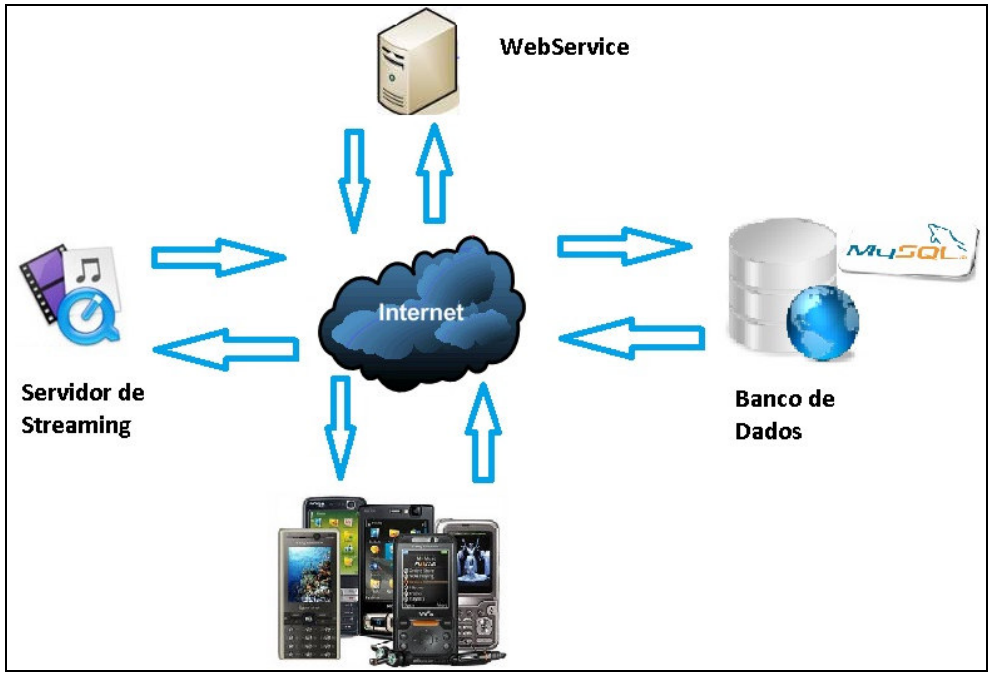

Figura 24 - Transmissão das informações na arquitetura do sistema

Como ilustrado na figura 20, esta arquitetura é composta por um servidor de *streaming* de vídeo, um *WebService* para busca dos vídeos na base de dados e o *Player* móvel desenvolvido com MMAPI para recepção das informações nos DM's.

3.4.4 Desenvolvimento

No desenvolvimento para DM foi utilizado também plataforma J2ME, porém o *Player* foi desenvolvido com a API opcional MMAPI, resolvendo assim o problema de sincronismos do áudio com o vídeo.

No desenvolvimento do *player* foram utilizados quatro *API's* opcionais que são: *LWUIT*, *FLOGGY, KSOAP e KXML*.

No intuito de uma melhor aparência gráfica na aplicação móvel foi utilizado o LWUIT, que tem como objetivo facilitar o desenvolvimento de aplicações móvel. Essa biblioteca é baseada no SWING, do Java, e permite ao desenvolvedor criar interfaces bem atraentes e ainda tem integração a ilustrações 3D e suporte a *touch Screen* (LWUIT:Home).

No caso do *FLOGGY*, esse tem como objetivo facilitar a persistência dos vídeos nos DM. Essa biblioteca, que foi desenvolvida por brasileiros (Floggy – J2ME persistence framework), abstrai os detalhes da persistência de dados para o desenvolvedor, reduzindo os esforços de desenvolvimento e manutenção com o RMS (Floggy – J2ME persistence framework).

O *KSOAP* e o *KXML* servem para o DM fazer pesquisa via *WebService* dos vídeos que existem no servidor, para assim serem tocados no *Player* (KSOAP; KXML).

O *Player Mobile* continua com a opção "*rec*", só que dessa vez os vídeos são persistidos com o auxilio da API opcional *FLOGGY*.

3.4.5. Limitações

O *player* corrigiu o erro de sincronismos e seguiu os requisitos do projeto, porém o conteúdo travava no medida em que era exibido ao cliente, isso sendo testado em uma rede interna.

3.4.6 Projeto de Interfaces

Na figura 25 pode ser visualizada a lista de vídeos enviada pelo servidor. Diferente da proposta anterior, essa foi desenvolvida utilizando a API opcional LWUIT. Para listagem dos vídeos foi utilizado o componente *List*, através de uma lista exclusiva, já para transição de tela foi utilizado o componente *command*, o qual executa um vídeo selecionado ou volta para o formulário principal da aplicação. Importante mencionar que a listagem dos vídeos foi recuperada do servidor através das API's: KSOAP e KXML.

Figura 25 - Lista de vídeo pesquisada pelo servidor e exibida com LWUIT

O *JDragon MMAPI Player Mobile* possui a mesma interface do *player* anterior, mas no desenvolvimento do *player* foi utilizado a biblioteca multimídia opcional MMAPI.

3.5 JDRAGON REAL*PLAYER MOBILE* (ESTUDO DE CASO 3)

Nesse terceiro estudo de caso continuou o uso do DSS e do WebService, contudo na plataforma *mobile* foi desenvolvida uma página Web que faz chamada do *Real Player*

Mobile, que é um aplicativo presente na maioria dos celulares que possui suporte ao protocolo RTSP.

3.5.1 Introdução

As transmissões dos vídeos utilizaram o mesmo protocolo do estudo de caso 2, mas no aplicativo móvel não foi utilizada a plataforma J2ME.

Nesse projeto foi desenvolvida uma página Web para DM, onde o usuário pode fazer pesquisas, através do WebService desenvolvido no estudo de caso 2. Fazendo a busca são exibidos os vídeos relacionados a essa pesquisa, onde o usuário pode escolher o conteúdo que será exibido pelo Real *Player*.

Como o Real *Player* não tem opções de gravação, nessa aplicação o usuário não poderá persistir os vídeos no DM, sendo assim o usuário só acessará o vídeo pelo browser do seu dispositivo, já que agora a aplicação é Web.

3.5.2 Recursos

Nesse projeto as tecnologias são open source ou no máximo *free software*, que é o caso do *Real Player Mobile*.

A aplicação servidora continuou a mesma do estudo de caso 2, a qual utiliza o protocolo RTSP com o DSS e o banco de dados *MySql* para armazenar URL's do sistema.

Para acesso a base de dados também foi utilizado o mesmo WebService do projeto anterior, porém a chamada foi feita através de um sistema Web e não foi preciso utilizar as API's opcionais do J2ME, até mesmo porque para esse caso o sistema é Web.

Na aplicação móvel, para exibição do conteúdo multimídia, foi utilizado o *Real Player Mobile*, que é um *player* de renome no mercado e muito eficiente.

3.5.3 Arquitetura

Nesse projeto foi utilizada a mesma arquitetura do estudo de caso 2, porém agora o cliente só consegue acessar os vídeos, do servidor, somente se estiver conectado à internet, pois o sistema é Web e ele só é acessado por intermédio de um *browser*.

3.5.4 Desenvolvimento

No desenvolvimento para DM foi empregada a linguagem Java para criação de uma página Web, onde o usuário poderá pesquisar e visualizar o conteúdo buscado.

O sistema Web requisita as URL's para o WebService e esse retorna uma lista para a requisição feita. Para visualizar o conteúdo no DM é necessário que o usuário clique no link desejado, e será chamado o *Real Player Mobile* de forma automática.

3.5.5 Limitações

Apesar da eficiência desse *player*, uma limitação é a impossibilidade de fazer gravações do conteúdo assistido, pois como Real *Player* não tem essa função ficou impossível fazer o uso desse recurso.

Outra limitação é a instalação de um player, pois para o sistema funcionar é obrigada a instalação do *Real Player Mobile* ou de um *player* mobile que tenha acesso ao protocolo RTSP.

3.5.6 Projeto de Interfaces

Na figura 26 pode ser visualizada a lista de vídeos enviada pelo servidor, que é uma página que utiliza a tecnologia JSP e foi personalizada para ser executada em DM. Nessa página foi desenvolvida uma pesquisa dos vídeos que estão persistidos no servidor, ao pesquisada cada vídeo possui um hiperlink, que ao clicá-lo a mídia é executada no *Real Player Mobile.*

Figura 26 - Lista de vídeo pesquisada pelo servidor e exibida pela Web

3.6 RESULTADOS OBTIDOS

Além do desenvolvimento do cliente J2ME, com o uso de *streaming* de vídeo utilizando o protocolo RTSP, foram implementadas e testadas duas formas de uso do protocolo HTTP. Suas respectivas características foram confrontadas na tabela 9.

A primeira foi o desenvolvimento de um falso *streaming* de vídeo, em que se utilizou técnica para extrair um *frame* e seu áudio específico, cujo foi explicada na primeira seção desse capítulo. A idéia é armazenar um *buffer* de áudio e outro de vídeo no DM e ir exibindo-o, na medida em que os quadros são acumulados. Abaixo o que se observou a partir dessa técnica:

- Testando em uma rede local, com um roteador sem fio *D-Link* e um celular da Nokia *n85*, o conteúdo foi exibido com uma velocidade adequada, em um faixa de aproximadamente 20 fps, porém a perda de informação foi muito alta de aproximadamente 10 frames por segundo;

- Ocorre erro no sincronismo entre o áudio e o vídeo, pois o áudio foi exibido com mais velocidade do que o vídeo, isso devido a erro de programação do player desenvolvido e as correções ficam para trabalhos futuros;

A segunda diz respeito ao carregamento da mídia pelo MMAPI através do protocolo HTTP, no qual o vídeo é dividido em vários trechos e é exibido na medida em que os trechos são totalmente carregados. Concluiu-se que:

- Para pequenos vídeos, o sistema funciona bem, mas, para um conteúdo maior, o DM não tem poder de armazenamento suficiente para gravar os dados antes de exibi-los no *player*;
- O sistema demonstra uma lentidão antes de fazer o *download* da mídia, ocasionando paradas inesperadas na reprodução do conteúdo, dando a impressão de que o aplicativo está com algum erro, isso devido ao carregamento dos trechos de vídeo. Em um vídeo de 5 MB dividido em 5 trechos de 1 MB, o sistema apresenta 5 paradas de aproximadamente 4 segundos, no que se refere a uma parada para cada trecho, quando o trecho de vídeo é carregado o mesmo é exibido em um velocidade de aproximadamente 24 fps. Esse teste foi feito com um celular *Nokia N85* e um modem sem fio *D-Link*;
- Suporta apenas a disponibilização de conteúdos (*on demand*) previamente gravados;
- O único benefício desse processo é que, como o protocolo utilizado é o HTTP, a perda de pacotes é mínima, tendo em vista que é feito o *download* do conteúdo multimídia para em seguida exibi-lo.

Já nos testes feitos utilizando o protocolo RTSP com MMAPI, cabe ressaltar o seguinte:

- O sistema teve um eficiente desempenho em emuladores do WTK, da Sony Ericsson e Nokia, os vídeos foram tocados em tempo real e a perda foi imperceptível.
- Como a medição nos emuladores não pode servir como parâmetro absoluto, também foi feita uma avaliação no aparelho celular *N85* da Nokia, por ser bastante utilizado no mercado e por possuir suporte para o protocolo RTSP. O desempenho não foi o mesmo do emulador executado no PC, pois quando os testes foram no celular n 85 houve paradas de aproximadamente 4 segundos a cada 30 segundos de vídeo e depois de 5 minutos de vídeo o celular travou.

Nos testes feitos utilizando o protocolo RTSP com o Real *Player com o Android Player Android Player*, cabe ressaltar o seguinte:

- Os testes foram feitos com as mesmas condições do *player* com o MMAPI, mas tanto o Real *Player Mobile* como o *Andriod* mostrou mais eficiência na transmissão do conteúdo, exibindo o vídeo em aproximadamente 30 fps com sincronismo entre áudio e vídeo, a arquitetura funcionou tanto em sistema operacional *Symbian* (celular N 85) como *Andriod* (celular Milestone).
- Vale ressaltar que foram feitos teste através de WebCam, placa de captura e streaming de vídeo, todos exibiram em aproximadamente 30 fps e com sincronismo de áudio e vídeo.

O Quadro 9 apresenta um quadro comparativo das três formas de transmissão e recepção de conteúdo multimídia para DM.

Quadro 8 - Comparativo das quatro formas de transmissão e recepção de conteúdo multimídia para DM

-	**FALSO STREAMING**	**MMAPI/HTTP (Dividindo o vídeo em Techos)**	**MMAPI/RTSP**	**RealPlayer e Android / RTSP**
Espaço em disco	Pelo menos 5 MB	Pelo menos 10 MB	Pelo menos 5 MB	Pelo menos 5 MB
Fluxo/ *download* prévio do conteúdo	À medida em que é feito o *download*	Somente após o *download*	À medida em que é feito o *download*	À medida em que é feito o *download*
Perda de pacotes	Perde muitos frames (≈ 10 fps)	Não perde	Perde pucos frames (≈ 1 fps ou nenhuma perda)	Perde pucos frames (≈ 1 fps ou nenhuma perda)
Tamanho do conteúdo	Não importa	Apenas pequenos conteúdos (No máximo 5 MB)	Não importa	Não importa
Partição do conteúdo	Não	Sim	Não	Não
Sincronismo áudio/vídeo	Sem sincrocismo	Com sincrocismo	Com sincrocismo	Com sincrocismo
Valocidade do vídeo	≈ 20 fps	≈ 24 fps	≈ 24 fps	≈ 30 fps
Compreensão do vídeo no emulador	Comprometida (O usuário não entende parte do vídeo)	Total (O usuário entende todo vídeo)	Total (O usuário entende todo vídeo)	Não foi testado em emulador
Compreensão do vídeo em um DM	Comprometida (O usuário não entende parte do vídeo)	Total (O usuário entende todo vídeo)	Comprometida (Travou em 5 minutos de vídeo)	Total (O usuário entende todo vídeo)

4 CONCLUSÃO

Em razão do aumento da capacidade de armazenamento, processamento e poder de banda dos DM's, é possível hoje a utilização de *streaming* de vídeo nos celulares. As operadoras investem cada vez mais em serviços multimídia, fornecendo inúmeras espécies de toques, papéis de parede, vídeos, entre outros.

O presente trabalho tem por foco principal o desenvolvimento e comparação de ambientes para transmissão de *streaming* de vídeo em DM. A partir do estudo realizado, a primeira conclusão que pode ser obtida é que, para o *streaming* de vídeo tocar em um *Player mobile* de forma eficiente, faz-se necessária a utilização de protocolos e de uma arquitetura bem arranjada, isso é a utilização de um formato de vídeo especifico, de um servidor para transmissão de streaming, de player mobile para exibir o conteúdo, de um protocolo para comunicação do player com o servidor de vídeo.

Outro conceito que se extrai é a eficiência do protocolo UDP no HTTP para transmissão de vídeo em tempo real. O UDP é um protocolo de transporte, pertencente à família de protocolos TCP/IP, o qual fornece um serviço mínimo de transporte em redes que usam o protocolo IP, permitindo que as aplicações tenham acesso direto aos serviços da camada de rede. UDP é tradicionalmente usado em redes TCP, mas, ao contrário do TCP, não garante o envio dos dados de maneira confiável, isto é, alguns pacotes podem não chegar ao receptor. Enquanto o HTTP é utilizado principalmente para a comunicação entre um servidor e um único cliente, o UDP é usado para o pacote de difusão ou *multicast*, em que os dados são enviados para todos os clientes na rede - caso do *streaming* de áudio ou vídeo. Já as conexões UDP trabalham com fluxos de dados em tempo real, que é a tecnologia de *streaming*. Embora apresente perda de alguns pacotes na transmissão, esse protocolo é ideal para uso de *streaming* de áudio e

vídeo, uma vez que a referida perda é pequena e a tecnologia possui vantagens devido a seu protocolo de transmissão.

Ainda no que se refere aos protocolos, pode-se elencar o *download* progressivo, também conhecido como HTTP *streaming*. Apesar de ser uma solução muito utilizada, esse tipo de *download* oferece algumas limitações, quando comparadas à *streaming* real, quais sejam:

a) Não permite gestão/otimização da largura de banda, provocando uma lentidão e paradas inesperadas na reprodução do conteúdo. Exemplo: o buffer utilizado não recebe a quantidade de informação necessária para acompanhar o ritmo de reprodução (Cunha,2007);
b) Não permite o controle de fluxo do conteúdo. Exemplo: não é possível mudar de faixa, sem que se tenha descarregado todo o conteúdo;
c) Não permite broadcast de eventos em tempo real, suportando apenas a disponibilização de conteúdos (on demand) previamente gravados;
d) O arquivo é transferido totalmente para o computador do receptor, requerendo espaço em disco para armazenar os conteúdos.

Os estudos dos ambientes resultaram em um levantamento de necessidades para construção de uma proposta para um novo ambiente para busca, recepção e transmissão de vídeo para DM utilizando uma arquitetura bem arranjada.

A primeira proposta foi o desenvolvimento de um sistema que transmitisse vídeos para micro computadores e DM, isso tudo transmitido da mesma aplicação servidora. Foi necessário o desenvolvimento de um *player* Web e outro *mobile*. Como existem inúmeras API's para micro computadores, os sistemas para essa plataforma foram eficientes, mas, para o segundo ambiente, o resultado não foi o mesmo, causando uma falta de sincronismo do vídeo com o áudio.

A partir desse problema foi apresentada uma segunda solução, que foi a utilização do MMAPI para a transmissão de vídeos nos DM e também da utilização do protocolo RTSP. Como o MMAPI tem suporte, para tal protocolo foi desenvolvido um sistema de busca e transmissão de vídeos a partir de um servidor de *streaming*. O escolhido foi DSS.

Foi desenvolvido também um sistema Web para DM, que faz busca de vídeos no servidor de *streaming* e transmite o conteúdo pela *Real Player Mobile*, que é um *player* bastante consolidado no mercado.

Esta dissertação implementou e testou aplicações para DM ligadas à transmissão e recepção de *streaming* de vídeos. Além disso, forneceu contribuições práticas e teóricas no que se diz respeito a implementações em DM's e servidores de vídeos. As contribuições práticas se caracterizaram por disponibilizar ambientes de transmissão e recepção de vídeos em DM. Já as teóricas, concretizaram-se na identificação das tecnologias necessárias para consulta, recepção e transmissão de vídeos em DM.

O conhecimento adquirido ao longo deste trabalho possibilitou, frise-se, a publicação de dois artigos em congressos científicos na área de ciências da computação (CORIOLANO; BUCK, 2009).

Para trabalhos futuros, projeta-se na melhoria da arquitetura cuja idéia é a transmissão do mesmo conteúdo de vídeo para diferentes plataformas, já que nesse trabalho a transmissão para o player em DM, nessa arquitetura, houve erro de sincronismo do áudio com o vídeo. Logo o propósito é desenvolver um servidor de vídeo que transmita o mesmo conteúdo tanto para celulares como para plataforma Web, desenvolver um player Web e *mobile* para receber esse vídeo e criar um portal para fazer *upload* do conteúdo a ser transmitido.

REFERÊNCIAS

[1] 4G IN USA. Disponível em: <http://spectrum.ieee.org/telecom/standards/4g-in-the-usa> Acesso em: 12 ago. 2010

[2] ANATEL - Agência Nacional de Telecomunicações. Disponível em: <url: http://www.anatel.gov.br>, Acesso em: 03 jul. 2009.

[3] ANDROMEDA. Disponível em: <url:http://www.turnstyle.com/andromeda/home.asp>, Acesso em: 05 jun. 2009.

[4] AUSTERBERRY, D. **The technology of video and audio streaming**. Focal Press 2004. V1.

[5] BRUNOZI, Anderson. et al. **Serviços e aplicações móveis.** CPqD Tecnologia, Campinas, 2006. v.1, p. 85-94

[6] BUQUARQUE, Carlos ; OLIVEIRA, Marcos ; LOPES, Marcos. **Sistemas celulares. Relatório teórico.** Universidade Pernambuco, 2009

[7] SANTOS, Celso; SANTOS, Alexandre; TAVARES, Tatiane. **Uma estratégia para a construção de ambientes para a descrição semântica de vídeos**. In: WEBMEDIA 2007, Gramado - SC. 2007 v.1. pp. 274--28.

[8] CHINA *MOBILE*. Disponível em:<url: http://www.china*mobile*.com/en/>, acesso em: 02 Jul. 2009.

[9] CORIOLANO, Fábio; BUCK, Thomas. **Streaming de Vídeo para dispositivos Móveis**. In: ERCEMAPI 2009, Parnaiba-PI.

[10] CORIOLANO, Fábio; BUCK, Thomas. **Uma estratégia para recepção de streaming de vídeo em dispositivos móveis.** In: C3N 2009, Foz do Iguaçu - PR.

[11] CRANLEY, N.; MURPHY, L.; PERRY, P. **Perceptual quality adaptation (PQA) algorithm for 3GP and multitracked MPEG-4 content over wireless IP networks.** In: 15th IEEE International Symposium, Singapore: 2007,v.3, pp 5-8.

[12] CUNHA, C. **Tecnologias de streaming em contextos de aprendizagem**. 2007. Dissertação de Mestrado, Universidade do Minho. Disponível em: <URI: http://hdl.handle.net/1822/6400>, Acesso em: 05 jun. 2009

[13] CURCIO, I. ***Mobile* streaming services in WCDMA networks**. In: Proceedings of the 10th IEEE Symposium on Computers and Communications. IEEE Computer Society Washington, DC, USA. 2008 pp. 231 – 236.

[14] CYBER UNIVERSITY. Disponível em: <url: http://www.cyber-u.ac.jp/index.html>, Acesso em: 08 mar. 2009.

[15] DARKICE, Disponível em: <url: http://dark-ice.co.uk/>, Acesso em 05/06/2009.

[16] DAVIS, M. Media Streams: An Iconic Visual Language for Video Representation. In: **Readings in human-computer interaction: Toward the year 2000.** Morgan Kaufmann Publishers, San Francisco: 1995.

[17] DOMINGUES, Mauro; LICHTNOW, Daniel. **Acessando métodos de uma WebServices de um aplicativo J2ME**. Monografia apresentada à Universidade Católica de Pelotas: 2008.

[18] ELSEN, I. et al. Streaming Technology in 3G *Mobile* Communication Systems. In**: IEEE Computer Society Press Los Alamitos**. California: 2001. v.34, pp 46 – 52.

[19] EVANGELISTA, Liliana; GUARDIEIRO, Paulo. **Provisionamento de QoS baseado em DiffServ na rede de núcleo de sistemas celulares 3G**. Relatório teórico. Universidade Federal de Uberlândia: 2008.

[20] FLOGGY – **J2ME persistence framework**. Disponível em: <http://floggy.sourceforge.net/>, Acesso em: 05 jun. 2009

[21] GERMANY EMBRACES 4G. Disponível em: <http://spectrum.ieee.org/tech-talk/telecom/wireless/germany-embraces-4g/>, Acesso em: 12 Ago. 2010.

[22] GIOVANNI, Gualdi; PRATI, Andrea; CUCCHIARA, Rita. **Video streaming for *mobile* video surveillance**. In: IEEE Multimedia, IEEE Transactions on Multimedia. 2008. pp:1142 – 1154 v.10.

[23] GOSH, S. **J2ME record management store. IBM developers works.** 2002. Disponível em: < http://www-128.ibm.com/developerworks/wireless/library/wi-rms/ >, Acesso em: 12 Ago. 2010.

[24] GOYAL, V. **Pro Java ME MMAPI: *mobile* media API for java micro edition**. Apress: 2006.

[25] GUBERT, Luis ; BAZZOTI, Ezequiel. **VOIP para computação móvel**. Relatório teórico. Universidade de Cruz do Sul - UNISC: 2007.

[26] GUO, Fang Mao; TALEVSKI, A. e CHANG, E.. **VOIP on *mobile* devices.** In: IEEE Multimedia, IEEE Transactions on Multimedia: 2007. pg:163 – 169 v. 10.

[27] SCHULZRINNE ,H. ; RAO, A. ; LANPHIER, R. **Real time streaming protocol**. In: International Multimedia Conference – ACM: 1998.

[28] HELIX SERVER. Disponível em: <url: https://helixcommunity.org/>, Acesso em: 10 nov. 2009.

[29] INSIGHT, Research corporation. Disponível em: <http://www.insight-corp.com/>, Acesso em: 02 jul. 2009

[30] HOWSTUFFWORKS, **Introduction to How Streaming Video and Audio Work**. Disponível em: <url: http://www.howstuffworks.com/>, Acesso em: 03 jul 2009.

[31] SJOBERG, J. et al. **Real-time transport protocol (RTP) payload format and file storage format for the adaptive multi-rate (AMR) and adaptive multi-rate wideband (AMR-WB) audio codecs**. International Multimedia Conference. ACM: 2002.

[32] JEHUN, H.; YOUNG, Jick. **A *mobile* streaming service support framework using SIP and sensing network**. In: IEEE Advanced Communication Technology, The 6th International Conference on Multimedia: 2004 pp: 470 - 473 v. 1.

[33] KSOAP. Disponível em: <http://ksoap2.sourceforge.net/>, Acesso em: 05 jun 2009.

[34] KXML. Disponível em: <http://ksoap2.sourceforge.net/>, Acesso em: 05/ jun. 2009.

[35] LIZHONG, Xu. et al. **A J2ME-based wireless intelligent video surveillance system using moving object recognition technology**. In: Congress on Image and Signal Processing. Sanya: 2008.

[36] LUGON, Priscila, ROSSATO, Thiago. **Floggy: framework de persistência para J2ME/MIDP**. Monografia apresentada à banca da Universidade Federal de Santa Catarina: 2005.

[37] LWUIT: Home. Disponível em:< https://lwuit.dev.java.net/>, acesso em 05 Jul. 2009.

[38] RANGANATHAN, M. et al. ***Mobile* streams - A middleware for reconfigurable distributed scripting.** In: IEEE Agent Systems and Applications, / Third International Symposium on *Mobile* Agents: 1999. pp:162 - 175.

[39] MADEIROS, Fábio ; PETROLI, Silvio. **Integração de sistemas através de webService.** Relatório teórico. Faculdade de Jaguariúna: 2008. Disponível em: <http://bibdig.poliseducacional.com.br/document/search.php>, Acesso em 05 Jul. 2010.

[40] MELO, Miguel. **Um framework para a consulta a repositórios de vídeos digitais**. Dissertação de Mestrado, Universidade Salvador: 2008.

[41] *MOBILE* MEDIA API SUN. Disponível em: <url: http://java.SUN.com/products/mmapi/>, acesso em 03 Mai. 2009.

[42] MONTEZ, C.; WILLRICH, R; PISTORI, J. BDMM - **A Biblioteca Digital Multimídia**. In: III WorkShop RNP2 , Florianópolis: 2001.

[43] MORIMOTO, Carlos. 2009. **Entendendo o 3G**. In: Guia do Hardware. Disponível em:<http://www.guiadohardware.net/comunidade/3g-entendendo/955336/>., acesso em 05 Out. 2009.

[44] MUCHOW, W.J. **Core J2ME technology e MIDP**. Makron Books: 2004.

[45] MULTIMEDIA SERVER PROTOCOL. Disponível em: <http://technet.microsoft.com/pt-br/library/cc753556(WS.10).aspx>, acesso em 05 Out. 2009.

[46] NOKIA, N95 User Guide. Disponível em: <URL: http://www.nokia.pt/apoio-e-software/apoio-produtos/nokia-n95-8gb/manuais>, acesso em: 10 Out. 2009.

[47] OLIVEIRA, Rafael; SOARES, Sergio. **Serviço de distribuição de conteúdo RSS para dispositivos móveis através de WebService**. Relatório Teórico -Escola Politécnica de Pernambuco: 2005.

[48] QUICK TIME, Streaming Server. Disponível em: <url: http://www.apple.com/br/quicktime/*streamings*erver/>, Acesso em 05 Jun. 2009.

[49] REAL NETWORKS. Disponível em: <url: http://www.realnetworks.com/products-services/helix/helix.aspx>, Acesso em 09 Jun. 2009

[50] RIES, M.; et al. **Audiovisual Quality Estimation for *Mobile***. IEEE: 2005. pp:173 – 177.

[51] RODRIGUES, M. **IPTV conceitos, padrões e soluções.** Pontifícia Universidade Católica do Rio de Janeiro – PUC: 2006 . Disponível em:< ftp://ftp.inf.puc-rio.br/pub/docs/techreports/06_05_rodrigues.pdf>

[52] ROY, S.; ANKCORN, J. ; WEE, S. **Architecture of a modular streaming media server for content delivery networks**. IEEE Page(s):III: 2003. pp 569-72 vol.3.

[53] SILVA, Jackson. **Evolução da comunicação celular**. Disponível em: http://pessoal.educacional.com.br, acesso em 03 Jul. 2009.

[54] SILVA, Pedro; BELLO, Julio. **Simulação computacional de modelos para previsão de cobertura de sistemas celulares modernos**. Relatório teórico. Universidade Federal Fluminense: 2009.

[55] SONY ERICSSON, **Java ME platform docs & tools**. Disponível em: <http://developer.sonyericsson.com/site/global/docstools/java/p_java.jsp>, acesso em 10 Out. 2009

[56] SUMIT, Roy. et al. **A system architecture for managing *mobile* streaming media services**. In: International Multimedia Conference: 2003. pp: 408.

[57] SUN, **Java wireless toolkit for CLDC**. Disponível em: <http://java.SUN.com/products/sjwtoolkit/>, acesso em 03 Mar. 2009.

[58] SUSIE, Wee. et al. **Research and design of a *mobile* streaming media content delivery network**. In: IEEE multimedia and expo, 2003. / ICME '03. Proceedings. 2003 International Conference: 2003. pg: I - 5-8 v.1.

[59] TANENBAUM, A.S. **Redes de computadores**. Editora Campus. São Paulo: 2003.

[60] TELECO - Artigos e Tutoriais. In: **Portal brasileiro de informação em telecomunicações**. Disponível em: <http://www.teleco.com.br/>, acesso em: 03 Jul. 2009.

[61] THIRD GENERETION, **Partnership project.** Disponível em: <http://www.3gpp.org>, acesso em: 08 Set. 2009.

[62] TOPLEY, K. **J2ME in a nutshell – A desktop quick reference**. Beijing: 2002.

[63] VAZQUEZ, M.; VICENT, P. A *mobile* audio messages streaming system. In: **Proceedings of the 2007 euro american conference on telematics and information systems**, Faro: 2007. N. 52.

[64] VLAN - VideoLan. **The VideoLAN non-profit organisation**. Disponível em: < http://www.videolan.org/ >, acesso em: 05 Jun. 2009.

[65] WANG, Y. et al. Mobile video applications and Standards. In: **International multimedia conference. ACM**, Augsburg: 2007. pp.1-6.

[66] WINDSON, Viana; ROSSANA, M. C. Uma proposta para a geração semi-automática de aplicações. In: **XX simpósio brasileiro de engenharia de software**. Florianópolis: 2006.

[67] YEO,B.; YEUNG, M. Retrieving and visualizing video. In: **Communication of the ACM** v. 40. n. 12. New York. 1997. pp. 43-52.

[68] GEEK. ***YouTube* growing with 20 hours of new video content per minute**. Disponível em: <http://www.geek.com/articles/news>, acesso em: 02 Jul. 2009.

[69] ZHIHUA, Huang; BENSHUN, Yi**. Adaptive transmission for *mobile* streaming over WCDMA networks**. Wuhan: Universidade de Wuhan, 2008. IEEE. pp. 1-4.