CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Estudo de Viabilidade Transmissão de Vídeo em Tempo Real com Tecnologia Android

ROBERTA PEREIRA DE MORAES

Marília, 2012

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Estudo de Viabilidade Transmissão de Vídeo em Tempo Real com Tecnologia Android

Monografia apresentada ao Centro Universitário Eurípides de Marília como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Fábio Dacêncio Pereira

Marília, 2012

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Roberta Pereira de Moraes

ESTUDO DE VIABILIDADE TRANSMISSÃO DE VÍDEO EM TEMPO REAL COM TECNOLOGIA ANDROID

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 6,0 (Seis)

Orientador: Fábio Dacêncio Pereira

1°. Examinador: Leonardo Castro Botega

2°. Examinador: Paulo Rogerio de Mello Cardoso

Marília, 03 de dezembro de 2012.

Sumário

Índice de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

2D – Dimensional.

3D – Tridimensional.

ADT - Android Development Tools,

API – Application Programming Interface, é de um conjunto de rotinas e padrões estabelecidos por um software.

CPU - Central Processing Unit, unidade central de processamento.

DoD - Departament of Defense

GSM - (Global System for Mobile – Sistema Global para Comunicações Móveis)

GSM - (Global System for Mobile – Sistema Global para Comunicações Móveis)

HTTP – HyperText Transfer Protocol, protocolo da camada de aplicação utilizado para transferir dados por intranets e pela World Wide Web.

HTTP/1.1 - Hypertext Transfer Protocol (HTTP), versão atual.

HTTPS – HyperText Transfer Protocol Secure, implementação do protocolo HTTP sobre uma camada SSL ou TLS.

IDE - Integrated Development Environment , Ambiente Integrado de Desenvolvimento, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo.

iOS - Sistema operacional para mobile da Apple.

IP – Internet Protocol, protocolo no qual envolve toda a infraestrutura da Internet.

IPE - Internal Player Engine

IPTV – IPTV ou TVIP, é um método de transmissão de sinais televisivos através do protocolo IP.

ISP – Internet Service Provider, são os fornecedores de acesso a internet.

JDK – Java Development Kit, é um conjunto de utilitários que permitem criar sistemas de software para a plataforma java. É composto por compilador e bibliotecas.

LAN – Local Area Network, rede de computadores de pequena dimensão, como por exemplo, uma área residencial, escritório ou mesmo de um pequeno grupo de edifícios.

MIB – Management Information Base, tipo de base de dados usada para gerir dispositivos em redes de comunicações.

MP3 - (MPEG Audio Layer-3 – MPEG Áudio Camada-3)

NAT – Network Address Translation, é uma técnica que consiste em alterar o endereço IP de origem de um pacote que passa por um roteador ou firewall de maneira que um computador de uma rede interna tenha acesso ao exterior (rede pública).

OSI – Open Systems Interconnection, arquitetura que define uma forma comum de conectar computadores.

PCM - (Phase Coding Modulation – Modulação por Código de Pulso)

QoS – Quality of Service, refere-se a capacidade de fornecer um serviço conforme as exigências.

RPC – Remote Procedure Call, trata-se de um processo de comunicação que permite que um programa local invoque remotamente a execução de outro programa.

RTCP – Real-Time Transport Control Protocol, é um protocolo de Internet para a troca de dados em pacote.

RTP – Real-time Transport Protocol, é um protocolo de redes utilizado em aplicações de tempo real como, por exemplo, entrega de dados áudio ponto-a-ponto, como Voz sobre IP.

RTSP – Real Time Streaming Protocol, é um protocolo no nível de aplicação para controle na transferência de dados com propriedades de tempo real.

SDK - Software Development Kit, Kit de Desenvolvimento de Software ou Kit de de Desenvolvimento de Aplicativos

SO – Sistemas Operacionais.

SSL – Secure Sockets Layer, protocolo utilizado para transmitir documentos de forma segura através da internet.

TCP – Transmission Control Protocol, é um dos protocolos sob os quais assenta o núcleo da Internet. Ele verifica se os dados são enviados de forma correta, na sequência apropriada e sem erros, pela rede.

TCP/IP – Corresponde a um conjunto de protocolos utilizados para comunicação entre computadores em rede.

UDP - User Datagram Protocol **,**é um protocolo simples da camada de transporte**.**

VoIP – Voice over Internet Protocol, tecnologia que permite transmissão de comunicações de voz sobre redes IP.

WAN – Wide Area Network, rede de computadores que cobre uma grande área.

Moraes, Roberta Pereira. **Estudo da Transmissão de Vídeo em Tempo Real: Aplicação em Robótica com Tecnologia Android**. 2012. 87 F. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2012.

RESUMO

Com o crescimento e diversificação das aplicações multimídia, sobre diversas plataformas, principalmente na plataforma Android, a transmissão de vídeo em tempo real vem se tornando cada vez mais frequente em dispositivos moveis, como *smartfones*, tablets, entre outros. Tornando cada vez mais possível integral tais tecnologias, de modo vantajoso, a outros tipos de aplicações, no caso robótica. Existindo uma gama de possiblidades de integração *mobile* e robótica. Essas tecnologias, a plataforma Android, streaming de vídeo e plataforma robóticas, são complexas de conteúdo abrangente em constante crescimento, necessitando um estudo aprofundado e detalhado. Para ter uma melhor visão do funcionamento da transmissão de vídeo em tempo real, foi feito um estudo a fundo da qualidade de serviço (QoS) em TCP/IP, um dos principais protocolos de transporte, abrangendo esse estudo para outros protocolos, deveras importante neste sentido, podendo então analisar as qualidades de serviços na transmissão multimídia e cálculo de largura de banda. Apesar de toda as referências contidas neste projeto, o trabalho correlato inicial estudado foi o DroidMinder, o qual contém uma detalhada descrição , que foi essencial para direcionar o estudo. Através de toda a documentação contida neste trabalho notou-se a complexidade na transmissão de vídeo em tempo real, e as diversas aplicações sobre a plataforma Android e em robótica, notando-se a falta de suporte das APIs do Android à QoS.

Palavras-chave: Transmissão de vídeo em tempo real, plataforma Android, TCP/IP, Qualidade de Serviço (QoS), Protocolos para transmissão de vídeo, Recepção de vídeo em Android.

Moraes, Roberta Pereira. **Estudo da Transmissão de Vídeo em Tempo Real: Aplicação em Robótica com Tecnologia Android**. 2012. 87 F. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2012.

ABSTRACT

With the growing and diversification of multimedia application, mainly on the Android one, the real time video transmission is becoming more common in mobile gadgets, such as smartphones, tablets, etc. Making much more possible to connect such technologies, in a advantageous way, to other platform types, in this case robotics. Existing a wide variety of possibilities to integrate robotics and mobile. These technologies, Android, video streaming and robotics platforms, are complex with constant growing content, requiring a deep and detailed study. To make a better vision of how does video streaming works, a deep study of the Quality of Service(QoS) in TCP/IP, one of the main transport protocols, reaching the study to other important protocols, turning it viable to analyze the qualities of the multimedia transmission and bandwidth calc. Meanwhile all the references in this project, the initial co-related work studied was The DroidMinder, which contains a detailed description, being essential to direct the research. Through all the documentation contained in this work, the complexity in video streaming, and in the many applications on the Android platform could be noticed, as also the lack of support of APIs of the Android to QoS.

Palavras-chave: streaming video, plataforma Android, TCP/IP, Quality of Service (QoS), transport protocols, streaming video of Android.

1- Introdução

Os telefones celulares, smartphones e dispositivos móveis ocupam cada vez mais tempo e espaço no cotidiano. Seus processadores estão mais velozes, há mais memória e um salto considerável no que diz respeito ao armazenamento foi realizado. A difusão dos smartphones atinge 14% da população, e esses proprietários de smartphones dependem cada vez mais de seus dispositivos. 73% acessam a Internet todos os dias no smartphone. (Google Inc.,2012)

No cenário atual existem no mercado vários Sistemas Operacionais (SO) para dispositivos móveis, grandes empresas desenvolvem seus próprios SO, que melhor se adapta com o aparelho fabricado por elas, sendo Nokia, Google, Apple, Microsoft, Samsung, Intel, entre outras.

De acordo com pesquisa realizada pelo *Gartner* e divulgada pelo *Business Insider* sobre as vendas no primeiro quadrimestre de 2012 (1Q12), Google e Apple dominam o mercado de Smartphones no mundo. Nota-se na Figura 1, a seguir, que além dos dois SOs , citados anteriormente, crescerem os outros tiveram uma grande queda no mercado.

Seis melhores Sistemas Operacionais para smartphones, os embarques, e participação de mercado, 2012 Q1 (unidades em milhões)

Sistema Operacional	1Q12 Unidades vendidas	1Q12 Fatia do Mercado	1Q11 Unidades vendidas	1Q11 Fatia do Mercado	Mudança Ano-a-ano
Android	89.9	59.0%	36.7	36.1%	145.0%
iOS	35.1	23.0%	18.6	18.3%	88.7%
Symbian	10.4	6.8%	26.4	26.0%	-60.6%
BlackBerry OS	9.7	6.4%	13.8	13.6%	-29.7%
Linux	3.5	2.3%	3.2	3.1%	9.4%
Windows Mobile 7/Windows Mobile	3.3	2.2%	2.6	2.6%	26.9%
Outros	0.4	0.3%	0.3	0.3%	33.3%
TOTAL	152.3	100%	101.6	100.0%	49.9%

Figura 1 Mercado de Dispositivos Móveis

Fonte: *International Data Corporation (IDC)*, Maio 2012

O Android está no topo da lista, com 59% do *marketshare*(fatia no mercado), tendo vendido 89,9 milhões de unidades no período, em segundo, o iOS com 35,1 milhões de unidades e 23% do *marketshare*.

Symbian e BlackBerry novamente perderam grande parte de sua participação no mercado. O Symbian caiu de 26% no *marketshare* no primeiro trimestre de 2011 para 6,8%, enquanto o BlackBerry caiu de 13,6% para 6,8%. O Windows Phone obteve um aumento de 26,9% no total de vendas, mas mesmo assim sua participação caiu de 2,6% para 2,2%. A seguir em resumo a historia de cada SO mobile importante no mercado:

- Simbyan: Parceria entre Nokia, Siemens, Samsung, Ericsson, Sony Ericsson e Panasonic, atualmente pertence a Nokia; foi fruto de um desenvolvimento em conjunto que resultou na formação da fundação Symbian, no ano de 1998, visando aproveitar ao máximo a convergência entre dispositivos portáteis do tipo PDA e de telefones celulares.
- BlackBerry – RIM : Empresa canadense Research in Motion (RIM), aparelho que deu origem à categoria dos *smartphones.*
- MeeGo : Parceria entre Nokia e Intel, de código aberto, com kernel Linux, foi desenhado para atuar como um Sistema Operacional para plataformas como netbooks, desktops, tablets, smartphones, sistemas de navegação automotiva , smart TVs e outros sistemas embarcados. O MeeGo é hoje hospedado pela Linux Foundation.
- Windows Phone: Desenvolvido pela Microsoft, sucessor da plataforma Windows Mobile, parceria na qual o Windows Phone 7 seria o sistema operativo principal dos *smartphones*, Nokia, Acer, Fujitsu e ZTE . Versão Atual: Windows Phone "Apollo", lançamento meio 2013.
- iOS (antes chamado de iPhone OS) : Sistema operacional móvel da Apple Inc. desenvolvido originalmente para o iPhone, atualmente usado em iPodTouch, iPad e Apple TV. Versão Atual iOS 5.1.1.
- Android : Inicialmente desenvolvido pelo Google e posteriormente pela Open Handset Alliance (aliança de diversas empresas com a intenção de criar padrões abertos para telefonia móvel, entre as empresas participantes estão Google, HTC, Dell, Intel, Motorola, Qual comm, Texas Instruments, Samsung, LG, T-Mobile e Nvidia), roda sobre o núcleo Linux, disponível como código aberto desde 21 Outubro de 2008. Versão Atual : Android 4.0: Ice Cream Sandwich .

O Android é uma plataforma de desenvolvimento para aplicativos móveis como *smarthphones, tablets*, entre outros. Contêm um sistema operacional baseado em Linux, uma interface visual rica, diversas aplicações já instaladas e ainda um ambiente de desenvolvimento robusto, inovador e flexível. A linguagem utilizada para desenvolver aplicações é a linguagem Java. Um dos IDEs mais utilizados no desenvolvimento de aplicativos Java é o Eclipse (IDE), que é de código aberto. Este conta também com o SDK (*Software Development Kit*) que disponibiliza as ferramentas e APIs necessárias para o desenvolvimento na plataforma Android (LECHETA,2010).

O Android será adotado neste projeto de conclusão do curso de bacharelado em ciência da computação como plataforma para recepção e tratamento de vídeos transmitidos em tempo real em uma rede sem fio. Esse recurso será integrado a uma plataforma de robótica que está sendo desenvolvida no COMPSI/UNIVEM, com o intuito de agregar uma importante funcionalidade ao sistema principal, a possibilidade de transmissão de vídeo e recepção em diferentes dispositivos que tenham como solução tecnológica a plataforma Android.

A Plataforma Robótica idealizado pelo COMPSI/UNIVEM é constituída de módulos independentes que são integrados a um sistema de processamento central que definirá a utilização ordenada dos módulos propostos. Inicialmente a plataforma irá conter os seguintes módulos sensores, atuadores, interfaces de comunicação, transmissão de imagens, posicionamento geográfico, sistema de memória e armazenamento e unidade de processamento.

1.1 Problemática e justificativa

A transmissão de imagens em tempo real consome uma largura de banda significativa. A recepção em dispositivos que muitas vezes tem limitação de comunicação e principalmente memória e processamento pode se tornar uma barreira. Estes são fatores que exige uma solução equilibrada para condicionar estas variáveis e possibilitar atingir os objetivos desse projeto.

Neste contexto, além da exploração da transmissão de vídeo em tempo real será realizado um estudo de fatores importante no dimensionamento do melhor cenário, garantindo uma qualidade boa para os vídeos transmitidos.

1.2 Objetivos

Estudar APIs e aplicações para plataforma Android para a recepção de vídeo em tempo real e balanceamento da qualidade de transmissão. Para atingir esse objetivo geral pode-se dividi-lo nos seguir objetivos específicos:

- Estudo de transmissão de vídeo em tempo real e recepção em Android
- Pesquisar trabalhos correlatos
- Definir métricas para avaliar transmissão e recepção
- Criar cenário de testes e avaliação

1.3 Metodologia de Pesquisa e Desenvolvimento

A metodologia do trabalho pode ser separada em quatro fases principais.

I. Pesquisa

Inicialmente devem ser estudadas as principais tecnologias envolvidas no projeto coma linguagem Java para Android e a transmissão de vídeo em tempo real. Na segunda será realizada uma revisão sistemática do tema para a seleção de trabalhos correlatos.

II. Desenvolvimento

Será adaptada aplicações para Android utilizando a linguagem Java e o ambiente de desenvolvimento Eclipse. O Eclipse é o ambiente de desenvolvimento adotado pelo Google, será utilizado o *plugin* ADT (Android Development Tools) para facilitar o desenvolvimento, os testes e a compilação do projeto.

III. Teste e validação

Serão definidas métricas de avaliação do meio de transmissão e recepção de vídeo em tempo real. Utilizando uma câmera IP e aplicativos para testes.

1.4 Organização do Trabalho

O documento está organizado em cinco capítulos. No cômputo geral, este documento pretende realizar uma introdução aos conceitos relativos à *streaming* de vídeo com a plataforma Android, além da exploração da transmissão de vídeo em tempo real é apresentado um estudo de fatores importante no dimensionamento do melhor cenário, QoS (Qualidade de Serviço) garantindo uma qualidade boa para os vídeos transmitidos.

No primeiro capítulo contém a introdução do projeto, com problemática e justificativa, objetivos do projeto e a metodologia de pesquisa e desenvolvimento, demonstrando a ideia principal deste trabalho de conclusão de curso.

No segundo capítulo será sobre a qualidade de serviço em TCP/IP, iniciando com tópico sobre *streaming*, as características de uma rede TCP/IP na transmissão de vídeo e em seguida um tópico apresentando os principais protocolos para transmissão de vídeo, perante o projeto, sendo eles HTTP,RTP, RTCP, RTSP.

Ainda no segundo capítulo, será apresentado a QoS qualidade de Serviços, sua importância, como funciona na transmissão multimídia em redes, a necessidade das aplicações (latência, *jitter*, *skew*, tabela comparativa) e qualidade de serviço (requisitos gerais para suporte a serviço de banda larga). Seguindo um próximo tópico sobre cálculo de largura de banda para produção de vídeo por streaming. O terceiro capítulo trata-se de um estudo aprofundado em protocolos, QoS e calculo de largura de banda, dando forma ao objetivo do trabalho de conclusão de curso , que é o estudo de transmissão de vídeo em tempo real.

No terceiro capítulo trata-se da recepção de vídeo na plataforma Android, será apresentado os desafios da execução de vídeos, tipos de mídias suportados no Android, protocolos de redes, codificações de vídeos recomendadas, todos esses estudos foram feitos para um melhor entendimento de como se comporta a tecnologia de desenvolvimento no âmbito do Android.

O quarto capítulo trata-se do DroidMinder, o principal trabalho correlato estudado para o desenvolvimento deste trabalho de conclusão de curso, ele conta com uma introdução, o estudo do software que ele foi baseado Zoneminder, o desenvolvimento Android, reprodução de fluxo de vídeos e o aplicativo DroidMinder.

Quinto capítulo apresenta os testes feitos com a câmera IP FOSCAM, sua instalação e funcionamento. Também contém testes com aplicativos para transmissão de vídeo no android. Os aplicativos neste item são IP cam Viewer Lite e DroidMinder, sendo o IP Cam Viewer Lite testado para verificar métricas de transmissão e o DroidMinder o foco encontra-se nos trabalhos correlatos, testes e código fonte.

Finalizando este trabalho com as conclusões do projeto e referências, obtidos através de toda pesquisa para o estudo de transmissão de vídeo em tempo real, propondo trabalhos futuros baseado em toda esta documentação contida neste trabalho de conclusão de curso.

2 - QoS Qualidade de Serviço em TCP/IP

A seguir será descrito Streaming de vídeo, o funcionamento dos protocolos TCP/IP na transmissão de vídeo, os protocolos para a transmissão de vídeo, sendo eles, HTTP, RTP, RTCP, RTSP.

O estudo dos protocolos será necessário para medir a qualidade de serviço QoS, sua importância no âmbito de transmissões multimídias em redes, a necessidade das aplicações (latência, *Jitter*, *Skew*) e uma tabela de comparações entre elas. Para finalizar esse capítulo apresenta-se o cálculo de largura de banda para a produção de vídeo por Streaming, explicando bit rate do vídeo, bit rate em multimídia.

2.1 Streaming

Streaming consiste na transmissão de dados – representada na Figura 2, de áudio ou vídeo, de um servidor para um cliente que descodifica e reproduz os dados à medida que são recebidos tentando preservar a relação temporal da fonte. Possibilita que o utilizador veja os conteúdos pretendidos imediatamente após um pequeno período de carregamento (*buffering*).

O *streaming* de conteúdos multimídia através da Internet foi popularizado pela Progressive Networks em 1995 com o seu formato proprietário RealAudio *Streaming*. Em 1997, na mesma altura em que alterou o seu nome para *Real Networks*, esta empresa iniciou uma parceria com a Netscape com o intuito de desenvolver o que mais tarde se tornou o standard RTSP para *streaming*.(Pereira, 2010)

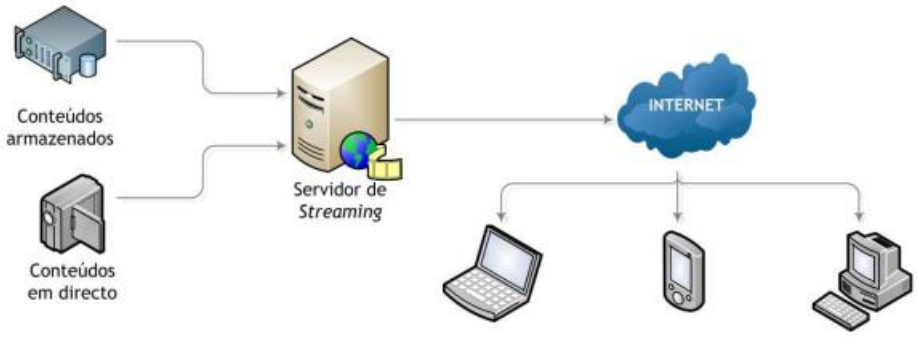

Figura 2 Configuração genérica de um sistema de streaming

Fonte: (Pereira, 2010)

Para Pereira (2010) ao contrário da Internet, onde o download é ainda a forma mais popular de distribuição de conteúdos, nos aparelhos móveis a distribuição de conteúdos multimídia está a convergir rapidamente para o *streaming* (particularmente nas gravações ao vivo ou conteúdos televisivos), pois este apresenta algumas das vantagens quando comparado com a reprodução local após download:

- Não existirão dados gravados permanentemente no cliente final (ideal para equipamentos móveis onde a dimensão da memória é mais reduzida);
- O utilizador não necessita de esperar que todo o conteúdo seja recebido para iniciar a reprodução;
- Possibilita a reprodução de programas ao vivo;
- Uma grande vantagem para os fornecedores de serviços é o fato de não ser possível ao utilizador final reencaminhar ou enviar os conteúdos para outros utilizadores, e por isso os conteúdos podem ser cobrados de cada vez que são consumidos.

Segundo Pereira(2010), o *streaming* inicia-se com a codificação de um arquivo ou sinal a transmitir e, em seguida, é feita a divisão do sinal em pequenos pacotes que serão enviados sequencialmente através da Internet. Quando os pacotes chegam ao seu destino (o terminal do utilizador que pediu o serviço), são descomprimidos e reorganizados de forma a serem reproduzíveis pelo dispositivo do utilizador. Para manter a ilusão de reprodução contínua, os pacotes são introduzidos num *buffer* (como uma fila de espera), para que alguns deles já estejam no sistema do utilizador antes do início da reprodução. Enquanto os pacotes guardados são reproduzidos, outros pacotes são recebidos, e introduzidos na fila de espera para posterior reprodução. Contudo, quando a recepção de pacotes é muito lenta possivelmente devido a problemas de rede, o dispositivo do utilizador não tem pacotes para reproduzir e, enquanto aguarda a recepção dos dados em falta, suspende a reprodução, degradando a QoS do utilizador.

2.2 Características de uma rede TCP/IP na transmissão de vídeo

Projetar uma rede que transmita dados multimídia não é trivial. Dados de áudio e vídeo precisam ser exibidos continuamente na mesma taxa em que foram gerados. Se os dados não chegam a tempo, o processo de exibição para os ouvidos e olhos humanos podem

facilmente perceber tal falha. Além da latência, congestionamentos na rede também têm efeitos sérios obre o tráfego de tempo real. Se a rede estiver congestionada, o único efeito sobre um tráfego que não seja tempo real é que a transferência levará mais tempo para completar, mas dados de tempo real se tornam obsoletos e serão descartados se não chegarem a tempo. Se nenhuma ação apropriada for tomada, a retransmissão dos pacotes perdidos agravaria a situação e comprimiria a rede. Apenas aumentar a largura de banda não resolverá o problema de transmissão em rajadas. Para a maioria das aplicações multimídia, o receptor tem um buffer de tamanho limitado. Se nenhuma medida for tomada para regular o fluxo de dados, ele pode gerar uma sobrecarga (ou um fluxo leve demais) no buffer da aplicação. Quando os dados chegarem muito rápido, o buffer irá sobrecarregar-se e alguns pacotes serão perdidos, resultando em uma qualidade pobre. (Tschoke, 2001)

Quando os dados chegarem muito lentos, o buffer ficará ocioso e a aplicação em espera. Como resolver esses problemas é um desafio que as redes multimídia precisam encarar.

Para rodar multimídia sobre a Internet, várias questões precisam ser resolvidas. Multimídia significa dados extremamente densos e tráfego pesado tendo o hardware que fornecer largura de banda suficiente. A Internet é uma rede de datagramas comutada por pacotes na qual eles são roteados independentemente ao longo das redes compartilhadas. Tecnologias atuais não garantem que dados de tempo real alcancem seu destino sem serem desordenados ou perdidos. Novos protocolos de transporte precisam cuidar de questões de temporização para que o áudio e o vídeo sejam exibidos continuamente com a sincronização e a temporização corretas. (Tschoke, 2001)

2.3 Protocolos para transmissão de vídeo

Existem dois pontos destacados no estudo de requerimento de serviços: considerações sobre o tempo e tolerância à perda de dados. Tempo é importante porque veremos que muitas aplicações são extremamente sensíveis ao atraso e um pequeno atraso na chegada de um pacote pode acarretar inutilidade a todo o processo já feito. Por outro lado, este tipo de aplicação é mais tolerante à perda de dados do que os serviços estáticos. Uma perda pode ser facilmente aliviada ou até anulada, por certos mecanismos da aplicação.

Para (Ferreira, 2007), há duas técnicas utilizadas para fornecer serviço de vídeo:

- *Download and play* – como o próprio nome sugere esta técnica precisa que todo o arquivo seja transferido para cliente e então seja possível sua visualização, esta técnica possui a vantagem de ser visualizada, pausada, avançada e retrocedida depois de armazenada, a qualquer momento;
- *Streaming* – aqui os dados são armazenados em *buffer* e são visualizados na mesma medida em que vão chegando. Para esse tipo de transmissão ser eficaz é necessário utilizar uma alta taxa de compressão, sendo o MPEG a mais utilizada. Técnicas do tipo streaming podem ainda se dividir em:
    - *Streaming* de vídeo armazenado – neste caso o *host* cliente solicita os dados que estão previamente armazenados em um servidor, dando ao cliente a liberdade de manipular o vídeo, isto é, poder pausar, retroceder, avançar, etc. Neste tipo de aplicação o conteúdo só acontece sob demanda e pode existir mais que um cliente conectado ao mesmo servido ao mesmo tempo.
    - *Streaming* de vídeo ao vivo – baseado em transmissões do tipo broadcast, os dados não são armazenados em um servidor, fazendo com que o usuário não tenha a liberdade assumida acima. A distribuição dessa transmissão pode ser do tipo *unicast* (conexão ponto-a-ponto entre o cliente e o servidor onde cada cliente recebe seu próprio *stream*) ou do tipo *multicast* (utilizado quando há o desejo de preservar a banda fazendo com que todos os clientes compartilhem o mesmo *stream*).
    - Vídeo interativo em tempo real – objetiva agregar muito mais que uma conversa entre duas pessoas, podendo ser expandida ao caso de uma reunião através de uma videoconferência.
- Reprodução contínua - Uma vez que o *playout* começa, ele deve ocorrer de acordo com o tempo original de gravação. O dado deve chegar ao destino a tempo de ser vista corretamente pelo cliente. (KUROSE E ROSS, 2001)

Nos próximos sub tópicos serão especificados os principais protocolos de transmissão de vídeos estudados para este trabalho de conclusão de curso.

2.3.1 HTTP – (Hypertext Transfer Protocol)

Figura 3 Ligação entre servidor e terminal, utilizando um protocolo do tipo download progressivo (sobre HTTP)

Fonte: Pereira, 2010

A Figura 3 a cima mostra a abordagem do tipo download progressivo que consiste no download de arquivos disponibilizados por um servidor web HTTP. Assim sendo, é suportado pela maioria das plataformas e aplicações de reprodução de conteúdos multimídia, tais como *Adobe Flash, Silverlight,* ou ainda *Windows Media Player*. O termo "Progressivo" enquadra-se neste tipo de abordagem uma vez que a maioria das aplicações de reprodução do lado do cliente permite que o conteúdo seja reproduzido enquanto o download ainda está a decorrer, ou seja, antes de o arquivo estar completo.

Clientes com suporte para HTTP 1.1 podem também escolher visualizar uma parte do conteúdo que ainda não esteja disponível no cliente. Isto é possível através da troca de mensagens entre o servidor e cliente, que envia pedidos denominados "*Byte Range Requests*". Esta é a solução adaptada por vários*websites* de partilha de vídeos, como o *Youtube,Vimeo,* ou *Myspace*, que utilizam quase exclusivamente download progressivo. (Pereira, 2010)

2.3.2 RTP (Real Time Protocol)

Utilizado para transportar mídias continuas no formato PCM (*Phase Coding Modulation* – Modulação por Código de Pulso), GSM (*Global System for Mobile* – Sistema Global para Comunicações Móveis) e MP3 (MPEG Audio Layer-3 – MPEG Áudio Camada-3) para som e MPEG e H.263 para vídeo. É otimizado para ser utilizado em UDP e são limitados a aplicações *unicast* e *multicast*, são transparentes aos roteadores e não fornecem

nenhum tipo de mecanismo de QoS ou entrega confiável de dados. Sendo assim ele simplesmente é definido como um cabeçalho que contém campos que será traduzido pelo receptor sobre qual a forma mais adequada de se decodificar a informação presente no datagrama que lhe foi entregue. Os campos do cabeçalho do protocolo RTP incluem tipo de carga útil, número de sequência, marca de tempo, identificador de sincronização da fonte e outros campos. Os dados são encapsulados pelo RTP, depois em um segmento UDP, que por sua vez são encapsulados em m datagrama IP, como mostra a Figura 4: (KUROSE E ROSS, 2001)

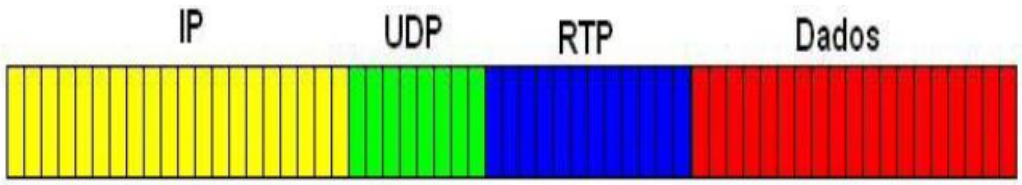

Figura 4 Estrutura datagrama IP

Fonte: KUROSE E ROSS, 2001

Um servidor de mídia contínua encapsula um "pedaço" da mídia num pacote RTP, depois esse pacote é encapsulado num segmento UDP e então entregue ao IP. O lado cliente extrai do UDP o pacote RTP, de onde extrai o pedaço da mídia e passa para o media player. Se uma aplicação usa RTP em vez de outro esquema proprietário para prover *payload time*[13], *timestamps*[14] ou números sequenciais, a aplicação vai facilmente interoperar com outras aplicações multimídia. O RTP não provê mecanismos para certificar de que o dado chegará dentro da data prevista ou outra qualidade de serviço. Nem garante a ordem da chegada dos dados, e nem mesmo a entrega deles. Os roteadores não têm como saber se um datagrama IP contém ou não um encapsulamento RTP. (KUROSE E ROSS, 2001)

O fato das aplicações RTP serem feitas na maioria das vezes em UDP/IP não significa que RTP requer UDP e IP. Ao passar para camada de transporte a mídia não comprimida é alocada em *buffers* que produzirão frames comprimidos de acordo com o algoritmo de compressão definido, esses frames são carregados nos pacotes RTP para envio e caso sejam muito grande eles poderão ser fragmentados em vários pacotes. Após o pacote ter sido enviado o transmissor não poderá descartar esses dados, pois talvez seja necessário para possíveis correções de erros.

O receptor por sua vez faz a validação, correção e inserem os pacotes em uma fila especifica de entrada, passando posteriormente a um *buffer* que é ordenado pela marca de

tempo e ali permanece até que um frame esteja completo, então os frames são montados e decodificados. (KUROSE E ROSS, 2001)

2.3.3 RTCP (Control RTP – Protocolo de controle RTP)

O RTCP é um protocolo de transmissão a ser utilizado em conjunto com o RTP, enviando de tempos em tempos dados estatísticos referentes a números de pacotes enviados, perdidos e variação de atrasos. Essas informações podem ser utilizadas pelos remetentes como forma a mudar suas taxas de transmissão e na determinação da localização do problema onde há o atraso ou a perda de pacotes. (KUROSE E ROSS, 2001)

A comunicação (representada na Figura 5) é feita utilizando dois tipos de mensagens:

- RE: Relatório de Emissor que contém estatísticas da transmissão e recepção de dados geradas por emissores ativos;
- RR: Relatório de Receptor que contém estatísticas da recepção de dados geradas por participantes que não são emissores ativos, ou combinados com RE para emissores ativos que informem sobre mais de 31 fontes.

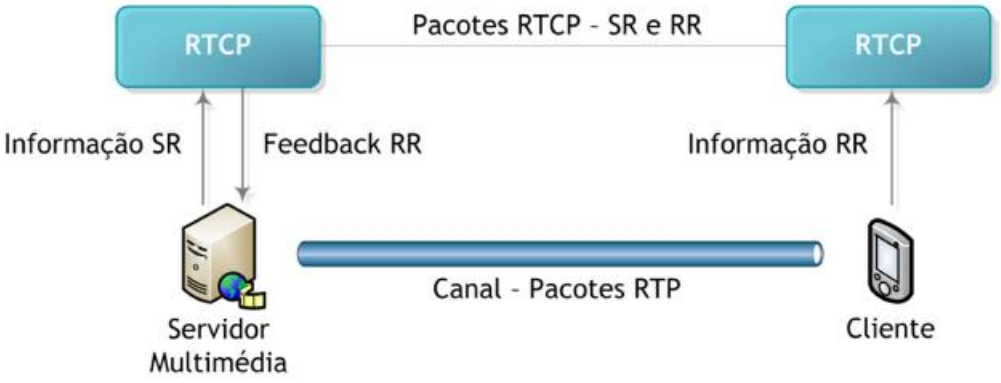

Figura 5 - Troca de relatórios entre cliente e servidor, utilizando o protocolo RTCP

Fonte: Pereira, 2010

Nos relatórios do receptor são enviadas informações sobre a recepção dos dados. Algumas das informações contidas nos relatórios são:

- Número cumulativo de pacotes RTP recebidos pelo cliente;
- Variação estatística do tempo de recepção dos pacotes RTP;

- Número de pacotes perdidos;
- Número de pacotes esperados.

Cinco tipos de pacotes RTCP são definidos pelo RTP:

- RTCP RR (*Report Receiver* RTCP – Relatório do Receptor RTCP) – O relatório estatístico que é um pacote RTCP é gerada a cada corrente RTP recebida e então o receptor envia àquele pacote para a árvore *multicast* a qual ele esta conectado.
- RTCP SR (*Sender Report* RTCP – Relatório do Transmissor RTCP) – São também informações estatísticas, contudo são enviadas pelo transmissor.
- RTCP SDES (*Description of the Source* RTCP – Descrição da Fonte RTCP) – são pacotes que contém identificação do participante, localização, *e-mail* etc.
- RTCP BYE (Gerenciamento de Membros) – esse pacote é gerado quando um participante deixa a sessão.
- RTCP APP (*Applications Packages* RTCP – Aplicações de Pacotes RTCP) – permite extensões definidas pela aplicação.

2.3.4 RTSP (Real Time Streaming Protocol)

O RTSP é um protocolo mais robusto para comunicação multimídia e controle de sessões em relação aos apresentados anteriormente. Esse protocolo serve para trocar informações do controle de reprodução, visto que aplicações de fluxo contínuo permitem ao usuário o controle da reprodução tal qual um aparelho de DVD. Ele é chamado de protocolo "fora da banda", pois suas mensagens são enviadas fora da banda, pois utiliza dois pares de portas cliente/servidor. (KUROSE E ROSS, 2001)

Seu funcionamento é baseado em *streamming* (transmissão) que segmenta os dados em vários pacotes onde o tamanho depende da banda disponível entre cliente e servidor, parâmetro avaliado através do parâmetro MTU da rede. A aplicação recebe estes pacotes e recompõe os dados antes de descomprimir e utilizar a informação inicial. Isto permite que fluxo seja ao vivo ou de dados armazenados. O RTSP é considerado mais próximo à um datagrama devido a esta característica de remontagem. A seleção dos canais de envio (UDP, TCP, IP *Multicast*) e de mecanismos de transmissão é baseado em RTP, tanto

para controlar como para garantir a entrega do conteúdo em tempo real. O RTSP é também utilizado com RSVP para solicitar e garantir banda para sessões.

Uma sessão RTSP não é amarrada a uma conexão do nível de transporte, por isso um cliente pode abrir e fechar várias conexões de transporte com o servidor. As mensagens trocadas são do tipo requisição e resposta que podem ser representadas para o usuário como formas de *setup* (alocação de recursos e inicio de sessão), *play* (inicio da transmissão de dados), *pause* (suspensão temporária do fluxo de transmissão) e *teardown* (libera recursos, encerrando a sessão), esses comandos necessitam de um protocolo confiável como o TCP. Na Figura 6 está representada a troca de mensagens durante uma sessão RTSP.

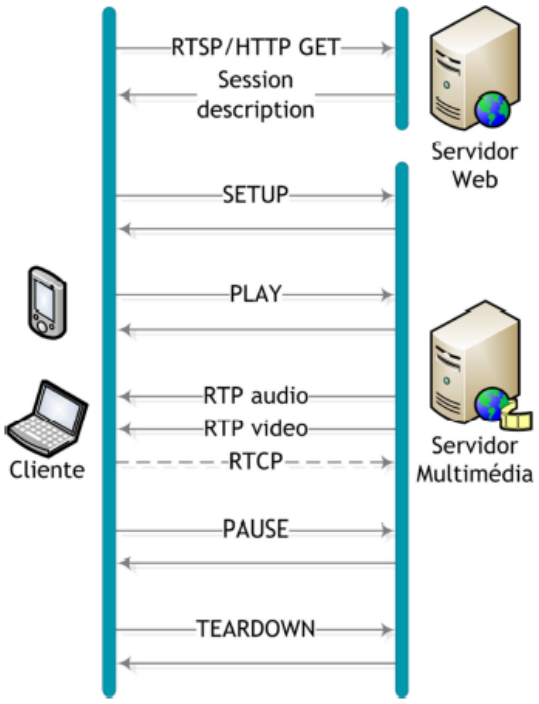

Figura 6 - Troca de mensagens durante uma sessão utilizando o protocolo RTSP

Fonte: Pereira ,2010

Entretanto o RTSP não define esquemas de compressão para áudio ou vídeo, não define como a mídia é encapsulada nem os cabeçalhos. Isto é tarefa do RTP, não restringe como o pacote deve ser transmitido (entre UDP ou TCP), não restringe como o media player armazenará (em *buffers*) o arquivo. (KUROSE E ROSS, 2001)

2.4 Qualidade de Serviços QoS

Outra forma de ver a camada de transporte, onde se localizam os protocolos estudados, é considerar que sua principal função seja melhorar a QoS (Qualidade de Serviço) oferecida pela camada de rede. Se o serviço de rede for perfeito, o trabalho da camada de transporte será fácil. No entanto, se o serviço de rede não for perfeito, a camada de transporte terá que servir de ponte para cobrir a distância entre o que os usuários de transporte desejam e o que a camada de rede oferece. (Pereira ,2010)

Para Pereira (2010) ainda que à primeira vista o conceito de qualidade de serviço seja vago (fazer com que todos concordem sobre o que significa um serviço "bom" não é uma tarefa simples), a QoS pode ser definida por um número específico de parâmetros. O serviço de transporte pode permitir ao usuário determinar os valores preferenciais, os valores aceitáveis e os valores mínimos para vários parâmetros de serviço no momento em que uma conexão é estabelecida. Alguns parâmetros também podem ser usados no transporte sem conexão. É tarefa da camada de transporte examinar esses parâmetros e, dependendo do(s) tipo(s) de serviço(s) de rede disponível(eis), determinar se é possível realizar o serviço solicitado. Os parâmetros típicos para a qualidade de serviço da camada de transporte são resumidos em:

- Retardo no estabelecimento da conexão
- Probabilidade de falha no estabelecimento da conexão
- *Throughput*
- Retardo de Trânsito
- Taxa de erros residuais
- Proteção
- Prioridade
- Resiliência

Observa-se que poucas redes ou protocolos oferecem todos esses parâmetros. Muitas apenas tentam reduzir a taxa de erros da melhor maneira possível. Outras têm arquiteturas de QoS mais elaboradas.

O retardo no estabelecimento da conexão é o tempo transcorrido entre a solicitação de uma conexão de transporte e o recebimento de sua confirmação pelo usuário do serviço de transporte. Nessa característica também está incluído o retardo do processamento

na entidade de transporte remota. A exemplo de todos os parâmetros que medem um retardo,quando menor o retardo, melhor o serviço.

A Probabilidade de falha no estabelecimento da conexão é a possibilidade de a conexão não se estabelecer dentro de um período máximo estabelecido devido a, por exemplo, um congestionamento na rede, à falta de espaço de tabela em algum lugar ou a outros problemas internos.

O parâmetro *throughput* calcula o número de bytes de dados do usuário transmitidos por segundo durante um determinado intervalo de tempo. O *throughput* é medido separadamente para cada direção.

O retardo de trânsito calcula o tempo transcorrido desde o envio de uma mensagem pelo usuário de transporte da máquina de origem até seu recebimento pelo usuário de transporte da máquina de destino. A exemplo do throughput, cada direção do transporte é analisada separadamente.

A taxa de erros residuais calcula o número de mensagens perdidas ou corrompidas em uma porcentagem do total enviado. Na teoria, a taxa de erros residuais deveria ser zero, pois o trabalho da camada de transporte é esconder os erros da camada de rede. Na prática, essa taxa pode apresentar um valor (baixo) finito.

O parâmetro de Proteção oferece uma forma de o usuário de transporte especificar seu interesse no fato de a camada de transporte fornecer proteção contra a leitura, ou a modificação, de dados por parte de terceiros (que se utilizam de “grampos” para violar a comunicação).

O parâmetro de Prioridade oferece ao usuário de transporte um modo de indicar que algumas conexões são mais importantes do que outras e, em caso de congestionamento, garantir que as conexões de maior prioridade sejam atendidas primeiro.

Por fim, o parâmetro de Resiliência oferece à camada de transporte a probabilidade de finalizar uma conexão espontaneamente devido a problemas internos ou a congestionamento.

O parâmetros QoS são especificados pelo usuário de transporte quando uma conexão é solicitada. Os valores mínimo e máximo aceitáveis podem ser fornecidos. Às vezes, ao conhecer os valores de QoS, a camada de transporte percebe imediatamente que alguns deles não podem ser alcançados. Nesse caso, ela informa ao responsável pela chamada que a tentativa de conexão falhou sem sequer tentar contato com o destino. O relatório da falha especifica o que a causou.

Em outros casos, a camada de transporte sabe que não pode alcançar o objetivo desejado (por exemplo, um *throughput* de 600Mbps), mas pode atingir uma taxa mais baixa, porém aceitável (por exemplo, 150Mbps). Em seguida, a camada de transporte envia a taxa mais baixa e a mínima aceitável para a máquina remota e solicita o estabelecimento de uma conexão. Se a máquina remota não puder administrar o valor sugerido, mas conseguir administrar qualquer valor acima do mínimo, a camada de transporte fará uma contraproposta. Se a máquina remota não puder trabalhar com qualquer valor acima do mínimo, ela rejeitará a tentativa de conexão. Por fim, o usuário de transporte da máquina de origem é informado do fato de que a conexão foi estabelecida ou rejeitada. Se a conexão tiver sido estabelecida, o usuário será informado dos valores dos parâmetros acordados. Esse procedimento é chamado de negociação de opção (*option negotiation*). Uma vez que tenham sido negociadas, as opções serão mantidas durante toda a conexão. Muitas concessionárias de serviços de melhor qualidade para evitar que seus clientes fiquem obcecados por esses detalhes.

2.4.1 Importância do QoS

Na internet e nas intranets atuais, a largura de banda é um assunto importante. Mais e mais pessoas estão usando a Internet por motivos comerciais e particulares. O montante de dados que precisa ser transmitido através da internet vem crescendo exponencialmente. Novos aplicativos, como RealAudio, RealVideo, Internet Phone e sistemas de videoconferência precisam cada vez de mais largura de banda que os aplicativos usados nos primeiros anos da Internet. Enquanto que aplicativos Internet tradicionais, como WWW, FTP ou Telnet, não toleram perda de pacotes, mas são menos sensíveis aos retardos variáveis, a maioria dos aplicativos em tempo real apresenta exatamente o comportamento oposto, pois podem compensar uma quantidade razoável de perda de pacotes mas são, normalmente, muito críticos com relação aos retardos variáveis.(Kamienski, 2000)

Isso significa que sem algum tipo de controle de largura de banda, a qualidade desses fluxos de dados em tempo real depende da largura de banda disponível no momento. Larguras de banda baixas, ou mesmo larguras de banda melhores, mas instáveis, causam á qualidade em transmissões de tempo real, com eventuais interrupções ou paradas definitivas da transmissão. Mesmo a qualidade de uma transmissão usando o protocolo de tempo real RTP depende da utilização do serviço de entrega IP subjacente.(Sadok, 2000)

Para Kamienski (2000), são necessários conceitos novos para garantir uma QoS específica para aplicativos em tempo real na Internet. Uma QoS pode ser descrita como um conjunto de parâmetros que descrevem a qualidade (por exemplo, largura de banda, utilização de buffers, prioridades, utilização da CPU etc.) de um fluxo de dados específico. A pilha do protocolo IP básica propicia somente uma QoS que é chamada de melhor tentativa. Os pacotes são transmitidos de um ponto a outro sem qualquer garantia de uma largura de banda especial ou retardo mínimo. No modelo de tráfego de melhor tentativa, as requisições na Internet são processadas conforme a estratégia do primeiro a chegar, primeiro a ser atendido. Isso significa que todas as requisições têm a mesma prioridade e são processadas uma após da outra. Não há possibilidade de fazer reserva de largura de banda para conexões específicas ou aumentar a prioridade de uma requisição especial. Assim, foram desenvolvidas novas estratégias para oferecer serviços previsíveis na Internet. Hoje em dia, há dois princípios básicos para conseguir QoS na Internet:

- Serviços integrados
- Serviços diferenciados

Os serviços integrados trazem melhoramentos ao modelo de rede IP para suportar transmissões em tempo real e garantir largura de banda para sequências de dados específicas. Neste caso, definimos um fluxo de dados (*stream*) como uma sequência distinguível de datagramas relacionados transmitidos de um único emissor para um único receptor que resulta de uma única atividade de usuário e requer a mesma QoS.

Por exemplo, um fluxo de dados poderia consistir de um *stream* de vídeo entre um par de host determinado. Para estabelecer a conexão de vídeo nas duas direções, são necessários dois fluxos de dados.

Cada aplicativo que inicia um fluxo de dados pode especificar a QoS exigida para esse fluxo. Se a ferramenta de videoconferência precisar de uma largura de banda mínima de 128 Kbps e um retardo de pacote mínimo de 100 ms para garantir exibição de vídeo contínua, essa QoS pode ser reservada para essa conexão. (Sadok, 2000)

O mecanismo de Serviços Diferenciados não usa sinalização por fluxo. Níveis diferentes de serviços podem ser reservados para grupos diferentes de usuários da Internet, o que significa que o tráfego todo será dividido em grupos com parâmetros de QoS´s diferentes. Isso reduz a carga extra de manutenção em comparação com os Serviços Integrados.

2.4.2 Transmissão multimídia em redes

As etapas de uma transmissão multimídia são mostradas na figura 7 a seguir:

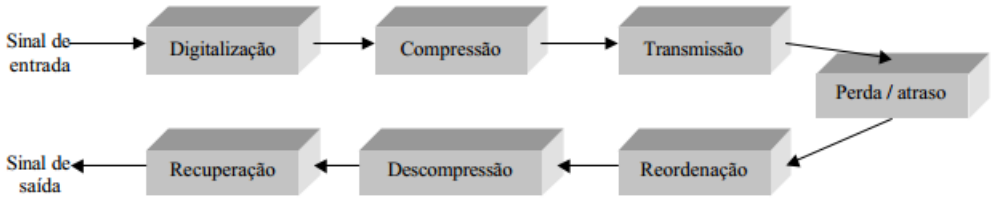

Figura 7 - Etapas de uma transmissão multimídia

Fonte: Roesler, 2001

O sinal gerado é inicialmente digitalizado, para então passar por um processo de compressão, que diminui seu tamanho, tornando-o viável para ser transmitido na rede. A rede insere alguns atrasos no sistema. No receptor, os pacotes são reordenados, descomprimidos e reconvertidos ao estado original, normalmente com perdas inseridas no processo de compressão.

Pode-se dividir a parte de transmissão multimídia em redes de computadores como mostra a Figura 8, ou seja, a parte de conferência (que requer interatividade) e a parte de transmissão de vídeo (que envolve apenas um lado transmitindo e vários clientes recebendo). Ambas possuem necessidades diferentes para funcionarem a contento, por exemplo, as aplicações de conferência normalmente possuem necessidades mais rígidas em relação ao atraso da rede, enquanto que a transmissão unidirecional pode trabalhar com um atraso maior. (Roesler, 2001)

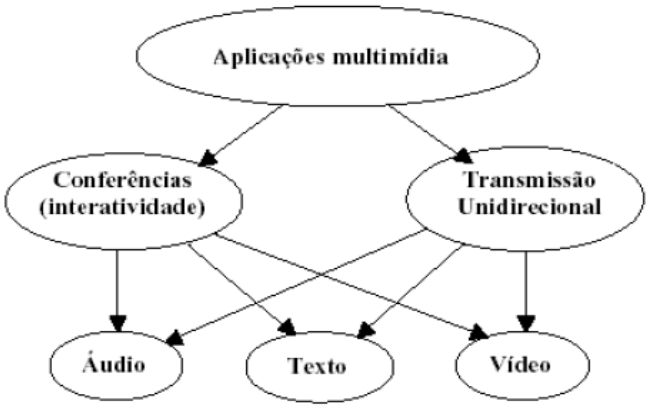

Figura 8 - Transmissão multimídia em rede

Fonte: Roesler, 2001

Apesar das aplicações possuírem necessidades diferentes, existe uma tendência atualmente para sua convergência em um único meio físico. Assim, se unificaria o meioísico, que compartilharia a transmissão de voz, vídeo, dados, imagens, músicas, e tudo que possa ser transformado em bits.

Entretanto, as aplicações têm características e requisitos bem diferentes umas das outras. Aplicações de teleconferência possuem necessidades mais rígidas em relação à latência e jitter do que aplicações de transmissão unidirecional. Da mesma forma, transmissões de vídeo necessitam uma largura de banda muito maior que transmissões de áudio ou texto.

A seguir serão definidos três conceitos fundamentais para o entendimento da transmissão multimídia nas redes de computadores: latência, jitter e skew. Em seguida esses conceitos serão comparados entre si dentro das necessidades das aplicações.

2.4.3 Necessidades das aplicações (Latência, Jitter, Skew, Tabela Comparativa)

Atualmente existe uma tendência de convergência de aplicações em um único meio físico, ou seja, voz, vídeo, dados, imagens, músicas, e tudo que possa ser transformado em bits utilizando o mesmo meio físico. Entretanto, as aplicações tem características e necessidades bem diferentes umas das outras, como por exemplo voz, que exige latência e jitter baixos, dados, que não tem tanta preocupação com latência e jitter, e videoconferência, que além de exigir latência e jitter baixos, ainda necessita de skew baixo, a fim de manter sincronizados voz e vídeo. A seguir será feita a definição desses termos, e depois será mostrada uma tabela comparativa das necessidades das aplicações.(Roesler, 2001)

2.4.3.1 Latência

Segunda Brun(2002), em redes de computadores, latência é o tempo que um pacote leva da origem ao destino. Caso esse atraso seja muito grande, prejudica uma conversação através da rede, tornando difícil o diálogo e a interatividade necessária para certas aplicações. Um atraso confortável para o ser humano fica na ordem de 100ms. Supondo duas pessoas conversando através da Internet. À medida que o atraso aumenta, as conversas tendem a se entrelaçar, ou seja, uma pessoa não sabe se o outro a ouviu e continua falando. Após alguns mili segundos vem a resposta do interlocutor sobre a primeira pergunta efetuada,

misturando as vozes. Num atraso muito grande, as pessoas devem começar a conversar utilizando códigos, tipo “câmbio”, quando terminam de falar e passam a palavra ao outro. Os principais responsáveis pela latência são o atraso de transmissão, de codificação e de empacotamento, que podem ser definidos da seguinte forma:

Atraso de transmissão: tempo após a placa de rede ter transmitido o pacote até ele chegar na placa de rede do computador destino. Esse tempo envolve uma série de fatores, como o atraso no meio físico (por exemplo, fibra ótica, UTP, wireless), processamento em cada roteador ou switch intermediário (por exemplo, para trocar o TTL do pacote e decidir sua rota), fila de espera em cada roteador e switch intermediário, e assim por diante;

Atraso de codificação e decodificação: sinais como voz e vídeo normalmente são codificados em um padrão, tipo PCM (G.711 a 64Kbps) para voz, ou H.261 para vídeo.

Essa codificação gasta um tempo de processamento na máquina. Alguns protocolos gastam menos, como o G.711, que ocupa menos de 1ms de codificação, porém, requer 64Kbps de banda. Alguns protocolos de voz, como o G.729, requerem 25ms de codificação, mas ocupam apenas 8Kbps de banda;

Atraso de empacotamento e desempacotamento: depois de codificado, o dado deve ser empacotado na pilha OSI a fim de ser transmitido na rede, e isso gera um atraso. Por exemplo, numa transmissão de voz a 64Kbps, ou 8000 bytes por segundo, tem-se que, para preencher um pacote de dados contendo apenas 100 bytes, vai levar 12,5ms. Mais 12,5ms serão necessários no destino a fim de desempacotar os dados. Além da latência, a existência do jitter é outro fator de atraso na comunicação entre duas pessoas.(Brun, 2002)

2.4.3.2 Jitter

Utilizar somente a latência não é suficiente para definir a qualidade de transmissão, pois as redes não conseguem garantir uma entrega constante de pacotes ao destino. Assim, os pacotes chegam de forma variável, como mostra a Figura 9, ocasionando o jitter, que nada mais é do uma flutuação na latência, ou variação estatística do retardo.

A consequência do jitter é que a aplicação no destino deve criar um buffer cujo tamanho vai depender do jitter, gerando mais atraso na conversação. Esse buffer vai servir como uma reserva para manter a taxa de entrega constante no interlocutor. Daí a importância de latência e jitter baixos em determinadas aplicações sensíveis a esses fatores, como videoconferência.

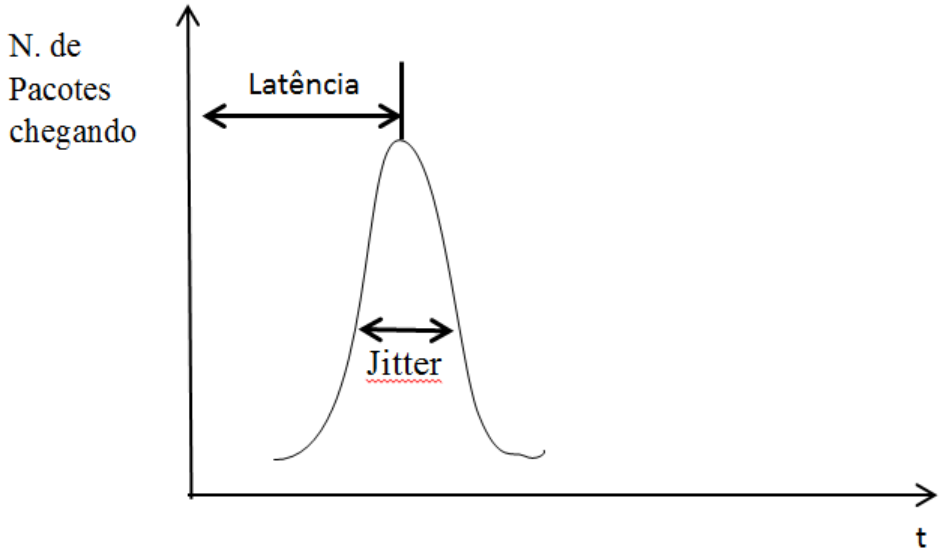

Figura 9 - Comparação entre latência e jitter

Fonte: Brun, 2002

2.4.3.3 Skew

O skew é um parâmetro utilizado para medir a diferença entre os tempos de chegada de diferentes mídias que deveriam estar sincronizadas, como mostra a Figura10. Em muitas aplicações existe uma dependência entre duas mídias, como áudio e vídeo, ou vídeo e dados. Assim, numa transmissão de vídeo, o áudio deve estar sincronizado com o movimento dos lábios (ou levemente atrasado, visto que a luz viaja mais rápido que o som, e o ser humano percebe o som levemente atrasado em relação à visão). Outro exemplo é quando tem-se uma transmissão de áudio explicativo e uma seta percorrendo a Figura 10 associada.

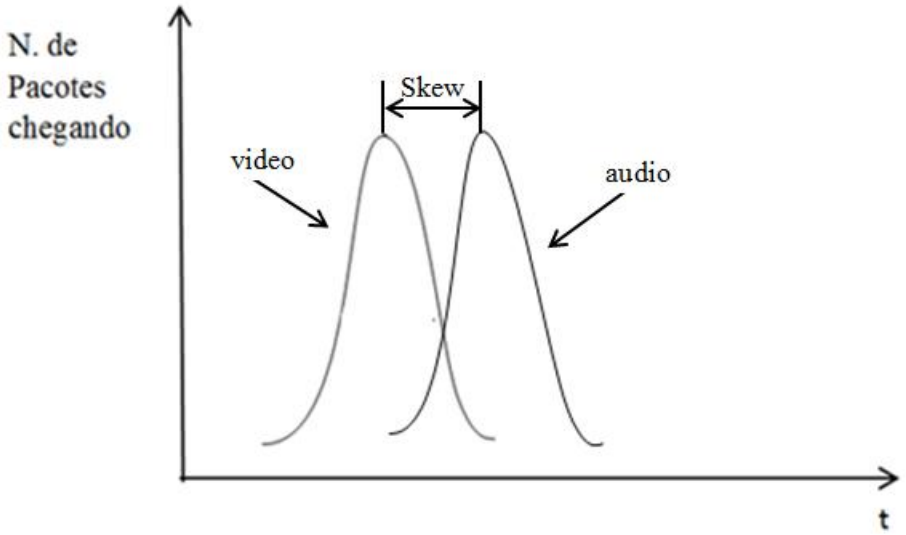

Figura 10 - Definição do Skew entre mídias diferentes

Fonte: Brun, 2002

Abaixo segue a tabela 1 mostrando algumas aplicações típicas em rede, bem como seus fatores críticos, em aplicações numa tendência de convergência nas redes.

Tabela 1 - Fatores críticos em aplicações numa tendência de convergência

	Telefone	Download	**TV**	Videoconferência
Latência	**Sensível**	**Insensível**	**Insensível**	Sensível
Jitter	**Sensível**	**insensível**	**Sensível**	Sensível
Skew	-	**Insensível**	**Sensível**	Sensível
Velocidade (largura banda)	**Baixa**	**Depende**	**Alta**	Alta

Aplicações de telefonia são sensíveis a latência e jitter. Caso estiverem associadas a sincronismo em alguma imagem, como por exemplo, um áudio explicativo associado a uma seta se movendo numa imagem, o áudio também é sensível a skew. Possuem velocidade baixa, de 64Kbps no padrão G.711, o mais comum em telefonia atualmente, mas pode-se chegar a apenas 8Kbps, usando a compressão no padrão G.729.

Aplicações de download de dados são insensíveis a latência, jitter e skew, podem variar em necessidades de velocidade, e possuem taxa variável. Entretanto, na maior parte das vezes esse tipo de mídia não pode sofrer perdas. Pode-se imaginar o problema que pode acontecer de perdas de pacotes numa transação bancária. Já em transmissões unilaterais de áudio e vídeo, como por exemplo TV, a latência não é tão importante, visto que não vai fazer muita diferença se a transmissão demorar 5 segundos para começar a passar. Entretanto, uma vez que começou, deve se manter até o final e com sincronismo entre áudio e vídeo, daí a necessidade de jitter e skew baixos. Aplicações de videoconferência são muito parecidas com aplicações de voz em termos de latência e jitter, entretanto, possuem alta largura de banda e devem manter um baixo skew, pois necessitam sincronização entre áudio e vídeo. (Roesler, 2001)

2.4.4 Qualidade de Serviço (Requisitos Gerais)

A camada de transporte tem como uma das principais funções a ampliação da qualidade de serviço (Quality of Service – QoS) fornecida pela camada de rede. A qualidade

de serviço pode ser caracterizada por uma série de parâmetros específicos (parâmetros QoS). Entre estes podemos citar:

- O retardo no estabelecimento da conexão.
- O retardo no encerramento da conexão.
- A probabilidade de falha no estabelecimento da conexão. Isto é, a probabilidade que uma conexão não seja estabelecida dentro do retardo máximo de estabelecimento.
- A probabilidade de falha na liberação da conexão. Isto é, a fração das tentativas de liberação de conexões que não se completaram dentro do retardo máximo de encerramento.
- A vazão em cada sentido da conexão, isto é, a taxa de bits transferidos por segundo.
- O retardo de transferência médio, também em cada sentido.
- A variação estatística do retardo, expressa, por exemplo, em termos da variância do retardo de transferência.
- A taxa de erro, expressa em porcentagem dos bits transmitidos.
- A prioridade de queda de uma conexão, isto é, a probabilidade de que a camada de transporte. O serviço de transporte permite ao usuário especificar valores preferencias, valores aceitáveis e inaceitáveis, quando do estabelecimento de uma conexão. Alguns dos parâmetros se aplicam tanto ao serviço com conexão quanto ao serviço sem conexão. É função da camada de transporte examinar os parâmetros requeridos e determinar se pode ou não fornecer o serviço.
- A definição da camada de transporte RM-OSI não determina a codificação ou os valores permitidos para os parâmetros QoS.

2.4.4.1 Requisitos Gerais para Suporte a Serviços de Banda Larga

Os requisitos básicos que uma rede de banda larga deve atender para dar suporte aos serviços especificados são identificados na recomendação I.211, que inclui categorias como as de: requisitos multimídia, qualidade de serviços (Quality of Service – QoS), temporização, sincronização e aspectos de sinalização.

A RDSI-FL deverá fornecer facilidades que permitam, em uma única chamada associada a um serviço, estabelecer um número de conexões que podem ser associadas a tipos

específicos de tráfego. Em março de 1993, no encontro de Helsinki, o ITU-T aprovou uma recomendação específica para a infraestrutura de uma rede de banda larga para dar suporte a serviços multimídia (recomendação I.374: *Framework Recomendation on NetworkCapabilities to Support Multimedia Services*).

A recomendação I.374 apresenta uma arquitetura funcional para serviços multimídia que define os elementos de controle de serviço: procedimentos executados, pelos participantes de uma chamada, para fornecer serviços multimídia. Uma chamada pode envolver a participação de várias conexões entre usuários, servidores e equipamentos de apresentação e armazenamento. As diferentes conexões podem apresentar diferentes características para atender a diferentes mídias. Os elementos de controle de serviços são utilizados para o controle das chamadas, o controle das conexões e o controle das mídias. O controle das chamadas inclui o estabelecimentos e liberação de chamadas. O controle das conexões dever permitir o estabelecimento de conexões entre dois ou mais usuários, a inclusão de novos usuários em uma chamada, o desligamento de um participante de uma chamada e a liberação de uma conexão pertencente a uma chamada. O controle das mídias inclui a alocação e liberação de mídias em uma chamada.

A qualidade de serviço(QoS) é, normalmente, um parâmetro negociado na fase de estabelecimento de uma conexão, muito embora, os procedimentos de sinalização permitam que a negociação possa ser feita também após o estabelecimento. Os parâmetros que definem uma determinada qualidade de serviço são definidos pela recomendação I.350. A recomendação I.211 reconhece que novos parâmetros podem vir a ser definidos, como a taxa de perda de células permitida (*cell loss ratio*). Adicionalmente, para alguns serviços, a recomendação I.211 reconhece poder ser necessária a indicação explícita, célula por célula, de uma prioridade para o descarte de células em caso de congestionamento. Essa capacidade é muito útil para minimizar a degradação da QoS para tráfegos do tipo voz ou vídeo.

2.5 Cálculos de largura de banda para produção de vídeo por streaming

Nos itens anteriores foi apresentado sobre protocolos de transporte, TCP/IP, UDP, protocolos específicos para a transmissão de vídeos e suas funcionalidades. Também foi descrito o significado e a importância do QoS e como utiliza-lo. Será descrito a seguir formas de calcular a largura de banda pra Streaming de vídeo.

2.5.1 Bit rate do vídeo

Para uma melhor experiência possível o tamanho e qualidade do ecrã ou dispositivo define qual o codec, formato, e bit rate a usar. O bit rate é medido em 'bits por segundo' (bps ou b/s), muitas vezes utilizado em conjunto com um prefixo SI (nome que precede uma unidade básica de medida), como kbps, Mbps, Gbps, etc., de acordo com o seguinte:

- 1.024 bps = 1 kbps (1 kilobit ou mil bits por segundo)
- 1.048.576 bps = 1 Mbps (1 megabit ou 1 milhão de bits por segundo)
- 1.072.741.824 bps = 1 Gbps (1 gigabit ou um bilhão de bits por segundo)

O bit rate útil de uma comunicação refere-se à capacidade de transferência de um canal excluindo os dados de controle transmitidos (para correcção de erros, etc).

O bit rate é independente do tamanho do vídeo, em altura e largura. Para uma maior resolução do vídeo, requer-se um bit rate maior para conservar a qualidade, ou seja, dois vídeos com o mesmo bit rate, que tenham a mesma duração, devem ocupar mais ou menos o mesmo em disco.

2.5.2 Bit Rate em multimídia

Em multimídia digital, o bit rate representa a quantidade de informação ou detalhe que está guardada por unidade de tempo numa gravação digital (áudio ou vídeo). Este bit rate depende de diversos fatores:

- O material original pode ser digitalizado com diferentes frequências de amostragem;
- As amostragens podem usar números de bits diferentes;
- Os dados podem ser codificados com diferentes técnicas;
- A informação pode ser comprimida com diferentes técnicas de compressão ou em graus diferentes;

Normalmente, estes fatores são escolhidos consoante os objetivos a que se destina o som/vídeo e tem que existir uma troca entre a qualidade (mais bit rate = mais qualidade = mais tamanho) e o tamanho (menos bit rate = menos qualidade = menos tamanho).

Alguns exemplos de bit rates em multimídia digital em Vídeo:

- 16 kbps — Qualidade videofone.
- 1.25 Mbps - Qualidade de VCD (Vídeo CD) (com compressão de vídeo MPEG-1)
- 1.34 Mbps - Qualidade de VCD (Vídeo CD) (com compressão de vídeo e áudio MPEG-PS)
- 5 Mbps – Qualidade de DVD (com compressão MPEG-2)
- 8 até 15 Mbps - Qualidade de HDTV (com compressão MPEG-4 AVC)
- 29.4 Mbps (no máximo) – Qualidade HD DVD
- 40 Mbps (no máximo) – Qualidade Blu-Ray

2.5.3 Produção do sinal de vídeo e bit rate

Para estimar as necessidades de largura de banda da conexão com a Internet, ou seja, a saída a Internet que use o computador que envie o vídeo ao servidor. Esse computador será o "produtor de sinal computador de vídeo".

Inicialmente verifica-se que dispositivo será dedicado o sinal, qual será a qualidade enviada, exemplo: dispositivos móveis, computadores de área de trabalho, transmissão em HD, etc. Após decidido os meios físicos que será enviado, pode-se fazer algumas estimativas:

- Para uma boa qualidade de vídeo em móveis, pode-se visar um bit rate de 200Kbps.
- Uma boa audiência para computadores de área de trabalho necessita-se um vídeo de maior qualidade, de modo que o bit rate seja em torno dos 400 Kbps.
- Paraum vídeo de alta qualidade a estimativa de bit rate seria 800 Kbps.
- Para vídeo em HD (alta definição), seria necessário um bit rate em torno de 1300 Kbps.

2.5.4 Cálculo da largura de banda

Para calcular a largura de banda é necessário conhecer o tamanho do arquivo a ser transmitido. O tamanho do arquivo é dependente de vários fatores, dentre eles, o tipo de compressão do vídeo (MJPEG, MPEG4, etc.), a resolução da Imagem (704x480, 352x240,

etc.), número de quadros por segundo (qps), número de usuários simultâneos. Segue abaixo dois exemplos básicos para visualizar melhor como funciona o consumo de banda:

Exemplo 1: - câmera usando a compactação MPEG-4 com 1qps, tem-se:

1 (câmera) X 25KB (tamanho da Imagem) X 1qps = 25KB/s (sempre convertendo KB/s para Kbits/s, caso contrário os resultados serão incorretos.)

25KB/s X 8 bits/Byte = 200 Kbits/s esta é a largura de banda necessária.

Exemplo 2:

- Dezesseis (16) câmeras com arquivos de 20KB a uma taxa de transmissão de 1qps por câmera que sejam assistidas por dois usuários, temos:

16 câmeras X 20KB X 1ps X 2 usuários X 8 bits por byte = 5,12 Mbits/s.

Dimensionamento do disco rígido para armazenar imagens:

Vários fatores devem ser considerados no cálculo da necessidade de espaço para armazenamento:

- Número de câmeras;
- Número de horas por dia que as câmeras estarão gravando;
- Quanto tempo os dados devem permanecer armazenados;
- Detecção de movimento (por evento) ou gravação constante;
- Taxa de quadros, compressão, qualidade e complexidade da Imagem.

O cálculo do exemplo abaixo não considera o espaço em disco para o sistema operacional, software de gerenciamento de vídeo, etc.

Motion JPEG No formato Motion JPEG onde arquivos individuais são recebidos, as exigências de armazenamento variam alterando a taxa de quadros, a resolução e a compressão. As câmeras 1, 2 e 3 na tabela 2 abaixo têm exigências diferentes de acordo com o número de quadros por segundo (qps) e a resolução.

Cálculo: Tamanho da Imagem x quadros por segundo x 3600s = KB por hora / 1000 = MB por hora MB por hora x horas de operação por dia / 1000 = GB por dia GB por dia x período de armazenamento = Necessidade de espaço de armazenamento

Câmera Resolução Tamanho da Imagem (KB)

Tabela 2 Quadros por segundo MB/horaHoras de operaçãoGB/dia

Nº. 1	CIF	13	5	234	8	1,9
Nº. 2	CIF	13	15	702	8	5,6
Nº. 3	4CIF	40	15	2160	12	26

Total de 3 câmeras com 30 dias de armazenamento = 1002GB.

MPEG-4 No formato MPEG-4 as imagens são recebidas em um fluxo contínuo de dados, isto é, não como arquivos individuais. É o bit-rate que determina o respectivo espaço de armazenamento. O bit-rate é um resultado da taxa de quadros, da resolução e da compressão, bem como do nível de movimento da cena.

Cálculo: Bit rate / 8 (bits em um byte) x 3600s = KB por hora / 1000 = MB por hora MB por hora x horas de operação por dia / 1000 = GB por dia GB por dia x período de armazenamento = Necessidade de espaço de armazenamento

Câmera Resolução Tamanho da Imagem (KB) .

Total de 3 câmeras com 30 dias de armazenamento = 204GB.

Como Escolher a Taxa de quadros para Visualização/Gravação.

Para cada aplicação existe uma relação de número de quadros por segundo mais adequada.

Escolhe-se da melhor configuração é dependente do tipo cena, sua complexidade e do nível de detalhe que se deseja monitorar. A tabela abaixo mostra algumas aplicações com as respectivas taxas de quadros.

Tabela 3 Quadros por segundo MB/horaHoras de operaçãoGB/dia

Nº. 1	CIF	170	5	76,5	8	0,6
Nº. 2	CIF	400	15	180	8	1,4
Nº. 3	4CIF	880	15	396	12	5

AplicaçãoNº. de Quadros

Monitoramento em geral, estoques, escritórios 1 – 2 qps

As imagens não terão movimentação, porém a maioria dos movimentos serão gravados.

PTZ em modo tour ou automático3 – 4 qps

As imagens apresentarão movimento e a movimentação do PTZ será suave.

Captura de placa veicular3,75 – 15 qps

É necessário determinar a velocidade dos veículos no campo de visão da câmera para determinar a taxa correta.

Caixas registradoras15 – 20 qps

Aplicação em tempo real são necessários no mínimo 15 qps.

Verificação de valores. Aplicações de alta segurança Identificar eventos

3 - Recepção de vídeo em Android

Uma das partes importantes para o desenvolvimento do trabalho foi o estudo da recepção de vídeo em Android, nesse capítulo será apresentado os desafios da execução de vídeos, tipos de mídias suportados no Android, protocolos de redes, codificações de vídeos recomendadas, todos esses estudos foram feitos para um melhor entendimento de como se comporta a tecnologia de desenvolvimento no âmbito do Android.

Além do entendimento sobre funcionamento da recepção de vídeo, será abordado a forma de desenvolver, programar, os métodos e classes para a recepção de vídeo, e um aprofundamento nos estados de um aplicativo MediaPlayer.

3.1 Os desafios da execução de vídeo

Há tempos, a execução de vídeos em computadores e dispositivos móveis apresentam desafios para desenvolvedores, pois o desenvolvimento com vídeo exige um entendimento de tecnologia e terminologia complexas.

Vídeo é a combinação de um fluxo de vídeo (ou imagem) e um ou mais fluxos de áudio, tudo compactado em um único arquivo. Em linguagem de vídeo comum: (Paris, 2012)

- Um contêiner de vídeo encapa arquivos de áudio e vídeo juntos em um único *archive* de arquivo. Existem muitos formatos de contêiner de vídeo e alguns populares são MPEG4, Flash Video, Ogg, WebM e Audio Video Interleave. O formato do contêiner é indicado pela extensão do arquivo, tal como mp4, flv, ogv, webm e avi, dentre outros.
- Um codec de vídeo Identifica o algoritmo de software pelo qual um fluxo de vídeo é codificado e decodificado (compactado e descompactado). Um reprodutor de vídeo deverá saber qual codec foi usado para poder decodificar e executar o fluxo de vídeo.
- Um codec de áudio é semelhante a um codec de vídeo, porém, para fluxos de áudio.

3.2 Tipos de Mídia suportados pelo Android

Este tópico descreve o codec de mídia, codificação de vídeo recomendada e suporte de protocolo de rede fornecida pela plataforma Android.

Para desenvolver aplicativos, usa-se qualquer codec de mídia, que está disponível em qualquer dispositivo Android, incluindo os fornecidos pela plataforma Android e aqueles que são específicas do dispositivo. No entanto, é uma boa prática usar perfis de codificação de mídia que são *device-agnostic*. (Android Developers, 2012, 2012)

3.2.1 Protocolos de rede e formatos de midias

Alguns dos seguintes protocolos de rede, já especificados no capítulo3,são suportados para a reprodução de áudio e vídeo:

- RTSP (RTP, SDP)
- HTTP / HTTPS streaming progressivo
- HTTP / HTTPS live streaming projecto de protocol:
    - MPEG-2 TS arquivos de mídia apenas
    - Protocol versão 3 (Android 3.0 e superiores)
    - Protocol version 2 (Android 3.x)
    - Não é suportado antes de Android 3.0

Nota: HTTPS não é suportado antes de Android 3.1.

A Figura 11 abaixo descreve o suporte ao formato de mídia incorporado na plataforma Android. Nota-se que um determinado dispositivo móvel pode fornecer suporte para formatos adicionais ou tipos de arquivos não listados na tabela.

Nota: Os codecs de mídia que não são garantidos para ser disponível em todas as versões da plataforma Android são, portanto, indicado entre parênteses, por exemplo "(Android 3.0 +)".

Type	Format	Encoder	Decoder	File Type(s) Supported
Audio	AAC LC/LTP	X	X	3GPP (.3gp) and MPEG-4 (.mp4, .m4a). No support for raw AAC (.aac)
	HE-AACv1 (AAC+)		X	
	HE-AACv2 (enhanced AAC+)		X	
	AMR-NB	X	X	3GPP (.3gp)
	AMR-WB	X	X	3GPP (.3gp)
	MP3		X	MP3 (.mp3)
	MIDI		X	Type 0 and 1 (.mid, .xmf, .mxmf). Also RTTTL/RTX (.rtttl, .rtx), OTA (.ota), and iMelody (.imy)
	Ogg Vorbis		X	Ogg (.ogg)
	PCM/WAVE		X	WAVE (.wav)
Image	JPEG	X	X	JPEG (.jpg)
	GIF		X	GIF (.gif)
	PNG	X	X	PNG (.png)
	BMP		X	BMP (.bmp)
Video	H.263	X	X	3GPP (.3gp) and MPEG-4 (.mp4)
	H.264 AVC		X	3GPP (.3gp) and MPEG-4 (.mp4)
	MPEG-4 SP		X	3GPP (.3gp)

Figura 11 - Formatos de Mídias

Fonte: Android Developers, 2012

Legenda Figura 25: (Type = Tipo, Format = formato, Encoder = Codificador, Decoder = decodificador, File Type(s) Supported = Tipos de arquivos suportados)

3.2.2 Codificação de Vídeos Recomendadas.

A tabela 4, abaixo, apresenta exemplos de perfis de codificação de vídeo e parâmetros que o Android *media framework* suporta para a reprodução. Em adição a estas recomendações dos parâmetros de codificação, os perfis de um dispositivo de gravação de vídeo disponíveis que podem ser usadas como um indicador de capacidade de reprodução de mídia. Estes perfis podem ser inspecionados através da classe *CamcorderProfile*, que está disponível a partir de API nível 8.

Tabela 4 Exemplos de parâmetros de codificação de vídeo suportados

	SD (Low quality)	SD (High quality)	HD (Not available on all devices)
Video codec	H.264 Baseline Profile	H.264 Baseline Profile	H.264 Baseline Profile
Video resolution	176 x 144 px	480 x 360 px	1280 x 720 px
Video frame rate	12 fps	30 fps	30 fps
Video bitrate	56 Kbps	500 Kbps	2 Mbps
Audio codec	AAC-LC	AAC-LC	AAC-LC
Audio channels	1 (mono)	2 (stereo)	2 (stereo)
Audio bitrate	24 Kbps	128 Kbps	192 Kbps

3.3 Media Playback

A plataforma Android tem embutido um sistema de codificação / decodificação para uma variedade de tipos de multimídia comuns, para se integrar vídeo, áudio e imagens nas aplicações.

O Android permite reproduzir áudio e vídeo de vários tipos de fontes de dados. Pode reproduzir áudio ou vídeo a partir de arquivos armazenados na pasta "raw" da aplicação, a partir de arquivos autónomos no sistema de arquivos, ou de um *stream* através de uma conexão de rede. Para reproduzir o áudio ou vídeo usa-se a classe *MediaPlayer*. (Android Developers, 2012)

A plataforma também permite gravar áudio e vídeo, quando suportado pelo hardware do dispositivo móvel. Para gravar áudio ou vídeo, usa-se a classe *MediaRecorder*.

Basicamente as seguintes classes são utilizadas para reproduzir som e vídeo no âmbito do Android:

MediaPlayer : Esta classe é a API primária para reproduzir som e vídeo.

AudioManager: Essa classe gerencia as fontes de áudio e saída de áudio em um dispositivo.

Antes de iniciar o desenvolvimento do aplicativo usando o *MediaPlayer*, é preciso verificar se o *Manifest* (manifesto) tem as declarações apropriadas para permitir o uso de recursos relacionados, sendo elas:

Permissão Internet – Usando o *MediaPlayer*para fluxo de rede baseada em conteúdo, o aplicativo deve solicitar acesso à rede.Exemplo abaixo:

```
<uses-permission android:name="android.permission.INTERNET" />
```

3.3.1 Usando o MediaPlayer

Um dos componentes mais importantes do *framework* de mídia é a classe *MediaPlayer*. Um objeto desta classe pode buscar, decodificar e reproduzir áudio e vídeo com configuração mínima. Ele suporta várias fontes diferentes de mídia, tais como:

- Recursos locais
- URIs internos, onde pode-se obter um *Content Resolver*.
- URLs externos (streaming)

Abaixo seguem alguns exemplos:

Exemplo de como reproduzir o áudio que está disponível como um *raw resource* local (salvo em na aplicação em res / diretório / raw):

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.sound_file_1);
mediaPlayer.start();
```

Exemplo de como reproduzir a partir de uma URL remota via HTTP *streaming*.

```
String url = "http://........";
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare();
mediaPlayer.start();
```

Nota: Para passar uma URL para um arquivo de *streaming* de mídia *on-line*, o arquivo deve ser capaz de *download* progressivo.

- **API Camera:** Classes relevantes para captura de imagem ou vídeos, através da API câmera ou câmera intent.
- **Camera:** Esta classe é a principal API para controlar câmeras de dispositivos. Essa classe é usada para tirar fotos ou vídeos quando se está construindo um aplicativo de câmera.
- **Suface View**: Essa classe é usada para apresentar uma pré-visualização de câmera ao vivo para o usuário.
    - .VideoView :extends SurfaceView

Implementa MediaController.MediaPlayerControl ; capaz de reproduzir fluxos de vídeos dentro de contêineres de mídia 3GP (3GPP, 2004) e MP4 (ISO, 2003).

- .MjpegView: extends SurfaceView :

  Capaz de reproduzir fluxo de vídeos em M-JPEG.

- **MediaRecorder:** Essa classe é usada para gravar o vídeo da câmera.

3.4 Diagrama de Estado

O controle de reprodução de áudio, vídeo e streaming são gerados como uma máquina de estado. O diagrama a seguir mostra o ciclo de vida e os estados de um objeto *MediaPlayer* e suporte de controle de operações dirigidas. As formas ovais representam os estados de um objeto MediaPlayer podem ter .Os arcos representam as operações de controle de reprodução que impulsionam a transição de estado do objeto. Existem dois tipos de arcos:

- Os arcos com uma única seta representam chamadas de métodos síncronos
- Os arcos com duas setas representam chamadas de método assíncrono.

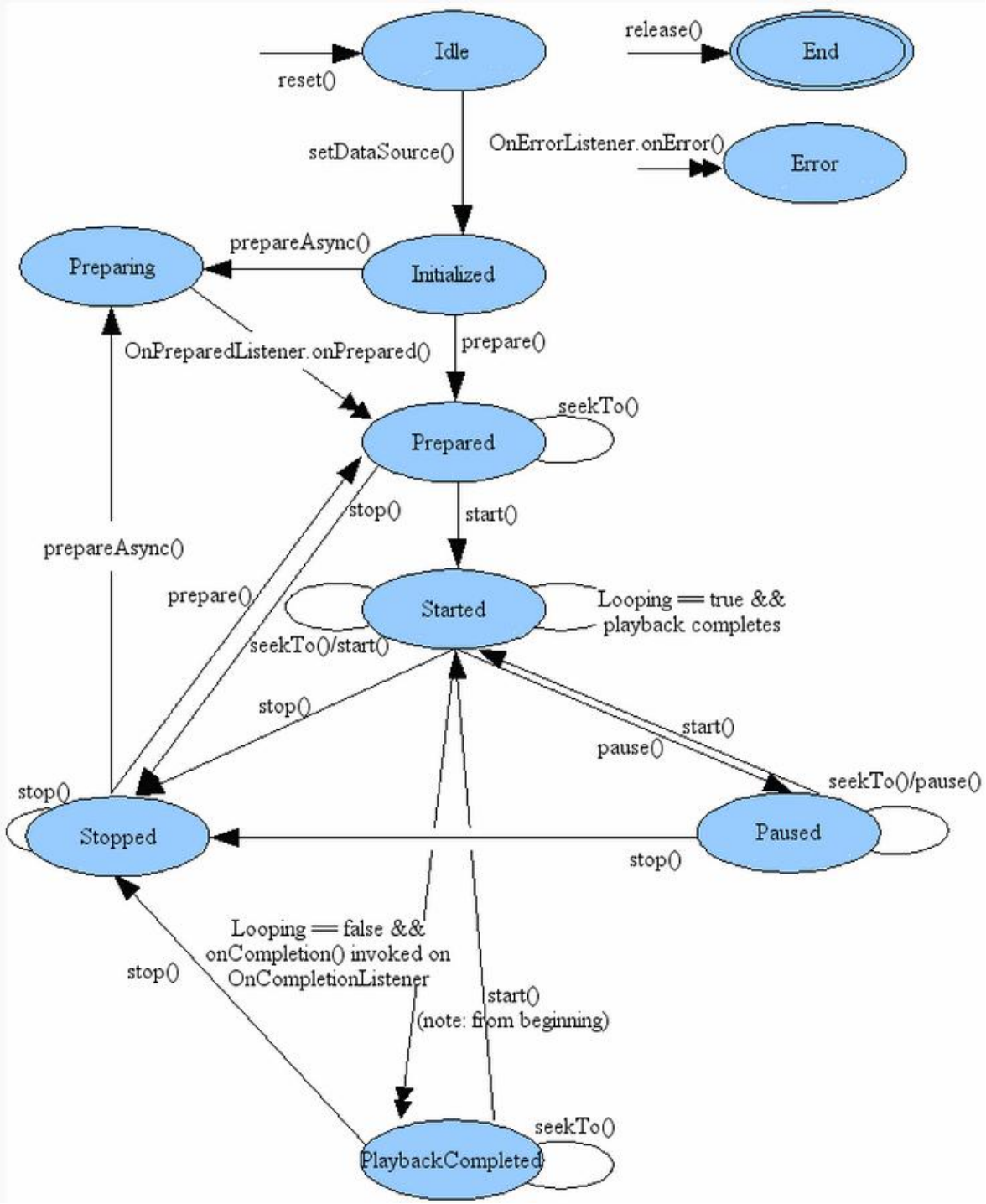

Figura 12 - Diagrama de estados do abjeto MediaPlayer

Fonte : Developer 2011

Estados do objeto MediaPlayer , através do diagrama de estados :

- Um objeto MediaPlayer é criado usando *new* e depois *reset*() é chamado, encontrando-se no estado ocioso, após chamar o *release*() chaga-se no estado final. Entre esses dois estados é o ciclo de vida do objeto MediaPlayer.

- Há uma diferença sutil, mas importante entre um objeto MediaPlayer recém-construído e o objeto MediaPlayer após ter chamado o *reset*() . Não é uma boa prática de programação invocar métodos como :*getCurrentPosition(), getDuration(), getVideoHeight(), getVideoWidth(), setAudioStreamType(int), setLooping(boolean), setVolume(float, float), pause(), start(), stop(), seekTo(int), prepare() ou prepareAsync()* no estado ocioso para ambos os casos. Se qualquer um desses métodos é chamado logo após um objeto MediaPlayer for construído, o usuário solicita um método de retorno *OnErrorListener.onError()* não será chamado pela máquina de estados e o objeto continua inalterado ; mas se esses métodos são chamados logo após o *reset()* o usuário solicita um método de retorno *OnErrorListener.onError()* será invocado pela máquina de estados interna e o objeto será transferido para o estado de Erro(*Error*).
- Recomenda-se também que uma vez que um objeto MediaPlayer não esta sendo mais usado chama-se imediatamente o *release()* , para que os recursos usados pela máquina de estados associados ao objeto Media Player possam ser liberados.
- Futuramente, os objetos criados usando o *new* no MediaPlayer estão no estado *Idle* (inativo) , aqueles criados com um dos métodos *create (ele é sobrecarregado)* não ficam do estado *Idle* (inativo).De fato se o objeto for criado usando o método *(de forma bem sucedida)* ele estará no estado*Prepared (preparado).*

- Em geral, uma operação de controle de reprodução pode falhar devido a várias razões, como formatos de áudio/vídeo não suportados, resolução muito alta, o tempo limite de streaming, entre outros. Relatório de erros e recuperação é uma preocupação importante nestas circunstâncias. As vezes, devido a erros de programação, invocar uma operação de controle do playback (reprodução) em um estado inválido pode ocorrer.
  - É importante notar que, uma vez que ocorre um erro, o objeto MediaPlayer entra no estado de *Error* (erro) (com a exceção indicada acima), mesmo se um erro não foi registrado pelo aplicativo.

- A fim de reutilizar o objeto MediaPlayer que se encontra no estado de Error (erro) e recuperá-lo , o *reset()* pode ser chamado para restaurar o objeto do estado *Idle* (inativo) .
- É boa prática de programação para ter sua aplicação registrar um *OnErrorListener* ,para poder ver a notificações de erro do IPE.
- *IllegalStateException* é acionada para evitar erros de programação, como chamar *prepare(), prepareAsync(),*ou de um dos métodos *overloaded* (sobrecarregados) *setDataSource* em um estado inválido.

- Chamando, *setDataSource(FileDescriptor),* ou *setDataSource(String),* ou *setDataSource(Context, Uri),* ou *setDataSource(FileDescriptor, long, long)* transfere um objeto MediaPlayer no estado *Idle*(inativo) para o estado *Initialized* (inicializado).
  - Um*IllegalStateException* é lançada se *setDataSource ()* é chamado em qualquer outro estado.
- Um objeto MediaPlayer deve primeiro entrar no estado *Prepared* (preparado) antes da reprodução ser inicializada.
  - Há duas maneiras (síncrono *vs* assíncrono) que o estado *Prepared* (preparado) podem ser alcançados: em vez de chamar *prepare()Synchronous* (síncrono), que transfere o objeto para o estado *Prepared* (preparado) ou chama *prepareAsync() (asynchronous)* que primeiro transfere o objeto para o estado *Prepared* (preparado) após o retorno de chamada (que ocorre de uma forma quase certa) enquanto o IPE continua trabalhando no resto da preparação até completá-la. Quando a preparação completa ou quando *prepare()* retorna a chamada, o IPE , então, chama um método de *callback* fornecida pelo usuário, *onPrepared ()* da interface *OnPreparedListener,* se um *OnPreparedListener* está registado previamente através *setOnPreparedListener(Android.media.MediaPlayer.OnPreparedListener).*
  - É importante notar que o estado *Prepared* (preparado) é um estado transitório, e o comportamento de chamar qualquer método com um efeito secundário, enquanto um objeto MediaPlayer está no estado *Prepared* (preparado) é indefinido.
  - Um IllegalStateException é lançada se *prepare()* ou *prepareAsync ()* é chamado em qualquer outro estado.

- Enquanto no estado *Prepared* (preparado) , propriedades, tais como áudio/ volume de som, *screenOnWhilePlaying*, *looping* pode ser ajustada invocando os métodos *set* correspondentes.

- Para iniciar a reprodução (playback), deve ser chamado *start()*.Depois que *start()*retornar com sucesso, o objeto MediaPlayer estará no estado *Started* (iniciado). *isPlaying()* pode ser chamado para testar se o objeto MediaPlayer esta no estado *Started* (iniciado).
  - Enquanto no *Started* (iniciado), o IPE chama um usuário fornecido *OnBufferingUpdateListener.onBufferingUpdate()*método de retorno se um *OnBufferingUpdateListener* foi registrado previamente via *setOnBufferingUpdateListener(OnBufferingUpdateListener)*. Este *callback* permite que os aplicativos para acompanhar o status de buffer durante a *streaming* (transmissão) de áudio e vídeo.
  - Chamar *start()*não tem nenhum efeito sobre o objeto MediaPlayer se ele já esta no estado *Started* (iniciado).
- O Playback (reprodução) pode ser pausado e parado, e a posição do *playback* (reprodução) pode ser ajustada. O *playback* (reprodução)pode ser pausado com *pause()*. Quando a chamada *pause()* retorna, o objeto MediaPlayer fica no estado *Paused* (pausado). Nota-se que a transição do estado *Started* e para o estado *Paused*e vice-versa acontece de forma assíncrona no *player engine*. Pode levar algum tempo antes que o estado seja atualizado em chamadas para *isPlaying()*,e pode ocorrer em segundos, no caso de o conteúdo *Streaming* (transmissão).
  - Chamar *start()* para retomar a *playback* (reprodução) de uma pausa do objeto MediaPlayer, e o *playback* (reprodução) é retomado na mesma posição que foi interrompido. Quando a chamada *start()* retorna, o objeto MediaPlayer pausado retorna ao estado *Started* (iniciado).
  - Chamando *pause()* não ocorre efeito no objeto MediaPlayer que já se encontra no estado *Paused* (pausado).
- Chamando *stop()*interrompe o *playback* (reprodução) e provoca no MediaPlayer nos estados *Started* (iniciado), *Paused* (pausado), *Prepared* (preparado) ou *PlaybackCompleted* (reprodução completa) a entrada do estado *Stopped* (parado).

- Uma vez no estado *Stopped*(parado), o playback(reprodução) não pode ser iniciada até chamar *prepare*() ou *prepareAsync*() para definir o objeto MediaPlayer para o estado preparado novamente.
- Chamando *stop()* não ocorre efeito no objeto MediaPlayer que já se encontra no estado *Stopped* (parado).

- A posição do *playback* (reprodução)pode ser ajustada com uma chamada para *seekTo(int).*
  - Embora o *asynchronuous* (assíncrono) chamando *seekTo(int)* retorna o caminho certo, a operação de busca atual pode demorar um pouco para terminar, especialmente para áudio e vídeo sendo transmitido. Quando a operação de busca atual termina, o IPE chama um usuário fornecido *OnSeekComplete.onSeekComplete()* se um *OnSeekCompleteListener* foi registrado previamente via *setOnSeekCompleteListener(OnSeekCompleteListener).*
  - Nota-se que*seekTo(int)* também pode ser chamado em outros estados, tal como,), *Prepared* (preparado), *Paused* (pausado) e *PlaybackCompleted* (reprodução completa).
  - Além disso, a posição real de reprodução atual pode ser recuperada com uma chamada para *getCurrentPosition(),*o que é útil para aplicações tais como um leitor de música que precisa para se manter a par do progresso da reprodução.
- Quando o *playback* (reprodução) atingir o fim do fluxo, ele estará completo.
  - Se o modo de *Looping* (repetição) estava sendo definido como *true* (verdadeiro) com *setLooping(boolean*), o objeto MediaPlayer permanecerá em estado inicial.
  - Se o modo de *Looping* (repetição)foi definido como *false* (falso), o *player engine* chama um método de retorno programado pelo usuário, *OnCompletion.onCompletion()* se um *OnCompletionListener* está registado previamente através *setOnCompletionListener(OnCompletionListener).* A invocação *call-back* (retorno de chamada) mostraque o objeto está agora no estado *PlaybackCompleted.*
  - Enquanto no estado *PlaybackCompleted*, chamar *start ()* pode reiniciar o *playback* (reprodução) desde o início da fonte de áudio e vídeo.

3.5 - Reprodução a Partir de um Arquivo ou Stream

Pode-se reproduzir ficheiros de mídia do sistema de ficheiros ou uma URL web:

1. Criar uma instância do *MediaPlayer* usando *new*

3. Chama *setDataSource()* com uma *String* contendo o caminho (ficheiro local ou URL) para o ficheiro que você quer reproduzir

3. Primeiro *prepare()* , em seguida, *start()* na instância:

```
MediaPlayer mp = MediaPlayer new ();
mp.setDataSource (path_to_file);
mp.prepare ();
mp.start ();
```

Para parar a reprodução, usa-se*stop()* . Se for preciso mais tarde fazer replay, deve fazer *reset()* e *prepare()* antes de chamar *start()* novamente.

Para parar momentaneamente a reprodução, usa-se *pause()* . Para retomar a reprodução a partir de onde parou usa-se *start()* . (Android Developers, 2012)

4 - Trabalho correlato (DroidMinder)

Este capítulo apresenta o DroidMinder, um aplicativo para dispositivos móveis munidos do sistema Android, que permite monitorar câmeras de vigilância conectadas a um servidor Zoneminder. que foi um dos principais trabalho correlato estudado para a escrita desta monografia.

4.2–Introdução

O mercado de dispositivos móveis está cada vez mais investindo nos telefones inteligentes (smartphones), os quais permitem navegar na Internet, ler e-mails, etc. através da rede 3G de telefonia celular e também através de redes sem fio Wi-Fi (IEEE, 2007). Tais telefones são caracterizados pelo poder de processamento, quantidade de memória, uma tela grande e principalmente por um sistema operacional mais elaborado capaz de gerenciar aplicações semelhantes com aquelas destinadas aos computadores pessoais. Diante dos diversos sistemas operacionais o Android (OPEN HANDSET ALLIANCE, 2008) tem-se mostrado como uma opção interessante. Trata-se de uma plataforma de código aberto que fornece além do sistema operacional um kit de desenvolvimento (Software Development Kit – SDK) que traz facilidades no desenvolvimento de aplicações para este sistema.

Diferentemente dos primeiros telefones capazes de navegar na Internet através do protocolo WAP (OPEN MOBILE ALLIANCE, 1998), os telefones inteligentes possuem navegadores capazes de exibir páginas web que não foram especificamente desenhadas para estes. Contudo, a navegação por tais páginas é um pouco incômoda devido ao tamanho e resolução da tela o que leva ao usuário recorrer a ferramentas de zoom.

Sistemas de vigilância com câmeras estão cada vez mais em uso. Sendo, hoje em dia, sua implantação muito simples e de baixo custo, necessitando-se apenas de um computador pessoal, uma placa de captura de vídeo e algumas câmeras. O que o torna também um sistema viável para implementar-se domiciliarmente. Esse sistema pode inclusive permitir a visualização das imagens filmadas pela internet.

Alguns softwares para vigilância disponibilizam uma interface web específica para telefone celular. Estas interfaces são projetadas para apresentar uma menor quantidade de informação para se adequar às telas pequenas e de baixa resolução. Porém, tais interfaces

ainda assim não oferecem uma boa experiência de uso, quando comparadas com a usabilidade das aplicações nativas dos telefones inteligentes.

Nesta seção é apresentado o DroidMinder, um aplicativo de código aberto para dispositivos Android 1.6 ou superior, que permite monitorar pela Internet câmeras conectadas a um sistema de vigilância baseado no Zoneminder.(Michael,2010)

4.3 -Zoneminder

O ZoneMinder (ZONEMINDER, 2004) é uma aplicação de código aberto para monitoramento de câmeras de vigilância que pode ser usado com câmeras conectadas a uma placa de captura de vídeo, câmeras USB ou mesmo câmeras IP. Provê suporte à captura, análise e gravação de imagens oriundas de uma ou mais câmeras, além de implementar protocolos da indústria para movimentar câmeras Pan/Tilt/Zoom (PTZ).

Concebido de forma de distribuída e fazendo uso das linguagens C++, Perl e PHP, o Zoneminder possui um agente centralizador, denominado servidor, o qual agrupa agentes de monitoração (p.e. câmeras) e a interação com o usuário se faz através de agentes clientes (aplicação de usuário). Como agente cliente o ZoneMinder apresenta uma página web que permite ao usuário adicionar câmeras, acompanhar ao vivo as imagens que estão sendo geradas e também ativar a gravação dessas imagens.(Michael. 2010)

(a) Interface padrão (b) Interface para dispositivos móveis

Figura 13 - Interface de usuário do Zoneminder

Fonte: ZoneMinder, 2004

Além da interface web padrão, o Zoneminder apresenta uma interface web simplificada para dispositivos móveis. Esta interface só permite o monitoramento das câmeras previamente cadastradas. A Figura 13a apresenta a interface web padrão e a Figura 13b apresenta a interface simplificada para dispositivos móveis.

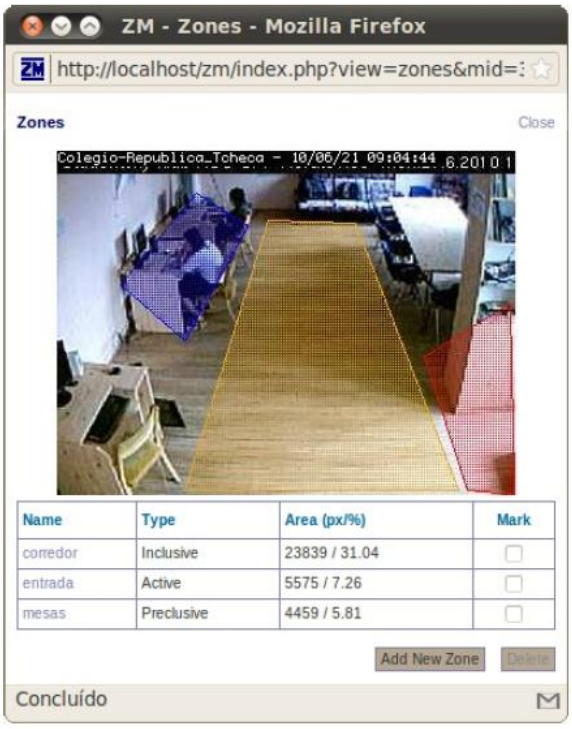

Figura 14 - Delimitação das áreas que serão monitoradas

Fonte: ZoneMinder,2004

Entre as características do ZoneMinder, a mais interessante é a detecção de movimentos. Isto permite disparar a gravação automática do vídeo mediante alguma movimentação no ambiente monitorado, garantindo um menor consumo de recursos, no caso espaço em disco. Também permite o envio de alertas por e-mail ou via mensagens de texto por celular (SMS). A Figura 14 ilustra a interface para seleção de zonas de monitoramento em uma determinada câmera. Neste exemplo existem três áreas hachuradas, corredor, entrada e mesas. É possível definir a ação que será tomada em cada área. Por exemplo, seria interessante disparar um alarme sempre que algum movimento for detectado na área "entrada". Por outro lado, movimentos na área "mesas" nunca deveria disparar um alarme por movimento. (Emerson, 2010)

Os vídeos gerados pelas câmeras são transmitidos como fluxos M-JPEG (ITU, 2005) sobre o protocolo HTTP. No caso, o servidor HTTP envia uma sequência de quadros JPEG e cabe ao navegador web reproduzi-los. Muitos dos atuais navegadores para computadores pessoais possuem suporte para reproduzir fluxos M-JPEG. Porém, em testes realizados notou-se problemas de reprodução dos vídeos quando acessados através de um navegador web de um telefone celular. (Michael,2010)

4.4 - Desenvolvimento Android

O kit de desenvolvimento de software (SDK) para o Android, composto por bibliotecas, depurador, documentação, códigos exemplos e um emulador de telefone celular, aliado ao ambiente de desenvolvimento integrado (AID) Eclipse1, formam uma ferramenta completa para escrita, compilação, depuração e execução de aplicações Android.

Uma aplicação Android típica é formada por dois arquivos: (1) descritor de interface com o usuário, escrito em XML; (2) classe Java a qual contém a lógica de programação e a associação desta lógica com cada elemento gráfico da interface com o usuário. Dentro do arquivo XML cada elemento gráfico na interface do usuário é chamado de View. Tem-se elementos para interação com o usuário como TextView, o qual exibe texto, EditView que permite inserir textos, e elementos que atuam como contêineres e agrupam outros elementos, como o LinearLayout e TableLayout. A Figura 15 apresenta as relações entre os elementos visuais para o usuário com sua descrição em XML e como são identificados dentro da classe Java responsável pela lógica funcional da aplicação.(Michael. 2010)

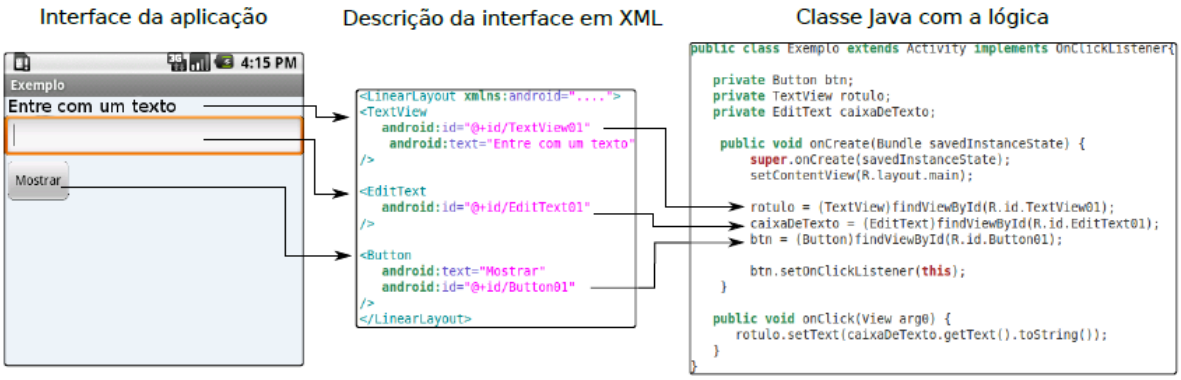

Figura 15 - Concepção de uma aplicação Android

Fonte: Michael,2010

4.5 - Reprodução de fluxos de vídeos

O Android fornece a VideoView a qual é capaz de reproduzir fluxos de vídeos dentro de contêineres de mídia 3GP (3GPP, 2004) e MP4 (ISO, 2003). Porém, conforme descrito na Seção 2 o Zoneminder por padrão envia fluxos M-JPEG e sendo assim o VideoView não está apto para reproduzi-los. Klös (2009) desenvolveu a classe MjpegView, derivada da classe Android SurfaceView, que é capaz de reproduzir fluxos M-JPEG. Esta classe foi essencial para o desenvolvimento do DroidMinder .(Emerson, 2010)

4.6 –DroidMinder

Para Michael , 2010, apesar do Zoneminder oferecer um agente cliente específico para aparelhos móveis, uma interface web simplificada, este não foi pensado para os novos telefones inteligentes. O DroidMinder consiste em uma aplicação para Android 1.6, ou superior, cujo objetivo é atuar como agente cliente para um sistema de vigilância ZoneMinder.

O Zoneminder implementa mecanismos de segurança para autenticação de usuários e para controle de acesso. Os usuários são classificados como administrador, com permissão total ao sistema; e como usuário comum, somente capaz de visualizar as câmeras de monitoramento. O uso de tais mecanismos garante que as imagens das câmeras só estejam disponíveis para usuários autênticos e autorizados. Contudo as informações trocadas entre o usuário, através de um navegador web, e o Zoneminder ainda estão susceptíveis a interceptação. Por estar rodando sobre o servidor web Apache, pode-se fazer uso dos protocolos SSL/TLS (FREIER; KARLTON; KOCHER, 1996; DIERKS; ALLEN, 1999) para garantir assim a confidencialidade dos dados.

No Zoneminder, informações sobre agentes de monitoramento (câmeras), usuários, alarmes, etc são armazenadas em um banco de dados relacional, no caso o MySQL3. O agente cliente padrão do ZoneMinder consiste de uma página em PHP que acessa diretamente essa base MySQL, usufruindo da facilidade de estar rodando na mesma máquina onde encontra-se a base de dados. O objetivo do DroidMinder é permitir aos usuários monitorar suas câmeras de vigilâncias através da Internet, estando essa aplicação desacoplada fisicamente da máquina onde está rodando o Zoneminder. Para o seu desenvolvimento duas abordagens poderiam ser exploradas:

1. Desenvolver um agente cliente que realize consultas diretas à base de dados MySQL;

3. Desenvolver um agente cliente que acesse a interface web do ZoneMinder e obtenha a partir desta os fluxos de vídeo das câmeras de vigilância.

Em termos de complexidade de desenvolvimento, a primeira abordagem seria a opção menos complexa, pois bastaria fazer uso de uma API para fazer consultas diretas à base MySQL. Porém se o ZoneMinder estiver rodando atrás de um firewall3 seria necessário liberar a porta para acesso ao MySQL, o que nem sempre é possível além de poder acarretar em problemas de segurança. Seria necessário ainda a criação de um usuário no MySQL que somente tivesse permissão de leitura em algumas poucas tabelas relacionadas às câmeras.

Na segunda abordagem a complexidade para o desenvolvimento seria um pouco maior se comparada com a primeira, pois a aplicação deveria interagir com uma página web do Zoneminder, sendo necessário enviar requisições HTTP e tratar as respectivas respostas para assim obter informações sobre os fluxos de vídeo das câmeras de vigilância.

Para o desenvolvimento do DroidMinder optou-se pela segunda abordagem pelo fato dessa ser a mais transparente para o usuário, pois bastaria este fornecer ao DroidMinder o endereço IP e porta onde está rodando o Zoneminder, para assim obter acesso às câmeras de vigilância cadastradas. Dessa forma não há necessidade de realizar modificações na base de dados MySQL ou mesmo criar novas regras no firewall4. A Figura 16 ilustra a interação entre o DroidMinder e o sistema de vigilância Zoneminder.(Michael,2010)

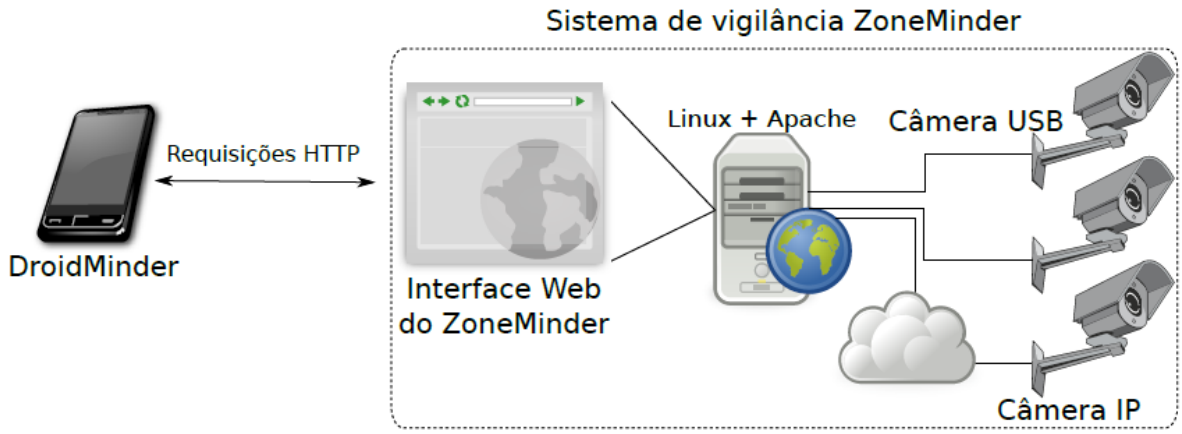

Figura 16 - DroidMinder interagindo com o Zoneminder

Fonte: Michael,2010

A Figura 17 apresenta as interfaces do DroidMinder. Na Figura 17b tem-se as informações necessárias para que o DroidMinder possa acessar o servidor do Zoneminder, como nome de usuário, senha, endereço IP e porta, caso o servidor web esteja rodando em uma porta diferente das padrões, 80 para HTTP e 443 para HTTPS (HTTP sobre o SSL/TLS).

Na primeira execução do DroidMinder, logo após fornecer as informações de conexão, o usuário é direcionado a interface da Figura 17c. Ali são listadas as câmeras presentes no Zoneminder, podendo o usuário alterar o nome de cada câmera e indicar quais dessas deseja monitorar. Cabe salientar que a listagem das câmeras é sempre obtida do Zoneminder à cada execução do Droidminder, visando assim garantir a corretude das informações ali apresentadas, uma vez que a lista de câmeras presentes no Zoneminder pode ser alterada a qualquer momento pelo administrador. Na Figura 17c são listadas 6 câmeras, mas somente 4 câmeras estão selecionadas para o monitoramento.

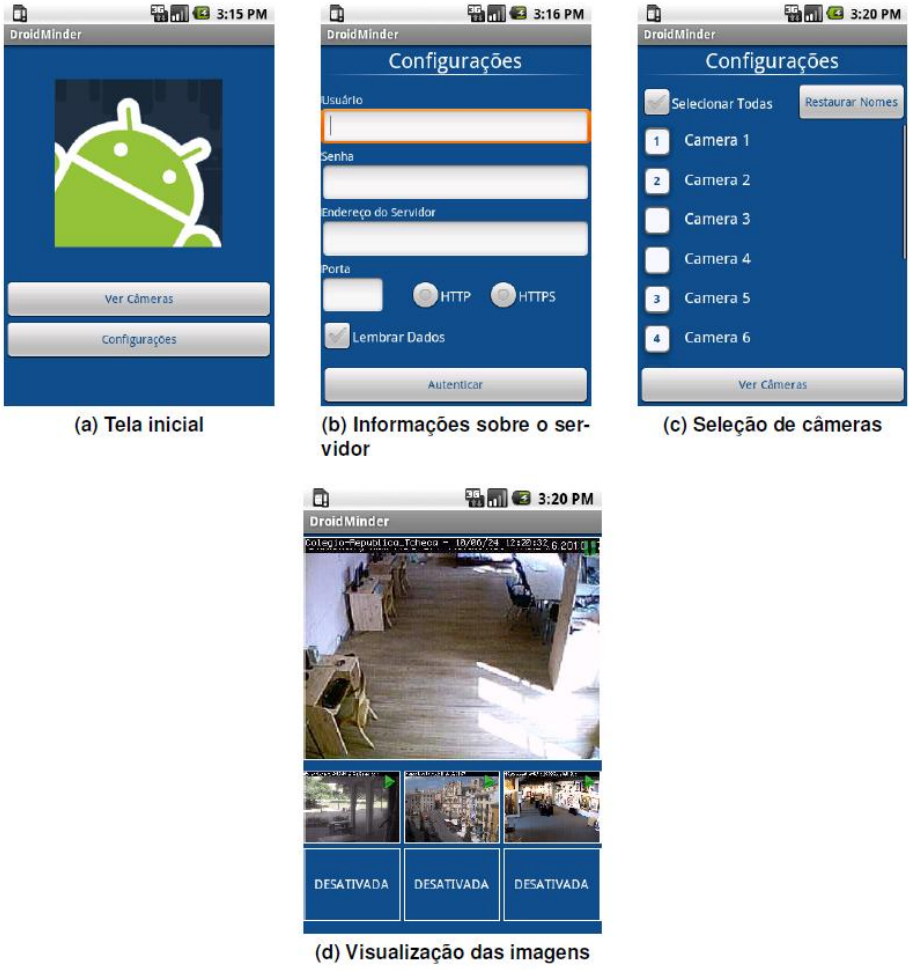

Figura 17 - Interfaces do DroidMinder

Fonte: Michael,2010

Após a configuração das câmeras pode-se então monitorá-las. A Figura 17d ilustra a interface de monitoramento com 4 câmeras ativas, sendo possível monitorar até 7 câmeras simultaneamente. Para a reprodução de cada fluxo de vídeo foi feito uso da classe *MjpegView* (KLÖS, 2009) e o usuário pode iniciar ou parar a reprodução de cada vídeo com um simples toque na tela. Nesta interface existe uma câmera em destaque, com uma área de exibição maior, e as câmeras secundárias. O usuário pode colocar em destaque qualquer câmera que desejar, bastando para isso dar um duplo toque sobre a janela secundária desejada e o vídeo desta trocará de lugar com o vídeo que está em destaque.(Michael,2010)

Apesar da interface permitir monitorar até 7 câmeras, o número de fluxos de vídeos que poderão ser reproduzidos simultaneamente será limitado pelo poder de processamento e memória de cada dispositivo Android. Nos testes realizados com o emulador do SDK Android 1.6 foi possível exibir os 7 fluxos simultâneos, o mesmo foi observado em um telefone Motorola A853 Milestone com o Android 3.0, porém com o aumento do número de fluxo simultâneos observou-se uma queda no desempenho da reprodução. .(Michael,2010)

5 - Resultados

Este capítulo trata-se dos testes feitos para definir a QoS para streaming de vídeo. Será descrito como foi realizado os testes. Iniciando os testes com a Câmera IP, em seguido dois aplicativos

5.1 Testes com Câmera IP FOSCAM modelo F18918W

Foi conectado primeiramente a câmera IP em um roteador, logo em seguida foi instalado através de um CD ROM o software da câmera. A Figura 18 á a câmera com o roteador, figura 19 mostra a conexão entre eles e a figura 20 o software da câmera instalado por um CD ROM.

Figura 18 - Roteador e Câmera IP (FOSCAM)

Figura 19 - Câmera IP (FOSCAM) conectada com o Roteador

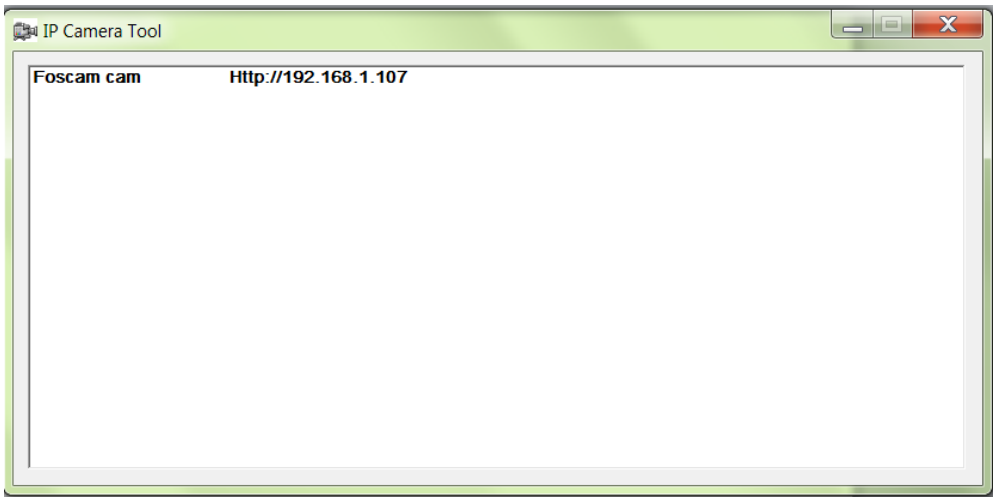

Figura 20 - Software Câmera IP (FOSCAM)

Para as configurações de rede, clica-se com o botão direito em cima do nome e ip que aparecem no software, no caso deste teste, foi configurado com o servidor DHCP. Como mostra a figura 21.

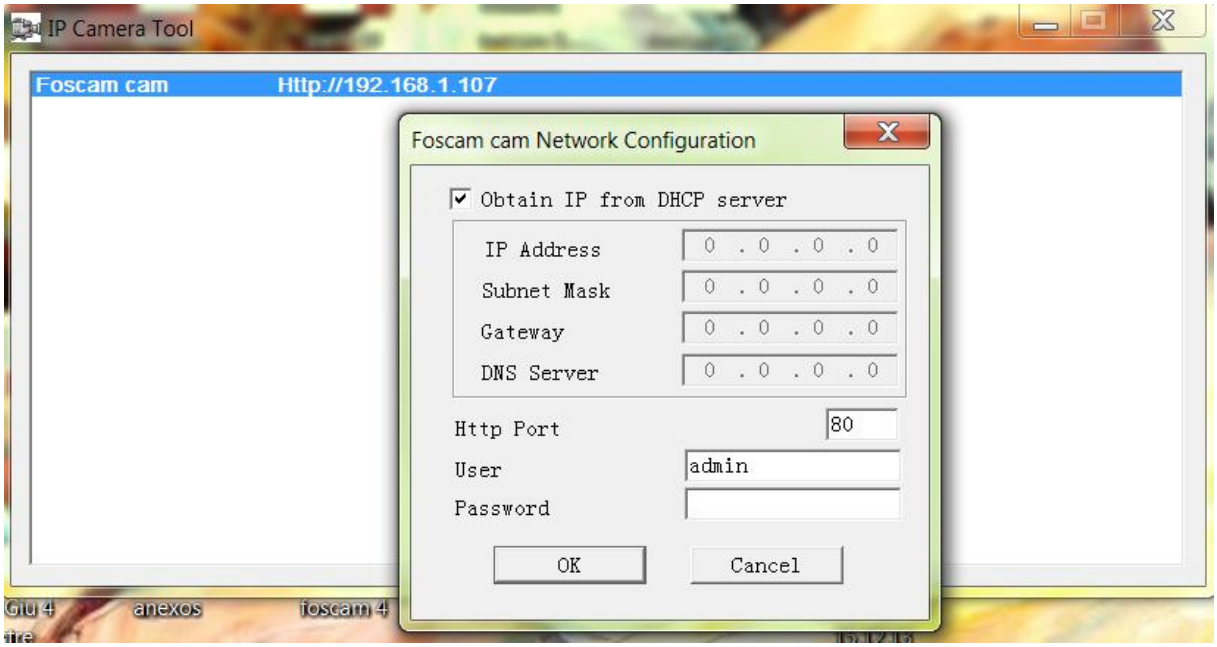

Figura 21 - Configuração de Rede

Dando dois clikes em cima do nome e IP da câmera, será aberto em uma pagina na web as imagens da câmera e o aplicativo onde serão realizadas suas configurações. Na figura 22 pode-se visualizar o painel onde é mostrada a imagem captada pela câmera e os menus de configuração.

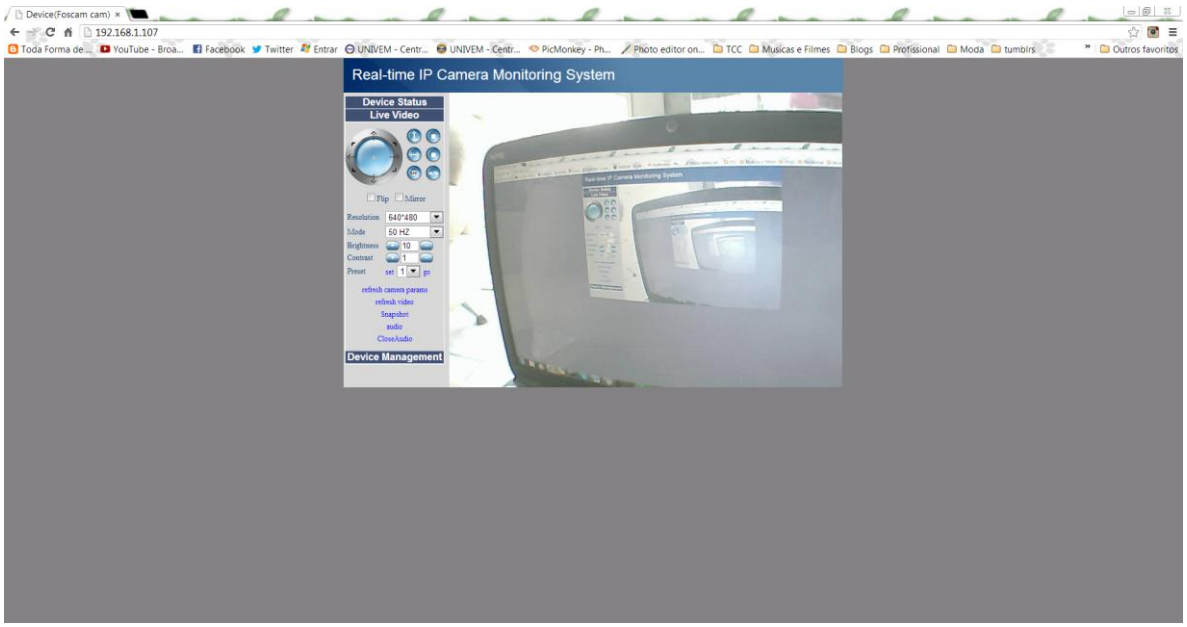

Figura 22 - Painel de visualização pelo navegador

No painel de visualização existem as opções para escolher resolução e frequência que o vídeo será enviado. Na figura 23 a resolução está sublinhado em vermelho e a frequência em verde.

Figura 23 - Configurações (resolução e Frequência)

5.2 Aplicativos para a transmissão de vídeo

Os aplicativo estudados foram:

- IP cam viewer Lite, baixado pelo google play, aplicativo Free.
- DroidMinder, aplicativo desenvolvido para pesquisa, código fonte aberto.

5.2.1 IP Cam Viewer Lite

Com este aplicativo (app), é possível remotamente visualizar e controlar câmera IP, DVR, Network Video Recorder, câmeras de trânsito, CCTV ou WebCam. Ele foi escolhido para o teste por ser gratuito e também é recomendado para as câmeras IP's da marca FOSCAM. A figura 24 mostra as primeiras configurações do app.

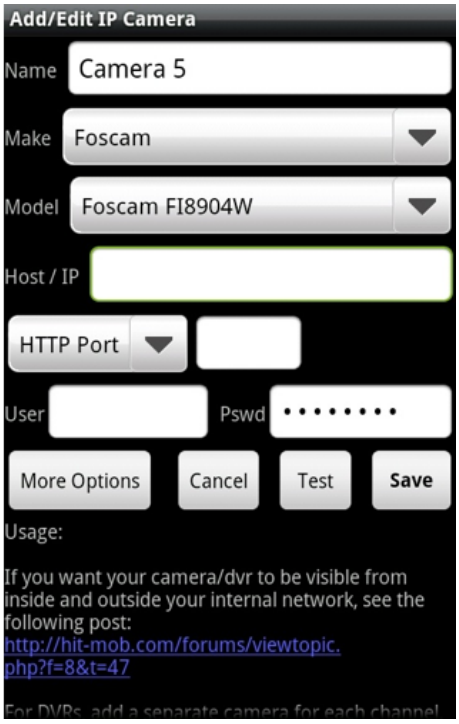

Figura 24 - Configurações app IP Cam Viewer Lite

Fonte : Google Play

Em seguida o app fará a conexão com a câmera IP e receberá as imagens capturadas por ela, como mostra a figura 25.

Figura 25 - Imagem da câmera IP pelo app IP Cam Viewer Lite

Fonte : Google Play

Conclusões

O levantamento de requisitos de qualidade de serviço feito neste trabalho de conclusão permitirá a parametrização dos servidores streaming e aplicações para equalizar a meio de transmissão a qualidade de imagem adquirida.

Para o desenvolvimento do trabalho foram feitos testes iniciais com a câmera IP FOSCAM, família FI8918W, que possui as seguintes características:

- Permite visualização e controle de movimentos através da internet ou celular;
- Pode ser conectada via wireless ou cabo de rede RJ45 em um roteador ou modem;
- Possui dispositivo de áudio embutido, que permite a comunicação entre quem controla e o ambiente monitorado;
- LED's infravermelhos automáticos para visão noturna até 8 metros;
- Sensor de movimento que envia um alerta para e-mail e upload de imagens para FTP;
- Permite cadastro de vários usuários (administrador/operador/visitante) com senhas para acesso.
- Ethernet: 10/100Mbps RJ-45
- Protocolos: HTTP, FTP, TCP/IP, UDP, SMTP, DHCP, PPPoE, DDNS, UPnP, GPRS

O aplicativo DroidMinder foi testado através da câmera Foscam, citada à cima, ela contém também um servidor streaming de onde foi possível acessar as imagens transmitida.

As APIs do Android não dão suporte a QoS na transmissão de vídeo, após o estudo notou-se a necessidade de investir na Qualidade de serviço.

Com a documentação contida neste trabalho de conclusão de curso, tem-se base para trabalhos futuros, como a implementações de testes, melhoria do aplicativoDroidMinder, criação de classes para QoS de streaming em Android.

Referências

Android Developers, http://developer.Android.com, Google, Android Developers, 2012, acessado em julho 2012.

Angelo R. V. ,**Análise de aspectos relativos à QoS de um dispositivo DVR.** Curso Superior de Tecnologia em Sistemas de Telecomunicações Instituto Federal de Santa Catarina, São José – SC 2009.

Brun A., Marta E. Gonçalves Vogt, Silveira A. M.,**QoS – Qualidade de Serviço em TCP/IP,** 2002.

Cbpf, UDP – ASPECTOS DE SEGURANÇA, Centro Brasileiro de Pesquisa Fisica,

Dias, A. F., **Concepção Conjunta Hardware/Software de Sistemas Embarcados de Processamento de Imagens**,CDTN/CNEN, Belo Horizonte, MG.2001.

Dierks, T.; ALLEN, C.**The TLS Protocol** – Version 1.0. [S.l.], jan 1999.

Eclipse**,** http://www.eclipse.org/, The Eclipse Foundation, Eclipse;

Elder E. S [1]., IVONEI F.[1], RAMIRO V. [2], **Plataforma Android**, [1]UNIOESTE -Cascavel, PR, [2] UNIVEL - Cascavel, PR, 2009.

Freier, A. O.; KARLTON, P.; KOCHER, P. C.**The SSL protocol** - v.3.[S.l.], Março 1996.

Google Inc., **Nosso Planeta Mobile: Brasil , Como entender o usuário de celular**, Maio 2013.Diponivel em <http://services.google.com/fh/files/blogs/our_mobile_planet_brazil_pt_BR.pdf>. acessado em fevereiro de 2012

IEEE Std 803.11-2007 (**Revision of IEEE Std** 803.11-1999), p. C1–1184, 12 2007.

IEEE.**Wireless lan medium access control (mac) and physical layer (phy) specifications**.

ISO.**Information technology – Coding of audio-visual objects** – Part 14: MP4 file format. [S.l.], 2003.

ITU. Recommendation T.802 :**Information technology** - JPEG 2000 image coding system: Motion JPEG 2000. [S.l.], jan 2005. Disponível em: <http://www.itu.int/rec/T-RECT.802/en>.

Kamienski C. A., Sadok D.**, Qualidade de Serviço na Internet,** Centro de Informática Universidade Federal de Pernambuco Belo Horizonte, 2000.

KLÖS, P. **Mjpegview – an Android view for mjpeg streams.**abr 2009. Disponível em: <http://www.anddev.org/multimedia-problems-f28/mjpeg-on-Android-anyone-1871.html>.

Lecheta , R. R, **Google Android Aprenda a criar aplicações para dispositivos móveis com o Android SDK**, Novatec, 2010.

Manuel, Pedro N. G. **Agente de Contexto para Dispositivos Móveis**, Dissertação realizada no âmbito do Mestrado Integrado em Engenharia Electrotécnica e de Computadores Major Telecomunicações, Julho de 2011.

Michael V. M. Euzébio, Emerson R. M. **DroidMinder – Monitoramento de câmeras de vigilância através de um telefone celular Android** , Instituto Federal de Santa Catarina (IFSC) - São Jose, SC – Brasil, 2010.

Oliveira, Alessandro F., **Linux Magazine** 48 ed., junho de 2009.

OPEN HANDSET ALLIANCE. **Android operating system**.2008. Disponível em: <http://www.Android.com>.

OPEN MOBILE ALLIANCE. **Wireless Application Protocol Architecture Specification**.[S.l.], abr 1998.

Oracle, http://www.java.com/, Oracle, Java.

Paris Bill, PAMIDALA Sreeni, HARITHAS Raghunandan K**, Usando o IBM Worklight para Desenvolver Aplicativos Híbridos de Execução de Vídeo HTML5 Multiplataforma**, IBM 2013. Disponível em <http://www.ibm.com/developerworks/br/websphere/techjournal/1208_paris/1208_paris.html#icomments>

Roesler V., **Transmissão multimídia em redes de computadores,** UNISINOS, 2009.

Sakuray F.**, Camadas do Modelo OSI e TCP-IP**, Redes de Computadores e Comunicação de Dados - 4ª Turma –2005

Santana H.**,A camada de transporte tem como uma das principais funções a ampliação da qualidade de serviço (Quality of Service – QoS),** Universidade Santa Cecília – Unisanta, 2005.

Tananbaum, Andrew S**.; Redes de Computadores**. Rio de Janeiro: Editora Campus, Tradução da terceira Edição, 2003.

Tschoke, Clodoaldo. **Criacao de Streaming de Vídeo para Transmissao de Sinais de Video em Tempo Real pela Internet**. 2001. Universidade Regional de Blumenau, Blumenau, 2001.

Wikipedia, **TCP/IP**, disponível em http://pt.wikipedia.org/wiki/TCP/IP, acessado em maio 2012.

Zoneminder, **Linux video camera and cctv security with motion detection.**2004. Disponível em: <http://www.zoneminder.com>.