Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Curso de Pós-Graduação em Engenharia de Eletricidade

Um Módulo de Monitoramento para uma Arquitetura de Transmissão de Mídia Contínua

Sadick Jorge Nahuz

São Luís
2008

Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Curso de Pós-Graduação em Engenharia de Eletricidade

Um Módulo de Monitoramento para uma Arquitetura de Transmissão de Mídia Contínua

Sadick Jorge Nahuz

Dissertação apresentada ao Curso de Pós-Graduação em Engenharia de Eletricidade da UFMA como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica.

São Luís
2008

Nahuz, Sadick Jorge.
Um módulo de monitoramento para uma arquitetura de transmissão de mídia contínua / Sadick Jorge Nahuz. – 2008.
72f.

Orientador: Zair Abdelouahab.
Impresso por computador (fotocópia).
Dissertação (Mestrado) – Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade , São Luís, 2008.

1.Servidor de video - monitoramento. 2. Streaming. 3. Transmissão de video via web. I. Abdelouahab, Zair, orient. II. Título.

CDU 004.4'242

UM MÓDULO DE MONITORAMENTO PARA UMA ARQUITETURA DE TRANSMISSÃO DE MÍDIA CONTINUA

Sadick Jorge Nahuz

Dissertação aprovada em14 de fevereiro de 2008.

Prof. Zair Abdelouahab, Ph.D.
(Orientador)

Prof. Mario Antonio Meireles Teixeira, Dr.
(Co-orientador)

Prof. Carlos Renato Lisboa Francês, Dr.
(Membro da Banca Examinadora)

Prof. Francisco José da Silva e Silva, Dr.
(Membro da Banca Examinadora)

“Filho meu, se aceitares as minhas palavras e esconderes contigo os meus mandamentos, para fazeres atento à sabedoria o teu ouvido e inclinares o teu coração ao entendimento, e, se clamares por inteligência e por entendimento alçares a voz, se buscares a sabedoria como a prata e como a tesouros escondidos a procurares, então entenderás o temor do Senhor e acharás o conhecimento de Deus. Porque o Senhor dá a sabedoria e da sua boca vem a inteligência e o entendimento.”
“Porquanto a sabedoria entrará no teu coração e o conhecimento será agradável a tua alma.”

Prov. 2:1-6, 10

Dedicatória

Aos meus pais, Sadick e Lélia Nahuz;

À minha noiva, Nicolle Matos;

Aos meus irmãos, Charles e Christianne Nahuz;

À minha avó, Mary Nahuz.

Agradecimentos

A DEUS, pela sua força, graça e misericórdia.

Aos meus pais, Sadick e Lélia Nahuz, pelo amor, confiança e incentivo.

À minha noiva, Nicolle Matos, pelo amor, força e companheirismo.

À minha avó, Mary, pelo carinho.

Aos meus irmãos, Charles e Christianne Nahuz, pela amizade.

Aos meus orientadores, Dr. Mário Meireles e Phd. Zair Abdelouahab, pela orientação precisa e pela dedicação, auxílio e amizade prestados no decorrer destes dois anos de trabalho.

A todos os componentes do Laboratório LabMint, que fizeram esses dois anos serem bem agradáveis.

A todos que contribuíram direta e indiretamente para conclusão deste trabalho.

Resumo

A Internet tem experimentado, nos últimos anos, um considerável aumento no uso de aplicações de áudio e vídeo, as quais promovem um acentuado consumo dos recursos disponíveis na rede e nos servidores. Desta forma, torna-se essencial o monitoramento e análise da utilização desses recursos, a fim de melhorar os serviços prestados aos usuários. Este trabalho descreve um módulo de monitoramento implementado em um servidor de vídeo, o qual é utilizado para acompanhar a transmissão de diferentes formatos populares de vídeo via streaming. Observou-se, por meio de experimentos, que um dos formatos apresenta um desempenho consideravelmente melhor que o outro, no que se refere à largura de banda alocada a cada sessão de usuário, o que corrobora a importância de módulos de monitoramento, tal como o desenvolvido nesta dissertação.

Palavras Chaves: Vídeo; Streaming

Abstract

The Internet has experienced a considerable increase in the use of audio and video applications, which provoke a large consumption of the resources available in the network and servers. Therefore, the monitoring and analysis of those resources becomes an essential task in order to enhance the service delivered to users. This work depicts a monitoring module implemented in a video server architecture, which is used to track the transmission of some popular video formats. Our experiments have demonstrated that one of the formats delivers a performance considerably better than the other, regarding the bandwidth allocated to each user session, what only reassures the importance of having such a monitoring module available in a server architecture.

Keys Word: Video; Streaming.

Lista de Tabelas

Lista de Figuras

Lista de Siglas

ACC	*Advaced Áudio Coding*
API	*Application Programming Interface*
ALF	*Application Layer Framing*
CSR	*Contributing Source*
DMIF	*Delivery Multimedia Framework*
DNS	*Domain Service Name*
DVD	*Digital Video Disc*
ES	*Elementary Stream*
FTP	*File Transfer Protocol*
GSM	*Golbal System Mobile*
HD	*Hard Disk*
HTTP	*HyperText Transfer Protocol*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
ISO	*International Organization*
JPEG	*Joint Photographic Experts Group*
MOV	*Movie*
MP4	*Movie Picture Experts Group*
MPEG	*Moving Picture Experts Group*
MTU	*Maximum Transmission Unit*
OD	*Object Descriptor*
PT	*Payload Type*
QoS	*Quality of Service*
RR	*Receiver Report*
RTCP	*Real Time Control Protocol*
RTP	*Real Time Protocol*
RTSP	*Real Time Streaming Protocol*
SDES	*Source Description*
SR	*Sender Report*
SSRC	*Synchronization Source*
TCP	*Transport Control Protocol*
TTS	*Text-to-Speech*
UDP	*User Datagram Protocol*
URL	*Uniform Resource Locator*

Sumário

Capítulo 1

Introdução

1.1 Contextualização

A Internet interconecta redes diferentes que não se comunicavam entre si e cumpre sua tarefa com sucesso. Redes acadêmicas, particulares e públicas podem ser facilmente integradas a ela [26].

A rede Internet é baseada no conjunto de protocolos definidos pela Arquitetura TCP/IP. Apesar de o IP ser o protocolo da camada de rede que provê o encaminhamento de pacotes IP e o TCP ser o protocolo da camada de transporte que oferece comunicação confiável, o termo TCP/IP [9] é geralmente usado para denotar o conjunto de protocolos que são utilizados em conjunto com o protocolo IP, tais como UDP, DNS, FTP, Telnet, entre outros [1].

Originalmente concebida para transmissão confiável de dados, com mínima ou nenhuma sensibilidade ao atraso, a Internet oferece um modelo simples de serviço do tipo "*best-effort*", ou serviço de melhor esforço, onde os dados são transmitidos tão rapidamente quanto possível e a preocupação principal reside em entregar os dados corretamente para a outra entidade participante da comunicação. Os protocolos TCP/IP foram projetados para este tipo de tráfego e trabalham muito bem neste contexto.

Nos últimos anos, houve um grande desenvolvimento e ampla disseminação das aplicações em rede que transmitem e recebem conteúdo de áudio e vídeo pela Internet. Essas aplicações têm requisitos de serviço distintos das aplicações tradicionais orientadas a dados.

Observa-se hoje grande demanda por uma classe de aplicações conhecida como Sistemas Multimídia na Internet. Esta classe de sistemas se distingue das outras pela necessidade de transmissão de dados multimídia, tais como áudio e vídeo, gerando grande fluxo de dados e promovendo um acentuado aumento no consumo de recursos da Internet [25].

Os sistemas multimídia provêem serviços voltados à execução de aplicações que necessitam de garantias de recursos para que seu desempenho não seja prejudicado, como aplicações interativas (conferência de áudio e vídeo, telefonia, ensino a distância, mídia contínua) e sob demanda (vídeo e áudio sob demanda), desencadeando uma demanda de tráfego bem mais exigente do que, por exemplo, uma aplicação de FTP para uma simples transferência de arquivo. Normalmente, aplicações multimídia requerem sincronização para ter uma perfeita inteligibilidade ao usuário final, e podem, geralmente, prescindir da complexidade do TCP para o tráfego dos seus dados utilizando assim uma estrutura de transporte bem mais simples.

Os sistemas multimídia são muito sensíveis ao atraso fim-a-fim e à variação do atraso, mas podem tolerar perdas de dados ocasionais [1]. Além disso, a maioria dos algoritmos de reprodução, que decodificam as mídias para posterior apresentação, podem tolerar perda de dados muito melhor do que atrasos prolongados causados por

retransmissões, não requerendo também transferência com seqüênciamento garantido. Algumas redes atuais não podem se adaptar aos serviços multimídias porque não foram projetadas para suportar esses serviços.

A atual arquitetura Internet não está preparada para o tráfego em tempo real de aplicações multimídia. Isto se justifica por causa de significativas perda de pacotes que, normalmente, são causadas por *buffers* cheios na rota entre fonte e destino. Grandes atrasos na entrega de cada pacote ao receptor, são suas variações, resultado de diferentes caminhos que pacotes de uma aplicação podem seguir, ora chegando adiantados, ora atrasados.

Além desses fatores, a falta de largura de banda necessária às aplicações multimídia também pode ser considerada. O problema principal deve-se ao fato de que todos os pacotes IP, independente de qual aplicativo os gerou, são processados da mesma forma. Uma arquitetura de rede baseada nos protocolos TCP/IP impõe limitações ao tráfego em tempo real principalmente quando a rede está congestionada. A camada de rede (IP) pode, em situação de congestionamento, descartar pacotes sem conhecer qual aplicação os gerou.

Atualmente, vários sites possuem um formato de vídeo para seus assinantes. Nos próximos anos, os serviços multimídia dominarão grande parte do fluxo da Internet, principalmente nas transmissões ao vivo, mas ainda existe muito a ser pesquisado e definido nessa área [2].

O acesso à Internet de alta velocidade aumentou progressivamente nas residências e os serviços multimídia entraram na rotina dos usuários [9]. Todos poderão acessar seus programas favoritos, independentemente dos horários em que serão transmitidos, pois sempre haverá um arquivo guardado em um servidor streaming, utilizando um protocolo de transmissão em tempo real [29].

1.2 Motivação

A expansão da banda larga e a redução do custo das conexões são fortes estímulos para a adoção do *streaming* de vídeo. Para uma empresa, é uma alternativa para se comunicar com várias pessoas ao mesmo tempo [30].

É importante notar, contudo, que todos estes desenvolvimentos foram muito além do projeto inicial da Internet e só foram possíveis devido à genialidade dos seus criadores, que fizeram uma estrutura capaz de receber tais inovações. Apesar de todas as inovações, a forma de se transmitir mídia (especificamente som e vídeo) na *Web* era até pouco tempo, por meio de *download* do arquivo. Somente quando este terminava de ser copiado na máquina do usuário é que poderia ser visto ou ouvido.

Ao optar por ouvir via *streaming*, o usuário aguarda alguns segundos e, após esta pequena espera, o som e o vídeo têm início. Estes primeiros segundos são usados para criação de um buffer na memória, uma espécie de depósito de dados que será utilizado pelo o *player*, caso a qualidade da conexão caia durante o *streaming*.

Para conseguir manter a taxa de *download* do *streaming* na mesma proporção que transmite o vídeo, os codificadores de *streaming* (ou *encoders*) precisam fazer compactações intensas no arquivo fonte, ocasionando, por vezes, uma perda de qualidade perceptível.

Uma peculiaridade do *streaming* é que o processo não visa fazer uma cópia da informação retirada da Internet no computador do usuário para depois tornar possível ver ou ouvir o arquivo. Os dados são tocados diretamente no player e não existe nenhum armazenamento em disco. Este fato promove maior segurança, diminuindo a possibilidade de cópias da mídia.

A qualidade da imagem do *streaming* melhorou após o surgimento de outros formatos de vídeo, como o MP4 e o MOV. Com estes formatos, é possível hoje assistir ao vídeo com uma qualidade similar ao DVD.

Uma vantagem adicional do *streaming* é que o criador do arquivo pode estabelecer uma banda máxima para o *stream*, diferentemente dos arquivos que sofrem *download* e usam toda a largura de banda disponível. Isto permite que usuários possam ouvir música de uma rádio na Internet e ainda assim naveguem pelas páginas da *Web*.

1.3 Objetivos

Este trabalho tem como objetivo implementar um módulo de monitoramento em uma arquitetura de servidor de mídia contínua (*streaming server*), o qual será utilizado para realizar um acompanhamento do desempenho desta arquitetura fornecendo dois formatos de arquivos de vídeo, o MP4 e o MOV.

Os objetivos específicos deste trabalho são os seguintes:

a) estudar a arquitetura do servidor de mídia contínua, o *Darwin Streaming Server*;
b) validar o módulo de monitoramento implementado no servidor;

c) realizar uma avaliação de desempenho do servidor, por meio do módulo desenvolvido;
d) avaliar o comportamento de diferentes formatos de arquivos de vídeo entregues por *streaming*;
e) desenvolver uma ferramenta de visualização do desempenho do servidor Darwin.

O servidor de *streaming* escolhido para o desenvolvimento deste trabalho é o servidor de código aberto *Darwin Streaming Server*, que, em seu estágio atual de desenvolvimento, não possui um módulo de monitoramento implementado em sua arquitetura.

1.4 Organização do Trabalho

O Capítulo 2 desta tese discute os protocolos utilizados atualmente para prover serviços de mídia contínua (*streaming*) na Internet, quais sejam: RTP, RTSP e RTCP. Também são abordados neste capítulo dois formatos de vídeos populares, comumente oferecidos via *streaming*, o MPEG-4 e MOV.

O Capítulo 3 aborda o *Darwin Streaming Server*, da Apple, que é um servidor de vídeo *open source* capaz de fornecer arquivos de mídia nos formatos QuickTime (MOV) e MPEG-4 (MP4), dentre outros, a clientes pela Internet, usando os protocolos padrão RTP e RTSP. O Darwin é baseado no código do antigo *Quicktime Streaming Server* e suporta transmissão ao vivo.

O Capítulo 4 detalha o *Módulo Monitor* implementado neste trabalho, o qual interage diretamente com o *kernel* do servidor Darwin, sendo utilizado para realizar uma avaliação de desempenho do *streaming* de arquivos nos formatos MOV e MP4.

O Capítulo 5 apresenta e discute os experimentos realizados e os resultados alcançados neste trabalho, a partir do uso do Módulo Monitor na arquitetura de servidor *streaming* supracitada.

Finalmente, o Capítulo 6 resume as conclusões e principais contribuições desta dissertação, pavimentando o caminho para trabalhos futuros.

Capítulo 2

Protocolos e Formatos de Mídia Contínua

2.1 Introdução

A recente tecnologia de mídia contínua foi projetada para tornar mais leve e rápido o *download* e a execução de áudio e vídeo na *web*. Esta tecnologia permite escutar e visualizar os arquivos enquanto ocorre o *download*. Se não for utilizada mídia contínua para mostrar um conteúdo multimídia na rede, tem-se que descarregar primeiramente o arquivo inteiro no disco rígido do computador para posteriormente executá-lo, e finalmente ver e ouvir o vídeo desejado [35].

A mídia contínua funciona da seguinte maneira: Inicialmente, o computador (cliente) conecta-se com um servidor de vídeo, depois de conectado o servidor começa a transmitir o arquivo. O cliente começa a receber o arquivo e constrói um buffer onde são salvos as informações dos vídeos. Quando o buffer recebe uma pequena parte do arquivo, o cliente passa a assistir ao vídeo e paralelamente permanece realizando o *download* do arquivo. O sistema é sincronizado para que o arquivo possa ser visto enquanto é baixado, de modo que quando o arquivo acaba de ser baixado, também acaba de ser visualizado. Se em algum momento a conexão sofre decréscimos de velocidade, é utilizada a informação que existe no buffer, assim tem-se um tempo para sincronizar as informações, não ocorrendo paradas na transmissão do vídeo. Se a comunicação é interrompida durante muito tempo, o buffer é esvaziado e a execução do arquivo pára até a restauração da transmissão dos formatadores (*players*).

Hoje, vários formatadores (*players*) permitem que recebam transmissões de mídia pela Internet. As transmissões podem ser originadas de fontes ao vivo, como câmeras de vídeo, *webcast*, ou fontes de áudio, como uma estação de rádio, ou a fonte pode ser um arquivo armazenado no servidor. Em todos os casos, não se está baixando um arquivo quando vê um filme transmitido. Os dados estão simplesmente sendo exibidos quando chegam pelo *plug-in* de um *player*. Os novos *players* avançaram em tecnologia de transmissão proporcionando acesso instantâneo à mídia sem a necessidade de esperar por downloads demorados ou o tempo do buffer. Com uma solução padronizada de ponta a ponta, a transmissão de um vídeo na Internet tornou-se muito acessível.

As transmissões de arquivos de mídia contínua envolvem aplicações de tempo real. Portanto, é de se esperar que demandem protocolos e formatos especificamente projetados para atender suas características. Nesta seção, são analisados os protocolos da Internet utilizados para a transmissão de mídia contínua: o RTP, RTSP e RTCP. E os formatos MP4 e MOV.

2.2 RTP

O protocolo de transporte em tempo real, conhecido como RTP, foi projetado para oferecer suporte ao tráfego de aplicações que transmitem dados em tempo real. É

normalmente integrado dentro da aplicação (modo usuário), não sendo implementado como parte do núcleo do sistema operacional. Definido na RFC 1889 [3], o protocolo RTP é um produto do Grupo de Trabalho de Transporte de Áudio e Vídeo (*Audio/Video Transport Working Group*) e foi apresentado formalmente em janeiro de 1996 pelo Grupo de trabalho de Redes (*Networking Working Group*) do IETF (*Internet Engineering Task Force*) com o objetivo de fornecer uma padronização de funcionalidades para os aplicativos de transmissão de dados em tempo real [22].

O RTP oferece funções de transporte fim-a-fim para aplicações que transportam dados em tempo real, como áudio e vídeo, sobre redes de serviço *unicast* ou *multicast*, caracterizando-se como um protocolo não orientado a conexão. Essas funções incluem identificação do tipo de dado a ser manipulado, numeração de seqüência, *timestamps* e monitoramento da transmissão de dados. Embora o RTP ofereça transferência fim-a-fim, ele não oferece todas as funcionalidades normalmente encontradas em um protocolo de transporte. Além disso, não reserva recursos da rede, não garante Qualidade de Serviço (QoS) [10] [33] para as aplicações, nem promove reordenamento ou retransmissão no caso de perda de pacotes, delegando essas responsabilidades às aplicações [11].

Normalmente, aplicações multimídia em tempo real não necessitam dos serviços de tratamento de falhas do protocolo TCP, por serem capazes de tolerar perda de dados. Assim, em vez de introduzir atrasos desnecessários com retransmissões, essas aplicações podem utilizar-se de outros protocolos para *streaming* de mídia. Um protocolo normalmente utilizado é o *User Datagram Protocol* (UDP), um protocolo "não confiável", pois não fornece garantias de que cada pacote possa alcançar o seu destino, nem que os mesmos cheguem na ordem em que foram enviados. O destinatário, portanto, precisa estar habilitado a compensar a perda de dados, pacotes duplicados e pacotes que cheguem fora de ordem. Aplicações multimídia, em qual, podem tolerar alguma perda de dados, desde que seja preservada a usabilidade da aplicação [34].

O padrão Internet para transporte de dados de mídia contínua é o RTP [39]. Tipicamente, um pacote UDP pode encapsular um único pacote RTP ou vários pacotes RTP, e um pacote RTP pode conter, por exemplo, um único frame de vídeo ou múltiplas amostras de áudio. Outros protocolos de transporte ou de rede também podem ser utilizados em conjunto com RTP e este utiliza, normalmente, o protocolo UDP para fazer uso dos serviços de multiplexação e verificação de erros. O protocolo RTP geralmente atua em conjunto com o protocolo de controle RTCP [41], que transmite dados estatísticos sobre a sessão, a serem potencialmente utilizados pela aplicação.

Ao contrário do HTTP e FTP, o RTP não baixa um filme ou música inteira para o computador do cliente. Em vez disso, envia uma transmissão de dados fina e unilateral a uma taxa constante que exibe a transmissão em tempo real (após alguns momentos iniciais de oscilação no armazenamento de dados). Por exemplo, a transmissão de um vídeo de um minuto, por exemplo, dura exatamente um minuto. Se a conexão tem largura de banda suficiente para suportar a transmissão de dados, o filme é transmitido sem problemas. Depois da exibição dos dados, eles são descartados. Os espectadores podem rever a transmissão, bastando solicitar para o servidor de transmissão.

Entretanto, apesar de utilizar o UDP e o IP, o RTP pode ser implementado em outros ambientes, já que necessita apenas de serviços de transporte não orientados à conexão. O protocolo de tempo real foi projetado para transmissão de dados em tempo real, possui as funções seqüênciamento e temporização. O seqüênciamento é feito no destino, através da numeração de cada pacote. Basicamente, o protocolo permite a especificação dos requisitos de tempo e conteúdo pertinentes à transmissão de multimídia, tanto no envio quanto na recepção através de:

a) Numeração seqüenciada;

b) Selo de temporização (*timestamp*);

c) Envio de pacotes sem retransmissão;

d) Identificação de origem;

e) Identificação de conteúdo;

f) Sincronismo.

2.2.1 Cabeçalho do Protocolo RTP

O cabeçalho do protocolo RTP tem o formato mostrado abaixo [40]. Qualquer pacote RTP possui pelo menos os doze primeiros octetos. As listagens dos CSRC (*Contributing Soucer*) somente estão presentes quanto inseridos por um *mixer*.

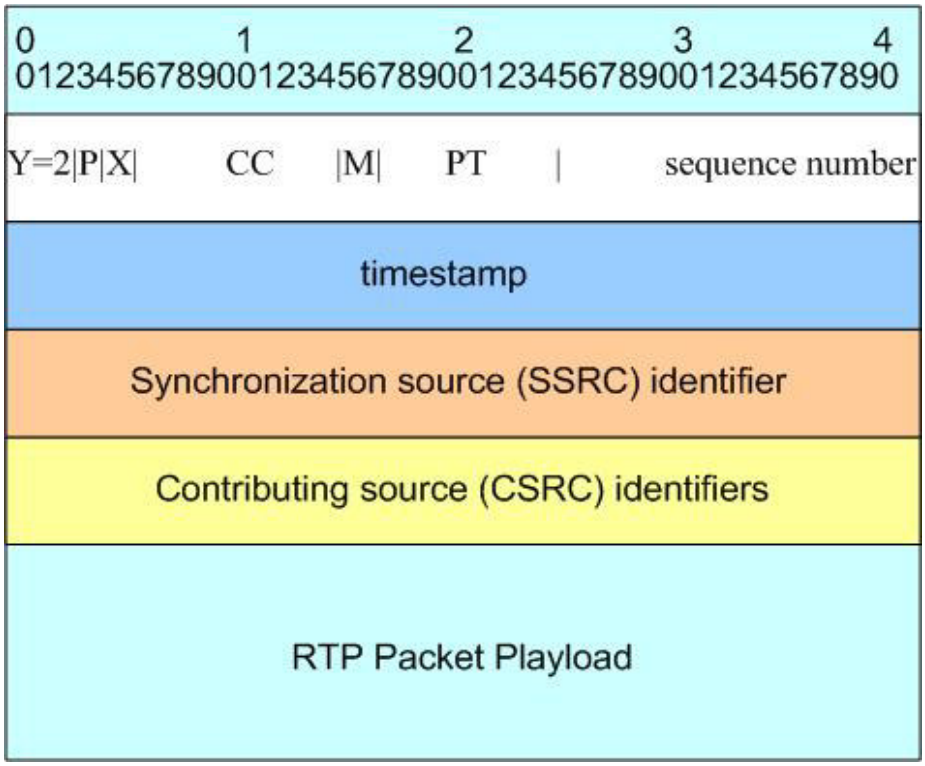

Figura 01. Cabeçalho RTP.

Segue abaixo a descrição dos campos do cabeçalho:

a) **Y – versão (2 bits):** Usado para especificar a versão do RTP.

Os primeiros dois bits indicam a versão do protocolo utilizada. Quando compatível com a RFC3550, seu valor é dois, por exemplo.

0 : especifica o primeiro protocolo utilizado na ferramenta de áudio.
1: especifica a primeira versão do RTP utilizada como teste.
2: identifica a versão do RTP especificada na RFC 1889.

b) **P – Preenchimento (*padding*) (1 bit):** Sinaliza a adição de octetos de preenchimento adicionais ao conteúdo da carga (*payload*), sem fazer parte da mesma. O último octeto do preenchimento contém a informação de quantos octetos foram inseridos. Este preenchimento adicional é normalmente utilizado para uso de algoritmos de criptografia, de tamanho de blocos fixos ou para transmissão de pequenos conteúdos.

c) **X – Extensão (*extension*) (1 bit):** Com esse bit marcado, é acrescentada uma extensão ao cabeçalho original.

d) **CC - Contador CSRC/CSRC (*count*) (4 bits):** Este campo contém o número de identificadores CSRC.

e) **M – marcador (*marker*) (1 bit):** Usado para identificar as fronteiras de um quadro em uma corrente de pacotes.

f) **PT – Tipo de carga (*payload type)* (7 bits):** Este campo identifica o formato da carga do pacote RTP, como também a sua interpretação pela aplicação.

g) **Numero de seqüência (*sequence number*), (16 bits):** O número de seqüência ajuda a ordenar os diversos pacotes RTP. A cada novo pacote, a numeração é incrementada de uma unidade. Esse ordenamento permite que o receptor detecte os pacotes perdidos, restaure na seqüência e identifique-os.

h) **Selo de temporização (*timestamp*), (16 bits):** Este campo reflete o instante de amostragem do primeiro octeto no pacote RTP. São usados trinta e dois bits para informar o instante em que foi amostrado o primeiro byte dos dados da mídia.

i) **Fonte de Sincronização (SSRC), (32 bits):** Este campo identifica a fonte de sincronização. Ele sincroniza as fontes que disponibilizam a mídia. Esta identificação é escolhida aleatoriamente, tencionando-se que duas fontes de sincronizações dentro da mesma sessão RTP não tenham o mesmo identificador SSRC. São usados trinta e dois bits para a fonte de seqüência de pacotes RTP. Esta identificação é importante para que a identificação da fonte fique independente dos protocolos inferiores da rede.

j) **Fonte de Contribuição (CSRC), (itens de 0 a 15, 32 bits cada)**. Este campo identifica a localização da fonte. A lista CSRC identifica a fonte contribuinte do conteúdo da carga (*payload*) de cada pacote. O número de identificadores é dado pelo campo CC. Se houver mais de 15 fontes contribuintes, somente 15 serão identificadas.

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

2.2.2 A Parte de Dados RTP

As aplicações RTP utilizam-se de uma estrutura conhecida como *Application Layer Framing* (ALF) [28]. Esta estrutura (Figura 02) possui regras básicas, operações e formato de mensagens. Existem vários padrões diferentes para codificar dados de vídeo, incluindo MPEG, JPEG e H.261, assim como dados de áudio, GSM, G723, entre outros. RTP oferece uma estrutura apropriada para qualquer um destes métodos de codificação. Um protocolo completo requer ambos: uma estrutura RTP e o formato *payload* de, por exemplo, MPEG. Além disso, faz parte da estrutura RTP uma informação de marcação de tempo (*timestamping*) que permite ao recebedor de um vídeo sincronizar voz e movimento de figuras.

Cada fonte de dado RTP marca cada mensagem com um *timestamp* indicando quando um evento acontece, dessa forma. O recebedor representa o dado recebido no mesmo tempo relativo. Isto garante que o áudio seja reconstituído no tempo original, após ser recebido pelo computador. Redes baseadas em TCP/IP não mantêm esta relação do tempo entre as mensagens, podendo chegar ao destino em intervalos irregulares, fora de ordem, ou ainda diferente da mensagem original [38].

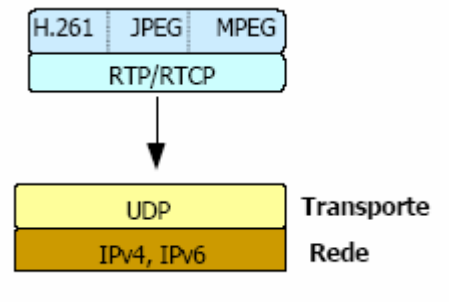

Figura 02. RTP combinado com um formato de *payload* para formar um protocolo completo.

Sistemas que utilizam RTP podem também agir como tradutores e *mixers*. Ambos localizam-se "no meio" da rede, entre transmissores e receptores, tendo como função processar os pacotes RTP que passam diretamente por eles. Não são necessários, mas são utilizados para que a rede possa suportar uma aplicação em tempo real [30]. Os Tradutores transformam um formato *payload* para outro, sendo que o novo formato pode oferecer vídeo com mais baixa qualidade, mas com largura de banda menor. Na Figura 03, cada *translator* gera 1Mbps de tráfego de vídeo. O tradutor aceita cada *stream* e converte para 256 Kbps cada. *Mixers* realizam um serviço semelhante ao dos tradutores, mas, ao invés de transformarem os fluxos da fonte individuais para um formato diferente, eles combinam múltiplos fluxos em um único, preservando o formato original. Essa abordagem pode ser particularmente utilizada em dados de áudio.

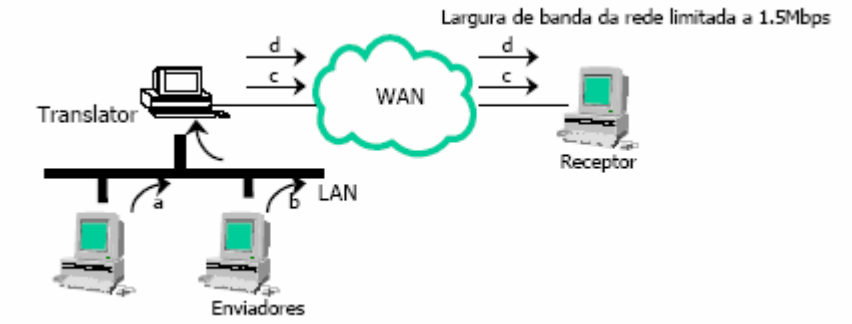

Figura 03. Um tradutor processando pacotes RTP em cenário de videoconferência.

Nem todas as aplicações podem suportar a operação de um *translator* (Figura 3). Fontes de vídeo múltiplas, por exemplo, não podem ser combinadas em uma. A abordagem trabalha muito bem em conferências de áudio, particularmente conferências com muitos participantes. A Figura 04 demonstra o cenário de uma audiconferência, onde cada enviador gera 64 Kbps de tráfego de áudio. O *mixer* aceita cada *stream* e combina todos em um único *stream* de 64Kbps.

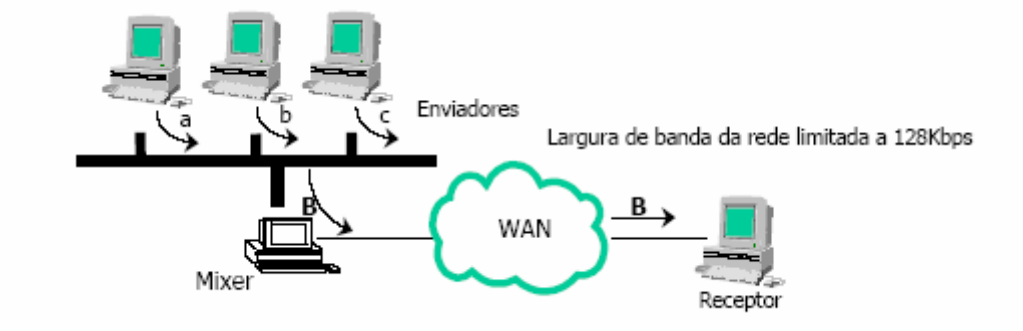

Figura 04. Operação básica de um *mixer*.

Atualmente, várias aplicações RTP, tanto experimentais quanto comerciais, têm sido implementadas. Essas aplicações incluem ferramentas de áudio e vídeo, juntamente com ferramentas de diagnóstico e monitores de tráfego. Muitos usuários já utilizam esta ferramentas [36].

Contudo, a atual arquitetura da Internet não pode ainda suportar a demanda potencial de serviços em tempo real. Serviços com alta largura de banda utilizando RTP, como vídeo por exemplo, podem potencialmente degradar seriamente a Qualidade de Serviço de outros serviços na rede. Dessa forma, implementadores de aplicações devem se precaver limitando a largura de banda utilizada em suas aplicações.

2.2.3 Operação do RTP

Na operação RTP, o usuário acessa cada apresentação através de uma URL RTSP que está armazenada em algum servidor na Internet ou rede local, por exemplo, rtsp://media.exemplo.com/vídeo, como mostra a figura 05. Uma apresentação pode conter vários *streams*.

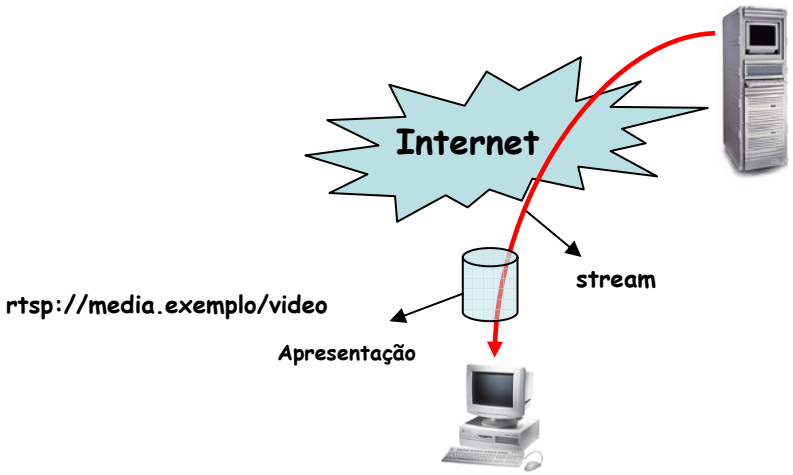

Figura 05. Operação RTP.

Um arquivo de descrição de mídia contém a descrição dos streams que compõem a apresentação, linguagem de codificação entre outros. Na requisição RTP o áudio e o vídeo trafegam em fluxos diferentes (Figura 06).

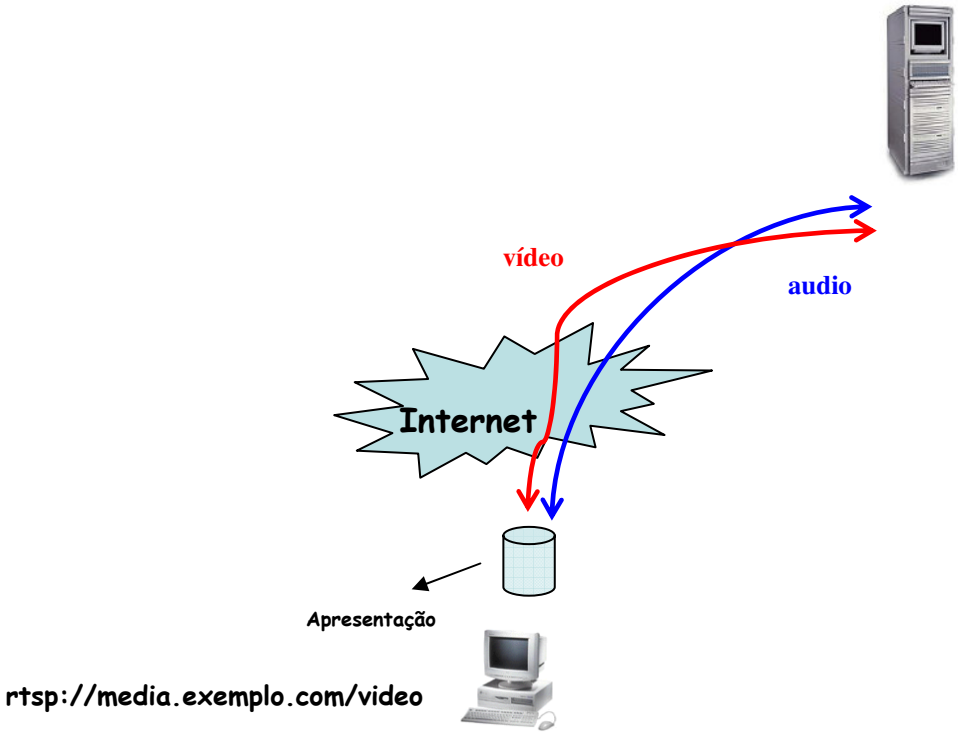

Figura 06. Operação RTP.

Existe um cenário em que o servidor de vídeo é separado do servidor de áudio. Para isso, deve existir uma definição de endereço, destino e porta (Figura 07). Precisa ser especificado se é *Unicast* ou *Multicast*, ou existir um convite do próprio servidor que gera um *broadcast* na rede para adição de mídia no usuário [22]. A sincronização do áudio e do vídeo é realizada pelo player.

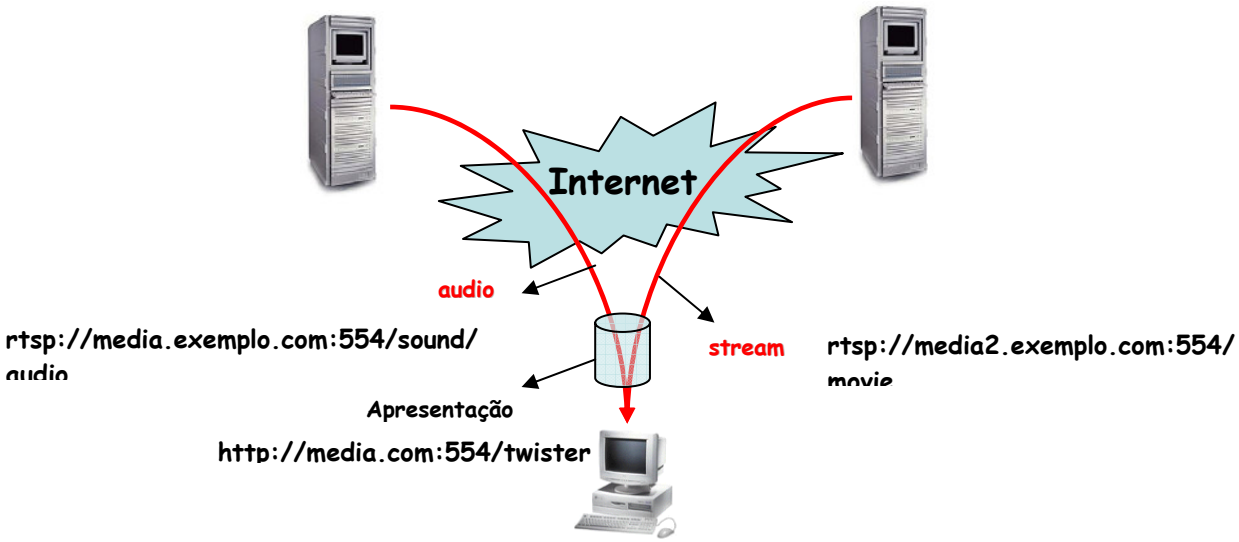

Figura 07. Operação RTP.

2.3 RTSP

O protocolo de mídia contínua em tempo real, conhecido como RTSP, é baseado na classe "*Stream Stored audio and video*". Ele opera na camada de aplicação e foi idealizado pela Real Networks, Netscape Communications e a Universidade de Columbia. Sua primeira RFC foi publicada pela IETF sob o número 2326 [4].

O *Real Time Streaming Protocol* (RTSP) é um protocolo em nível de aplicação para controle na transferência de dados com propriedades de tempo real. RTSP torna possível a transferência, sob demanda, de dados em tempo real como áudio e vídeo. Ele serve para estabelecer e controlar um único ou vários *streams* sincronizados de mídias contínuas pertencentes a uma apresentação [32].

O conjunto de *streams* a ser controlado é definido por uma descrição de apresentação (*Source Description* [22]), normalmente um arquivo, que pode ser obtido por um cliente usando HTTP ou outro meio como e-mail, e não necessariamente precisa estar armazenado em um servidor de mídia.

Uma descrição de apresentação contém informações sobre um ou mais fluxos que compõem uma apresentação, como endereços de rede e informações sobre o conteúdo da apresentação (assunto, e-mail do responsável pela sessão, tempo da apresentação), além de parâmetros que tornam possível ao cliente escolher a combinação mais apropriada das mídias. Na descrição da apresentação, cada fluxo é individualmente identificado por uma URL RTSP, a qual aponta para um servidor de mídia que trata aquele fluxo particular e dá um nome ao fluxo armazenado naquele servidor. Vários fluxos (áudio e vídeo) podem ser localizados em servidores diferentes para fins de balanceamento de carga. Além disso, a descrição da apresentação também descreve quais métodos de transporte o servidor é capaz de oferecer.

Vários modos de operação são utilizados, como *unicast* e *multicast*. Em relação ao funcionamento de RTSP, não existe a noção de uma conexão RTSP. Em vez disso, um servidor mantém uma sessão indicada por um identificador. Uma sessão RTSP não está ligada a uma conexão em nível de transporte, como acontece numa conexão TCP. Durante uma sessão RTSP, um cliente RTSP pode abrir e fechar conexões de transporte para o servidor emitir requisições RTSP. Geralmente, o controle RTSP pode acontecer em uma conexão TCP, enquanto o fluxo de dados segue via UDP ou RTP. Entretanto a operação de RTSP não depende do mecanismo de transporte utilizado para o transporte das mídias contínuas.

O protocolo suporta as seguintes operações: recuperação de mídia de um servidor de mídia, convite de um servidor de mídia para uma conferência (apresentação ou registro de um, ou um subconjunto de mídias da conferência) e adição de mídias a uma apresentação existente.

O RTSP é um protocolo de domínio público que permite a interação cliente-servidor entre a fonte do fluxo de mídia a taxa constante (servidor) e o usuário (transdutor) [23]. Essa interatividade vem da necessidade do usuário obter um maior controle sobre a reprodução da mídia no transdutor. As funcionalidades do RTSP resumem-se às manipulações de execução do arquivo similarmente às funcionalidades

que um aparelho reprodutor de DVD disponibiliza para se ouvir vídeo gravado, permitindo que um transdutor controle a corrente de mídia através de comandos.

O RTSP tem o estabelecimento e controle de um ou vários fluxos de áudio e vídeo sincronizados [21]. É importante esclarecer que o RTSP não é responsável pelo transporte do conteúdo da mídia em si. Para realizar o transporte, o RTSP relaciona-se com outro protocolo, o RTP.

O RTSP funciona através de trocas de mensagens, requisições e respostas, entre o cliente e o servidor. Seu funcionamento é independente do protocolo que transporta-rá a mídia. Não existe o conceito de conexão RTSP. O servidor identifica cada sessão através de um identificador único. Isto porque o servidor precisa guardar o estado dos clientes. Existem três estados possíveis: setup, play-record e pause, o recebimento de mensagens implica na transição entre eles.

É preciso descrever um processo comum de interação cliente e servidor para um melhor entendimento global do protocolo. O processo é bem simples do ponto de vista do usuário. Um usuário digita uma URL em seu aplicativo de vídeo, do tipo *rtsp://servidor.com/video*, e o servidor iniciará a transmissão do vídeo para o cliente, ficando atento para novos comandos. Sustentando este processo simples, há várias trocas de mensagens de diferentes protocolos.

O RTSP tem relacionamento com o protocolo HTTP, utilizando na execução de uma função, ou possui características semelhantes. O primeiro a ser descrito é o protocolo de transporte TCP. Para que as mensagens RTSP sejam trocadas entre cliente e servidor, uma conexão *full-duplex* TCP é estabelecida. Em seguida o RTSP precisa se relacionar com outro protocolo de transporte de dados, para que a mídia seja transportada. Na grande maioria das vezes, o transporte dos dados de mídia é feito através do protocolo RTP. O RTP criará uma conexão UDP unidirecional para o transporte da mídia e o RTCP cria duas conexões UDP em sentidos opostos para controle de sincronismo no cliente e informação de perda de pacotes para o servidor [21].

É possível também que o transporte seja realizado por outros protocolos. Um exemplo é o protocolo proprietário da RealNetworks, o RDT (*Real Data Transport*) [3]. Em termos de conexões estabelecidas, a diferença ocorre nas conexões UDP onde as duas são unidirecionais, uma no sentido do servidor e outra no sentido do cliente. A conexão no sentido do servidor serve para o cliente requisitar o re-envio de pacotes perdidos [17].

O RTSP é usado quando os espectadores se comunicam com um servidor *unicast*. O RTSP permite comunicação bilateral, isto é, os espectadores podem comunicar-se com o servidor de transmissão e dar alguns comandos, como retroceder o filme, pular o capítulo, e assim por diante. Os *players* traduzem automaticamente a interação do espectador com o controle do filme na tela dentro da requisição adequada ao RTSP. Em contrapartida, o RTP é um protocolo unilateral usado para enviar transmissões ao vivo ou arquivadas, do servidor para o cliente.

O RTSP é intencionalmente semelhante ao HTTP, porém com algumas características a mais:

a) Introduz novos métodos e possui identificador de protocolo diferente; Ex: rtsp://media.example.com:554/video

b) Tanto um cliente quanto um servidor RTSP podem emitir requisições de conexão;

c) Possui uma máquina de estados para o controle da mídia;

d) A reprodução de uma mídia só é iniciada quando o buffer possui uma quantidade mínima desejada de dados;

e) O RTSP reutiliza os mecanismos de segurança e autenticação do HTTP;

f) Independência da camada de transporte: opera sobre TCP ou UDP (via RTP), sendo o segundo para aplicações de tempo real;

g) O cliente pode negociar o método de transporte para TCP e UDP, usando "rtsp:" e "rtspu:";

h) Negociação de banda (bps).

2.3.1 Máquina de Estados RTSP

O RTSP necessita manter um estado de sessão, pois qualquer alteração que o usuário faça em uma apresentação afetará individualmente cada fluxo.

Funcionalmente, o RTSP não encapsula os comandos no mesmo pacote de transmissão de mídia, envia-os paralelamente, em portas diferentes, semelhantemente ao FTP. Assim, o RSTP é chamado de protocolo fora da banda. A banda é considerada como sendo o canal de transmissão da mídia.

Os métodos modificam o estado do servidor e do cliente (Alocação de Recursos):

a) SETUP: Servidor aloca recursos para um *stream*;

b) PLAY e RECORD: inicia a transmissão no canal alocado;

c) PAUSE: Paralisa sem liberar os recursos;

d) TEARDOWN: Libera os recursos e encerra a sessão.

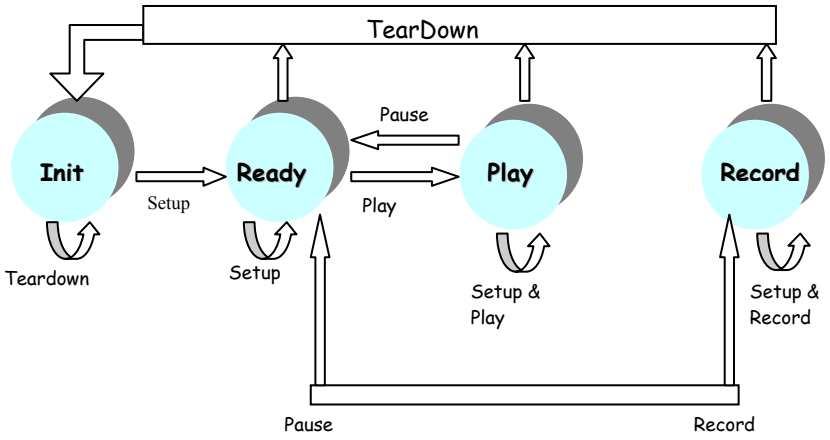

Figura 08. Diagrama de estados RTSP.

Estados do Cliente	**Estados do Servidor**
Init: SETUP enviado, aguardando resposta.	**Init**: Estado inicial, aguardando SETUP.
Ready: Confirmação de SETUP recebido ou PAUSE recebido enquanto reproduzia ou gravava.	**Ready** : SETUP recebido, enviando confirmação. Ou PAUSE recebido, enviando confirmação.
Play : Mensagem PLAY confirmada.	**Play**: PLAY recebido e confirmado. Os dados começam a ser transmitidos.
Record: Mensagem RECORD confirmada.	**Record** : O servidor está gravando.

Tabela 01. Estados RTSP.

Cada sessão RTSP possui um identificador atribuído pelo servidor (Tabela 01). O cliente inicia a sessão RTSP com uma requisição de SETUP ao passo que o servidor a responde com um identificador (requisição RTSP SETUP). O cliente então fornecerá esse identificador para cada requisição que fizer até que feche a sessão com uma requisição TEARDOWN (Figura 09). Em tempo, o início da transmissão da mídia acontece quando o transdutor envia uma requisição de RTSP PLAY (Figura 08).

Como exemplo da utilização de RTSP temos mídia sob demanda (Figura 09). Em mídia sob demanda, um cliente requisita um filme de um servidor de áudio (audio.example.com) e de um servidor de vídeo (video.example.com). A descrição da mídia é armazenada em um servidor Web e o cliente estabelece uma conexão com o servidor, solicitando o arquivo de descrição da apresentação. O servidor Web responde com OK e mais informações (tipo de conteúdo a ser apresentado, o endereço onde o

áudio e o vídeo se encontram). Logo depois, o cliente envia um comando para o servidor de áudio e para o servidor de vídeo pedindo para iniciar uma sessão (método SETUP) e em seguida para inicializar a representação de áudio e vídeo (método PLAY). Os servidores enviam respostas de OK e, posteriormente, transportam o áudio e vídeo ao cliente, através de RTP ou outro protocolo.

Cabe ressaltar que RTSP se faz presente aqui fornecendo informações de sincronização para o cliente e informações de perda de pacotes para o servidor, entre outras. Por fim, o cliente solicita aos servidores para liberar recursos através do comando TEARDOWN, encerrando a sessão RTSP com uma resposta de OK dos servidores.

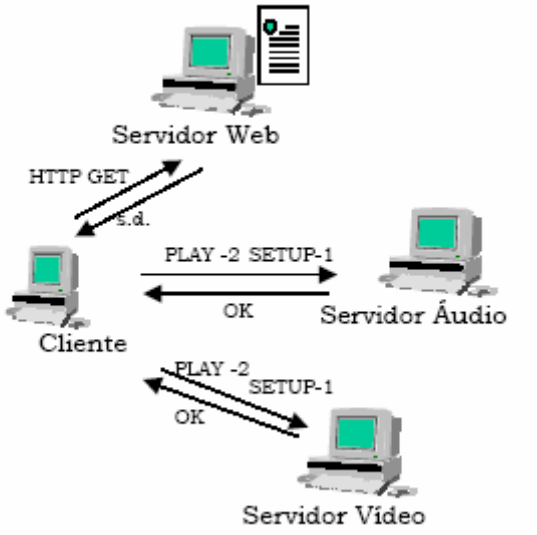

Figura 09. Mídia sob demanda em RTSP.

Já existem implementações de RTSP disponíveis para download na Internet e aplicações diferentes para as quais RTSP é apropriado, como mídia sob demanda. Dentre as áreas de aplicação estão o Rádio na Internet, o controle de dispositivos, além de registro remoto, o qual é apropriado para videoconferência.

Em termos de aplicabilidade, o RTSP tem um bom desempenho ao atuar sob servidores sob demanda e apresentar inúmeras aplicações.

2.4 RTCP

Apesar das características vantajosas para multimídia do RTP, ele ainda não satisfaz nenhuma das necessidades de controle e QoS [37]. O protocolo de controle de transporte em tempo real conhecido como RTCP (*Real Time Control Protocol*), definido na RFC 1889, por sua vez, implementa funções de controle na troca de informações entre as fontes e os destinos.

São cinco as mensagens de controle implementadas no RTCP:

a) **SR (*Sender Report*)** – São mensagens geradas pelos usuários que estão enviando os pacotes. Elas descrevem, além da quantidade dos dados transmitidos, as informações de sincronismo entre os diferentes tipos de transmissão.

b) **RR (*Receiver Report*)** – É emitida pelos receptores das informações, revelando a qualidade na recepção do fluxo. Isso faz com que a fonte possa realizar alterações na transmissão baseando-se nas mensagens RR que recebem dos destinos. O destino recebe, de tempos em tempos, uma descrição do tráfego gerado pela fonte (SR) e contabiliza os dados recebidos. Um pacote RR carrega informações sobre a diferença entre o tráfego gerado e o tráfego recebido, possibilitando o cálculo do impacto destes dados sobre a rede. Informações contidas nestes pacotes RTCP RR contêm o maior número de seqüência recebido, número de pacotes perdidos, atraso e variações de atraso entre pacotes. A fonte pode ou não alterar suas características de transmissão em função dos relatórios recebidos dos destinos.

c) **SDES (*Source Description*)** – Nessa mensagem seguem informações adicionais sobre cada participante de uma sessão RTP (e-mail, telefone, localização geográfica, etc.) visando exclusivamente sua identificação. São emitidas por fontes visando suprir mais informação sobre as mesmas. Exemplos incluem CNAME (Canonical Name), um identificador único global semelhante ao formato dos endereços de e-mail. O CNAME é usado para resolver conflitos no valor SSRC (o enviador original da mensagem) e associar diferentes fluxos de mídia gerados pelo mesmo usuário. Pacotes *Source Description* também identificam os participantes diretamente pelo nome, e-mail e número de telefone, localização geográfica do emissor, aplicação que está gerando o fluxo e um texto adicional. Aplicações cliente podem mostrar o nome e informações de e-mail na interface do usuário permitindo aos participantes da sessão conhecer outros participantes. Ele também permite aos participantes obter informações de contato (como e-mail e telefone) para realizar outras formas de comunicação (como iniciar uma sessão de conferência separada utilizando SIP).

d) **BYE** – Enviando por uma das fontes quando está saindo da sessão RTP.

e) **REPORTING INTERVAL** – Mensagem contendo informações sobre a qualidade do fluxo de dados recebido e enviado, fornecida a todos os participantes de uma sessão RTP.

Através desses pacotes com informações de controle, RTCP oferece os seguintes serviços:

a) Monitoração de QoS e Controle de Congestionamento: Esta é a função primária do RTCP. O RTCP fornece um retorno de informações para uma aplicação sobre a qualidade da recepção dos dados e contém informações necessárias para monitoração de QoS [17]. As informações de controle são apropriadas para os enviadores dos dados RTP. Os recebedores podem determinar se um congestionamento está sendo local, regional ou global. Administradores de rede podem avaliar o desempenho da rede para distribuição multicast.

b) Identificação da fonte: Em pacotes de dados RTP, fontes são identificadas por identificadores gerados aleatoriamente. Esses identificadores são diferentes para cada mídia particular gerada pela fonte.

c) Sincronização intermídia: Pacotes RTCP SR contêm uma identificação de tempo real (NTP Timestamp) e um correspondente *timestamp* RTP (RTP Timestamp). Esses dois valores permitem sincronização de diferentes mídias, como por exemplo, sincronização labial de áudio e vídeo.

d) Escalabilidade das informações de controle: Pacotes RTCP são enviados periodicamente entre participantes. Quando o número de participantes aumenta, as informações de controle são balanceadas e o tráfego de controle é limitado. RTCP recebe a designação técnica como um protocolo separado porque as mensagens utilizam uma porta UDP diferente da utilizada pelo tráfego RTP. O valor da porta default RTCP (5005) é um número a mais do que o correspondente tráfego RTP (5004).

O RTP e o RTCP foram projetados para ser independentes das camadas subjacentes de rede e transporte. Enquanto o RTP é responsável pelo transporte das mídias contínuas (áudio e vídeo), o RTCP controla as informações retornadas pelos usuários que receberam os dados, informando sobre a qualidade da recepção e transferência dos dados, o suporte e a sincronização de diferentes fluxos de mídia [31].

2.5 Formatos para Transmissão de Mídia Contínua

2.5.1 MPEG-4

MPEG, que significa Moving Picture Experts Group, é o nome dado a uma família de padrões internacionais que são utilizados na codificação das informações de áudio e vídeo em um formato digital comprimido. O grupo surgiu em 1987 com o propósito de desenvolver tais algoritmos e ferramentas. Como conseqüência, tem-se hoje uma série de padrões que incluem o MPEG-1, o MPEG-2 e o MPEG-4, formalmente conhecidos como ISO/IEC-11172, ISO/IEC-13818 e ISO/IEC-14496.

O padrão MPEG-1 [12] [14] foi o primeiro membro dessa família, tendo sido disponibilizado no início dos anos 90. O MPEG-1 caracteriza-se por ser voltado a aplicações de vídeo para CD-ROM a uma taxa de 1.5 Mbps. O segundo membro dessa família, o MPEG-2 [14], é uma evolução do MPEG-1 e teve sua primeira versão pronta em 1994. Esse padrão foi desenvolvido visando aplicações como TV Digital, DVD e TV por Satélite, a uma taxa variando de 3 a 8 Mbps. O padrão MPEG-2 incorporou o conceito de perfil e níveis, até então não existentes na família MPEG. Entende-se por perfil (*profile*) um conjunto de ferramentas algorítmicas, representando uma tendência particular de desempenho e de recursos de consumo, que suportam os requisitos de um conjunto específico de aplicações.

O nível (*leve*l) de um perfil descreve os parâmetros de desempenho (tamanho máximo do quadro, macroblocos por segundo, taxa em bits por segundo etc.), especificando os limites inferiores de capacidade necessárias para o decodificador MPEG [15].A primeira versão do padrão MPEG-4 foi lançada em 1998, tornando-se oficialmente internacional em 1999 [12] [14]. O MPEG-4 surgiu com o propósito de ser mais do que um simples codificador de áudio ou vídeo, a idéia foi de definir um sistema de codificação e distribuição de cenas audiovisuais. Para atender a esse propósito, o grupo de desenvolvimento do MPEG-4 buscou, e continua buscando, adicionar novas funcionalidades aos padrões MPEG anteriores. Atualmente, o padrão MPEG-4 encontra-se na versão 2, que foi oficializada no início de 2000 e subdividida em 17 partes. É importante ressaltar que o padrão MPEG-4 respeita a compatibilidade entre suas versões, como também com os padrões MPEG-1 e MPEG-2, como ilustra a Figura 10.

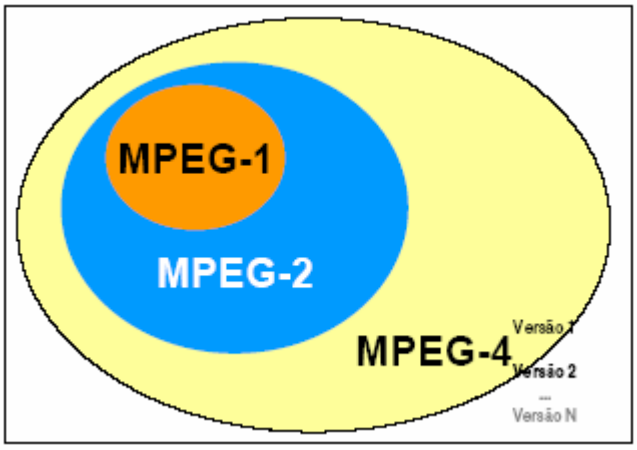

Figura 10. Padrão MPEG.

O MPEG-4 é o padrão global de multimídia, transmitindo áudio e vídeo de qualidade profissional por vários tipos de largura banda, de telefones celulares a banda larga e outros. O áudio e vídeo são os centros das especificações do MPEG-4, mas ele também suporta objetos 3D, texto e outros tipos de mídia. O MPEG-4 também evoluiu para transportar mídia a qualquer taxa de transmissão, com uma qualidade compatível desde conexões de linha discada até redes de banda larga [15].

Basicamente, o padrão MPEG-4 possui como principais componentes: BIFS e DMIF. O BIFS (*Binary Format for Scenes*) é um formato binário para descrição de uma cena MPEG-4, definindo o relacionamento que há entre todos os objetos da cena. O DMIF (*Delivery Multimedia Framework*), abrange desde o armazenamento até a transmissão (transporte) de uma cena MPEG-4.

O MPEG-4 evoluiu para ter um envio escalonável, com vários avanços em tecnologia para oferecer melhor desempenho. Por exemplo, o codificador oferece controle de taxas de dados, podendo ser ajustado para uma determinada taxa de transmissão de dados que garanta reprodução apropriada para um mecanismo particular de envio. O versátil codificador pode usar o controlador de taxa de transmissão VBR (*variable bit rate*) *single-pass* para maximizar a precisão de saída na maior qualidade ou maximizar a velocidade para a codificação mais rápida possível. Além disso, o MPEG-4 apresenta rigoroso gerenciamento de cores, com um gerenciador de alto desempenho e estimativa de movimento otimizada para maior precisão e velocidade. O decodificador também oferece uma etapa de pós-processamento otimizada para remover artefatos de codificação.

O MPEG-4 foi projetado para entregar vídeos com qualidade de DVD (MPEG-2), a baixas taxas de transmissão e com arquivos de tamanhos menores, foi desenvolvido também o novo codificador *Advanced Audio Coding* (AAC), oferecendo uma compressão mais eficiente do que o MP3, com qualidade equiparável a de um CD sem compressão.

2.5.1.1 Características do MPEG-4

Para conseguir oferecer tecnologias para diferentes "públicos-alvo", o MPEG-4 padronizou a forma de codificação e composição de seus objetos de mídia, a multiplexação e sincronização dos dados associados a esses objetos, como também a forma de interação que deve haver entre a cena desenvolvida e o usuário final. O processo de codificação de um objeto de mídia está diretamente relacionado com a forma que esses objetos encontram-se na composição hierárquica [16].

O MPEG-4 padroniza um número de objetos de mídia primitivos, capazes de representar tanto conteúdos do tipo sintético quanto natural, e ainda tanto objetos bidimensionais quanto tridimensionais. Adicionalmente, o MPEG-4 define uma representação codificada de objetos como textos e gráficos, textos associados usados para sintetizar falas e animar faces, animação de corpos em conjunto com as faces, e áudio sintetizado. Um objeto de mídia na sua forma codificada consiste de elementos descritivos que possibilitam o gerenciamento do objeto em uma cena audiovisual como também no fluxo de dados associado, se necessário. É importante destacar que na sua forma codificada, cada objeto de mídia pode ser representado de forma independente da sua vizinhança ou background.

De forma mais geral, o MPEG-4 provê uma maneira padronizada para descrever uma cena, permitindo, por exemplo: colocar os objetos de mídia em alguma posição de um dado sistema de coordenadas; aplicar transformação para modificar a aparência geométrica e acústica dos objetos de mídia; agrupar os objetos de mídia primitivos objetivando formar as composições dos objetos de mídia; aplicar dados de fluxo (*streamed*) aos objetos de mídia buscando modificar os seus atributos (por exemplo, um som, uma textura movimentada pertencendo a um objeto; parâmetros de animação dirigindo uma face sintética); e modificar, interativamente, a visualização e pontos de audição do usuário em qualquer lugar da cena O padrão MPEG-4 buscou também definir um modelo comum para a multiplexação e a sincronização dos dados associados aos objetos de mídia. Esses objetos precisam ser tratados (encapsulados) em fluxos de dados que devem ser transportados em um ou mais fluxos elementares (*ES - Elementary Stream*). Um descritor de objeto (OD – *Object Descriptor*) identifica todos os fluxos associados com um objeto de mídia, permitindo, assim, o gerenciamento hierárquico dos dados codificados, como também a associação de metainformação ao conteúdo (chamado de *object content information*) e a gerência dos direitos de propriedade intelectual associados a eles.

Cada fluxo é caracterizado por um conjunto de descritores para informações de configuração, por exemplo, determinar os recursos do decodificador necessários e a precisão das informações temporais codificadas. Além disso, os descritores também carregam sugestões de qualidade de serviço (QoS) que são necessárias para a transmissão (por exemplo: taxa máxima de bit, taxa de erro de bit, prioridade, etc).

A sincronização de cada fluxo elementar é alcançada através da marcação de tempo (*timestamp*) de unidades de acesso individuais dentro dos fluxos elementares. A camada de sincronização gerencia a identificação de tais unidades de acesso e os *timestamp*. Independentemente do tipo de mídia, essa camada permite a identificação do tipo da unidade de acesso nos fluxos elementares (por exemplo, quadros de vídeo ou de áudio, comandos de descrição de cena), recuperados na base de tempo dos objetos de mídia e dos descritores de cena, habilitando a sincronização entre eles. A sintaxe nessa camada pode ser configurada de várias formas, possibilitando o uso em um largo espectro de sistemas. Por fim, o MPEG-4 especificou a forma de interação do usuário com os objetos de mídia.

Essa interação pode ser separada em duas principais categorias: interação pelo lado do cliente e interação pelo lado do servidor. A interação do lado do cliente é gerenciada pelo usuário final e envolve a manipulação de conteúdo. Dependendo do grau de liberdade definido pelo autor da cena MPEG-4, o usuário final pode interagir com essa cena de forma a modificar os seus pontos de visualização e de escuta, arrastar os objetos da cena para diferentes posições, desencadear eventos em cascata através do clique em um objeto específico, ou ainda, selecionar um idioma desejado quando múltiplos idiomas estão disponíveis para seleção. Já a interação pelo lado do servidor implica na manipulação de conteúdo que ocorre no início da transmissão, inicializada por alguma ação do usuário. Essa interação necessita que um canal de retorno (*back-channel*) esteja disponível [13].

2.5.1.2 Funcionalidades no MPEG-4

As principais funcionalidades e as diferentes partes do padrão MPEG-4 oferecem:

a) Transporte - O MPEG-4 não define camadas de transporte. Em vários casos, uma adaptação é feita sobre uma camada de transporte já existente, sobre o Fluxo de Transporte MPEG-2 e o Transporte sobre IP (em cooperação com IETF – *Internet Engineering Task Force*). O transporte "MPEG-4 sobre MPEG-2" (conhecido como MPEG-2 TS) é feito através da emenda 13818-1 no padrão MPEG-2 Systems [14] [15]. Já o transporte "MPEG-4 sobre IP" foi desenvolvido juntamente com o grupo de trabalho IETF AVT, incluindo um framework e várias especificações no formato RTP payload. O framework é especificado tanto na Parte 8 do Padrão MPEG-4 (ISO/IEC 14496-8) quanto como uma RFC informativa no IETF. As especificações no formato RTP payload são padronizadas apenas com uma RFC no IETF.

b) DMIF - Outra funcionalidade descrita no MPEG-4 é o DMIF (*Delivery Multimedia Integration Framework*), que atua como uma interface entre a aplicação e o transporte. Basicamente, o DMIF é um protocolo de sessão para o gerenciamento dos fluxos multimídia sobre tecnologias genéricas de entrega. Com isto, esse protocolo permite que o desenvolvedor de uma aplicação MPEG-4 não tenha o transporte como uma preocupação, pois uma única aplicação pode executar em diferentes camadas de transporte, caso a mesma seja suportada pela instanciação do DMIF adequado. A princípio, o DMIF é bastante parecido com o FTP (*File Transfer Protocol*), tendo como única e essencial diferença o fato de que o FTP retorna os dados, enquanto que o DMIF retorna ponteiros indicando para onde se deve ir a fim de recuperar o fluxo de dados. Quando o DMIF é executado, a primeira ação disparada é a de estabelecer uma sessão com o lado remoto. Uma vez feito isso, fluxos são selecionados e o DMIF envia uma requisição para que eles comecem a ser enviados (seja feito o *streaming* deles). A camada DMIF, por sua vez, irá retornar ponteiros para as conexões onde os fluxos serão recuperados, estabelecendo assim a comunicação entre os dois lados [17].

c) System - Uma outra funcionalidade bastante importante no MPEG-4 está relacionada com a Parte 1 do padrão: *System*. Essa parte preocupa-se com a relação que pode haver entre os componentes de vídeo e áudio que constituem uma cena. A parte *System* define uma caixa de ferramentas de algoritmos de compressão avançados para informação de áudio e visual. Os fluxos elementares resultantes do processo de codificação podem ser transmitidos ou armazenados separadamente (um fluxo para os elementos visuais e um fluxo para o áudio), mas precisam ser combinados para criar a apresentação multimídia desejada no lado do receptor. As relações que precisam ser definidas entre os elementos visuais e de áudio podem ser descritas em dois principais níveis: BIFS e ODs. As BIFS descrevem as ligações (relações) espaço-temporal dos objetos da cena MPEG-4. Em um nível mais baixo estão os ODs (*Object Descriptor*) que são os descritores de objeto que definem a relação entre os ESs (fluxos elementares) pertinentes a cada objeto (por exemplo, o fluxo de áudio e o fluxo de vídeo de um participante em uma vídeo-conferência). Os ODs também fornecem

informações adicionais tais como as URLs necessárias para acessar os fluxos elementares, as características necessárias para os decodificadores para interpretar esses fluxos, propriedades intelectuais, entre outros aspectos. A parte Systems do padrão MPEG-4 também especifica um formato de arquivo padrão que suporta a troca e autoria de conteúdo MPEG-4, conhecido como MP4 [12] [15]. As questões de interação com o usuário, as APIs do MPEG-J, a ferramenta FlexMux, para o mecanismo de intercalação de múltiplos fluxos em um único fluxo, e bases de dados cobrindo identificação de direitos de propriedade intelectual relacionados aos objetos de mídia são também cobertas na Parte 1 do padrão.

d) Visual - A Parte 2 do padrão MPEG-4, Visual, permite a codificação híbrida de imagens naturais (baseadas em pixels) e de vídeo, em conjunto com cenas sintéticas (geradas pelo computador). Essa característica torna possível, por exemplo, a presença virtual de participantes em uma vídeo conferência. O Visual também compreende ferramentas e algoritmos para suporte à codificação de imagens estáticas naturais (baseadas em *pixels*) e seqüências de vídeo, bem como ferramentas para o suporte à compressão de parâmetros geométricos de gráficos bidimensionais e tridimensionais sintéticos. Esta seção tem o propósito de apresentar, de forma resumida, as principais funcionalidades e características existentes nas ferramentas e nos algoritmos definidos na parte Visual do padrão MPEG-4. Os objetos de mídia visuais no MPEG-4 podem assumir o formato de vídeo progressivo como também de vídeo entrelaçado, possuindo uma taxa de bit variando de 5 kbit/s a uma taxa que pode ser superior a 1 Gbit/s. Os quadros MPEG-4 podem se apresentar em uma resolução tipicamente sub-QCIF (128x96) [15] até uma resolução de estúdio (4kx4k *pixels*).

e) Audio - A parte 3 do padrão MPEG-4 Audio, fornece facilidades para uma grande diversidade de aplicações para tratamento do áudio, variando desde uma fala de boa compreensão a um áudio multicanal de alta qualidade, como também dos sons naturais aos sons sintetizados. Em particular, MPEG-4 Audio suporta a representação eficiente dos objetos de áudio pertencentes aos seguintes sinais: sinais genéricos de áudio, sinais de fala, áudio sintético e fala escalável sintetizada. Para os sinais genéricos de áudio existe suporte à codificação de sons genéricos em um intervalo de baixas taxas de transmissão (*low bitrate*) até alta qualidade. Essa variação de taxas tem início em um valor de 6 kbit/s e largura de banda abaixo de 4 kHz, indo até uma qualidade de áudio broadcast (do mono a multicanal). O padrão afirma que a alta qualidade pode ser alcançada com baixos retardos. O *Parametric Audio Coding* permite manipulação de som em baixas velocidades. Já os sinais de fala podem ter sua codificação usando taxas que variam de 2 kbit/s a 24 kbit/s. Taxas mais baixas, tais como numa média de 1.2 kbit/s, também são possíveis quando uma codificação à taxa variável é permitida. Essas taxas baixas são permitidas (toleradas) para aplicações de comunicação. O áudio sintético é representado através do MPEG-4 *Structured Audio*, que é uma linguagem para descrever instrumentos (nome dado a pequenos programas que são utilizados para gerar som) e scores (nome dado à entrada que gerencia (dirige) tais objetos). Esses objetos não são necessariamente instrumentos musicais, essencialmente eles são fórmulas matemáticas que podem gerar sons de piano, de uma queda d'água, ou algum outro som da natureza, por exemplo. Por fim, para suporte a fala escalável

sintetizada o padrão sugere o uso de codificadores TTS (Text-To-Speech). As taxas (bitrate) de codificadores TTS variam de 200 bit/s a 1.2 Kbit/s. Nesse grupo é possível trabalhar tanto com um texto puro ou com um texto que possua parâmetros de prosódia (entonação de contorno, duração de fonemas etc.) como entrada, gerando como saída uma fala sintética inteligível.

2.5.2 MOV

O MOV, outro formato de mídia bastante popular, é utilizado para armazenar seqüências de vídeo e foi criado pela Apple Computers. O QuickTime Player foi apresentado pela Apple como uma alternativa à plataforma Windows Media, da Microsoft, apostando na variedade de formatos disponíveis para a distribuição de conteúdo. A polivalência do QuickTime aparece tanto no iTunes, que vende músicas e grava CDs no formato AAC Audio, como na popular reprodução de trailers no site da Apple, feita totalmente em formato MOV. Este formato de vídeo ganhou ainda mais popularidade quando suas especificações foram escolhidas pelo consórcio MPEG para uma parceria.

O Quick Time Movie (formato MOV) suporta qualquer quantidade de canais de vídeo, sendo apropriado para criação de efeitos especiais em tempo real. O Quick Time Movie permite pegar vários canais de áudio, de diferentes freqüências e diferentes tipos de arquivos e combiná-los em um único arquivo.

2.5.2.1 Arquitetura do QuickTime

QuickTime é um formato de arquivo (.mov), um conjunto de aplicativos e plugins, e uma biblioteca de software. É possível pensar no QuickTime como uma série de funções e de estruturas de dados passíveis de utilização em aplicações para controle de mudança e combinação de dados e mídias. No QuickTime, um conjunto de mídias dinâmico é denominado simplesmente como *movie* (filme, em português) [7].

Originalmente, o QuickTime foi desenvolvido para trazer movimento para dentro do computador e permitir a execução de vídeo no próprio desktop. Mas conforme ele foi se desenvolvendo, baseado principalmente na profusão da multimídia nos computadores pessoais, foi ficando evidente que não apenas vídeos estavam envolvidos. Elementos foram adicionados para que as apresentações estáticas pudessem ser organizadas ao longo de uma linha de tempo conforme trocava dinamicamente as informações apresentadas.

O conceito de mídia dinâmica inclui não apenas filmes, mas também animações em 3D, música, seqüências sonoras, ambientes virtuais e ainda dados, informações e mídias de todos os tipos. Desta forma, o QuickTime transformou o conceito de *movie*, que originalmente era um formato dedicado a execução de vídeo, em um conceito mais amplo. Tornou-se uma mídia que contém diversas outras e onde se pode especificar e controlar todas as outras em uma exibição baseada em uma linha temporal.

O processo de construção do *movie* do QuickTime é exemplificado na figura 11:

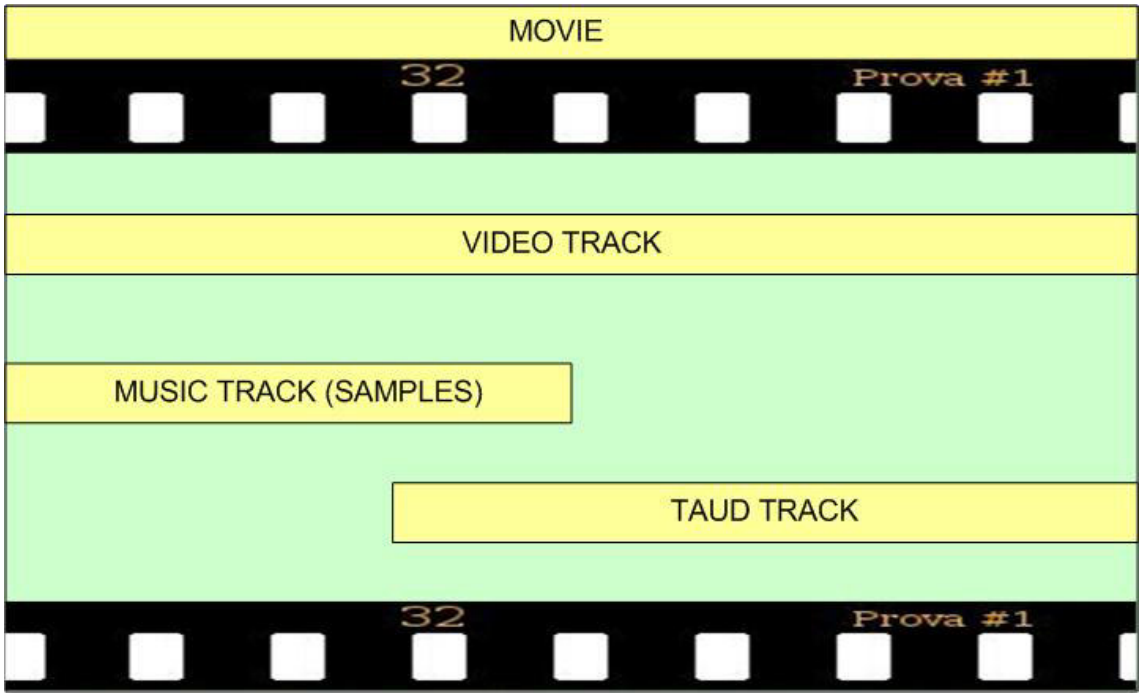

Figura 11. Construção do Movie.

Um arquivo QuickTime Movie pode conter diversas trilhas. Cada uma dessas trilhas referem-se a estrutura de dados de uma determinada mídia, contendo referências dessa mídia, que pode estar armazenada como som ou imagens no HD, CD-ROM ou DVD. Através do QuickTime, qualquer tipo de mídia (auditiva ou visual, ou ambas) podem ser organizadas no formato movie. Com ele é possível organizar, editar, copiar, comprimir e reproduzir praticamente qualquer tipo de mídia. Apesar dos detalhes serem um pouco mais complexos, as idéias centrais são poucas e relativamente simples:

a) Os arquivos QuickTime Movie são estruturas de catalogação de dados. Eles contêm toda a informação necessária para organizar esses dados temporalmente, mas não os arquivos de dados propriamente ditos;

b) Os arquivos QuickTime Movie são compostos por trilhas (figura 12). Cada trilha refere-se e organiza a seqüência de dados de um mesmo tipo de mídia na ordem em que deverão ser executadas.

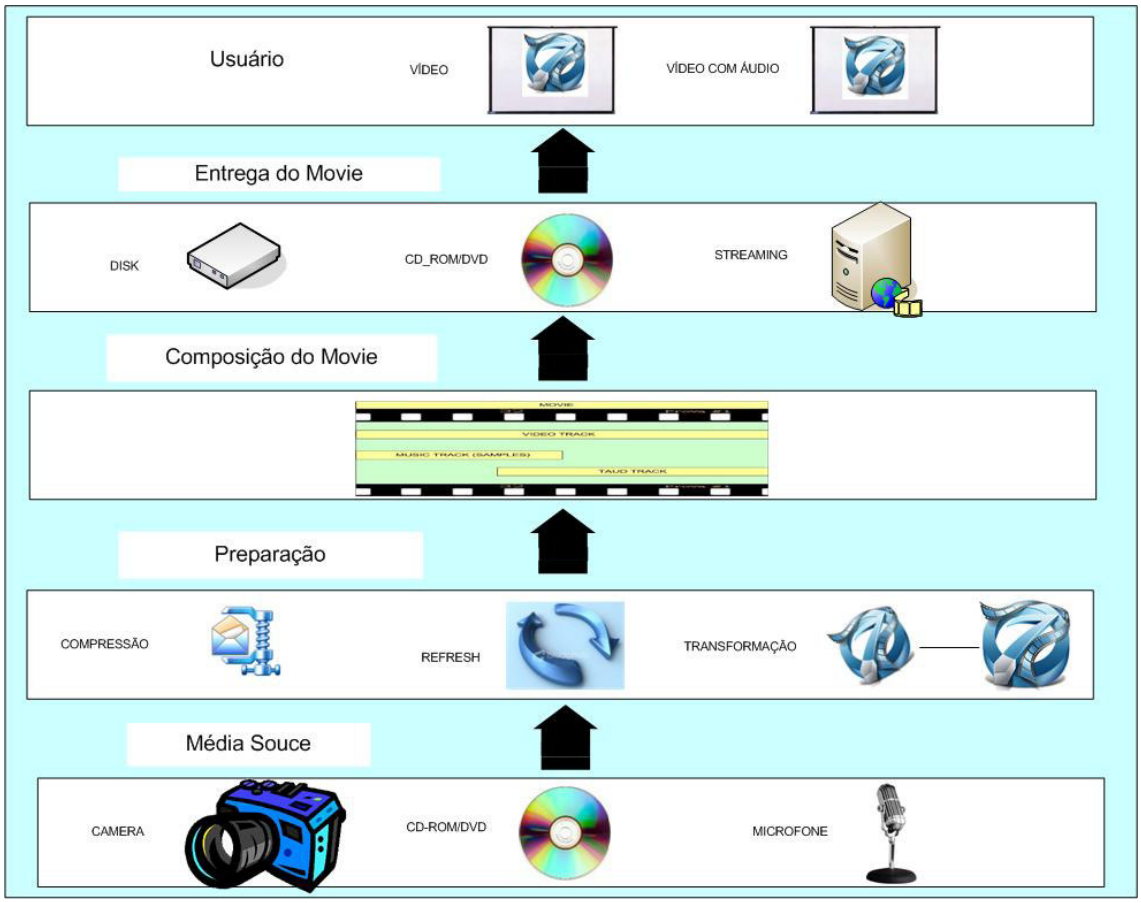

Figura 12. Reprodução do Movie.

Um arquivo movie, tipicamente, contém o arquivo *movie* em si e as mídias a que ele se refere, de forma a permitir o transporte ou o download de todos os elementos necessários para a execução juntos, chamado *self-contained*, conforme a figura abaixo. O modo *self-contained* salva em partes.

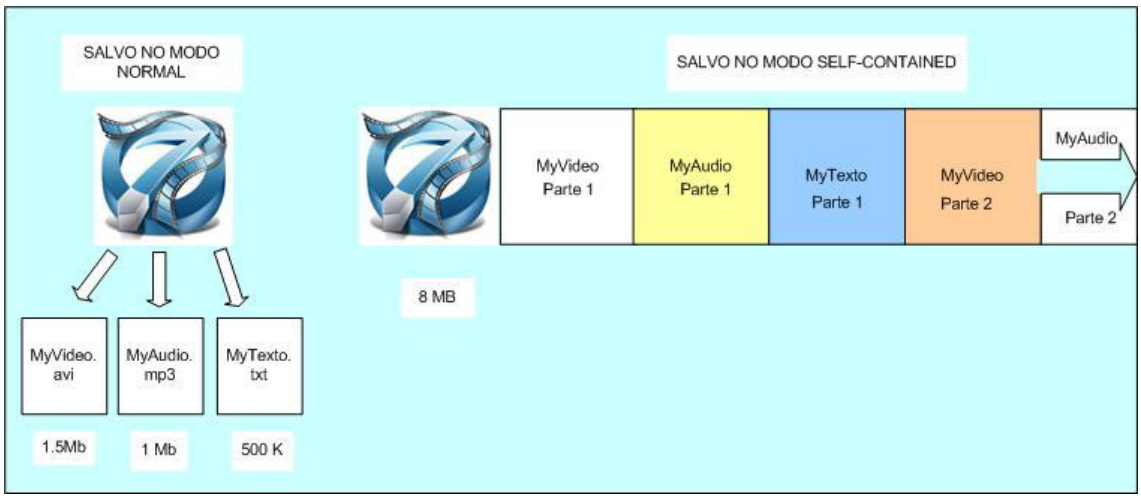

Figura 13. Modo Self-Contained.

O formato de arquivo nativo do QuickTime é o QuickTime Movie, e cuja extensão correspondente no Windows é .mov, visto que o Mac não utiliza extensões de

nome nos arquivos. O formato QuickTime Movie pode ser distribuído em CDs, DVDs ou pela Internet, podendo ser distribuído a partir de qualquer sistema operacional utilizado pelo servidor Web: Unix, Solaris, Linux, Mac OS ou Windows. Os arquivos QuickTime Movie são estrutura de dados extremamente versáteis que podem conter praticamente qualquer combinação de mídias utilizando praticamente qualquer forma de compressão de dados.

Os aplicativos e plug ins que compõe QuickTime são: o QuickTime Player, o QuickTime Plug in e o Picture Viewer. Eles possibilitam ao usuário executar arquivos QuickTime Movie e visualizar imagens estáticas.

A diferença básica é que quando o usuário está navegando na Internet, o plug in fornece a visualização do conteúdo dentro da janela no navegador e QuickTime Player oferece uma forma de visualizar o conteúdo fora do navegador. O Picture Viewer permite a visualização de imagens paradas. A aparência do Plugin é um pouco diferente do Player.

As interações proporcionadas pelo QuickTime fora do navegador como um todo são executadas pelo aplicativo QuickTime Player. Este aplicativo pode executar filmes, músicas, fotos, entre os cerca de 200 formatos compatíveis, que podem ser executados a partir da Internet, de uma rede local, do HD do computador, de um CD ou um DVD. O QuickTime player pode executar *Internet streams* e *web multicasts* sem necessidade do uso do browser.

2.5.3 H.264

O H.264 é um novo formato de vídeo que oferece uma boa transmissão com taxas de dados muito baixas. Ratificado como parte do padrão MPEG-4 (MPEG-4 Part 10), essa tecnologia oferece bons resultados através de uma grande variedade de largura de banda, de 3G para aparelhos móveis passando pelo iChat AV para videoconferência até HD para transmissões e DVDs (Figura 14).

Figura 14. Resolução H.264.

O H.264 utiliza a tecnologia de compressão de vídeo para oferecer qualidade a partir da menor quantidade de dados de vídeo. Isso significa assistir a vídeos com definição e clareza em arquivos muito menores, poupando largura de banda e custos de armazenamento, em comparação às gerações anteriores de codificadores de vídeo. O H.264 apresenta a mesma qualidade que o MPEG-2, com um terço ou metade de taxa de dados, e até quatro vezes o tamanho do frame do MPEG-4 Part 2, com a mesma taxa de dados.

A tabela 02 mostra a capacidade de transmissão do H.264. Com uma conexão de modem de 56Kbps é possível transmitir um vídeo de qualidade com boa resolução.

Cenário de Uso	Resolução e Taxa de Frame	Exemplo de Taxa de Dados
Conteúdo Móvel	176x144, 10-24 fps	50-60 Kbps
Internet/Definição Padrão	640x480, 24 fps	1-2 Mbps
Alta Definição	1280x720, 24p	5-6 Mbps
Alta Definição em Tela Cheia	1920x1080, 24p	7-8 Mbps

Tabela 02. H.264.

2.5.4 AAC

Por sua qualidade e desempenho , o Advanced Audio Coding (AAC) é utilizado nos formatos MPEG-4, 3GPP e 3GPP2 e é o codificador de áudio para arenas de transmissão digital pela Internet e sem fio.

O AAC foi desenvolvido pelo grupo MPEG que inclui Dolby, Fraunhofer (FhG), AT&T, Sony e Nokia , empresas que também se envolveram na programação de codificadores de áudio, como MP3 e AC3 (também conhecido como Dolby Digital). O codificador AAC no QuickTime carrega uma nova tecnologia de processamento de sinal do Dolby Laboratories e apresenta codificação de dados a uma taxa de dados variável para o QuickTime.

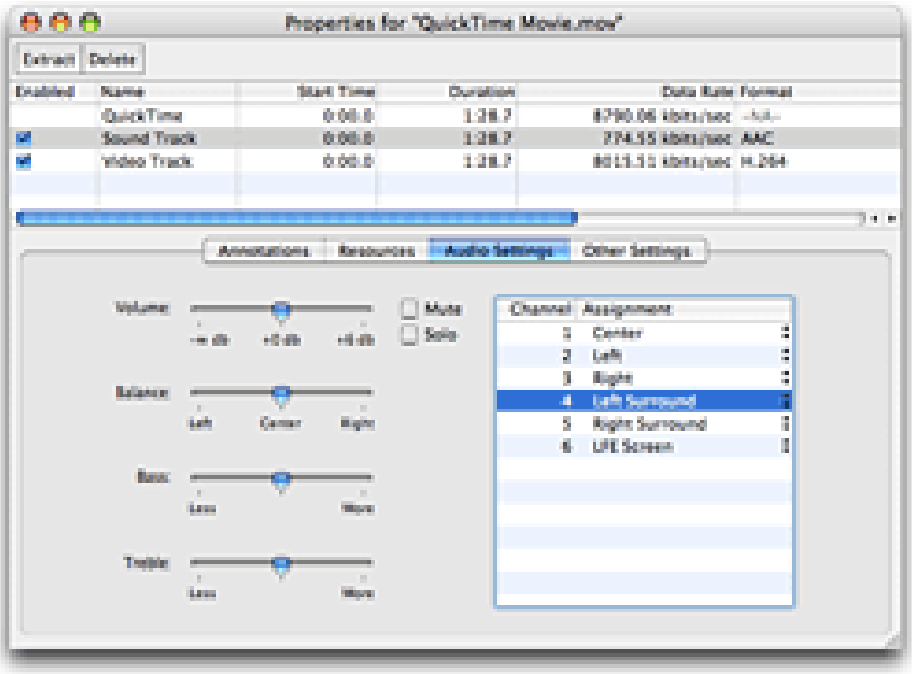

Figura 15. QuickTime StreamServer.

Por sua capacidade em amplas taxas de transmissão de dados e ratificação como padrão industrial, o áudio AAC está sendo adotado por uma grande parcela do mercado. Por exemplo, todas as músicas vendidas na iTunes Music Store usam o formato AAC, para reprodução em desktop ou iPod. O AAC também foi adotado como padrão para grandes organizações, incluindo o ISO MPEG (MPEG-4), 3GPP and 3GPP2 [27]. Como resultado de seu suporte a esta tecnologia o áudio que se cria com o QuickTime é interoperável com uma ampla variedade de aparelhos e players de mídia.

Vantagens do ACC:

a) Compressão aprimorada que oferece resultados de alta qualidade em arquivos menores;
b) Suporte a múltiplos canais de áudio, oferecendo até 48 canais de freqüência;
c) Áudio de alta resolução, fornecendo amostras de taxas de até 96 kHz;
d) Eficiência de decodificação aprimorada, exigindo menor poder de processamento para decodificação.

Por fim este capítulo detalhou os avanços dos sistemas multimídias, que são utilizados pelo servidor Darwin que será mostrado no próximo capitulo.

Capítulo 3

O Servidor Darwin

O Darwin Streaming Server da Apple é um servidor de vídeo de código aberto que pode servir vídeos QuickTime (MOV) e MPEG4 para clientes pela Internet usando os protocolos padrão RTP e RTSP. É baseado no código do antigo QuickTime Streaming Server. O Darwin também suporta transmissão ao vivo, por usar um codificador de vídeo, chamado de mp4live, que é parte integrante do MPEG4IP [5]. Este faz a captura em tempo real do vídeo, gerando um fluxo a ser transmitido por unicast ao servidor Darwin, que fará a distribuição do mesmo usando os protocolos RTP e RTSP.

Na comunidade do Darwin, a maioria dos integrantes são funcionários ou colaboradores da Apple. O projeto encontra-se bastante avançado, mas existem várias lacunas a serem pesquisadas e estudadas. A Apple disponibiliza no site do Darwin toda a documentação interna e externa do código. Embora o Darwin e o QuickTime Streaming Server compartilhem o mesmo código base, eles não compartilham as mesmas funções de interface com o usuário. O QuickTime Server apresenta uma série de avanços como resultado dos serviços disponíveis no servidor Mac OS X.

O QuickTime Server, mesmo tendo sido projetado para o servidor Mac OS X, também está disponível no projeto de código fonte aberto Darwin, que oferece alto nível de personalização para praticamente qualquer rede. Compartilhando o mesmo código base do QuickTime Streaming Server, o Darwin Streaming Server pode ser executado em vários sistemas operacionais incluindo Windows, Linux, FreeBSB e Solaris.

O Darwin é um servidor dirigido por eventos e funciona como um conjunto de processos que executam os protocolos padrões de transmissão de mídia contínua RTSP, RTP e RTCP. O servidor suporta vários formatos de arquivos *streaming* compatíveis e suporta até dois mil ou mais usuários on-line [6] [18].

O Darwin Streaming Server é baseado na linguagem PERL e implementado em C e C++. Não é um sistema muito amigável, pois, por ser baseado em scripts, todo o processo de configuração deve ser realizado através de documentos de texto um tanto complexos.

O Darwin Streaming Server disponibiliza uma aplicação web de gerenciamento como mostra a Figura 16 a seguir. Por meio desta interface, é possível iniciar e parar o serviço *streaming* de vídeo e alterar suas principais configurações. Esta aplicação é acessada através da porta 1220 do servidor. Depois de iniciado, o servidor aguarda por requisições na porta 554, referente ao protocolo de aplicação RTSP [16].

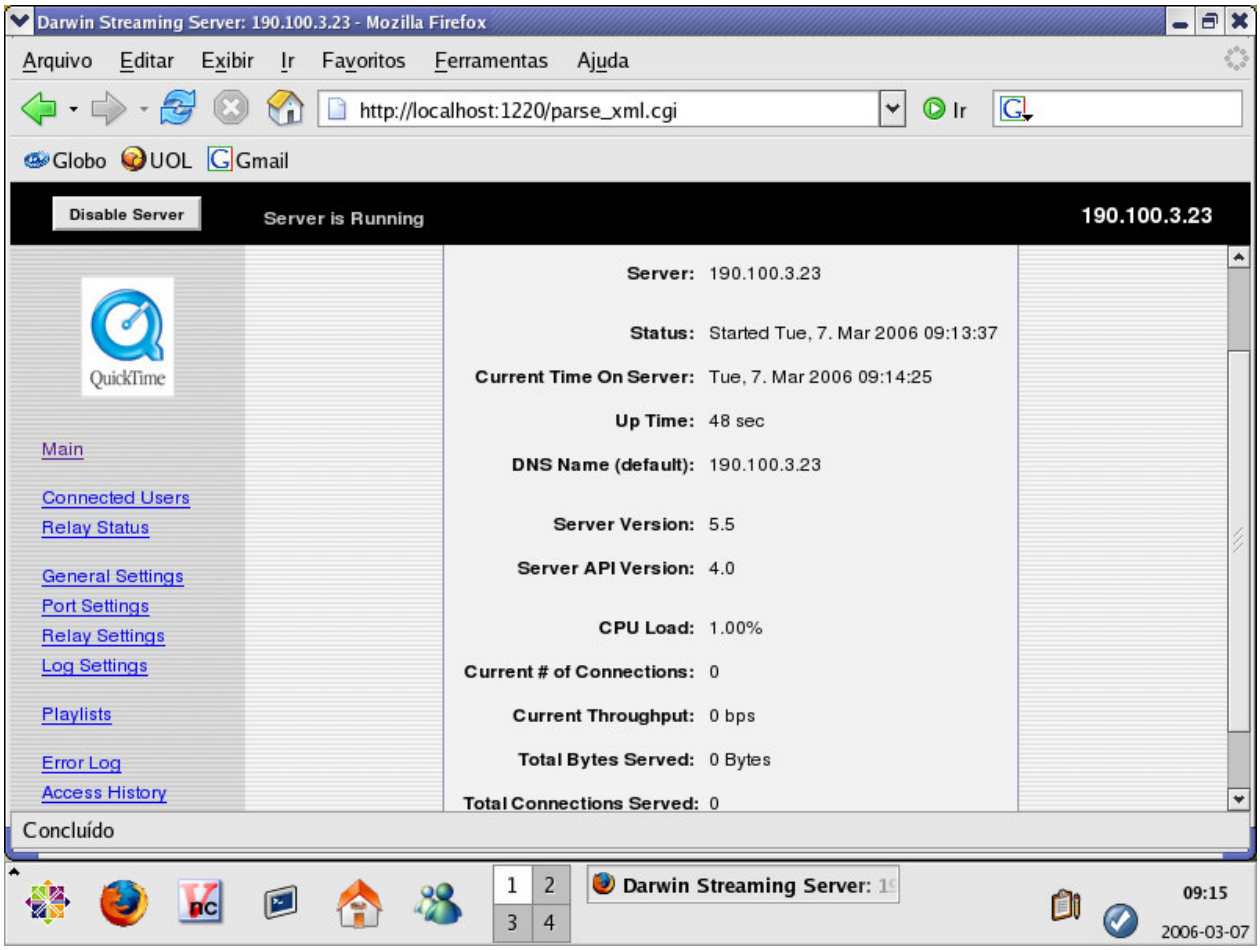

Figura 16. Interface de gerenciamento do Darwin streaming server

Para transmissões de mídia contínua via RTP, é necessário anexar uma *hint track* no início de cada arquivo. A *Hint track* adciona no cabeçalho do arquivo a escala de tempo do RTP e o tamanho máximo de um pacote (*Maximum Transmission Unit* – MTU). É importante ressaltar que as informações contidas na *hint track* podem influenciar diretamente o desempenho do serviço de *streaming*.

3.1 Arquitetura do Servidor

O servidor Darwin consiste em um processo pai que cria um processo filho, que é o núcleo do servidor. O pai espera o processo filho finalizar. Se na saída do processo filho ocorrer um erro, o processo pai cria um novo processo filho.

O núcleo do servidor (*core server*) age como uma interface entre os módulos do servidor e os clientes na rede, que usam os protocolos RTP e RTSP para enviar pedidos e receber respostas (Figura 17). Os módulos do servidor processam pedidos e enviam pacotes ao cliente.
O núcleo do servidor faz seu trabalho criando quatro threads:

A **Thread Principal (Main thread)** gerencia o servidor, verificando se este precisa finalizar, gera um log de status ou imprime estatísticas (Figura 17).

A **Thread de Tarefas Ociosas (Idle Task Thread)** administra uma fila de tarefas que ocorrem periodicamente. Há dois tipos de filas de tarefas: tarefas de timeout e sockets (Figura 17).

A **Thread de Eventos (Event Thread)** escuta os sockets para receber requisições de eventos RTSP ou pacotes RTP e os despachar para as Task Threads (Figura 17).

As **Thread de Tarefas (Task Threads)** recebem as requisições RTSP e RTP da Event Thread. Posteriormente a Task Thread pede ao módulo de servidor apropriado para processar e enviar pacotes ao cliente (Figura 17).

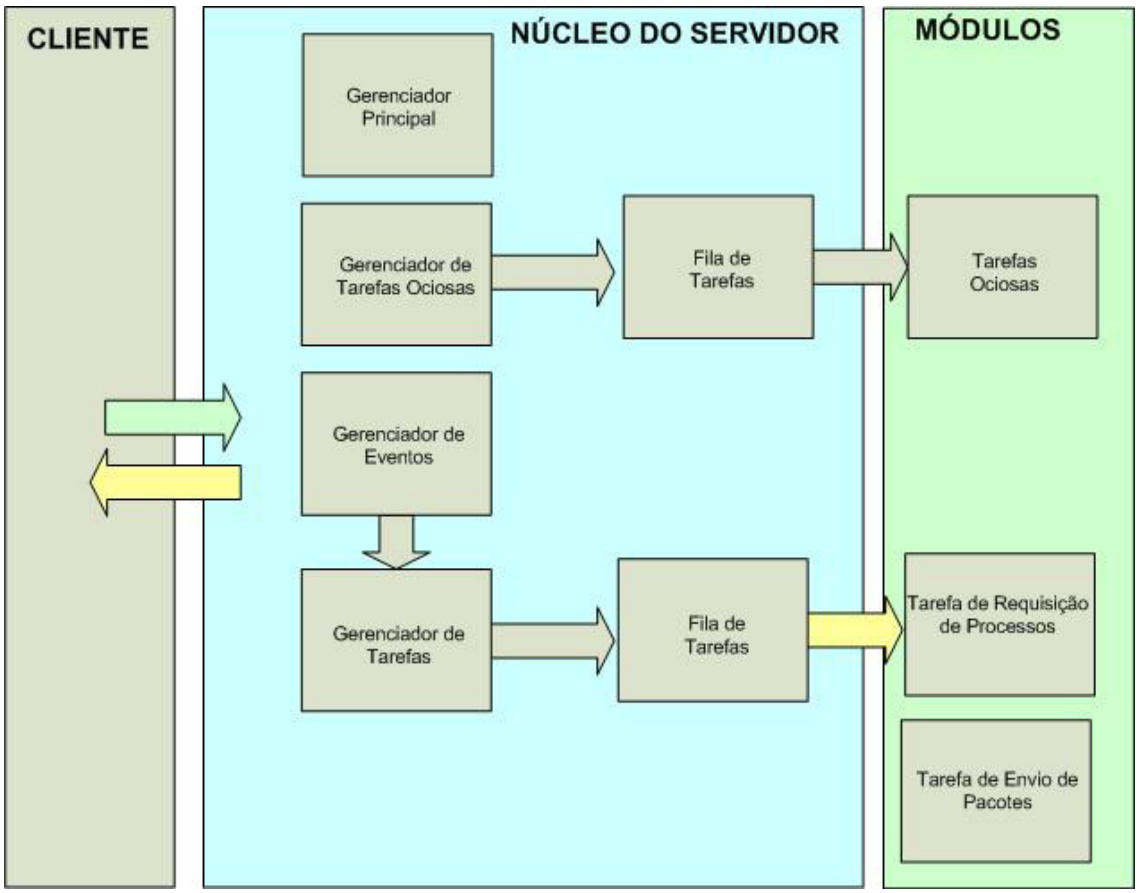

Figura 17. Arquitetura do Servidor.

O servidor Darwin é complexo e trabalha de forma assíncrona, utilizando de um mecanismo de comunicação baseado em eventos. Por exemplo, quando um socket é usado em uma conexão RTP para adquirir dados, algum módulo, precisa ser notificado de forma que os dados possam ser processados [19].

Cada objeto *Task* tem dois métodos principais: *Signal* e *Run*. O método *Signal* é chamado pelo servidor para enviar um evento a um objeto *Task. Run* é chamado a fim de escalonar a *Task* para processar o evento.

A meta de cada objeto *Task* é implementar alguma funcionalidade no servidor usando pequenas fatias de tempo que não são bloqueadas. O *Run* é uma função virtual chamada quando um objeto *Task* tem eventos para processar. Dentro da função *Run*, o objeto *Task* pode chamar *GetEvents* para receber automaticamente todos os pedidos que estejam na fila e previamente sinalizar eventos. A função *Run* nunca é reentrada se um objeto *Task* chama o *GetEvents* dentro da função *Run* e sinaliza antes da função se completar, ou seja, a função *Run* só será chamada para um novo evento depois que ela for encerrada. De fato, a função *Run* será chamada repetidamente até que todos os eventos do objeto *Task* sejam limpos do *GetEvents.*

Este conceito de núcleo de tarefas e eventos ativos é integrado em quase todo subsistema do Servidor Streaming. Objetos Task tornam possível usar o Servidor Streaming em uma única thread, rodando todas as conexões default em um único processo do sistema.

3.2 Módulos

O Servidor Streaming usa módulos para responder pedidos e tarefas completas. Há três tipos de módulos:

a) Modulo de Administração de Conteúdo (*Content-Managing*): Administra as requisições e as respostas do RTSP relacionados às fontes de mídia, como um arquivo ou um broadcast. Este módulo é responsável por interpretar o pedido do cliente, lendo e analisando os arquivos suportados ou os destinos da rede, e respondendo via RTSP e RTP. Em alguns casos, o módulo mp3 *streaming* usa HTTP.

Os submódulos do *content-managing* são QTSSFileModule (Controla os Arquivos), QTSSReflectorModule (Controla toda parte Gerencial), QTSSRelayModule (Controla os Atrasos), e QTSSMP3StreamingModule (Controla o fluxo do Mp3) [ANEXO I].

b) Módulo Suporte do Servidor (Server-Support): É responsável pelo controle de desempenho dos dados e pelas funções de agrupamento e autenticação no servidor.

Os submódulos do Server-Support são QTSSErrorLogModule (Registra os Logs de erros), QTSSAccessLogModule (Registra todos os Acessos), QTSSWebStatsModule (Registra estados do servidor), QTSSWebDebugModule (Modo Debug via Web), QTSSAdminModule (Controle do Administrador), e QTSSPOSIXFileSystemModule (Registra a parte de Controle) [ANEXO I].

c) Módulo Controle de Acesso (*Access Control*): Provê autenticação e autorização de funções, como também trata o caminho da URL.

Os submódulos do access control são QTSSAccessModule (Responsável pelo controle de acesso), QTSSHomeDirectoryModule (Responsável pelo diretório principal), QTSSHttpFileModule (Responsável pela pasta de arquivos Http), e QTSSSpamDefenseModule (Defensor de Spams) [ANEXO I].

3.3 Classes do Servidor

No Servidor Dawin existem quatro classes, que são responsáveis por funções básicas como interfaces e requisições de conexão.

As principais classes do Darwin são:

a) **Interface do Servidor (*QTSServerInterface*)** – Responsável pela interface do servidor utilizando o objeto QTSS_ServerObject;

b) **Interface da Sessão RTSP (*RTSPSessionInteface*)** – Responsável pela interface da sessão RTSP, utilizando o objeto QTSSRTSPSessionObjectType;

c) **Interface da Sessão RTP (*RTPSessionInterface*) -** Responsável pela interface da sessão RTP, utilizando o objeto QTSS_Client SessionObject;

d) **Interface de Requisição RTSP (*RTSPRequestInterface*) -** Responsável pela interface da requisição RTSP, utilizando o objeto QTSS_RTSP RequestObject.

Como mostra a figura 18, o objeto *serve*r tem um dicionário de preferências (Preferências do Servidor). O servidor possui uma lista de módulos, cada um com dicionário para as suas preferências. O servidor possui uma lista de sessões de clientes RTP, cada cliente pode ter uma ou mais sessões RTSP ou um ou mais fluxos RTP.

O QTServer é um objeto do núcleo do servidor, que é acessado pela API e a classe base dele é QTSServerInterface na classe Servidor.

O Dicionário de preferências é um armazenamento de dados de uma classe básica que implementa uma chave e acessa dados dos objetos . Esta é a classe base que é herdada por todos os objetos do servidor definidos pela API.

Módulos é uma classe que controla os módulos. Cada instância do módulo é responsável por carregar, inicializar, e executar um módulo estático ou dinâmico da API.

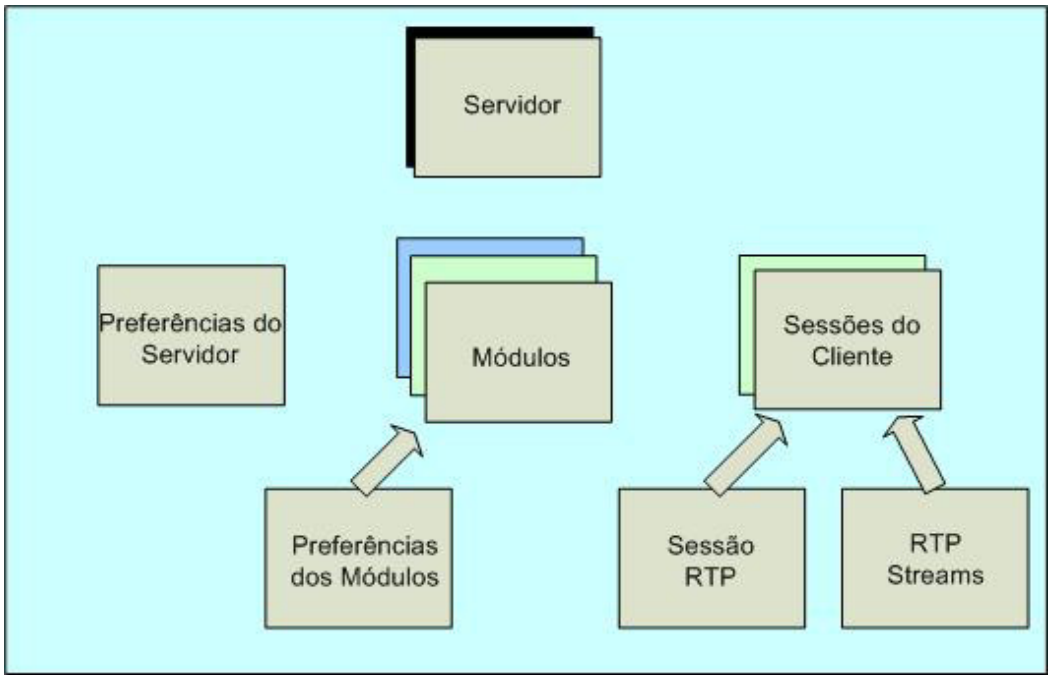

Figura 18. Modelos dos Objetos de Dados do Servidor.

3.4 Funcionamento do Servidor

O Darwin Streaming Server trabalha com módulos para processar pedidos de clientes invocando módulos com um papel particular. Cada papel é projetado para executar uma tarefa em particular. Esta seção descreve como o servidor trabalha com papéis quando se inicia e se desativa e como o servidor trabalha com papéis quando processa pedidos de cliente.

3.4.1 Inicialização e Término do Servidor

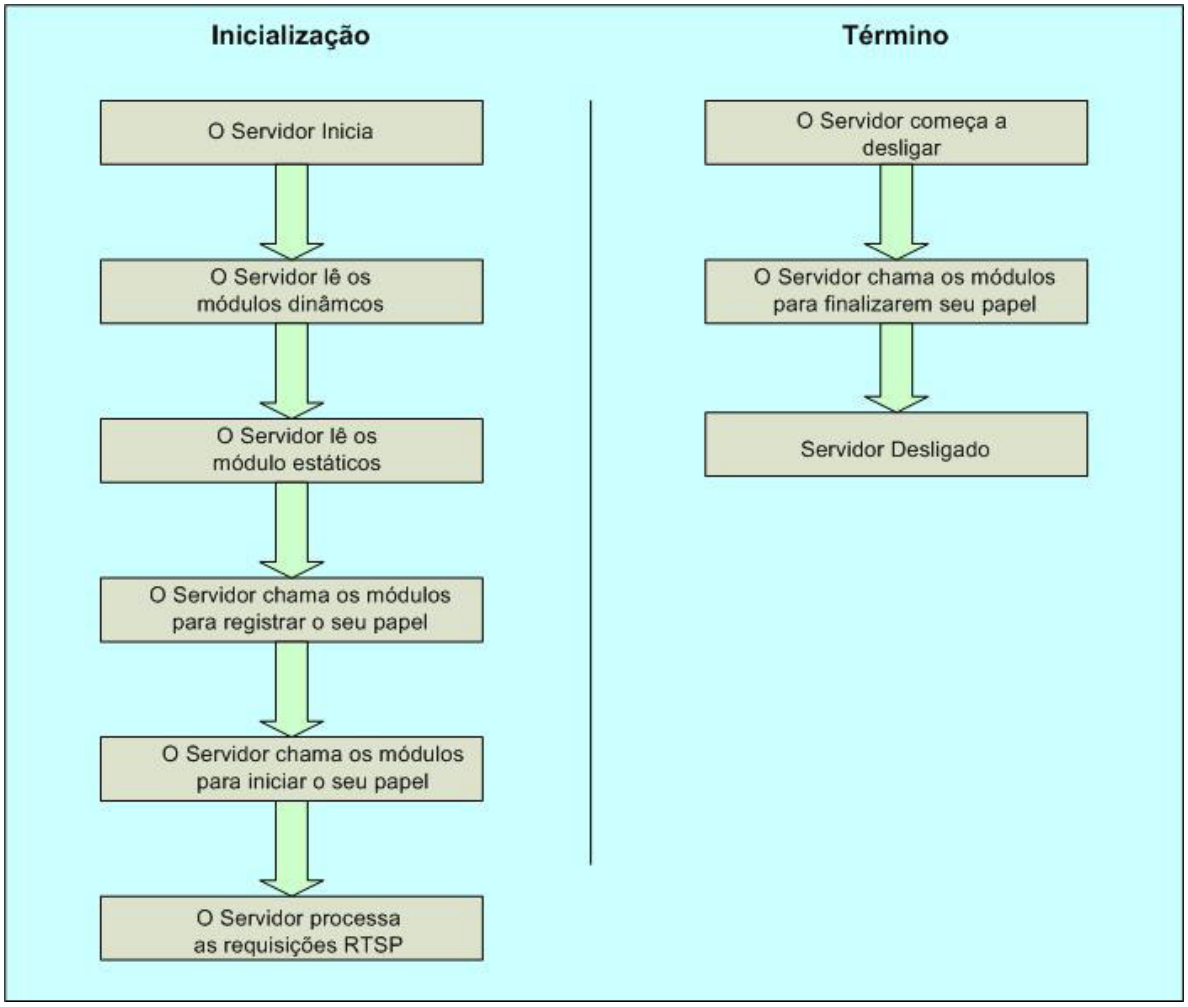

Figura 19. Inicializar e Desativar o Servidor.

Quando o servidor inicia, primeiro ele carrega módulos que não são compilados no servidor (módulos dinâmicos) e então carrega módulos que são compilados no servidor (módulos estáticos). Então o servidor invoca cada módulo do QTSS com a funcionalidade de Registro (role), que é um papel que todo módulo tem que suportar. Então o módulo chama QTSS_AddRole para especificar os outros papéis que ele pode suportar.

Depois, o servidor invoca o papel Initialize para cada módulo que foi registrado naquele papel. O Initialize executa qualquer tarefa de inicialização que o módulo requer, alocando memória e inicializando estruturas de dados globais.

Quando o servidor desativa-se, invoca o papel *shutdown* para cada módulo que foi registrado naquele papel. Quando executa um *shutdown*, o módulo deve executar uma limpeza e liberar estruturas globais.

O Darwin *Streaming Server* opera fazendo um *broadcast* do conteúdo a todos os clientes que solicitaram o arquivo, ou servindo sob demanda. Quando o cliente solicita o

arquivo, o número dos pacotes emitidos a cada cliente depende do tempo relativo ao começo da transmissão (broadcast) e do momento em que o cliente estabeleceu a conexão com o servidor [17]. Cada pedido de um cliente recebe uma transmissão completa do arquivo [8].

O servidor possui uma arquitetura modular que facilita a criação de novas funcionalidades por meio de módulos independentes. O servidor possui um processo principal, chamado de *Core Server*, que é, na verdade, uma interface entre as requisições dos clientes e os módulos. Os módulos devem ser constituídos por uma estrutura específica, obrigatoriamente possuindo dois métodos, um chamado pelo servidor para iniciá-lo e outro chamado pelo servidor para executar uma determinada tarefa. Para saber que módulos devem ser chamados para um determinado evento, cada módulo deve explicitar seu papel. Sendo assim, cada módulo tem que possuir uma lista de papéis, denominados “Roles”.

O Darwin Streaming Server, portanto, trabalha com módulos para processar pedidos de clientes, invocando-os através de papéis. O servidor trabalha com Papel para ser iniciado e desativado e para processar pedidos de cliente.

3.5 Processamento de Requisições RTSP

Depois que o servidor chama cada módulo que fez registro no papel Inicialize, o servidor está pronto para receber requisições de clientes. Estes pedidos são feitos por meio de requisições RTSP.

Uma amostra de requisição RTSP:

```
DESCRIBE rtsp://streaming.site.com/foo.mov RTSP/1.0
CSeq: 1
Accept: application/sdp
User-agent: QTS/1.0
```

Figura 20. Amostra de requisição RTSP.

A figura 20 descreve uma requisição de um vídeo via RTSP, quando o servidor recebe uma requisição RTSP, ele cria um objeto para a requisição RTSP, que é uma coleção de atributos que descrevem o pedido. Neste momento, o qtssRTSPReqFullRequest attribute é o único atributo que tem um valor, e este valor consiste nos conteúdos completos da requisição RTSP.

Em seguida, o servidor chama módulos com papéis específicos de acordo com uma sucessão pré-determinada.

A figura 21 a seguir mostra toda a seqüência:

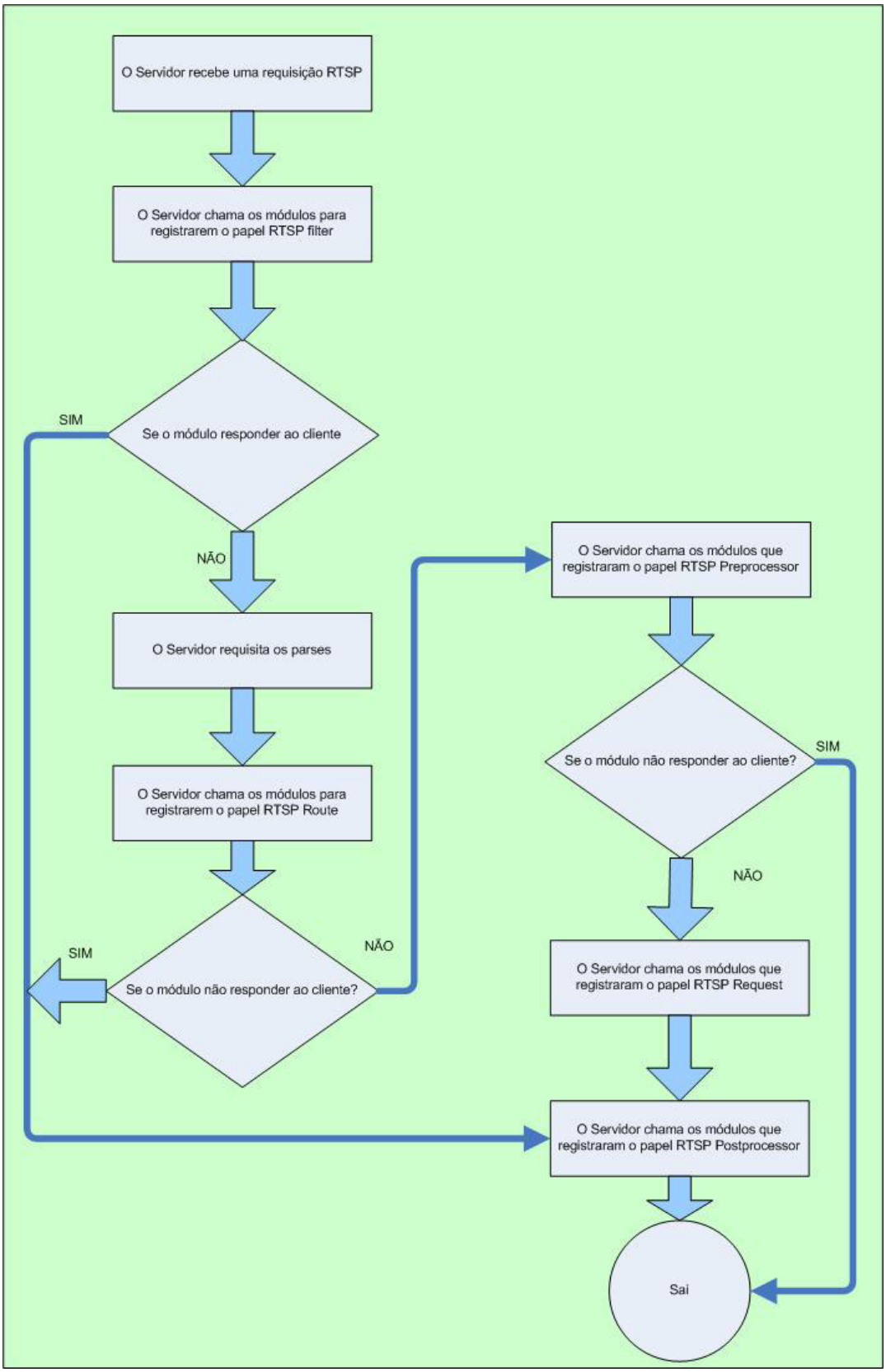

Figura 21. Resumo do processo de pedido RTSP.

Quando é processada uma requisição RTSP, o primeiro papel que o servidor chama é o RTSP Filter, O servidor chama cada módulo que registrou o papel para RTSP Filter e os analisadores XML para a requisição RTSP. Para cada módulo com papel RTSP Filter, tem-se a opção de mudar o valor do atributo qtssRTSPReqFullRequest. Por exemplo, um papel RTSP Filter poderia mudar /musica/musica.mov para /audio/audio.mov, mudando a pasta e o nome do arquivo que será usado para satisfazer este pedido.

Qualquer módulo que assume o papel RTSP Filter e responde às solicitações do cliente, o servidor passa para outros módulos que registraram o papel RTSP Filter, e passa para módulos que registraram outros papéis de RTSP, e imediatamente o servidor chama o papel RTSP Postprocessor para o módulo que está respondendo. Uma resposta para um cliente é definida como qualquer tipo de dado que o módulo pode enviar ao cliente.

Quando todos os papéis de RTSP *Filter* forem invocados, o servidor envia os pedidos para serem processados. O *parsing* de requisições consiste em vários atributos do objeto RTSP e cria duas sessões:

a) Uma sessão RTSP, que inicia quando há um pedido particular do cliente e fecha quando o cliente encerra sua conexão RTSP com o servidor.

b) Uma sessão cliente que é associada com a conexão do cliente que originou o pedido e o cliente que está transmitindo um fluxo.

Depois da requisição do *parser*, O servidor chama o papel RTSP *route* para cada módulo que registrou aquele papel e passa o objeto RTSP. Cada papel RTSP *Route* tem a opção de usar valores de certos atributos para determinar se é preciso mudar o valor do atributo do qtssRTSPReqRootDir, assim muda a pasta que é usada para processar este pedido. Por exemplo, se o tipo de idioma é inglês, o módulo poderia mudar o qtssRTSPReqRootDir para atribuir a uma pasta de papéis que contém a versão inglesa do arquivo pedido.

Depois que o papel RTSP Route foi chamado, o servidor chama o papel RTSP PreProcessor para cada módulo que registrou-se para aquele papel. O papel RTSP PreProcessor tipicamente usa o atributo qtssRTSPReqAbsoluteURL para determinar se o pedido é igual ao tipo de pedido processado no módulo.

Se o pedido for processado, o papel RTSP PreProcessor responde ao pedido chamando QTSS_Write ou QTSS_WriteV para enviar dados ao cliente. É enviado uma resposta standard, o módulo pode chamar QTSS_SendStandardRTSPResponse, ou QTSS_AppendRTSPHeader e QTSS_SendRTSPHeaders.

Se nenhum papel RTSP Preprocessor responde o RTSP request, o servidor invoca o papel RTSP request do módulo que teve sucesso no registro desse papel. O papel RTSP request é responsável por responder todas as requisições RTSP, e não é manipulado por módulos registrados no papel RTSP PreProcessor.

Depois que o papel RTSP Request processa o pedido, o servidor chama módulos que registraram o papel RTSP Postprocessor. O papel RTSP Postprocessor tipicamente executa contabilidade das tarefas, e anota a informação estatística.

Um módulo que manipula o RTSP Preprocessor ou o papel RTSP Request pode gerar os dados de mídia para uma sessão de um cliente em particular. Para gerar dados de mídia, o módulo chama QTSS_Play que chama o módulo a ser invocado no papel RTP Send Packets (figura 22).

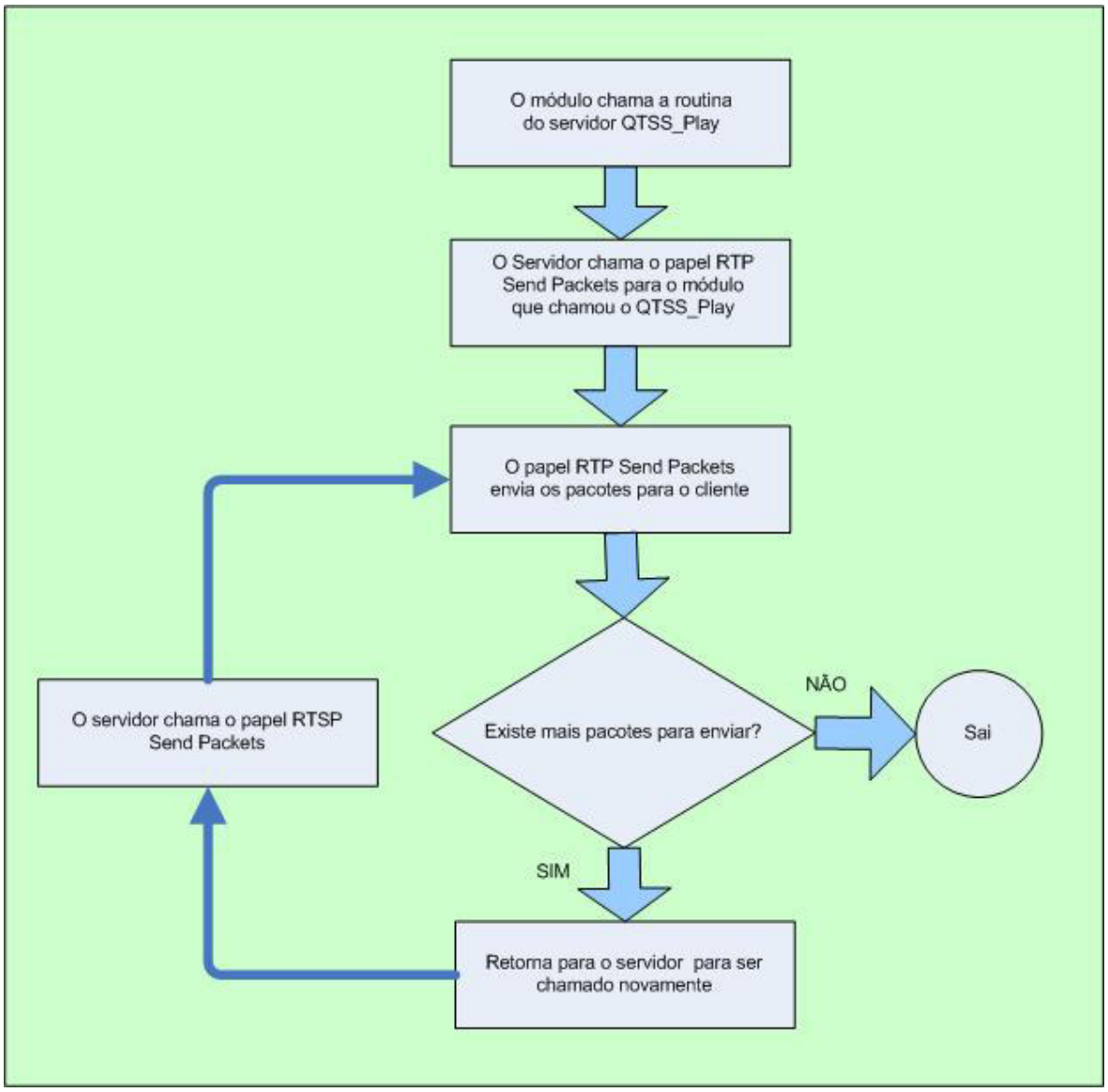

Figura 22. Resumo do RTSP Preprocessor e dos papéis RTSP Request.

O papel RTP Send Packets chama QTSS_Write ou QTSS_WriteV para enviar dados ao cliente sobre a sessão RTP. Quando o papel RTP Send Packets enviou alguns pacotes, ele devolve ao servidor e especifica o tempo que vai demorar antes que o servidor chame o módulo RTP Send Packets. Este ciclo se repete até que todos os pacotes de mídia sejam enviados ou até que os pedidos das sessões dos clientes abertos terminem.

Capítulo 4

Monitoramento do Servidor Streaming

4.1 Módulo Monitor

O servidor Darwin, no estágio atual de desenvolvimento, não possui um módulo de monitoramento na sua arquitetura. O seu *kernel* calcula o fluxo de saída de *streaming*, mas isso é uma função totalmente transparente para o usuário.

Neste trabalho, desenvolvemos um **módulo de monitoramento** responsável por obter o uso da banda e a taxa de perda de pacotes para cada sessão de usuário.

O Módulo Monitor interage diretamente com o kernel do Servidor Darwin (Figura 23). Ele faz uma coleta dos dados no núcleo do servidor, os quais são processados on-line nos atributos de um objeto do servidor.

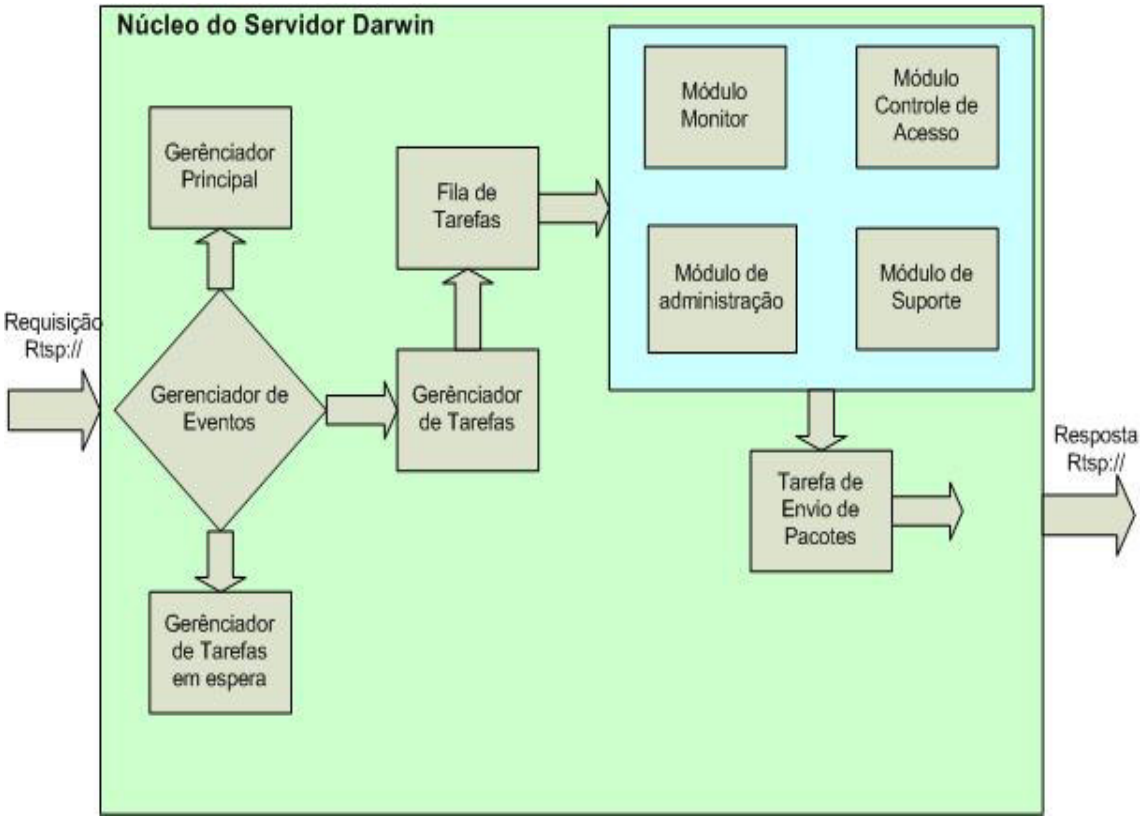

Figura 23. Diagrama de Blocos com o Módulo Monitor.

O Servidor Darwin recebe uma requisição RTSP, essa requisição vai para o gerenciador de eventos, dependendo da quantidade de processos o gerenciador repassa a requisição para o gerenciador principal ou uma fila de tarefas. Quando a requisição é liberada ela vai para o gerenciador de tarefas, esse gerenciador de tarefas envia a requisição para uma fila de tarefas. O Gerenciador Principal, faz os controles de todos os gerenciadores e de logs.

Após a requisição chegar à fila de tarefas, ela é processada pelos módulos, onde existe o módulo de administração que processa a requisição e a resposta RTSP, ele interpreta o pedido do cliente lendo e analisando os arquivos e os destinos de resposta na rede. Após analisada e processada a requisição, ela é enviada à Tarefa de Envio de Pacotes, onde é enviada ao cliente via RTSP. O Módulo Monitor trabalha junto a todos os módulos coletando os atributos diretamente no núcleo do servidor.

Esses atributos trabalham diretamente com a estrutura do servidor, calculando constantemente várias informações necessárias para avaliar o seu desempenho [10] [17] [21]. Essas informações ficam ocultas dentro do próprio Darwin, o Módulo Monitor foi implementado para coletar essas informações e criar uma interface para a visualização dos dados.

O Módulo Monitor trabalha da seguinte forma: existe uma classe no servidor chamada QTSServer (Figura 24) que controla todas as funções básicas, é a classe principal. Toda vez que o servidor entra em utilização, essa classe ativa a QTSSMonitorModule, que coleta todas as informações da classe QTSS_ServerObject, que está funcionando dentro do núcleo do Servidor. Essa classe QTTS_ServerObject é responsável por calcular vários atributos, dentre eles as taxas de transferência e perda de pacotes.

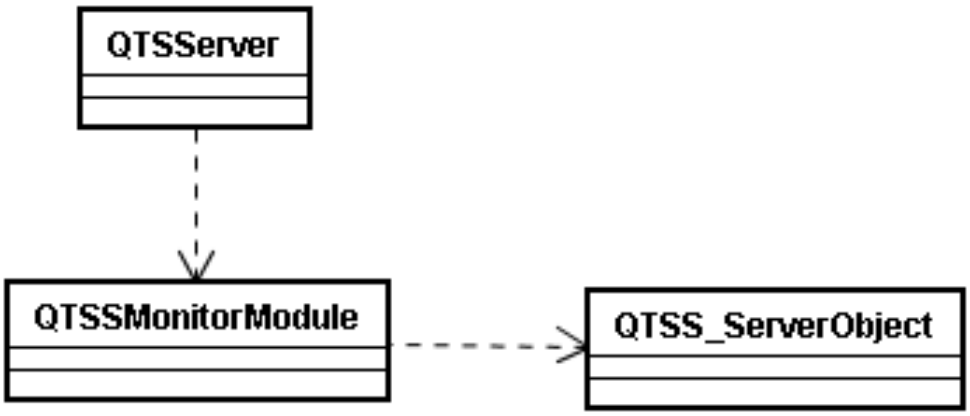

Figura 24. Diagrama de Classe do Módulo Monitor.

O Módulo Monitor coleta as informações deste objeto e salva em um arquivo texto para ser utilizado por outro programa desenvolvido em Java.

O Módulo Monitor implementa duas rotinas básicas: a Main Routine e a Dispach Routine.

Na Main Routine, todos os módulos QTSS têm que prover uma rotina Main. O servidor chama a rotina Main para ser iniciado e usa o papel Initialize para o QTSS *stub library* (carrega as bibliotecas), assim o servidor pode invocar o módulo posteriormente. Para módulos que são compilados no servidor, o endereço da rotina principal do módulo deve ser passado à rotina de inicialização de módulo do servidor.

O corpo da Main Routine é:

```
QTSS_Error QTSSMonitorModule_Main(void* inPrivateArgs)
      {
        return_stublibrary_main(inPrivateArgs,
QTSSMonitorModuleDispatch);
      }
```

QTSSMonitorModuleDispatch é o nome da rotina de despacho do módulo na qual é aberta mais uma sessão do cliente.

Na Dispatch Routine, todo módulo QTSS tem que prover uma rotina Dispatch.O servidor chama a rotina de dispatch quando invoca um módulo para uma tarefa específica, passando à rotina de dispatch o nome da tarefa e um bloco de parâmetros específicos para a tarefa.

O protótipo da Dispach Routine do Módulo Monitor implementado é:

```
static QTSS_Error QTSSMonitorModuleDispatch(QTSS_Role inRole,
QTSS_RoleParamPtr inParams);
```

QTSSMonitorModuleDispatch é o nome especificado como o nome da rotina de dispatch pelo módulo de rotina Main. O parâmetro inRole é o nome do papel para o qual o módulo está sendo chamado e inParams é uma estrutura que contém valores de interesse para o módulo.

O módulo Monitor implementa três regras:

- QTSS_Register_Role;
- QTSS_Initialize_Role;
- QTSS_RTSPFilter_Role.

Quando a regra QTSS_Register_Role é invocada pelo servidor, o módulo registra as regras que ele deseja que sejam notificadas, as quais são QTSS_Initialize_Role e QTSS_RTSP Filter_Role.

Quando é invocada a QTSS_Initialize_Role, o módulo realiza as inicializações dos objetos globais, por exemplo: obter a referência ao QTSS_ServerObject (objeto que representa o servidor). E quando o servidor invoca a regra QTSS_RTSPFilter_Role, o módulo realiza o processamento da requisição.

Quando o módulo Monitor é notificado através da regra QTSS_RTSPFilter_Role, ele obtém os valores da taxa de transferência total do servidor no atributo qtssRTPSvrCurBandwidth, do objeto QTSS_ServerObject. Em seguida, para cada sessão de cliente, obtém a taxa de transferência no atributo qtssCliSesCurrentBitRate, bem como o endereço do cliente no atributo qtssCliRTSPSessRemoteAddrStr e a quantidade de pacotes perdidos, no atributo qtssCliSesPacketLossPercent do objeto que representa a sessão de cada cliente. Posteriormente, coleta os cálculos de uso da banda e armazena os resultados em um arquivo.

4.2 Visualização dos Resultados

Também desenvolvemos um programa em Java para ler o arquivo de resultados e representá-los de forma gráfica. Existem dois modos de visualização, um que mostra a largura da banda e outro que mostra a perda de pacotes.

O programa desenvolvido em Java coleta todas as conexões ativas no momento, existe uma thread responsável por verificar cada conexão ativa. Essa Thread fica lendo um arquivo de texto, esse arquivo de texto é atualizado a cada segundo pelo núcleo do Servidor Darwin, esse arquivo contém a velocidade dos pacotes, o endereço Ip e a quantidade de pacotes perdidos. Esse arquivo tem um tamanho variado, pois depende do numero de conexões e a quantidade de tráfego acessado.

Esse procedimento é feito em tempo real, para poder visualizar o conteúdo através de frames no programa.

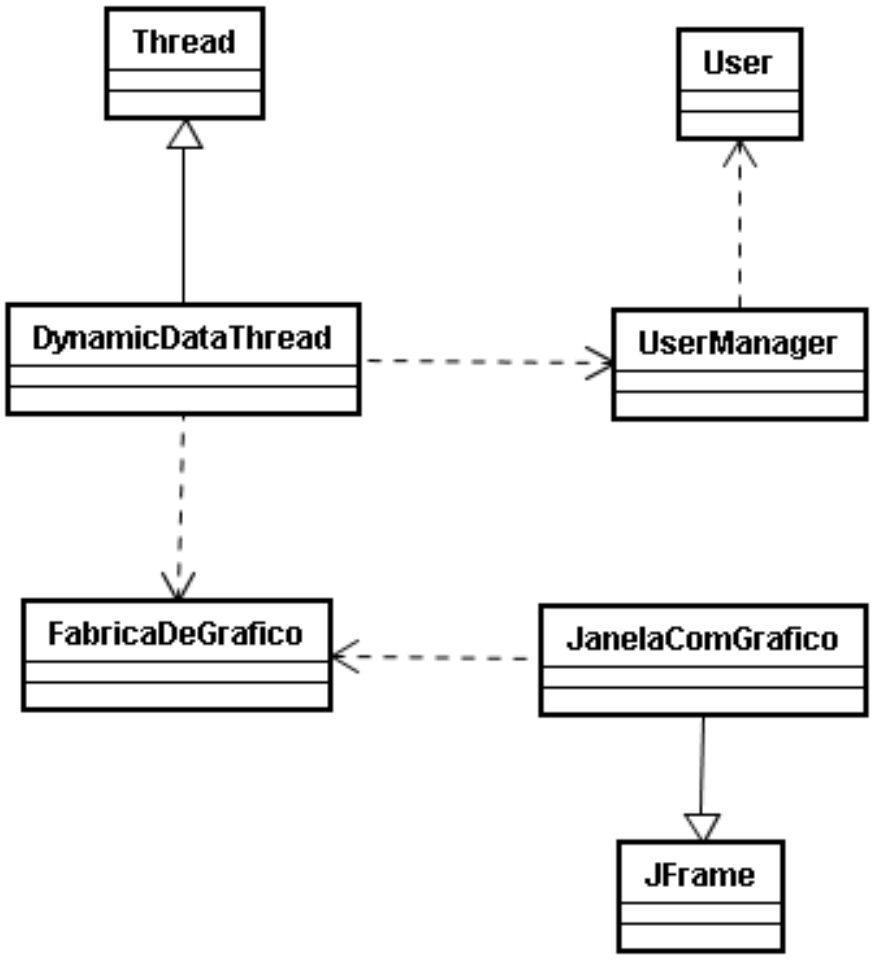

Figura 25. Diagrama de Classe do Programa Gerador Gráfico.

Após ler o arquivo, a classe DynamicDataThread (Figura 25) envia os dados para serem processados pela classe FabricaDeGrafico, essa classe cria todos os gráficos com os dados obtidos do núcleo do servidor. Após criados esses gráficos, são coletados

pela classe JanelaComGrafico, essa classe trabalha diretamente com a classe JFrame que mostra os frames variando on-line conforme a velocidade de transmissão. Os frames são dinâmicos e mudam a cada segundo, eles mudam conforme os dados coletados no arquivo de texto gerado pelo núcleo do servidor.

Após o gráfico ser gerado, ele volta para a classe DynamicDataThread, esta envia para uma classe chamada UserManager que por fim envia para visualização na tela, para que o monitoramento possa ser acompanhado em tempo real pelo o usuário.

Todo esse processo mostra para o usuário de forma clara e explicativa, a velocidade de transmissão do fluxo do Servidor Darwin e a taxa de perda de pacotes, por sessão de usuário. Os resultados obtidos estão detalhados no próximo capítulo.

Capítulo 5

Experimentos e Resultados

5.1 Cenário dos Experimentos

Utilizando-se o Módulo Monitor aqui desenvolvido, foram conduzidos experimentos com o servidor Darwin em uma rede LAN de 100 MBps com mais de 20 computadores. Todas as máquinas acessaram um único servidor Darwin de vídeo (Figura 26). A rede estruturada continha uma conexão wireless 54 Mbps, que era acessada por notebooks. Os computadores e notebooks acessaram simultaneamente o servidor Darwin, reproduzindo dois formatos de vídeos diferentes.

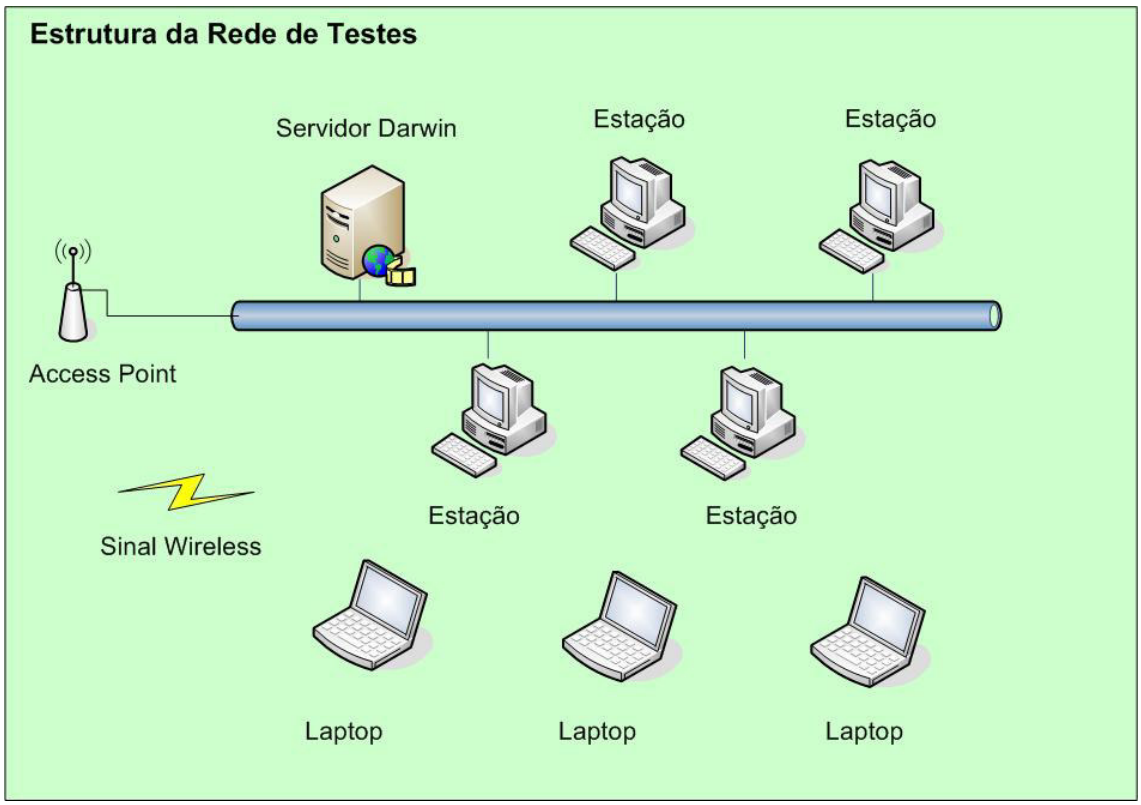

Figura 26. Estrutura da Rede.

Nessa fase dos experimentos, o Módulo Monitor implementado foi testado com sucesso, pois o mesmo coletou as informações dentro do núcleo do servidor e gerou o gráfico detalhado das conexões. Os experimentos foram divididos em duas etapas.

Os testes tiveram seis meses de duração, todos os arquivos testados foram estressados e várias vezes acessados, para que os testes de monitoramento comprovassem de forma exata a grande diferença de transmissão dos dois formatos.

5.2 Transmissão de Arquivos MPEG-4

Na primeira etapa foram utilizados arquivos de vídeo no formato MP4, com tamanho até 50 MB. Observou-se que o servidor Darwin tenta manter os arquivos sempre com a mesma taxa de saída de dados, mas com isso o servidor acabou deixando pouca banda para cada sessão e deixou de usar uma enorme banda que ficou ociosa, e que poderia ser escalonada para acelerar a transmissão [20].

Na reprodução dos vídeos nos clientes, observou-se um excelente desempenho e ótima qualidade de imagem, sendo utilizado o QuickTime Player [7] e o protocolo RTP para acessar o servidor. Para que os arquivos MP4 possam ser servidos via streaming, é preciso adicionar uma hint track, uma adição de marcas de tempo no cabeçalho do pacote. O programa selecionado para este fim foi o mpeg4IP, um programa open source que, além de adicionar a hint track, faz transmissão ao vivo convertendo um vídeo capturado em MPEG-4 [21].

A Figura 27 mostra o gráfico de monitoramento do servidor Darwin com arquivos de vídeo MP4. É importante ressaltar que o servidor está utilizando apenas 15% de toda sua capacidade de banda disponível, de modo que a taxa de saída total de dados é de 15 MBps. O servidor Darwin, portanto, deixa 85% da banda totalmente livre e sem utilização.

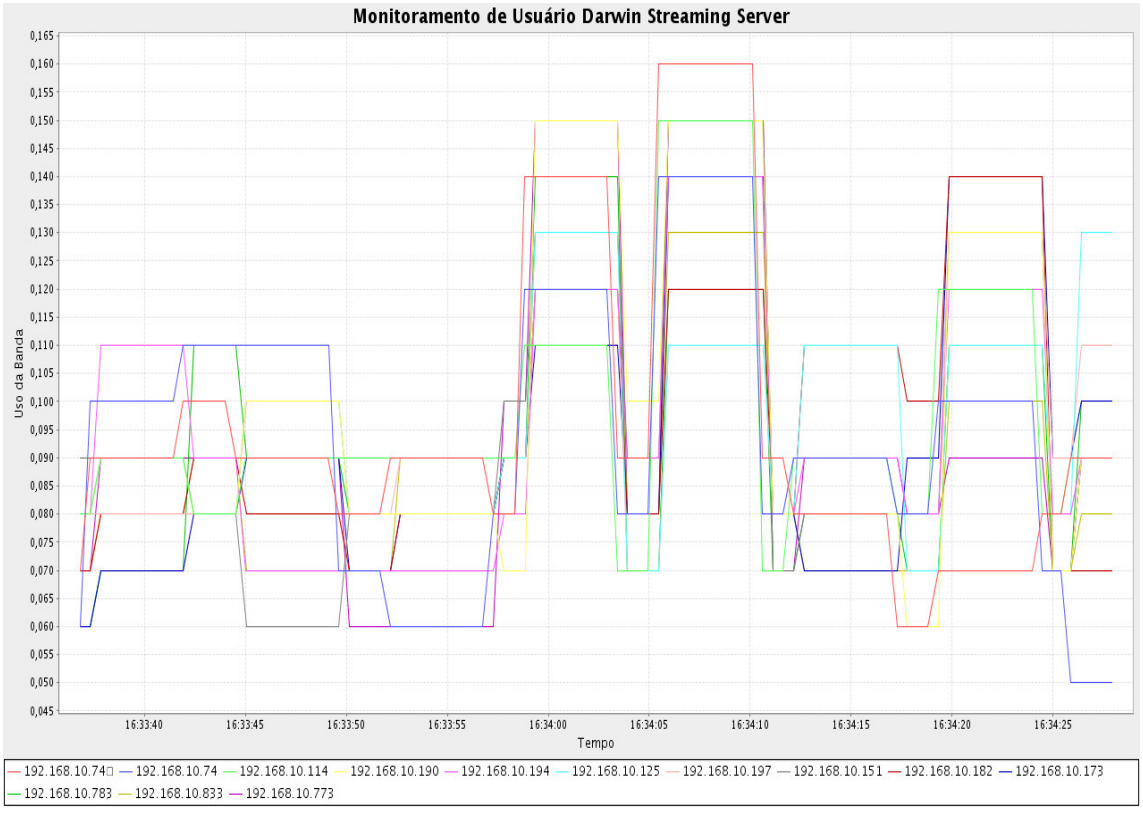

Figura 27. Monitoramento da transmissão de arquivos MP4.

A Figura 28 mostra o gráfico de monitoramento do servidor Darwin também com arquivos de vídeo MP4, nesse gráfico foram utilizados arquivos maiores, com 90

MB, são arquivos com qualidade superior e foi observado que a banda de saída do servidor foi reduzida para 12% e ficando com 88% de banda totalmente livre.

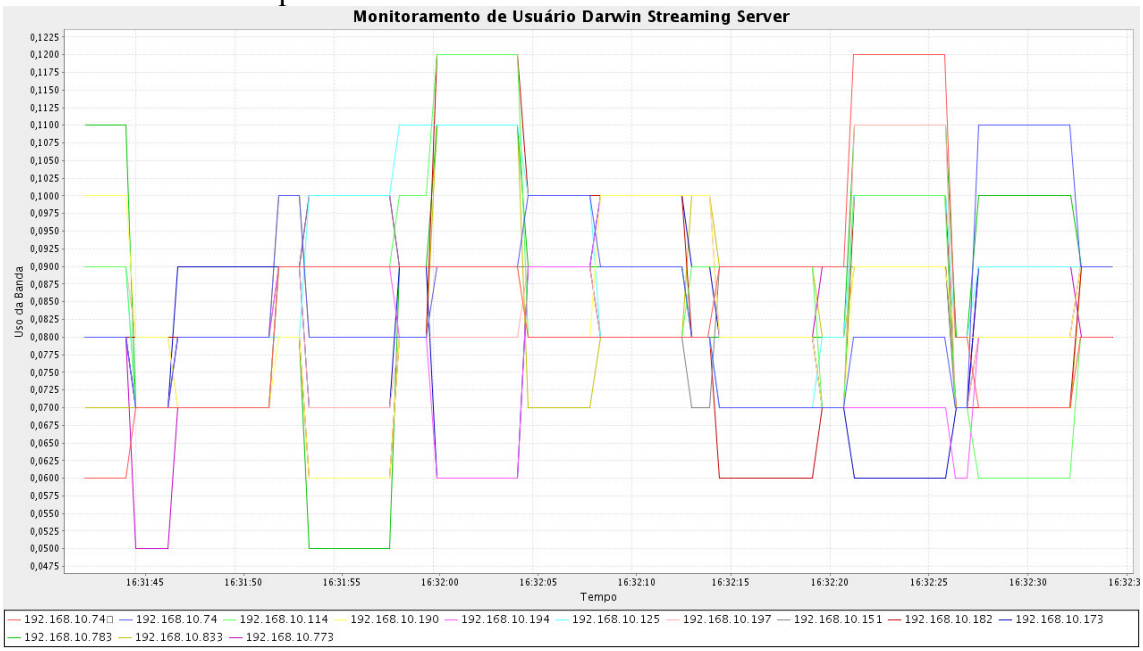

Figura 28. Monitoramento da transmissão de arquivos MP4 com arquivos 90 MB.

A Figura 29 mostra o gráfico de monitoramento do servidor Darwin também com arquivos de vídeo MP4, nesse gráfico foram utilizados arquivos menores, com 20 MB, são arquivos com qualidade inferior e foi observado que a banda de saída do servidor foi aumentada para 15% e ficando com 85% de banda totalmente livre. Observa-se que a variação da taxa de transmissão é mais longa, nos arquivos maiores essa variação é muito maior (Figura 28), o intervalo de tempo é mais curto.

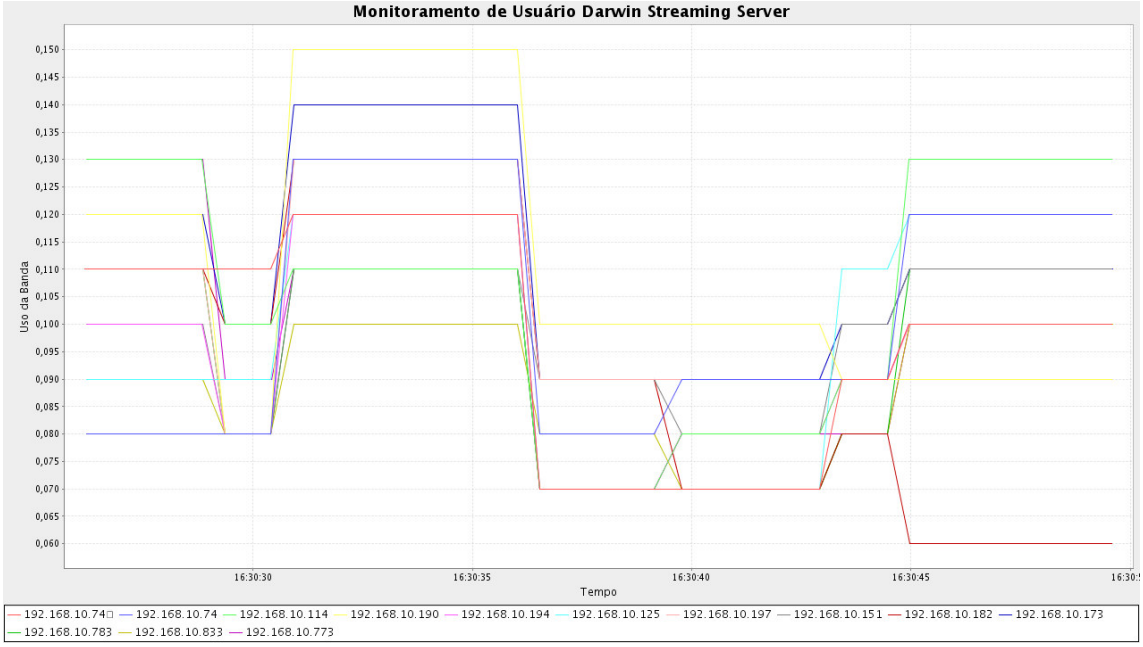

Figura 29. Monitoramento da transmissão de arquivos MP4 com arquivos de 20 MB.

5.3 Transmissão de Arquivos MOV

Na segunda etapa dos experimentos, foram utilizados arquivos de vídeo no formato MOV, com até 60 MB. Observou-se que o servidor Darwin trata os arquivos MOV de forma diferenciada, já que nem todos os arquivos apresentam a mesma taxa de saída de dados, pois, dependendo da qualidade de cada arquivo de vídeo, o servidor disponibiliza maior ou menor banda para a sessão.

O formatador utilizado foi novamente o QuickTime, com o protocolo RTP para acessar o servidor. O próprio QuickTime faz adição da hint track aos arquivos MOV para funcionar no servidor streaming, mas essa operação não é automática, sendo necessário fazer uma exportação do arquivo MOV no formato *hinted track*.

A Figura 30 mostra o gráfico de monitoramento do servidor Darwin com arquivos de vídeo MOV. Diferentemente da etapa anterior, com arquivos MP4, o servidor Darwin utiliza desta vez 90% da banda disponível, alcançando uma taxa total de saída de dados de 90 MBps, como se pode observar no gráfico. Em alguns momentos, alguns vídeos ocupam toda a banda disponível no servidor (Figura 31).

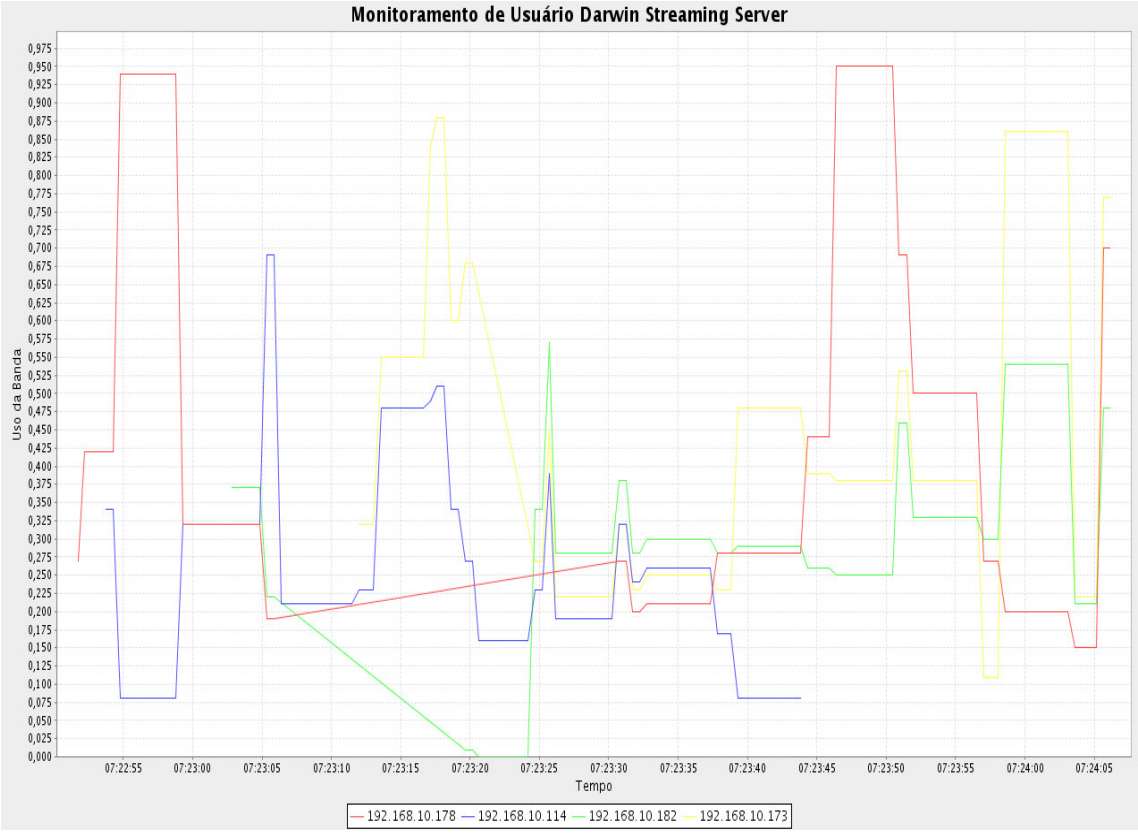

Figura 30. Monitoramento da transmissão de arquivos MOV.

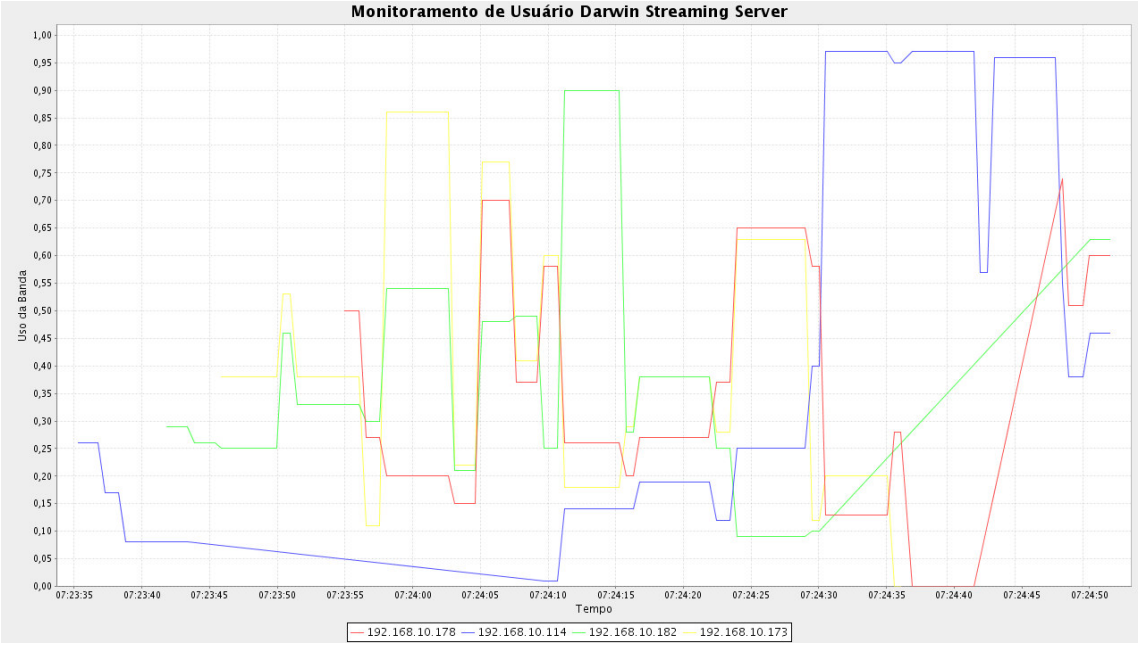

Figura 31. Monitoramento da transmissão de arquivos MOV com largura de banda total.

A Figura 32 mostra vários arquivos Mp4 sendo transmitidos e somente um arquivo MOV, observa-se que o arquivo da cor preta tem uma grande diferença na transmissão do seu *streaming*, foram alocados 40% de banda a mais para esse arquivo.

O Servidor Darwin tem uma forma de escalonamento diferenciada para os arquivos MOV, sua velocidade e qualidade de transmissão é visivelmente superior em relação aos arquivos MP4.

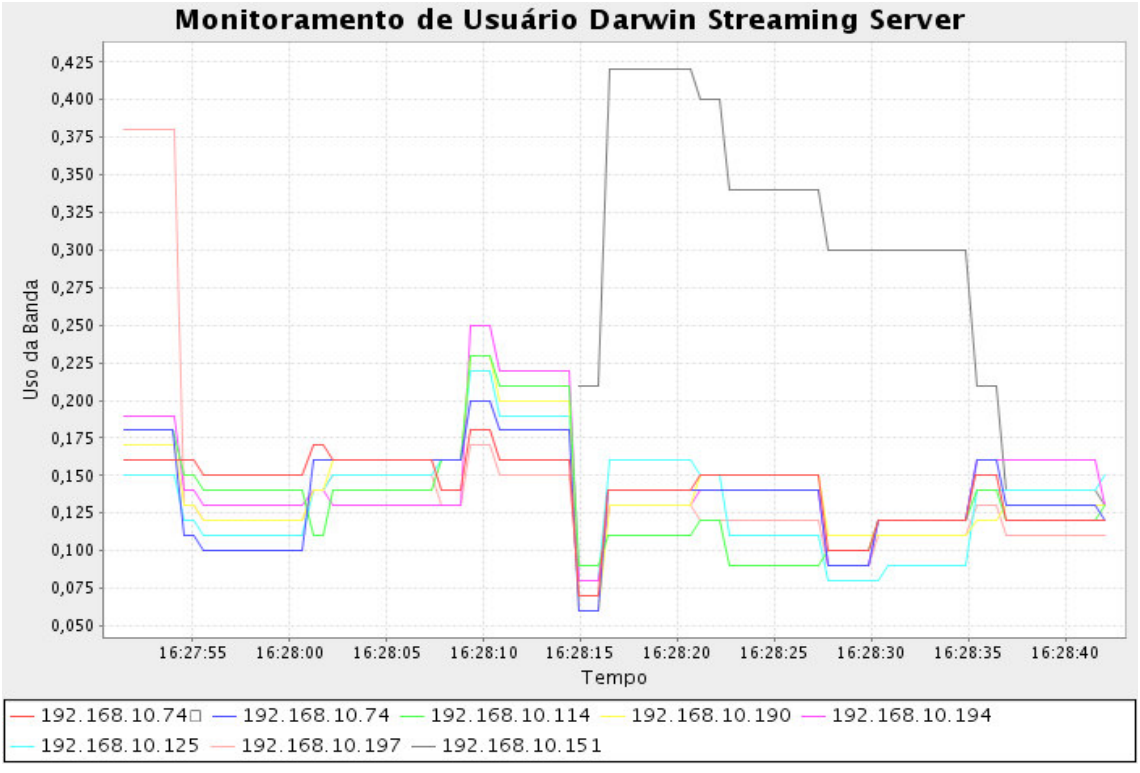

Figura 32. Monitoramento da transmissão de arquivos MP4 e MOV.

A Figura 33 mostra o menu de administração do servidor Darwin, onde se vê um resumo das sessões ativas. Detectamos, analisando as sessões ativas, que os arquivos MOV apresentam uma taxa de saída cinco vezes maior que a dos arquivos de vídeo MP4.

Pelos experimentos, conclui-se que, servir arquivos MP4, em modo streaming, consome bem mais tempo de processamento do servidor Darwin. Por este motivo, este não os consegue entregar com a mesma taxa de saída dos arquivos MOV. Acreditamos que tal fato ocorra não por causa de alguma deficiência "oculta" no servidor Darwin, mas sim pelas próprias características inerentes a cada formato de arquivo, sendo os do tipo MOV mais naturalmente propensos à transmissão via streaming, pela sua própria natureza e origens históricas.

Connected Users Sun, 0. Jan 1900 00:00:153970688

Type▲	IP Address	Bit Rate	Bytes Sent	% Packet Loss	Time Connected	Connected To
	192.168.10.77	234.617Kbps	5.774 MB	0.00%	3 min 8 sec	/mp4/mario.mp4
	192.168.10.197	170.508Kbps	5.568 MB	0.00%	2 min 59 sec	/mp4/mario.mp4
	192.168.10.83	202.867Kbps	5.443 MB	0.00%	2 min 54 sec	/mp4/mario.mp4
	192.168.10.74	1391.984Kbps	55.324 MB	0.00%	2 min 46 sec	/mov/spider.mov
	192.168.10.178	257.336Kbps	4.098 MB	0.00%	2 min 3 sec	/mp4/mario.mp4
	192.168.10.151	0 bps	39.418 MB	0.00%	1 min 44 sec	/mov/spider.mov
	192.168.10.114	5016.562Kbps	34.577 MB	26.76%	1 min 27 sec	/mov/spider.mov
	192.168.10.78	208.578Kbps	3.004 MB	0.00%	1 min 21 sec	/mp4/mario.mp4
	192.168.10.194	191.211Kbps	2.867 MB	0.00%	1 min 15 sec	/mp4/mario.mp4
	192.168.10.190	1033.938Kbps	36.845 MB	0.00%	1 min 8 sec	/mov/spider.mov
	192.168.10.173	1006.961Kbps	36.539 MB	0.00%	1 min 6 sec	/mov/spider.mov
	192.168.10.182	4595.344Kbps	35.198 MB	0.00%	1 min 2 sec	/mov/spider.mov

Figura 33. Informações sobre as sessões ativas.

Com isso, nota-se claramente a importância de se ter um módulo de monitoramento como o aqui desenvolvido e implementado na arquitetura do servidor Darwin, pois, sem um adequado acompanhamento das sessões de usuário, tal discrepância na transmissão dos dois tipos de arquivos poderia facilmente passar despercebida.

Capítulo 6

Conclusões

6.1 Visão Geral

Como se pôde constatar, a transmissão de conteúdo multimídia na Internet constitui-se um tema relativamente novo e possuidor de algumas dificuldades a serem solucionadas, considerando-se o período de existência da rede e o fato da estrutura existente atualmente não atender aos propósitos de transmissão de informação em tempo real.

Embora muitos estudos estejam sendo realizados no sentido de adequar a Internet à transmissão desse tipo de informação, as tecnologias utilizadas no processo carecem de um maior amadurecimento, de modo a incrementar sua utilização prática.

Propostas têm sido elaboradas com o intuito de oferecer a solução mais adequada para os problemas impostos. Este trabalho concentrou-se no monitoramento de servidores *streaming* de vídeo, visando um melhor acompanhamento da transmissão em tempo real de arquivos desse tipo, utilizando-se os protocolos RTP e RTSP.

6.2 Dificuldades

A principal dificuldade, foi a implementação, gastou-se muito tempo com estudos para conseguir implementar funcionalidades na atual arquitetura do Darwin, uma arquitetura modular, com vários módulos fazendo papeis diferentes, e com comunicação XML.

O Servidor Darwin é opensource, mas a escassez de documentação para implementação dificultou muito, a documentação distribuída pelo fabricante é muito breve, não discrimina o papel que os módulos assumiam, e nem como era sua comunicação.

O tempo levado para estudo do servidor durou seis meses, após isso veio à implementação do módulo monitor e os testes.

6.3 Principais Resultados e Contribuições

Esta dissertação descreveu um módulo de monitoramento implementado em um servidor streaming de vídeo, no caso o *Darwin Streaming Server*, da Apple, um servidor de código aberto que tem como base do seu código o *QuickTime Streaming Serve*r, do mesmo fabricante.

Foram analisadas as peculiaridades da transmissão multimídia na Internet, bem como os protocolos RTP, RTCP e RTSP utilizados para este fim. Dois formatos

populares de arquivos de vídeo, o MPEG-4 e o MOV foram escolhidos para os experimentos e também relatados.

A arquitetura do servidor Darwin foi analisada em detalhes, assim como a arquitetura do nosso *Módulo Monitor*, o qual foi implementado como um módulo do Darwin.

Os resultados experimentais obtidos, devidamente acompanhados por meio do Módulo Monitor, mostram uma diferença significativa na taxa de entrega dos arquivos MP4 e MOV, com a primazia deste último. Conclui-se que, servir arquivos MP4, em modo streaming, consome bem mais tempo de processamento do servidor Darwin. Por este motivo, este não os consegue entregar com a mesma taxa de saída dos arquivos MOV. Os arquivos MOV tiveram uma taxa de transmissão até 75% maior que os arquivos MP4.

Com isso, constata-se a importância de um módulo de monitoramento como esse, o qual pode tornar-se uma ferramenta importante para o administrador do sistema que deseja acompanhar o desempenho do mesmo.

6.4 Trabalhos Futuros

O Módulo Monitor aqui relatado ainda é um projeto em andamento. Pretende-se futuramente torná-lo capaz de capturar outras informações além da largura de banda utilizada e da taxa de perda de pacotes de cada sessão de usuário, aumentando-se assim sua aplicabilidade.

Outro trabalho futuro seria a implementação de um mecanismo de *tuning* semi-automático no servidor *streaming,* a fim de permitir um controle minucioso de todas as transmissões ativas, podendo-se assim melhor controlar a qualidade do serviço oferecido e a largura de banda fornecida a cada usuário.

Referências Bibliográficas

[1] J. Kurose, K. Ross. *Redes de Computadores e a Internet*, Addison Wesley, 2003.

[2] Tanembaum A., *Sistemas Operacionais Modernos*, 2.ed., Prentice Hall, 2003.

[3] H. Schulzrinne, *"A Transport Protocol for Real-Time Applications. (RTP)"*, http://www.ietf.org/rfc/rfc1889.txt, 1996.

[4] H. Schulzrinne, A. Rao, R. Lanphier, *"Real Time Streaming Protocol (RTSP)"*, http://www.ietf.org/rfc/rfc2326.txt, 1998.

[5] MPEG4ip, 2007. *Mpeg4ip Commuty Site* , http://mpeg4ip.sourceforge.net.

[6] Darwin , 2007. *Darwin Project Site*, http://developer.apple.com/darwin/projects/streaming.

[7] QuickTime Apple Web Site, http://www.apple.com/ quick time/. Acesso em março, 2007.

[8] Ferguson P., Huston G., 1998, *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*. John Wiley.

[9] D. E. Comer, *Internetworking with TCP/IP: Principles, Protocols and Architecture*, vol. 1, 4.ed, Prentice Hall, 2000.

[10] I. Busse, B. Deffner, H. Schulzrinne, "*Dynamic QoS Control of Multimedia* Applications based on RTP*"*, *Computer Communications*, vol. 19, pp. 49-58, jan., 1996.

[11] Stallings, W., 2002, *High-Speed Networks and Internets: Performance and Quality of Service*, 2.ed., Prentice Hall.

[12] MPEG Site, 2007. *Moving Picture Experts Group*, http://www.mpeg.org.

[13] Gnustream, A P2P Media Streaming System Prototype, 2003.

[14] Banerjee, S., 2003, *Scalable Resilient Media Streaming*. Technical Report, CS-TR 4482, Univ. of Maryland, USA.

[15] Wakamiya, N. et al, 2001, .MPEG-4 Video Transfer with TCP-Friendly Rate Control. *LNCS*, Vol. 2216, pp. 29.

[16] Radha, H. M. et al, 2001. The MPEG-4 Fine-Grained Scalable Video Coding Method for Multimedia Streaming over IP. *IEEE Transactions on Multimedia*, Vol. 3, No. 1, pp. 53-68.

[17] Wei Zhao, Satish K. Tripathi *Bandwidth-Efficient Continuous Media Streaming Through Optimal Multiplexing*, 1999.

[18] Sen, S. et al., 1999, *Proxy Prefix Caching for Multimedia Streams*. *INFOCOM*, pp. 1310-19.

[19] Chen, S., et al., 2005. Fast Proxy Delivery of Multiple Streaming Sessions in Shared Running Buffers. *IEEE Transactions on Multimedia,* Vol. 7, No. 6.

[20] Anastasiadis, S. V. et al., 2001. Server-Based Smoothing of Variable Bit-Rate Streams. *ACM Multimedia* , pp. 147-58.

[21] Zhao, W., Tripathi, S. K., 1999, Bandwidth-Efficient Continuous Media Streaming Through Optimal Multiplexing. *ACM Performance Evaluation Review*, Vol. 27, No. 1, pp. 13-22.

[22] Schulzrinne H. et al., 1996, RTP: A Transport Protocol for Real-Time Applications. *RFC 1889*.

[23] H. Schulzrinne, A. Rao, and R. Lanphier, "*Real time streaming protocol (RTSP), request for comments 2326*," April 1998.

[24] CEREDA, R. *ATM O Futuro das Redes.* São Paulo, Makron Books, 1997.

[25] FLUCKINGER, F. *Understanding Networked Multimedia- Applications and Technology.* Prentice Hall, 1995.

[26] VINE, P. *Video On Demand Via ATM Networks*, 1997.

[27] 3GPP. Transparent end-to-end Packet-switched Streaming Service. V 6.8.0. 3GPP, 2006.

[28] IBE, Oliver C. *Essentials of ATM Networks and Services*. 2 ed. Massachusetts, Addison Wesley Longman, 1998.

[29] HALSALL, Fred. *Multimedia Communications (Applications, Networks, Protocols and Standards)*. Editora Addison – Wesley, 1a edição.

[30] Comer, D. E., 2000 *Internetworking with TCP/IP: Principles, Protocols and Architecture*, Vol. 1, 4.ed, Prentice Hall.

[31] J. Kurose, K. Ross, James F. *Computer Neworking: a top-down approach featuring the Internet*. Boston, Addison Wesley, Inc., 2001.

[32] FAFALI,V., PATRIKAKIS,Ch e MINOGIANNIS, N. Commercial Video Streaming Servers, Rate Control and Stream Switching Techniques, *Olympic Project, 11/07/2003.*

[33] P. Ferguson and G. Huston. *Quality of Service: Delivering QoS on the Internet and in CorporateNetworks.* John Wiley, 1998.

[34] Soares, L.F.G; Lemos, G. e Colcher, S. *Redes de Computadores: das LANs MANs e WANs as Redes ATM*. Segunda Edição. Editora Campus. Rio de Janeiro, 1995.

[35] AMMAR, M., LI, X., PAUL, S. "*Video Multicast over the Internet", IEEE Network,*
New York, 1999.

[36] BERSON, S., LINDELL, R., BRADEN, R. "*An Architecture for Advance Reservations in the Internet", Technical Report, UCS Informations Science Institute*, 1998.

[37] Busse, I., B., Deffner, Schulzrinne, H., 1996, Dynamic QoS Control of Multimedia Applications based on RTP. *Computer Communications*, vol. 19, pp. 49-58.

[38] LIAO, W., LI, V. O. K. "*Distributed Multimedia Systems*". New York, vol. 85,nº 7, 1997.

[39] SCHULZRINNE, Henning, et al. "*RTP: A Transport Protocol for Realtime Applications*", Internet RFC 1889, Internet Engineering Task Force, 1996.

[40] SCHULZRINNE, Henning. "*RTP Profile for Audio and Video Conferences with Minimal Control*", Internet RFC 1890, Internet Engineering Task Force, Jan. 1996.

[41] WOLF, L. C., GRIWODZ, C., STEIRMETZ, R. "*Multimedia Communications*", vol 85, nº 12, 1997.

[42] RTPMON, 2008. *Rtpmon Project Site*, Darwin , 2007. *Darwin Project Site*, http://developer.apple.com/darwin/projects/stre aming.

[43] SAFIRE RTP TRAFFIC ANALYSER, 2008. *Safire RTP Project Site,* http://www.safire-world.com/datasheets/ds_9226_01_rtp_rtcp_signaling_tester.htm.

[44] H. Schulzrinne, S. Casner, V. Jacobson, R. Frederick, *"RTP: A Transport Protocol for Real-Time Applications"*, http://www.ietf.org/rfc/rfc3550.txt, 2003.

Anexo I

Funções da API do Darwin

QTSSFileModule: Gerencia os arquivos.
QTSSReflectorModule: Gerencia controle básicos.
QTSSRelayModule: Gerencia os atrasos.
QTSSMP3StreamingModule: Gerencia os streams e MP3.
QTSSErrorLogModule: Gerencia os logs de erro.
QTSSAccessLogModule: Gerencia os logs de acesso.
QTSSWebStatsModule: Controla todos os logs de estados.
QTSSWebDebugModule: Debug do módulo Web.
QTSSAdminModule: Gerencia todo o Darwin.
QTSSPOSIXFileSystemModule: Controla os arquivos de sistema posix.
QTSSAccessModule: Gerencia os Acessos.
QTSSHomeDirectoryModule: Controla o diretório home no controle de acesso.
QTSSHttpFileModule: Gerencia os arquivos HTML.
QTSSSpamDefenseModule: Módulo de prevenção de Spam.
QTSServerInterface: Gerencia a interface do servidor.
QTSS_AddRole: Rotina que informa ao servidor que seu módulo pode ser chamado para uma regra especificada por esta rotina.
QTSS RTSPReqFullRequest: String que contém o caminho completo da requisição RTSP.
QTSS RTSPReqRootDir: a raiz do diretório para uma requisição.
QTSS RTSPReqAbsoluteURL: Controla os pedidos processados.
QTSS_Write: Escreve um buffer de dados em uma stream.
QTSS_SendStandardRTSPResponse: Rotina que escreve uma resposta padrão. A resposta depende do método.
QTSS_AppendRTSPHeader: Rotina que adiciona um cabeçalho ao cabeçalho RTSP. Após invocar esta rotina deve-se chamar QTSS_SendRTSPHeaders.
QTSS_SendRTSPHeaders: Rotina que envia um cabeçalho RTSP.
QTSSMonitorModule: Modulo desenvolvido para monitorar o trafego do servidor.
QTSS_RTSPFilter_Role: Regra invocada no módulo cliente para realizar modificações no conteúdo da requisição RTSP.
QTSS RTPSvrCurBandwidth: Banda corrente saindo do servidor em bits por segundo.
QTSS_ServerObject: Consiste dos atributos que contêm as informações globais do servidor.
QTSS CliSesCurrentBitRate: Taxa de transferência atual de um determinado cliente.
QTSS CliRTSPSessRemoteAddrStr: Endereço de IP do cliente.
QTSS CliSesPacketLossPercent: Representa a percentagem de pacotes perdidos por um determinado cliente.