Eduardo Carneiro da Cunha

Uma Estratégia de Criação e Apresentação de Documentos Multimídia Adaptativos em Rede

Orientadores:
Prof. Luiz Fernando Rust da Costa Carmo, Dr. UPS
Profª Luci Pirmez, Dra.

Núcleo de Computação Eletrônica - NCE
Instituto de Matemática - IM
Universidade do Estado do Rio de Janeiro - UFRJ

Rio de Janeiro, Outubro de 2000.

UMA ESTRATÉGIA DE CRIAÇÃO E APRESENTAÇÃO
DE DOCUMENTOS MULTIMÍDIA ADAPTATIVOS EM REDE

Eduardo Carneiro da Cunha

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE MATEMÁTICA/NÚCLEO DE COMPUTAÇÃO ELETRÔNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM INFORMÁTICA.

Aprovada por:

_____
Prof. Luiz Fernando Rust da Costa Carmo, Dr. UPS.

_____
Profª. Luci Pirmez, Dra.

_____
Prof. José Ferreira de Rezende, Dr.
.

_____
Prof.

RIO DE JANEIRO, RJ - BRASIL

OUTUBRO DE 2000

FICHA CATALOGRÁFICA

CUNHA, EDUARDO CARNEIRO DA.

Uma Estratégia de Criação e Apresentação de Documentos Multimídia Adaptativos em Rede [Rio de Janeiro] 2000

IX, 84 p. 29,7 cm (IM/NCE/UFRJ, M.Sc., Informática, 2000)

Dissertação - Universidade Federal do Rio de Janeiro, IM/NCE

1. Agentes Móveis

I. IM/NCE/UFRJ II. Título (série)

AGRADECIMENTOS

A Deus e à minha família, sempre presentes em minha vida.

A Andréia Rosa Coelho, por ter estado sempre ao meu lado nos momentos decisivos. Obrigado pelo amor, pelos incentivos e pela paciência.

Aos meus amigos que estimularam e desejaram o meu completo sucesso no decorrer desta pesquisa.

Aos meus orientadores pela disponibilidade de tempo, dicas e sugestões fornecidas no decorrer do presente trabalho. Obrigado pela assistência e pelas soluções nos momentos de incertezas.

Agradeço ao Núcleo de Computação Eletrônica pelas condições de trabalho fornecidas e pelo auxílio relacionado à publicação de artigos durante a realização deste trabalho.

Resumo da Tese apresentada ao IM/NCE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

Uma Estratégia de Criação e Apresentação
de Documentos Multimídia Adaptativos em Rede

Eduardo Carneiro da Cunha

Outubro/2000

Orientadores: Luiz Fernando Rust da Costa Carmo
Luci Pirmez

Programa: Informática

A apresentação de um documento multimídia distribuído envolve a recuperação de objetos a partir de um ou mais servidores e a apresentação destes nos sistemas clientes. A recuperação dos objetos multimídia a partir dos servidores de documentos é influenciada por fatores como a largura de banda e o atraso máximo oferecidos pela rede, e deve ser realizado de acordo com a especificação dos relacionamentos entre os objetos. Este trabalho busca desenvolver um ambiente propício para a criação e recuperação adaptativa de documentos multimídia em redes corporativas. O objetivo final consiste em possibilitar uma apresentação coordenada de um documento multimídia, garantindo sempre a preservação da coerência entre os diferentes fluxos de mídias, mesmo quando o processamento é confrontado com uma insuficiência temporária de recursos oferecidos pela rede de comunicação. Desta forma, o ambiente define uma estratégia de composição de documentos multimídia que trata os requisitos de QoS da apresentação do ponto de vista da aplicação e que pode ajudar no gerenciamento de variações nos recursos disponíveis na rede através de um monitoramento em tempo-real desses relacionamentos entre os objetos.

Abstract of Thesis presented to IM/NCE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

An Authoring and Presentation Strategy for
Adaptive Multimedia Documents in Computer Networks.

Eduardo Carneiro da Cunha

October/2000

Advisors: Luiz Fernando Rust da Costa Carmo
Luci Pirmez

Department: Informatics

A distributed multimedia document presentation involves retrieval of objects from one or more document servers and their presentation at the client system. The retrieval of multimedia objects from the document servers is influenced by factors such as throughput and maximum delay offered by the network, and has to be carried out in accordance with the specification of object relationships. This work intends to develop an appropriate environment for creation and retrieval of multimedia documents in corporate networks. The final purpose consists of providing a coordinate multimedia document presentation, always guaranteeing the preservation of coherence between the different media, even when the process is confronted with a temporary lack of communication resources. Thus, the environment defines a strategy for multimedia documents composition that address QoS from the application's point of view and can help in handling variations in network resources availability through a real-time monitoring over these object relationships.

SUMÁRIO

LISTA DE FIGURAS

Capítulo 1

Introdução

O surgimento das redes de computadores de alta velocidade, das tecnologias de armazenamento e de comunicação, e dos equipamentos periféricos de áudio e vídeo digitais incentivou o rápido crescimento da pesquisa e desenvolvimento de sistemas multimídias. O uso das tecnologias de multimídia veio permitir o gerenciamento e a rápida disseminação da informação de modo eficiente, indicando que os sistemas multimídia distribuídos interconectados via rede serão, em um futuro próximo, um componente crucial da infra-estrutura de comunicação de dados.

1.1 Qualidade de Serviço e Qualidade de Percepção

Os sistemas multimídia são sistemas computacionais que manipulam de forma integrada vários tipos de mídias de representação da informação. Frequentemente, sistemas multimídia são distribuídos, isto é, seus componentes estão localizados em diferentes nós de processamento numa rede local ou de longa distância. Desta forma, a apresentação de um documento multimídia distribuído envolve a recuperação de objetos a partir de um ou mais servidores e a apresentação destes nos sistemas clientes. A recuperação dos objetos multimídia a partir dos servidores de documentos é influenciada por fatores dinâmicos, como atraso, variação de atraso e perdas de pacotes, que são verificados no interior da rede de comunicação, ao longo do caminho que interconecta os clientes e servidores.

1.1.1 Qualidade de Serviço

A qualidade de serviço neste contexto pode ser definida intuitivamente como uma medida de quão satisfeito está o usuário com respeito a um serviço prestado por um sistema multimídia distribuído (SMD). Embora a noção de QoS seja intuitiva, várias definições são encontradas na literatura: um conjunto de características qualitativas e

quantitativas de um sistema multimídia distribuído necessárias para adquirir as funcionalidades requeridas por uma aplicação [2]; um conjunto de parâmetros que define as propriedades de *streams* [3]; uma descrição quantitativa de quais serviços oferecidos pelo sistema satisfazem as necessidades da aplicação expressos como um conjunto de pares de parâmetros [4].

Do ponto de vista do sistema computacional, Paul Ferguson [5] descreve os seguintes parâmetros de QoS:

"*Atraso* é o tempo decorrido para que um pacote seja passado do emissor, através da rede, até o receptor. Quanto maior o atraso entre o emissor e o receptor, maior o esforço exigido do protocolo de transporte para que este opere eficientemente e menos atuante se torna a malha de realimentação. Consequentemente, o protocolo se torna menos sensível às mudanças dinâmicas de curta duração que ocorrem na carga da rede".

"*Jitter* é a variação no atraso fim-a-fim (em termos matemáticos ele é medido como o valor absoluto da primeira diferencial da sequência de medições individuais do atraso). Altos níveis de *jitter* são inaceitáveis em situações onde a aplicação é baseada em tempo real (áudio e vídeo). Nestes casos, o *jitter* torna o sinal distorcido, podendo ser recuperado apenas com um aumento no buffer do receptor, o que afeta o atraso, tornando as sessões interativas muito difícieis de serem mantidas".

"*Largura de banda* é a taxa de transferência máxima que pode ser sustentada entre dois nós da rede. Constata-se que a largura de banda é limitada não apenas pela infraestrutura física da rota dentro das redes de trânsito (o que fornece um limite superior para a banda), mas também pelo número de outros fluxos que compartilham os componentes comuns da rota selecionada".

"*Confiabilidade* é comumente conceituada como uma propriedade do sistema de transmissão, e neste contexto, pode ser vista como a taxa média de erro do meio de transmissão".

A maioria das arquiteturas de SMD desenvolvidas até o momento trata a questão da qualidade de serviço do ponto de vista do sistema computacional (ou provedor de acesso) e

empregam políticas efetivas de monitoramento e gerência de recursos para prover suporte à qualidade de serviço [6]. Porém, estas arquiteturas pecam por não elevarem o conceito de QoS até o nível do usuário, fazendo com que a especificação da QoS seja feita forçosamente através dos parâmetros do nível do sistema computacional.

1.1.2 Qualidade de Percepção

As abordagens tradicionais para se prover qualidade de serviço às aplicações multimídia têm focado formas de assegurar e gerenciar, dentro das redes de comunicação, diferentes parâmetros técnicos, tais como os citados anteriormente. Entretanto, para o usuário de multimídia, estes parâmetros possuem pouco significado ou impacto imediato. Embora ele possa se sentir levemente incomodado com a falta de sincronização entre os fluxos de áudio e vídeo, é altamente improvável que ele perceba, por exemplo, a perda de um quadro de vídeo dentre os 25 que podem ser apresentados dentro de um único segundo.

O fator que mais interessa aos usuários finais é a possibilidade de usufruir de uma apresentação multimídia ao mesmo tempo em que assimilam seus conteúdos informacionais. Alguns trabalhos interessantes [7, 8] foram realizados com o objetivo de verificar a satisfação do usuário com relação às aplicações multimídia que foram apresentadas com seus níveis de qualidade de serviços deliberadamente modificados. O trabalho apresentado em [9] incrementou a tradicional visão da QoS com uma definição feita ao nível do usuário chamada QoP (*Quality of Perception*). A QoP corresponde a uma medida que engloba não apenas a satisfação do usuário com respeito a um serviço prestado por um sistema multimídia distribuído (SMD), mas também, à potencialidade do usuário perceber, sintetisar e analisar os conteúdos informacionais de uma apresentação.

Um fator importante é que, devido à natureza heterogênea das infraestruturas de redes globais dos dias de hoje, e à sua condição variável, pode não ser possível satisfazer completamente os requisitos de QoP e, portanto, de QoS, dos usuários. A reserva de recursos e as técnicas de admissão convencionais não são capazes de garantir uma qualidade de serviço sem um considerável desperdício de recursos e uma ineficiente utilização dos mesmos [5]. Para resolver este problema e evitar qualquer impacto adverso sobre o usuário final, as aplicações distribuídas e suas infraestruturas precisam ser

adaptativas. Desta forma, as aplicações devem tolerar flutuações na disponibilidade dos recursos ou a infraestrutura de rede deve poder se moldar às mudanças dinâmicas nos requisitos das aplicações.

1.2 Recuperação adaptativa

O processo de adaptação freqüentemente acontece como resultado de notificações de QoS normalmente emitidas por mecanismos de monitoramento de QoS. Estas notificações indicam mudanças no serviço que está sendo oferecido pela rede e que normalmente é afetado através da disponibilidade de recursos fim-a-fim. As notificações podem indicar falta de recursos, e uma consequente redução na qualidade (degradação de QoS), ou uma falha para manter um serviço de qualidade através de uma perda completa de recursos (falha de QoS). Pode ser considerada como falha tanto uma perda total da rota utilizada pelo fluxo, quanto uma impossibilidade de se manter a QoS dentro de uma faixa de requisitos de qualidade que tenha sido especificada pela aplicação. Sejam degradações ou falhas que venham a acontecer, a adaptação faz-se necessária. Esta adaptação pode ajustar a utilização de recursos de maneira a manter uma degradação de QoS quase não sentida pela aplicação informando ao usuário sobre a necessidade de alterar o nível de serviço para um nível mais baixo ou reduzindo os valores dos parâmetros de QoS de forma transparente ao usuário.

Cristina Aurrecoechea [6] descreve uma variedade de mecanismos de adaptação da qualidade de serviço. Por exemplo, filtros de QoS localizados em pontos específicos na rede que transcodificam os fluxos que passam por eles de acordo com a capacidade do *link* e os recursos do cliente; mecanismos de adaptação da taxa de envio do emissor, e camadas incrementais de QoS enviadas por multicast (*Layered Multicast*).

Entretanto, todos eles se preocupam apenas com parâmetros de tráfego específicos e com a percepção direta do usuário com relação a mídias isoladas. A apresentação de um documento multimídia, considerando todos os seus fluxos de mídias distintas e suas relações semânticas, não é tratada como uma documentação complexa. O significado desta documentação, que foi construído através da combinação de diversos objetos de informação multimídia, deve ser transmitido e preservado ao longo da apresentação.

1.3 Objetivos e organização do trabalho

Este trabalho faz parte de um projeto de pesquisa na área de redes de computadores que se insere no contexto de ensino à distância. Basicamente, o projeto Servimídia [1] trata da autoria e armazenamento de documentos multimídia e da infraestrutura de rede de comunicação para realizar a recuperação remota destes documentos. O objetivo básico está na necessidade de se acomodar em um mesmo ambiente multimídia de ensino, baseado no modelo cliente/servidor, usuários com diferenças substanciais na disponibilidade dos recursos de comunicação. O fator crucial é que um mesmo documento, recuperado por diferentes usuários, pode ser interpretado de uma maneira completamente diferente de acordo com a qualidade da apresentação (percepção) que é verificada pelo cliente. A qualidade de percepção, por sua vez, está diretamente relacionada com a disponibilidade dos recursos encontrados na rede ao longo da rota que chega ao cliente, pois alguns dos fluxos multimídias que compõem o documento podem ser perdidos ou parcialmente danificados durante o processo de recuperação do documento a partir do servidor.

Neste trabalho, é investigada uma estratégia de autoria, armazenamento e recuperação de documentos multimídia que permita gerar diferentes formatos de apresentação de um mesmo documento para satisfazer às necessidades e limitações dos diferentes usuários. Os diferentes formatos de apresentação devem ser gerados a partir de uma única especificação do documento incrementada com informações que permitem às aplicações gerenciar e adaptar a apresentação do documento. As aplicações podem se basear em diferentes parâmetros com o objetivo de transformar (adaptar) um documento que está sendo enviado ao cliente. Porém, este trabalho se concentra nos parâmetros relacionados à disponibilidade de recursos de comunicação verificada na rede que interconecta servidores e clientes.

O ponto chave nesta estratégia é que, para cada transformação do documento ao longo do processamento, as propriedades semânticas das diversas mídias que o compõem devem ser preservadas. Em outras palavras, o usuário que estiver assistindo à apresentação do documento deve ser capaz de absorver o conteúdo informacional contido no documento, mesmo quando este sofrer alguma degradação de qualidade devido à falta de recursos verificada na rede.

Este trabalho está estruturado em cinco capítulos. O capítulo 2 apresenta uma visão geral da área de comunicação multimídia nas redes de computadores atuais, os desafios e as soluções propostas para se realizar a transmissão de dados multimídia com um certo grau de qualidade de serviço. O capítulo 3 introduz alguns conceitos de autoria e de adaptação de conteúdo dos documentos multimídia. O capítulo 4 discute a adaptação de conteúdo destes documentos e apresenta a proposta deste trabalho para uma estratégia de composição de documentos multimídia adaptativos através de relacionamentos semânticos entre suas diversas mídias. A metodologia empregada e uma ferramenta desenvolvida para se implementar esta estratégia são também descritas no capítulo 4. O capítulo 5 define a arquitetura de comunicação do ambiente ServiMídia e apresenta os mecanismos de monitoramento da transmissão e de adaptação do documento que foram implementados com o objetivo de se avaliar a proposta deste trabalho. Por fim, o capítulo 6 apresenta algumas conclusões e oferece sugestões de trabalhos futuros com o objetivo de dar continuidade ao Projeto ServiMídia.

Capítulo 2

Multimídia em Rede

As redes de computadores foram desenvolvidas para interconectar computadores em diferentes localidades de maneira que estes possam compartilhar dados e se comunicar. Há alguns anos, a maioria dos dados que trafegavam na rede eram de origem textual. Hoje, com o crescimento das tecnologias de redes, a multimídia tem se tornado uma ferramenta indispensável na Internet. Produtos do tipo *Internet telephony*, *Internet TV*, e video-conferência têm aparecido no mercado. No futuro, as pessoas irão desfrutar de outras aplicações multimídia nas áreas de ensino a distância, simulação distribuída, trabalho cooperativo entre outros. Para os pesquisadores, o desafio é desenvolver a infraestrutura de *hardware* e *software*, bem como as ferramentas e aplicações para suportar a transmissão destes dados multimídia nas redes de computadores.

2.1 Multimídia na Internet: objetivos e desafios

A transmissão multimídia em redes, porém, não é uma tarefa trivial. Sendo uma rede compartilhada baseada em datagramas, a Internet não é naturalmente indicada ao tráfego de tempo real. Contrariamente às características solicitadas pelos dados multimídia (grande banda passante, tráfego em tempo real e em rajada), a Internet é compartilhada por milhões de usuários, tendo banda limitada, além de atraso e disponibilidade de recursos imprevisíveis. Como solucionar estes conflitos é o desafio que os pesquisadores têm que enfrentar.

Novas arquiteturas estão sendo propostas buscando oferecer o suporte necessário para que as aplicações multimídia tenham seus requisitos de qualidade de serviço atendidos. Essas arquiteturas são amplas em escopo e cobrem as redes e os sistemas finais; diferem em vários aspectos, os quais são resultados das diferentes comunidades em que foram desenvolvidas. Alguns trabalhos que vêm sendo realizados para endereçar os

problemas quanto ao oferecimento de QoS para as aplicações são: a arquitetura apresentada por Campbell [10] que é semelhante ao modelo em camadas OSI (*Open Systems Interconnection*), porém, além de camadas, é composta de planos que são responsáveis por tratar a QoS das aplicações multimídia; uma estrutura para suportar QoS no modelo OSI [11]; uma estrutura baseada em CORBA para administração de QoS [12]; uma arquitetura oferecendo um Servidor de Reserva Avançada, onde reservas de recursos são realizadas antecipadamente e armazenadas no servidor [13]; um mecanismo para ajuste dinâmico de requisitos de largura de banda para aplicações multimídia [14]; uma estrutura dinâmica para controle de qualidade de serviço de vídeo para aplicações multimídia distribuídas [15]; um modelo de especificação de QoS para apresentações multimídia distribuídas que permite adaptação das aplicações ao que a rede oferece [16]; uma estrutura para provisão de QoS em ambientes genéricos de processamento e comunicação [17].

Para realizar a transmissão de dados multimídia sobre a Internet, algumas questões devem ser respondidas.

Primeiro, o custo associado às redes de meio compartilhado está no fato da disponibilidade dos recursos não poder ser assegurada. Porém, as aplicações de tempo real necessitam de garantias de banda passante quando a transmissão é realizada. Portanto, deve existir algum mecanismo que permita às aplicações reservar recursos ao longo do caminho da transmissão.

Segundo, as aplicações multimídia estão geralmente relacionadas à comunicação multicast, isto é, o mesmo fluxo de dados, e não múltiplas cópias, é enviado a grupos de receptores. Os protocolos desenvolvidos para uso nas aplicações multimídia devem considerar a comunicação multicast de forma a reduzir o tráfego na rede.

Terceiro, na Internet os pacotes são roteados independentemente através das redes intermediárias. Não existem garantias para que os pacotes não cheguem atrasados, que cheguem ordenados ou que não sofram *jitter*. Algum protocolo de transporte deve ser usado para cuidar das questões de temporização de forma que os dados de áudio e vídeo possam ser apresentados continuamente com a correta temporização e sincronização.

Por fim, devem existir operações padronizadas que permitam às aplicações gerenciar e controlar o envio e o andamento da apresentação dos dados multimídia.

As repostas para as questões acima estão nos protocolos e padrões definidos no âmbito da Internet e que são discutidos na sessão a seguir.

2.2 Os protocolos para comunicação multimídia na Internet

A Internet transporta todo tipo de tráfego, cada tipo tendo suas características e requisitos específicos. A solução genérica para a multimídia sobre IP envolve a classificação dos tráfegos, a alocação de prioridades para diferentes aplicações e a implementação da reserva de recursos.

Dois modelos de serviços avançados para a Internet, criados pelo IETF, e denominados Serviços Integrados [18] e Serviços Diferenciados [19] buscam habilitar as redes baseadas em IP a prover qualidade de serviço para as aplicações multimídia. O protocolo de reserva de recursos (RSVP) [20] em conjunto com o RTP (*Real-time Transport Protocol*) [21], o RTCP (*RTP Control Protocol*) [21] e o RTSP (*Real-Time Streaming Protocol*) [22], fornecem a base fundamental para os serviços de tempo real solicitados pela maioria das aplicações multimídia. Os modelos de serviços do IETF permitem às aplicações configurar e gerenciar uma infraestrutura comum tanto às aplicações multimídia quanto às aplicações tradicionais.

2.2.1 Real Time Transport Protocol (RTP)

O RTP é tanto um padrão proposto pelo IETF (RFC1889) quanto um padrão do ITU (H.225.0). Este protocolo descreve o formato dos pacotes para os fluxos de dados multimídia, sendo utilizado pelo RTSP e pelo H.323 para a comunicação de dados.

O RTP provê serviços de transferência fim-a-fim para dados com características de tempo real tais como áudio e vídeo interativo. Estes serviços incluem identificação de tipo de conteúdo, numeração de seqüência, estampas temporais e monitoramento da transferência. O protocolo RTP é incrementado por um protocolo de controle (RTCP) para

permitir o monitoramento da transferência de uma forma escalável para grandes redes multicast, e para prover funcionalidades de controle e identificação.

Os protocolos RTP e RTCP são independente das camadas de rede e de transporte envolvidas. As aplicações tipicamente rodam o RTP no topo do protocolo UDP para fazer uso dos seus serviços de multiplexação e *checksum*. O RTP suporta transferência de dados para múltiplos destinos usando a distribuição multicast se esta for oferecida pela rede. Ele pode ser usado para um transporte unidirecional tal qual em uma aplicação de *video-on-demand*, bem como para serviços interativos tal qual em aplicações de telefonia na Internet.

A especificação do RTP [21] consiste de duas partes que estão diretamente relacionadas: o protocolo de transporte de tempo real (RTP), para transportar os dados que têm propriedades de tempo real; e o protocolo de controle do RTP (RTCP), para monitorar a qualidade do serviço e para transmitir informação sobre os participantes de uma sessão em andamento.

Operação do RTP

A Internet é uma rede de datagramas compartilhada. Pacotes enviados na Internet sofrem atrasos e *jitter* imprevisíveis. Porém, as aplicações multimídia requerem uma temporização apropriada na transmissão e na apresentação dos dados. O RTP provê alguns mecanismos para cuidar destas questões de temporização.

A ***estampa de tempo*** (*timestamp*) é a informação mais importante para as aplicações de tempo real. O emissor atribui uma estampa de tempo de acordo com o instante no qual o primeiro octeto do pacote foi amostrado. As estampas são incrementadas pelo intervalo de tempo ocupado por um pacote. Após receber os pacotes de dados, o receptor usa as estampas para reconstruir a temporização original e apresentar os dados na taxa correta. As estampas de tempo também são usadas para sincronizar diferentes fluxos. Entretanto, o RTP não é responsável pela sincronização. Esta tarefa deve ser feita no nível da aplicação.

O protocolo UDP não entrega os pacotes em ordem temporal, então, ***números de sequência*** (*sequence numbers*) são usados para ordenar os pacotes que chegam. Eles também são usados para detectar a perda de pacotes. Note que em alguns formatos de vídeo, quando um quadro de vídeo é quebrado em vários pacotes RTP, todos podem ter a mesma estampa de tempo. Assim, apenas a estampa de tempo não é suficiente para colocar os pacotes em ordem.

O identificador do ***tipo de conteúdo*** (*payload type*) especifica o formato dos dados bem como os esquemas de codificação/decodificação. A partir dele, a aplicação receptora sabe como interpretar e processar os dados. Tipos padronizados (*profiles*) são especificados em [23]. Como exemplo temos, *PCM, MPEG1/MPEG2 audio and video, JPEG video, Sun CellB video, H.261 video streams*, entre outros. Em qualquer instante da transmissão, um emissor RTP pode apenas enviar um tipo de formato de conteúdo, embora o tipo de formato de conteúdo possa mudar durante uma transmissão, por exemplo para se ajustar a um congestionamento na rede.

Outra função é a ***identificação da fonte*** (*source identification*), que permite à aplicação receptora identificar de onde os dados estão vindo. Por exemplo, em uma audio-conferência, a partir da identificação da fonte um usuário pode saber quem está falando.

Os mecanismos acima são implementados através do cabeçalho do pacote RTP que possui os campos apresentados no apêndice A. A figura 2.1 mostra um pacote RTP encapsulado em um pacote UDP/IP.

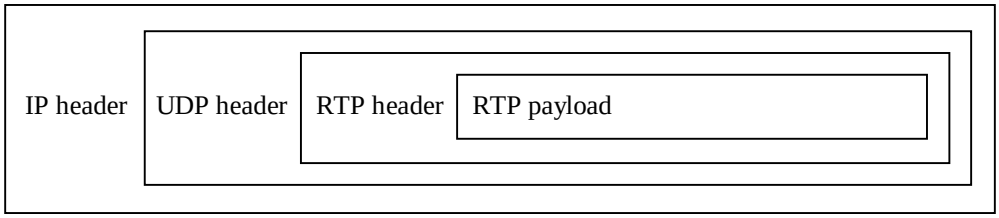


Figura 2.1 - Dados RTP em um pacote UDP/IP

O RTP representa um novo estilo de protocolo seguindo os princípios de estruturação ao nível da aplicação e de processamento em camada integrada [24, 25]. Em outras palavras, o RTP busca ser maleável para prover a informação requisitada por uma

aplicação em particular e está freqüentemente integrado ao processo da aplicação ao invés de ser implementado como uma camada separada. O protocolo RTP é um *framework* que foi deixado deliberadamente incompleto. Sua documentação especifica aquelas funções que devem ser comuns a todas as aplicações para as quais o RTP seria apropriado. Na prática, o RTP é geralmente implementado dentro da aplicação. Muitas questões, como a recuperação de pacotes perdidos e o controle de congestionamento, têm que ser implementadas no nível da aplicação.

Para estabelecer uma sessão RTP, a aplicação define um par de endereços de transporte de destino (um endereço de rede mais um par de portas para RTP e RTCP). Em uma sessão multimídia, cada mídia é transportada em uma sessão RTP separada, figura 2.2, com seus próprios pacotes RTCP reportando a qualidade da recepção para aquela sessão. Por exemplo, áudio e vídeo viajam em sessões RTP separadas, habilitando o receptor a selecionar uma mídia específica para recepção.

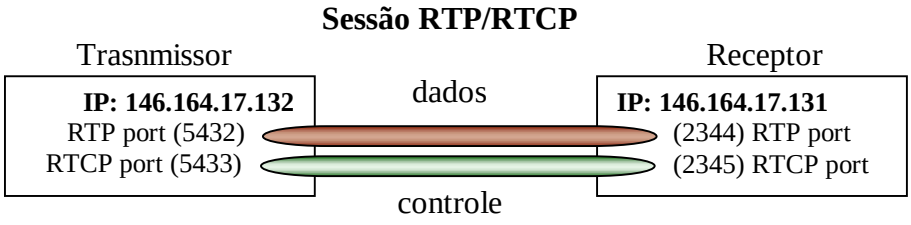


Figura 2.2 - Representação de uma sessão RTP/RTCP

RTP Control Protocol (RTCP)

O protocolo de controle do RTP, o RTCP, é baseado na transmissão periódica de pacotes de controle para todos os participantes de uma dada sessão, usando o mesmo mecanismo de distribuição dos pacotes de dados. Sua especificação [21] define 5 tipos de pacotes RTCP para carregar informações de controle. O apêndice A contém a descrição dos formatos destes pacotes.

1. **RR's (*receiver reports*)** são gerados por participantes que são receptores passivos. Eles contêm *feedback* sobre a qualidade da recepção dos dados transferidos, incluindo o número de sequência mais alto dentre os pacotes recebidos, o número de pacotes

perdidos, o *jitter* entre chegadas, e as estampas de tempo para calcular o atraso de ida e volta entre o emissor e o receptor.

2. **SR's (*sender reports*)** são gerados pelos emissores ativos. Em adição ao *feedback* sobre a qualidade da recepção encontrado nos RRs, eles contêm uma seção de informação do emissor, provendo informações sobre sincronização inter-mídia, contadores cumulativos de pacotes, e número de bytes enviados.

3. **SDES's (*source description items*)** contêm informações que descrevem as fontes.

4. **BYE's** indicam o fim da participação de um membro.

5. **APP's (*application specific functions*)** definidos para uso experimental conforme novas aplicações e novas ferramentas forem sendo desenvolvidas.

Através destes pacotes com informações de controle, o RTCP realiza quatro funções:

1. A função primária é prover *feedback* da qualidade da distribuição dos dados. A qualidade da distribuição corresponde a uma parte fundamental do objetivo do RTP (enquanto protocolo de transporte), e está relacionado às funções de controle de fluxo e de congestionamento encontradas em outros protocolos de transporte. Esta função de *feedback* é realizada pelas mensagens *RTCP sender report* e *RTCP receiver report*.

2. O RTCP carrega um identificador persistente para a fonte RTP chamado CNAME. Os receptores precisam do CNAME para manter informações de cada participante e para associar os diversos fluxos de um dado participante em um conjunto de sessões RTP relacionadas, por exemplo para sincronizar áudio e vídeo.

3. As duas primeiras funções requerem que todos os participantes enviem pacotes RTCP, portanto, a freqüência de envio deve ser controlada para que o RTP se adapte a um grande número de participantes. Como cada participante envia seu pacote de controle para todos os outros, cada um tem a informação do número de participantes. Este número é utilizado para calcular a freqüência na qual estes pacotes são enviados.

4. Uma quarta função, que é opcional, é transmitir informações mínimas de controle de sessão, como por exemplo, identificação dos participantes para ser mostrada na interface do usuário.

2.2.2 Real Time Stream Protocol (RTSP)

O RTSP [22] é um protocolo cliente-servidor de controle de apresentações multimídia. Ele é destinado a influenciar a infra-estrutura atual da web e funciona bem tanto para grandes conferências quanto para situações em que um único usuário solicita uma mídia (*media-on-demand*).

O RTSP provê serviços de transferência de fluxos de áudio e vídeo da mesma forma que o HTTP faz para os textos e figuras. Ele foi desenvolvido intencionalmente para ter a sintaxe e as operações similares ao HTTP de forma que muitas das extensões e mecanismos do HTTP possam ser adicionados ao RTSP.

Operação do RTSP

O RTSP é um protocolo de controle (sinalização) que inicia e gerencia os pedidos e as entregas de fluxos de mídias contínuas sincronizados no tempo, tais como áudio e vídeo, a partir de servidores de mídias. Este protocolo, tipicamente, não transfere os fluxos de dados, utilizando para isto um protocolo da camada inferior (UDP,TCP ou RTP). Em outras palavras, ele age como um controle remoto dos servidores multimídia, sendo apelidado de *"Internet VCR remote control protocol"*. A figura 2.3 mostra como estes protocolos estão relacionados.

Não existe a noção de uma conexão RTSP. Porém, o servidor mantém controle de uma sessão através de um identificador da sessão. Uma sessão RTSP não está associada a uma única conexão de nível de transporte tal como uma conexão TCP. Durante uma sessão RTSP, um cliente pode abrir e fechar várias conexões de transporte para o servidor e solicitar requisições RTSP. Alternativamente, um cliente pode usar um protocolo de transporte não orientado a conexão.

O conjunto de fluxos a serem controlados é definido por uma descrição da apresentação. Cada apresentação e fluxo multimídia pode ser identificado por uma RTSP URL. Uma apresentação pode conter vários fluxos de mídias. A apresentação e as propriedades das mídias que a compõem são definidas por um arquivo de descrição da apresentação. O arquivo de descrição da apresentação pode ser obtido por um cliente usando o protocolo RTSP, o protocolo HTTP, ou até mesmo outros meios como e-mail, podendo não estar, necessariamente, armazenado no servidor das mídias.

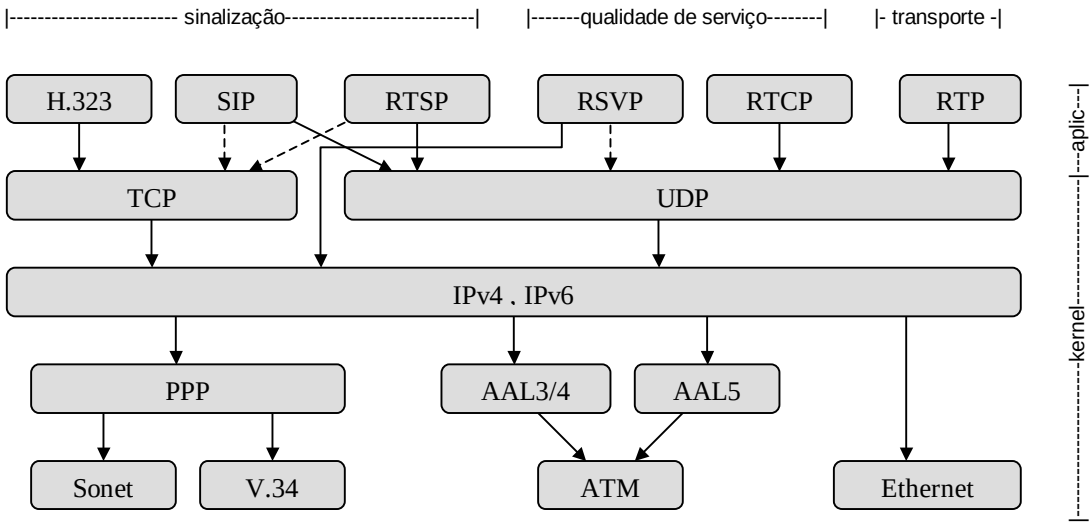


Figura 2.3 - Pilha de protocolos multimídia

O arquivo de descrição da apresentação contém uma descrição dos fluxos de mídias que constituem a apresentação, incluindo suas codificações, linguagens, e outros parâmetros que permitem ao cliente escolher a combinação das mídias mais apropriadas. Nesta descrição, cada fluxo, que é controlado individualmente pelo RTSP, é identificado por uma RTSP URL que aponta para o servidor de mídias responsável por aquele fluxo em particular e nomeia o fluxo armazenado naquele servidor.

Além dos parâmetros das mídias, o endereço e a porta de destino na rede precisam ser determinados. Três modos de operação podem ser relacionados.

- *Unicast* - A mídia é transmitida para a origem da requisição RTSP, com o número da porta escolhido pelo cliente. Alternativamente, a mídia pode ser transmitida no mesmo

fluxo das mensagens RTSP. Porém, o RTSP foi desenvolvido como um protocolo que transmite os dados “fora da banda” através de um protocolo de transporte diferente.

- *Multicast (servidor escolhe o endereço)* - O servidor seleciona o endereço e a porta multicast. Este é o caso típico de uma transmissão ao vivo ou *near-media-on-demand*.

- *Multicast (cliente escolhe o endereço)* - O endereço multicast e a porta são dados pela descrição da conferência quando o servidor decide participar de uma conferência multicast existente.

O RTSP, assim como o RTP, é um protocolo implementado no nível da aplicação, fazendo com que as aplicações não precisem de interfaces para se comunicar com camadas de protocolos diferentes. O RTSP é um protocolo baseado em texto, com as linhas sendo terminadas pelos caracteres CRLF. Ele é constituído basicamente de mensagens de requisição (*request messages*) e de resposta (*response messages*).

Uma mensagem de requisição de um cliente para um servidor, ou vice versa, inclui, dentro da sua primeira linha (*request-line*), um método a ser aplicado em um recurso, o identificador do recurso e a versão do protocolo em uso, todos separados por um caracter de espaço. A tabela B.1 do apêndice B resume os principais aspectos da mensagem de requisição do RTSP.

Uma mensagem de resposta é semelhante à mensagem de requisição sendo que sua primeira linha (*status-line*) consiste da versão do protocolo seguido por um código numérico de *status* e uma frase textual associada ao código de *status*, todos separados por um caracter de espaço. A tabela B.2 do apêndice B resume os principais aspectos da mensagem de resposta do RTSP.

As requisições RTSP podem ser transmitidas de diferentes maneiras. O tipo de conexão de transporte é definido pela RTSP URI. Se for usado “rtsp://”, uma conexão persistente é assumida, enquanto que se for usado “rtspu://”, as requisições serão enviadas sem o estabelecimento de uma conexão.

Um cliente pode encadear suas requisições, isto é, pode enviar múltiplas requisições sem esperar pelas respectivas respostas. O servidor deve enviar suas respostas para aquelas requisições na mesma ordem em que elas são recebidas. A figura 2.4 apresenta a operação do RTSP com as principais mensagens de requisição de um fluxo multimídia (“*video*”).

Primeiramente, uma mensagem de requisição (DESCRIBE) solicita a descrição da apresentação. Em seguida, baseado no arquivo de descrição recebido, uma mensagem (SETUP) solicita a configuração de uma sessão RTP para o fluxo “*video*”. Uma nova mensagem de requisição (PLAY) é enviada para solicitar o início da transmissão do video. Uma sessão RTP é estabelecida e os pacotes de dados e de controle começam a trafegar na rede. Então, uma mensagem (PAUSE) solicita a parada da transmissão e uma outra mensagem (TEARDOWN) solicita o fechamento da sessão RTP.

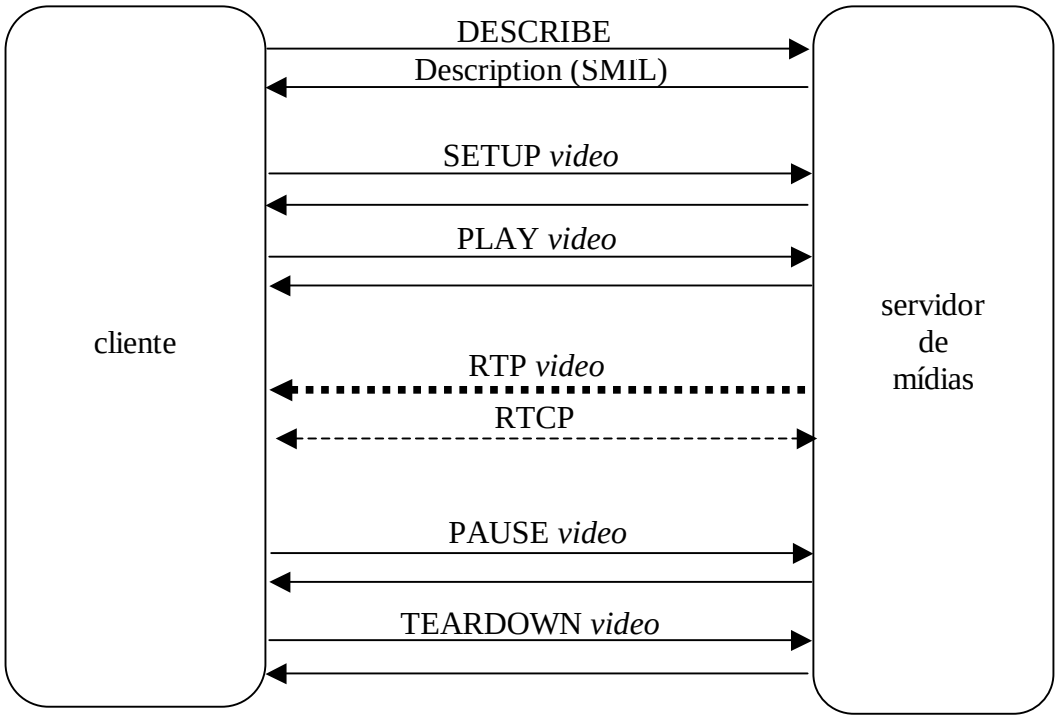


Figura 2.4 - Operação básica do RTSP

2.3 Resumo

Este capítulo, apresenta uma visão geral da área de comunicação multimídia nas redes de computadores atuais. Os desafios e as soluções propostas para se realizar a transmissão de dados multimídia com um certo grau de qualidade de serviço, seja pela extensão do modelo através de novas arquiteturas de comunicação, seja pela definição de novos protocolos de comunicação para atender às aplicações multimídia, são descritos. Este capítulo focaliza os protocolos chamados de tempo real (RTP/RTCP e RTSP) padronizados pelo IETF com o objetivo de apresentar os seus conceitos fundamentais de operação. Esses conceitos são de grande valor quando da definição da arquitetura de comunicação do ambiente ServiMídia e da apresentação dos mecanismos de monitoramento da transmissão e de adaptação do documento sendo restituído, ambos no no capítulo 5.

Capítulo 3

Autoria de Documentos Multimídia

A especificação de documentos multimídia é realizada pelos sistemas de autoria atuais com base em três aspectos fundamentais: estruturação lógica da apresentação, estabelecimento de relações temporais e definição espacial entre os objetos multimídia que a compõem. A estruturação lógica se preocupa em estabelecer mecanismos de abstração, objetivando a obtenção de uma visão ampla e estruturada da apresentação. A especificação do comportamento temporal, por sua vez, implica na definição de relações de sincronização temporal (instantes iniciais e durações) entre os objetos envolvidos. A sincronização espacial, por fim, prima por ajustar o posicionamento destes mesmos objetos de acordo com os dispositivos de saída. Uma linguagem de descrição é utilizada para especificar os documentos multimídia com base nos três aspectos acima. A linguagem SMIL é uma destas linguagens e suas principais características, com relação à adaptabilidade dos documentos, são abordadas neste capítulo. As idéias por trás da adaptação de conteúdo dos documentos multimídia também são apresentadas através do conceito de acesso universal

3.1 Estruturação lógica

A complexidade das apresentações multimídia está diretamente relacionada com o número de objetos multimídia envolvidos e, consequentemente, com os diversos relacionamentos estabelecidos entre eles. Esta é a razão fundamental pela qual a especificação destas apresentações em um único plano é inadequada, devendo ser adotada um forma de estruturação lógica da apresentação. Para resolver este problema, foram adotados por certas normas e ferramentas proprietárias os conceitos de cenas, cenários, páginas e grupos. Na norma MHEG-5 [26], por exemplo, as apresentações são organizadas como um conjunto de cenários relacionados por eventos que provêm a navegação entre eles.

3.2 *Layout* da apresentação

A sincronização espacial permite ao autor organizar o posicionamento dos objetos das mídias visíveis de uma apresentação, isto é, o autor pode configurar o *layout* da apresentação através da definição de regiões na área de apresentação onde os *clips* devem ser apresentados. Se a apresentação só apresenta um *clip* de cada vez, não é necessária a criação de um *layout* para a mesma. Na maioria das vezes, cada *clip* é automaticamente apresentado na janela principal da ferramenta de apresentação (*player*) e a janela se ajusta automaticamente para cada *clip* novo. Se for desejado que o tamanho da área de *playback* permaneça estável, ou se a apresentação apresentar vários *clips* ao mesmo tempo, é interessante definir áreas de *playback* (regiões) dentro da janela principal.

3.3 Sincronização temporal

Um documento multimídia consiste de diferentes tipos de objetos de mídias que devem ser apresentados em diferentes instantes de tempo com diferentes durações. Os instantes de início e as durações das apresentações destes objetos podem ser especificados de uma maneira rígida ou flexível. No caso de uma especificação rígida, estes instantes e durações são fixos. Em uma especificação flexível, é permitido que estes variem desde que eles respeitem certos relacionamentos especificados no documento.

A vantagem do uso de especificações temporais flexíveis é a de facilitar o uso de técnicas de relaxamento ou aceleração das apresentações com propósitos de sincronização, auxiliando na derivação de um escalonamento da apresentação, como discutido em [27].

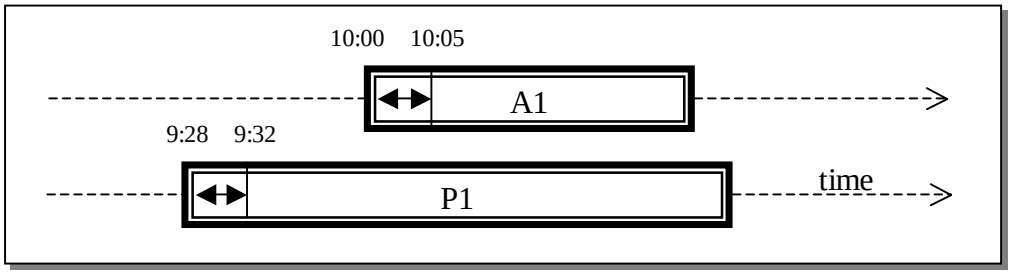


Figura 3.1 - Especificação temporal flexível

A figura 3.1, por exemplo, mostra dois objetos posicionados nas suas respectivas linhas de tempo. Esta figura ilustra a seguinte especificação:

- *comece a tocar o áudio A1 em algum instante entre 10:00 e 10:05; e*
- *comece a apresentar a figura P1 em algum instante entre 9:28 e 9:32.*

3.4 Linguagens de descrição de apresentações multimídia

Com relação à criação de apresentações multimídia, não há um consenso ou padrão amplamente aceito para a especificação dos documentos a serem recuperados e/ou apresentados via servidores remotos. Algumas linguagem de descrição de apresentações multimídia têm se destacado no âmbito da Internet graças a ferramentas desenvolvidas por companhias especializadas em multimídia como a *Macromedia* e a *RealNetworks*, e ao empenho do *W3Consortium* em definir recomendações e padrões para a área de multimídia e Internet.

A *RealNetworks* definiu as linguagens RealPix e RealText para criar apresentações que podem ser transferidas a partir de servidores remotos. Os arquivos RealPix e RealText são criados com suas respectivas linguagem de marcação (*markup*), simples e semelhantes ao HTML. Com a RealPix podem ser criadas apresentações no estilo *slide-show* combinando imagens em diferentes formatos (GIF89a, GIF87, JPEG, Iterated StiNG e Bitmap) através de efeitos especiais de transição entre duas imagens. Já com a RealText podem ser criadas apresentações textuais que se comportam de acordo com um estilo escolhido (Generic, ScrollingNews, TickerTape, Marquee ou TelePrompter).

3.4.1 A linguagem SMIL

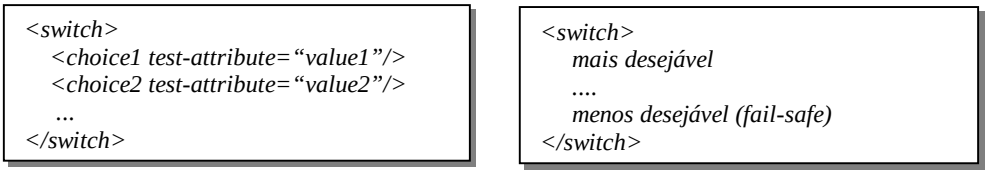
Recentemente, surgiu uma proposta que tem sido guiada por grandes nomes da área de multimídia e Internet, como *Lucent/Bell Labs, RealNetworks, Netscape, CWI, Phillips*, entre outros. Esta proposta é a linguagem SMIL (*Synchronized Multimedia Integration Language*) que está sendo desenvolvida pelo W3C, o qual liberou a especificação de sua versão 1.0 através de uma recomendação em junho de 1998 [28].

A linguagem SMIL (se pronuncia *smile*) permite integrar um conjunto de objetos multimídia independentes em uma apresentação multimídia sincronizada através de uma especificação textual, com uso de *tags* e muito similar ao HTML, sendo também baseada em XML [29]. A SMIL vem ganhando espaço muito rapidamente com o apoio de algumas ferramentas de edição e apresentação que já estão sendo disponibilizadas.

Desta forma, a linguagem SMIL foi adotada como base para a especificação dos documentos multimídia criados no ambiente do Projeto ServiMídia. Entretanto, certas características que compõem os objetivos deste projeto, como a especificação de uma adaptabilidade dinâmica para os documentos, não são consideradas pela linguagem SMIL.

Na verdade, SMIL define a *tag* <switch> que permite um grau limitado de adaptação. Com a *tag* **<*switch*>,** o autor pode descrever múltiplas opções dentre as quais a ferramenta de apresentação (*player*) deve escolher uma para ser apresentada. O grupo **<*switch*>** especifica qualquer número de opções de escolha no formato apresentado na figura 3.2a.

A escolha de qual *clip* apresentar é feita da seguinte forma: o *player* examina as opções na ordem em que aparecem, avaliando cada atributo de teste e seus valores para selecionar um *clip* válido. Desta forma, o autor deve ordenar as alternativas partindo da mais desejável para a menos desejável. Além disso, o autor deve especificar uma opção que estaria relativamente livre de falhas como a última opção da lista, figura 3.2b. A *tag* <***switch***> realiza a escolha com base em variáveis estáticas que são configuradas ou armazenadas pelas ferramentas de apresentação dos clientes. Dentre os atributos de teste definidos na especificação da linguagem SMIL [28] constam: *system-bitrate, system-caption, system-language, system-screen-size, system-screen-depth,* entre outros.



(a) (b)

Figura 3.2 - Estrutura do grupo <*switch*>

A seguir, são apresentados os conceitos mais abordados na literatura com relação à adaptação dos documentos multimídia. Na maioria das vezes, estes documentos são especificados através de alguma linguagem de integração como a SMIL, ou da combinação de mais de uma.

3.5 Em busca do acesso universal

O conceito de acesso universal (Universal Multimedia Access – UMA) é adotado pelo grupo de desenvolvimento do MPEG-7 [30, 31]. A idéia básica do acesso universal é permitir que os clientes com capacidades limitadas de comunicação, processamento, armazenamento e apresentação, possam acessar os ricos conteúdos multimídia disponíveis atualmente na Internet. O usuário requer acesso universal às informações em qualquer lugar, a qualquer hora, usando qualquer tipo de cliente e de rede de comunicação. O objetivo dos sistemas de acesso universal é criar diferentes apresentações da mesma informação para atender diferentes formatos, equipamentos e redes, a partir de uma mesma base de informação. Assim, a tarefa é disponibilizar a mesma informação através da escolha apropriada dos elementos de conteúdo. Para realizar esta tarefa e disponibilizar o UMA a qualquer cliente é necessário um mecanismo de adaptação para adaptar os conteúdos multimídia às diferentes capacidades dos clientes, como mostrado na figura 3.3.

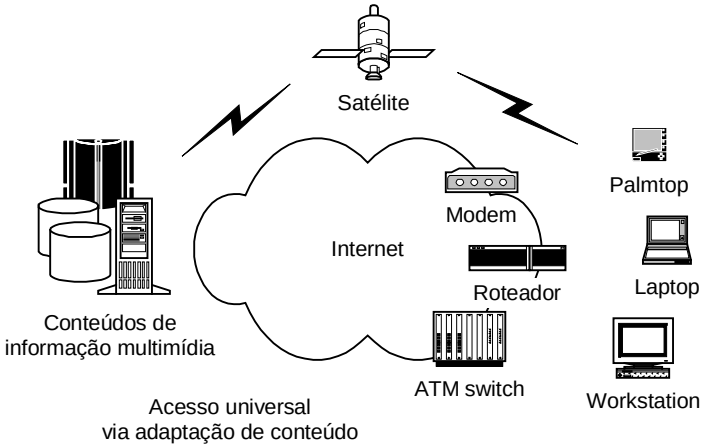


Figura 3.3 - Acesso universal aos conteúdos multimídia pela Internet

Recentemente, algumas soluções têm se focado na adaptação dos conteúdos multimídia.

Bulterman [32] propôs o conceito de canais para lidar com a adaptação de apresentações multimídias centradas em decisões do usuário. Um canal agrupa várias trilhas de mídias independentes que o usuário pode ligar ou desligar provendo assim alguma forma de adaptação. O conceito imita os *TV broadcasts*, tendo trilhas de som em diferentes idiomas – o usuário pode escolher uma trilha desejada. Os canais são úteis, entretanto, eles são limitados ao fato de que as trilhas devem ser independentes e correr em paralelo, ocupando recursos na rede de comunicação.

Quando uma conferência é feita em multicast no MBONE, importantes benefícios de performance podem ser alcançados com o uso de "mídia em camadas" (*Layered Multicast*) [33, 34, 35], onde a codificação é baseada no princípio de sucessivos refinamentos da qualidade. A informação básica é enviada primeiro, então, camadas de refinamento adicionam mais informação para incrementar a qualidade da apresentação. Em geral, uma árvore multiponto é utilizada para cada camada gerada pela fonte. Destinos (clientes) podem se associar a um ou mais grupos multipontos de acordo com os seus requisitos e com a disponibilidade de banda. Mais uma vez, o usuário decide quais camadas ele deseja receber, provendo certo grau de adaptação.

No contexto dos servidores de dados multimídia, quando recuperando os resultados de uma busca através de uma apresentação multimídia sincronizada, esta pode ser criada de acordo com as preferências do usuário [36].

Uma forma de se alcançar o acesso universal é proposta pela IBM através de um framework [37] e de um mecanismo para se descrever as características dos usuários [38]. Estas características são utilizadas na adaptação dos conteúdos dos documentos.

3.5.1 Framework para a adaptação de conteúdo

Em [37], um grupo de pesquisadores da IBM apresenta um *framework* para a adaptação de conteúdo na Internet. Os autores seguem uma abordagem bastante comum e que tem se mostrado a mais efetiva. O acesso universal é alcançado através da conversão dos itens de conteúdo dos documentos em modalidades e resoluções alternativas, de maneira que possam se adequar às características dos clientes. Isto é, são oferecidas

múltiplas versões de cada objeto multimídia através da transcodificação destes objetos em diferentes modalidades (e.g., vídeo, imagem, texto e áudio) e em diferentes fidelidades (resoluções), como mostrado na figura 3.4.

Os itens de conteúdo podem ser transcodificados em diferentes versões antecipadamente e armazenados nos servidores da maneira desejada. No instante da entrega, as versões transcodificadas a serem transmitidas são então selecionadas. Alternativamente, os conteúdos também podem ser transcodificados durante a transferência. Segundo este *framework*, a adaptação de conteúdo pode ser realizada em vários nós da rede: no servidor, no cliente, em um *proxy* entre o servidor e o cliente, ou distribuída entre o servidor, o cliente, e os *proxies*.

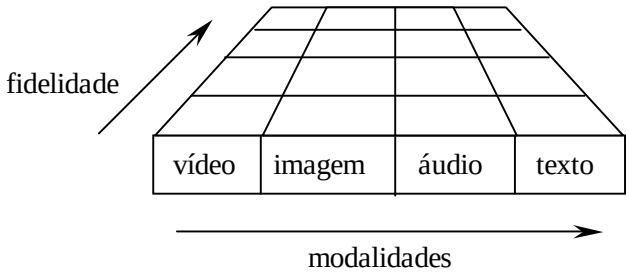


Figura 3.4 - Diferentes versões de um objeto multimídia

O *framework* é composto pelos componentes descritos a seguir.

***Análise* –** o conteúdo é analisado para se extrair meta-dados, como requisitos de recursos, além de tipo e propósito do conteúdo. Estes meta-dados são usados como guia para os subsequentes processos de transcodificação e seleção.

***Transcodificação* –** baseado nas características dos clientes, diferentes módulos de transcodificação são empregados para gerar versões dos conteúdos em diferentes modalidades e resoluções.

***Seleção* –** um módulo de seleção escolhe as versões mais indicadas às características do cliente, uma vez que cada versão de conteúdo requer diferentes níveis de recursos do cliente.

Renderização **–** o conteúdo selecionado é então *"renderizado"* em um formato de entrega adequado (por exemplo HTML ou outra linguagem de integração baseada em XML) para ser enviado ao cliente.

Descrição **–** um esquema ou modelo de descrição dos conteúdos deve ser usado para representar as múltiplas modalidades e resoluções dos dados transcodificados, em conjunto com quaisquer meta-dados associados.

3.5.2 Descrevendo as características do usuário

Em [38] é apresentado o desenvolvimento de um serviço de perfis (*profiles*) para descrever as características (capacidades) e preferências dos clientes conectados na Internet. Um perfil CC/PP (*Composite Capability/Preference Profile*) é uma coleção de características e preferências associadas a um cliente e aos agentes utilizados pelo cliente para acessar a *web*. Estes agentes incluem a plataforma de *hardware* e de *software* e as aplicações utilizadas pelo cliente. As características e preferências destes agentes podem ser vistas como metadados ou descrições das propriedades dos agentes de *hardware* e de *software*.

Esta proposta [38] está focalizada no desenvolvimento de um serviço de perfis baseado no XML/RDF. O RDF (*Resource Description Format*) [39] foi desenvolvido pelo *W3Consortium* para descrever as chamadas "*machine understandable properties"* dos conteúdos *web*. A proposta dos CC/PPs explora o uso do RDF para descrever as características e preferências associadas aos clientes e aos agentes de *hardware* e de *software* utilizados para acessar a *web*. Ela está principalmente direcionada para permitir a adaptação de conteúdo e o acesso universal aos conteúdos da *web* por diversos equipamentos, como os da nova geração de celulares e *palmpilots* (G3). Espera-se que o uso de uma tecnologia comum para codificar tanto o conteúdo quanto as características do cliente encoraje a adoção desta tecnologia e simplifique o uso de metadados na *web*. Ferramentas poderosas para lidar com XML e RDF, algumas das quais já estão em pleno desenvolvimento, estarão disponíveis em breve.

O modelo básico de dados para um perfil CC/PP é uma coleção de tabelas. Na forma mais simples, cada tabela em um perfil CC/PP é uma coleção de declarações RDF com propriedades atômicas (*propriedade = valor*). O exemplo a seguir mostra alguns componentes principais (*hardware*, *software* e aplicação) de um perfil:

Hardware Platform
Memory = 64mb
CPU = PPC
Screen = 640*400*8
BlueTooth = Yes
Software Platform
OS version = 1.0
HTML version = 4.0
WML version = 1.0
Sound = ON
Images = Yes
Email
Language = English
...

Alguns conjuntos de propriedades e seus valores podem ser comuns a um componente particular. Por exemplo, um modelo específico de *"smart phone"* (cliente) pode vir com uma CPU, tamanho de tela e memória padrão de fábrica. Combinando estas propriedades padrão em um recurso RDF distinto, torna-se possível a recuperação destas propriedades de forma independente das demais e o seu armazenamento em *cache*. Assim, ao invés de enumerar cada conjunto de propriedades, uma referência remota pode ser usada para um conjunto de propriedades padrão, tais como as propriedades de *hardware*. Isso se mostra muito útil quando o *link* entre o *gateway/proxy* e o cliente é lento e o *link* entre o *gateway/proxy* e o site referenciado remotamente é rápido – um caso típico quando o cliente está conectado via *modem* ou é um *"smart phone"*.

O exemplo a seguir mostra um perfil CC/PP especificado em XML/RDF que utiliza referências remotas para descrever algumas das propriedades padrão de um equipamento (cliente), as modificações e as preferências adicionadas pelo usuário:

```
-----------------------------------------------------
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:prf="http://www.w3.org/TR/WD-profile-vocabulary#">
  <rdf:Description about="HardwarePlatform">
      <prf:Defaults>
            <rdf:li resource="http://www.nokia.com/profiles/2160"/>
      </prf:Defaults>
       <prf:Modifications
            Memory="32mB"/>
   </rdf:Description>
   <rdf:Description about="SoftwarePlatform">
     <prf:Defaults>
            <rdf:li resource="http://www.symbian.com/profiles/pda"/>
     </prf:Defaults>
```

```
     <prf:Modifications
            Sound="Off"
            Images="Off" />
   </rdf:Description>
   <rdf:Description about="UserPreferences">
     <prf:Defaults
            Language="English" />
</rdf:Description>
</rdf:RDF>
-----------------------------------------------------
```

O perfil fornecido pelo fabricante do hardware
(em http://www.nokia.com/profiles/2160):

```
-----------------------------------------------------
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <rdf:Description
      Vendor="Nokia"
      Model="2160"
      Type="PDA"
      ScreenSize="800x600x24"
      CPU="PPC"
      Keyboard="Yes"
      Memory="16mB"
      Bluetooth="YES"
      Speaker="Yes" />
</rdf:RDF>
-----------------------------------------------------
```

O perfil fornecido p/ o sistema de software
(em http://www.symbian.com/profiles/pda):

```
-----------------------------------------------------
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <rdf:Description
      OS="EPOC1.0"
      HTMLVersion="4.0"
      JavaScriptVersion="4.0"
      WAPVersion="1.0"
      WMLScript="1.0" />
</rdf:RDF>
-----------------------------------------------------
```

3.6 Resumo

Neste capítulo, mostra-se como a especificação de documentos multimídia é realizada através de uma linguagem de descrição, como a linguagem SMIL, que aborda três aspectos fundamentais: estruturação lógica da apresentação, estabelecimento de relações temporais e espaciais entre os objetos multimídia que a compõem. Os conceitos básicos envolvidos com a adaptação de conteúdo também são apresentados através do framework proposto em [37] e do método proposto em [38] para descrever as características dos usuários.

Capítulo 4

Uma Estratégia de Composição de Documentos Multimídia Dinâmicos

As aplicações multimídia sobre a Internet necessitam que serviços adicionais sejam oferecidos para prover o suporte adequado ao tráfego das diversas mídias. O principal problema abordado neste trabalho é a diferença verificada na disponibilidade de recursos de comunicação durante a fase de apresentação de um documento multimídia distribuído em um ambiente de ensino à distância, visto que um certo grau de indisponibilidade pode degradar uma mídia de tal forma que a mesma se torna incompreensível, perdendo seu sentido.

Neste capítulo, as idéias propostas pelo *framework* [37] apresentado no capítulo anterior são discutidas, bem como as características e deficiências da linguagem SMIL no que diz respeito à adaptação de conteúdo. Uma estratégia de autoria adaptativa, que constitui a principal contribuição deste trabalho, é apresentada. Esta estratégia tem o objetivo principal de descrever formas alternativas para a apresentação do documento e os critérios de seleção, baseados em relacionamentos semânticos, que definem os momentos de ativá-las.

4.1 Discutindo a adaptação de conteúdo

Embora as idéias presentes no *framework* [37] busquem adaptar os documentos multimídia às diferentes características dos clientes, deve-se notar que esta adaptação é realizada apenas no instante de requisição dos documentos. Quando um documento multimídia é solicitado ao servidor, as mídias que o compõem são analisadas e a estrutura de descrição é percorrida em busca das versões que melhor se adaptem às características do cliente. O documento é então processado com as versões selecionadas e enviado ao cliente solicitante. A partir deste momento, nenhuma outra adaptação é realizada.

Esta abordagem pode ser útil para documentos que contêm mídias discretas, como figuras e textos, visto que uma mídia discreta é transferida integralmente (no instante da requisição) antes de ser apresentada. Já para documentos que contêm mídias contínuas, como áudio e vídeo, um conjunto de elementos (mídias) selecionado no instante de requisição do documento pode ser o mais adequado apenas naquele instante. Visto que uma mídia contínua é transferida gradativamnente conforme é apresentada, em um determinado instante ao longo da apresentação, o conjunto de mídias mais adequado pode ser diferente daquele que foi selecionado no instante de requisição do documento. Esta abordagem pode ser útil para documentos como páginas HTML que apresentam, em sua maioria, mídias discretas. Por outro lado, esta abordagem não se mostra efetiva para documentos como apresentações SMIL que possuem, em sua maioria, mídias contínuas.

Outro ponto interessante a se destacar é que as mídias que compõem o documento são analisadas e suas versões são selecionadas independentemente uma das outras. Uma vez que estas adaptações não seguem nenhuma especificação de relacionamentos entre as mídias, elas podem resultar em um documento onde a combinação das versões selecionadas alcança uma baixa qualidade de percepção por parte do usuário.

É importante notar que a preocupação com a manutenção da coerência de uma apresentação (QoP), associada ao controle de degradação da QoS da mesma, não tem sido explorada pelas arquiteturas de sistemas multimídia distribuídos atuais.

O módulo de descrição do *framework* [37] corresponde a uma estrutura de dados capaz de representar e associar as diferentes versões de um conteúdo multimídia armazenado no servidor. Algumas informações semânticas, como a importância ou prioridade dos elementos, são mantidas nesta estrutura com o objetivo de seleção. Porém, estas informações continuam sendo isoladas para cada elemento, não havendo informações que relacionem os diferentes elementos de conteúdo de um documento.

Esta descrição pode corresponder a uma estrutura gerenciada por um banco de dados ou a um arquivo de anotação. No primeiro caso, o banco de dados é pesquisado em busca das versões e a adaptação é feita pelo servidor. No segundo caso, o arquivo de anotação é transferido ao cliente junto com o documento que ele referencia e a adaptação é

feita pelo cliente. O segundo caso mostra uma melhor escalabilidade, flexibilidade e performance em relação ao primeiro, visto que o processamento, análise e seleção dos elementos podem ser distribuídos, além da possibilidade de *cache* em nós de rede intermediários (*proxy*). Porém, dependendo do número de versões que tenham sido produzidas (transcodificadas) para cada objeto, descrever as diversas alternativas tornaria o arquivo de anotação bastante extenso e difícil de ser especificado pelo autor.

Com relação à linguagem SMIL, o elemento *switch* corresponde a uma evolução do conceito de *"choice composite"* presente no modelo de documentos CMIF [40]. Entretanto, com o *switch*, a escolha de uma alternativa depende de um atributo de teste. A adaptação também é limitada ao início da apresentação de um objeto. Na verdade, a utilização de uma condição fixa para se selecionar a apresentação de uma informação não é a forma ideal de especificar uma adaptação, pois as condições dependem da plataforma e do tipo de conexão que um usuário pode ter com o servidor de mídias. Se estas condições se basearem em variáveis fixas que são definidas pelo usuário na sua ferramenta de apresentação, o número de combinações possíveis dos atributos pode ser muito grande. Como as condições da apresentação são conhecidas somente durante o decorrer da mesma, uma adaptação da apresentação deve se dar com base em condições dinâmicas a serem monitoradas ao longo da apresentação.

A maioria dos trabalhos que lidam com a questão da adaptação de apresentações multimídia realiza a adaptação através de decisões tomadas pelo usuário, seja no instante de pedido do documento ao servidor (*content negotiation*), seja na verificação de variáveis fixas (*user's profiles*), seja na seleção de canais ou filtros (*proxys*). Porém, pode-se verificar uma deficiência com relação à especificação de uma adaptabilidade baseada nos requisitos de QoS e de QoP dos objetos que compõem o documento. Este trabalho propõe a especificação combinada dos requisitos de QoS e da possível adaptação que o documento pode sofrer. A adaptação é especificada através de relacionamentos condicionais (*links* condicionais) que descrevem a coerência do documento. Esta combinação de requisitos de QoS e *links* condicionais tem como objetivo realizar uma adaptação coerente da apresentação resultando na sua correta interpretação pelo usuário (QoP).

4.2 Relacionamentos condicionais

A maioria das propostas de adaptabilidade trata os objetos de uma mesma apresentação de forma independente. Essas adaptações ao nível dos objeto, ou ao nível da codificação, são limitadas a uma estrutura lógica estática do documento. Além disso, da maneira como estas alternativas têm sido especificadas, a adaptação é realizada ao se apresentar uma ou mais alternativas de um objeto, mas não é possível interromper outros objetos como parte desta mesma adaptação.

Para realizar uma adaptação no nível da aplicação, realizando uma reestruturação coerente no formato do documento (de forma a preservar a QoP do documento), este trabalho propõe a especificação de dependências (relacionamentos) condicionais entre os diferentes objetos. As dependências condicionais foram propostas [41] com o objetivo de se tirar vantagem do conhecimento de relações semânticas entre os diversos objetos de mídias. As dependências condicionais expressam relações de causalidade entre os objetos ajudando a descrever os requisitos de seleção de um objeto com relação a outros objetos. Para um melhor entendimento, considere o exemplo abaixo:

- comece a tocar o áudio A1 em algum instante entre 10:00 e 10:05 e comece a tocá-lo apenas se a figura P1 já foi iniciada;

- comece a apresentar a figura P1 em algum instante entre 9:28 e 9:32 e continue mostrando até que o áudio A1 termine.

Nos dois casos acima existe um relacionamento condicional entre os objetos A1 e P1. Na primeira declaração, tem-se uma especificação relacionada ao instante de início da apresentação de P1. Desta forma, a apresentação de A1 está condicionada ao fato da apresentação de P1 ter sido iniciada. Na segunda declaração, tem-se uma especificação relacionada ao instante de fim da apresentação de A1. Se, por exemplo, a figura P1 não puder ser apresentada por algum motivo, a apresentação do áudio A1 será descartada, economizando os recursos da rede. Note que o descarte da informação é realizado com base nas especificações fornecidas pelo autor do documento, resultando em uma

apresentação coerente ao mesmo tempo que não sobrecarrega a rede com informações que se tornaram desnecessárias (perderam o significado).

Uma especificação temporal flexível é obtida com o estabelecimento de margens de tolerância para o início da apresentação dos objetos, isto é, estabelecendo um intervalo temporal no início de um objeto que indique que a apresentação deste pode ser iniciada em qualquer instante dentro deste intervalo. Desta forma, os aspectos temporais são expressos por uma margem de tolerância para o início da apresentação mais a duração desejada para o respectivo objeto. Este trabalho não se preocupa com a definição explícita de relacionamentos temporais entre as unidades de informação internas ao conteúdo dos objetos, o que seria uma tarefa complicada para os autores da apresentação, que visualizam o documento em um nível mais abstrato.

4.3 A Metodologia de adaptação proposta

Com o objetivo de suavizar o impacto das adaptações sobre os usuários e, ao mesmo tempo, garantir que a grande variedade de alternativas disponíveis para cada objeto possam ser utilizadas, são adotados os dois níveis de adaptação descritos a seguir.

4.3.1 Níveis de adaptação

No primeiro nível ocorre uma adaptação ao longo do eixo das fidelidades (resoluções) de um certo objeto. Neste nível, uma informação codificada em um conjunto de diferentes resoluções de um mesmo tipo de mídia pode, durante a apresentação do documento, ter a sua apresentação chaveada entre as resoluções deste conjunto. Neste nível de adaptação, os usuários não são incomodados pelo processo de adaptação pois a estrutura inicial do documentos se mantém e apenas a qualidade das mídias é alterada.

No segundo nível ocorre uma adaptação ao longo do eixo das modalidades (tipos de mídia) de um certo objeto. Neste nível de adaptação, os usuários são capazes de perceber que houve uma adaptação, pois, ao se alterar o tipo de mídia de uma informação, dificilmente é mantida a estrutura inicial do documento. Porém, para que ocorra uma

adaptação coerente da estrutura do documento, as relações de dependência condicional devem ser especificadas entre as diferentes alternativas.

Uma “alternativa” é constituída por uma modalidade (adaptação de nível 2) incrementada por um descritor dos requisitos máximos e mínimos de fidelidade solicitados por aquela modalidade. Este descritor descreve a faixa que pode ser percorrida ao longo do eixo das fidelidades (adaptação de nível 1). A figura 4.1 apresenta os dois níveis de adaptação possíveis. Note que uma modalidade pode ser repartida em mais de uma faixa de fidelidade (a modalidade B, por exemplo, possui duas faixas de fidelidade). Desta forma, as alternativas especificadas podem referenciar diferentes faixas de fidelidade dentro da mesma modalidade.

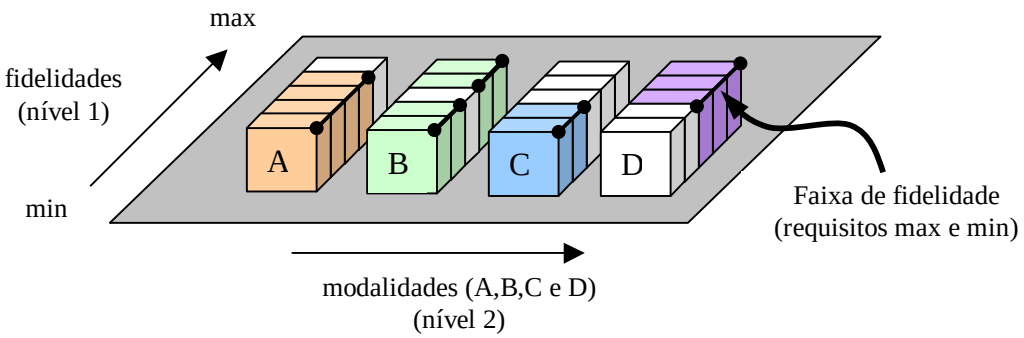


Figura 4.1 - Dois níveis de adaptação onde cada par (modalidade + req. de fidelidade) representa uma alternativa que pode ser usada na adaptação do documento.

As relações de dependência são especificadas através da definição de *links* condicionais que interconectam os objetos. Estes *links* formam uma malha de causalidade que descreve a consistência desejada para o documento ao mesmo tempo que possibilita uma adaptação coerente da apresentação.

Um objeto *Link* consiste de um *LinkSource* e um *LinkTarget*. O *LinkSource*, por sua vez, consiste de um conjunto de condições associadas a diferentes objetos. O *LinkTarget* é associado a um único objeto que é encapsulado por uma *LinkAction* e uma *LinkDexpr*. A *LinkDexpr* combina as condições em uma expressão de dependência para descrever os requisitos a garantir durante a apresentação do objeto. Quando um *link* é disparado, sua *LinkDexpr* é verificada e quando esta se torna verdadeira a *LinkAction* é executada. A figura 4.2 representa um *link* com um *LinkSource* onde existem duas

condições (uma associada à modalidade B do objeto “quadrado” e outra associada à modalidade C do objeto “círculo”) e um *LinkTarget* com uma certa *LinkAction* associado à modalidade A do objeto “cruz”.

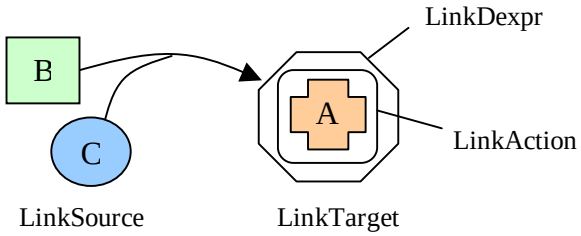


Figura 4.2 - Representação simbólica de um *link* condicional

Dois tipos de *links* foram definidos: ***startlink*** e ***stoplink***, com as *LinkAction* definidas como “apresentar objeto” e “interromper objeto”, respectivamente. Em uma apresentação multimídia, a sincronização temporal também deve ser considerada, logo, os *links* são disparados uma única vez durante a apresentação do documento. Assim, quando o instante de início da apresentação de um objeto for atingido, seu *startlink* é disparado e todas as condições do *startlink* são verificadas durante o intervalo de tolerância para o início deste objeto. Se o *startlink* é disparado com sucesso, o *stoplink* é então disparado em seguida e passa a ter as suas condições verificadas. Tanto a sincronização temporal quanto a causal devem ser respeitadas.

Condições		
Estados	***Descrição***	***Exemplo***
started	verdadeiro se a apresentação do objeto foi iniciada	`(P1:started)`
stopped	verdadeiro se a apresentação do objeto foi interrompida	

LinkDexpr	
Descrição	***Exemplo***
corresponde a uma expressão Booleana que combina as condições descrevendo os relacionamentos condicionais a serem respeitados para a ativação do respectivo *link*.	`(A1:stopped) and (P1:started)`

Links		
Tipos	***Descrição***	***Exemplo***
startlink	estabelece um *link* condicional para o início da apresentação de um objeto.	`startlink = “(A1:stopped) and (P1:started)”`
stoplink	estabelece um *link* condicional para que se interrompa a apresentação de um objeto.	

Tabela 4.1 – Especificando relacionamentos causais (*links*)

A tabela 4.1 apresenta os possíveis valores (estados) que podem ser testados através dos *links* condicionais, como estes podem ser combinados através da LinkDexpr, a sintaxe empregada em cada caso e um exemplo de especificação.

4.3.2 Gerando arquivos de anotação

Um documento multimídia original, criado para ser apresentado a um cliente específico (por exemplo, usando uma estação multimídia e estando conectado através de uma LAN) pode estar associado a anotações que fornecem dicas de como adaptar este documento para outros clientes menos favorecidos, seja pelo sistema local ou pela conexão de rede. O objetivo da anotação de um documento multimídia é disponibilizar dicas para que uma política de adaptação tome decisões mais inteligentes com relação à adaptação de conteúdo, dadas as características do cliente. As anotações podem ir desde as mais simples, como a importância dos elementos (elementos pouco importantes podem ser ignorados quando o espaço de apresentação for pequeno), até as mais complexas, como a especificação de mídias diferentes para cada tipo de cliente.

Uma abordagem possível é definir um conjunto de novas *tags* e atributos para anotação e combiná-los diretamente dentro da especificação do documento multimídia. Em [42], são definidas extensões da sintaxe da linguagem SMIL com o objetivo de implementar os relacionamentos condicionais. Entretanto, esta abordagem possui algumas limitações. As linguagens utilizadas para se especificar os documentos (como a SMIL) são normalmente padrões internacionais estabelecidos, além de já serem suficientemente complicadas. A tarefa de estender estas linguagens e incorporar as extensões nos seus respectivos padrões se mostra extremamente difícil. Ainda, se tais extensões fossem aceitas, as aplicações atuais (*players e browsers*) não seriam capazes de tratar estas novas *tags*.

O uso de anotações externas aos documentos exige tarefas adicionais de gerenciamento mas, em contra-partida, ele tem a vantagem de não exigir nenhuma modificação nos conteúdos já existentes. Assim, a anotação externa é muito importante para que os documentos multimídia já publicados na web possam ser adaptados na busca

pelo acesso universal. Por fim, o uso de anotação externa pode oferecer vantagens em termos de latência e banda passante através de *caching* nos elementos de rede (*proxies*).

Ao longo do desenvolvimento deste projeto, as vantagens verificadas no uso dos arquivos de anotação fizeram com que a estratégia de anotação externa fosse adotada. Assim, os mesmos *Links* definidos em [42] para especificar os relacionamentos condicionais são utilizados, agora, dentro dos arquivos de anotação.

Os arquivos de anotação possuem informações que estão ligadas aos elementos do documento original e podem descrever as alternativas àqueles elementos e os critérios de seleção para estas alternativas. Assim, um documento, ou qualquer conjunto dos seus elementos, pode ser fornecido com representações alternativas.

Os arquivos de anotação podem ser otimizados ao se descrever apenas as alternativas (*modalidade + req.de fidelidade/QoS*), pois estas descrevem intrinsicamente os dois níveis de adaptação. No arquivo de anotação, os *links* condicionais podem ser especificados entre as diferentes alternativas dos objetos multimídia que compõem o documento, como na figura 4.3.

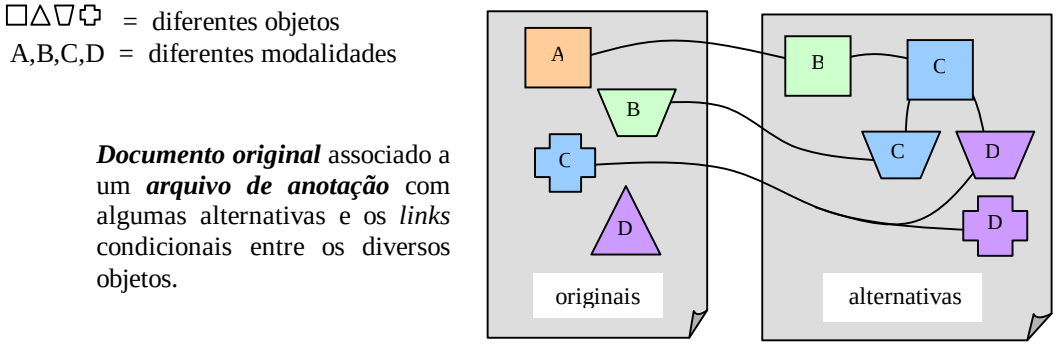


Figura 4.3 - *Links* condicionais entre os objetos multimídia

Assim, o arquivo de anotação é utilizado da seguinte forma: através dos requisitos de fidelidade especificados junto com cada alternativa ele guia a adaptação de nível 1; enquanto que, através dos relacionamentos semânticos entre as próprias alternativas, ele guia a adaptação de nível 2.

Desta forma, tem-se o seguinte andamento para a apresentação do documento: (i) os objetos do documento original iniciam a apresentação e seus requisitos de recursos começam a ser monitorados; (ii) através de um mecanismo de monitoramento e sinalização [43], uma degradação da qualidade é realizada ao longo da faixa de fidelidade especificada; (iii) se o limite mínimo for ultrapassado, uma alternativa deve ser selecionada para o respectivo objeto. Neste momento, a seleção de uma alternativa pode gerar, através dos *links* condicionais, a seleção de outras alternativas para os outros objetos do documento original, ou ainda, interromper a apresentação de algum objeto. Isso possibilita uma adaptação (seleção) mais coerente da estrutura do documento, ao invés de selecionar as alternativas de cada objeto independentemente.

4.3.3 A linguagem de anotação proposta (SMAL)

A SMIL 1.0, uma linguagem declarativa para especificação de apresentações multimídia, foi escolhida para especificar os documentos multimídia originais. Estes documentos apresentam a estrutura lógica ideal que seria apresentada por um sistema contendo todos os recursos solicitados.

Uma nova linguagem, denominada *SMIL Annotation Language* (SMAL), foi definida neste projeto para possibilitar a especificação dos arquivos de anotação. A SMAL permite a definição de uma estrutura de informações referentes às dependências condicionais entre os objetos. A SMAL é uma linguagem baseada em XML. O XML (*eXtensible Markup Language*) [29] é uma linguagem padrão reconhecida pelo W3C e uma maneira de formatação dos dados para o intercâmbio de documentos estruturados. Várias bibliotecas de classes e ferramentas de desenvolvimento têm sido construídas para suportar as definições de *parsers*, geradores e outras aplicações baseadas em XML.

A figura 4.4 apresenta um exemplo de um arquivo de anotação com o vocabulário da linguagem SMAL. O grupo **<controls>** é formado pelas informações de controle de cada objeto especificado por uma *tag* **<description>**. Os limites máximos e mínimos, que definem as faixas de fidelidade dentro das quais é realizada a adaptação de nível 1, são especificados através das *tags* **<maxRequirements>** e **<minRequirements>**. Além disso, a *tag* **<stoplink>** especifica um *link* condicional

descrevendo as condições que devem ocorrer para que o respectivo objeto seja removido da apresentação.

Um objeto, ou um conjunto de objetos, pode ser fornecido com representações alternativas que são descritas no arquivo de anotação. O grupo **<alternatives>** especifica um conjunto de representações alternatives para vários objetos da apresentação. Cada elemento alternativo é especificado em uma *tag* **<replace>**. Esta *tag* possui um elemento **<startlink>** associado que especifica sob que condições a alternativa deve ser ativada. O elemento **<resourceToSubstitute>** especifica o recurso (alternativa) que deve ser apresentado no lugar de outro recurso especificado pelo elemento **<target>** da *tag* **<replace>**. O elemento **<regionToAdd>** especifica uma nova região a ser adicionada ao *layout* da apresentação para que a alternativa possa ser apresentada caso esta necessite.

```
<smal>
  <controls>
    <description about=”a2”>
      <minRequirements bw=”8kbps”>
      <maxRequirements bw=”16kbps”>
      <stoplink expr=”(v2:stopped)”>
    </description>
  </controls>
  <alternatives>
    <replace target=”v2”>
      <startlink expr=”(v2:stopped)”>
      <regionToAdd>
        <![CDATA[
          <region id=”rp3” title=”rp3” ... /> ]]>
      </regionToAdd>
      <resourceToSubstitute>
        <![CDATA[
          <animation id=”p3” region=”rp3” ... /> ]]>
      </resourceToSubstitute>
    </replace>
  </alternatives>
</smal>
```

Figure 4.4 - Emprego da linguagem SMAL em um arquivo de anotação

4.4 Exemplo de aplicação

Um exemplo simples na área de treinamento à distância é descrito nesta seção. A aplicação de treinamento está dividida em duas fases (dois arquivos SMIL): uma fase de introdução (os servidores enviam informações aos estudantes de forma a apresentar um resumo do treinamento, os autores e *copyrights*, etc); e uma fase de treinamento (os servidores enviam informações de forma a apresentar o material do treinamento).

Na fase de introdução, um vídeo de abertura (“v1”) apresenta um resumo do treinamento a ser apresentado e a equipe responsável pelo seu desenvolvimento. O vídeo foi repartido em duas faixas de fidelidade (“v1A”e “v1B”). Na primeira faixa (“v1A”), a degradação da qualidade é feita gradualmente até que o limite desta faixa seja atingido. A partir deste ponto, um *link* condicional inicia a apresentação do vídeo na segunda faixa de fidelidade (“v1B”). Entretanto, os autores decidiram que, uma vez iniciada a faixa (“v1B”), uma informação textual complementar deve acompanhar o vídeo de forma a não prejudicar a QoP. Assim, outro *link* condicional conecta a faixa (“v1B”) ao texto (“t1”) para que este seja apresentado. A figura 4.5 mostra o processo de adaptação na fase de introdução e a figura 4.6 mostra o arquivo de anotação correspondente.

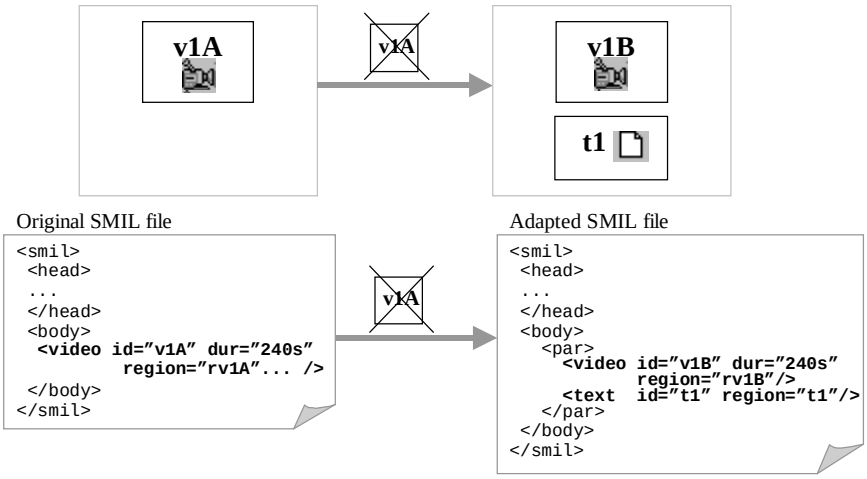


Figura 4.5 - Processo de adaptação (nível 2) na fase de introdução

```
<smal>
<controls>
  <description about=”v1A”>
    <minRequirements width=”360” height=”240” bpp=”16” tx=”20fr/s”/>
    <maxRequirements width=”360” height=”240” bpp=”16” tx=”30fr/s”/>
  </description>
  <description about=”v1B”>
    <minRequirements width=”360” height=”240” bpp=”8” tx=”5fr/s”/>
    <maxRequirements width=”360” height=”240” bpp=”16” tx=”10fr/s”/>
  </description>
</controls>
<alternatives>
  <replace target=”v1A”>
    <startlink expr=”(v1A:stopped)”>
    <regionToAdd>
      <![CDATA[
        <region id=”rv1B” title=”rv1B” ... />
        <region id=”rt1” title=”rt1” ... /> ]]>
    </regionToAdd>
    <resourceToSubstitute>
      <![CDATA[
        <par>
          <video id=”v1B” dur=”240s” region=”rv1B”/>
          <text  id=”t1” region=”t1”/>
        </par>        ]]>
    </resourceToSubstitute>
  </replace>
</alternatives>
</smal>
```

Figura 4.6 - Arquivo de anotação da fase de introdução

O arquivo de anotação da segunda apresentação SMIL, onde o material do treinamento é apresentado, especifica um cenário um pouco mais sofisticado, mostrado na figura 4.7. Na fase de treinamento, a apresentação contém um vídeo (“v2”), que apresenta o conteúdo da aula, um áudio (“a2”) relacionado ao vídeo, e uma figura (“p2”) que apresenta alguns diagramas e imagens. Nesta fase existe um forte requisito pela apresentação do vídeo. Em outras palavras, apresentar apenas a informação do áudio e da figura é considerada sem utilidade do ponto de vista da aplicação (ou do ponto de vista do autor). Então, se o vídeo não puder ser apresentado dentro do limite de fidelidade especificado, ele será substituído por uma figura (“p3”) mais um texto (“t1”) e o áudio (“a2”) será interrompido. Se apenas o áudio (“a2”) não puder ser apresentado, o vídeo será mantido e o texto (“t1”) será acrescentado. Porém, se o vídeo falhar após a falha do áudio, apenas a figura (“p3”) será acrescentada. A figura 4.8 mostra o arquivo de anotação que descreve os relacionamentos condicionais e os requisitos de QoS envolvendo os objetos desta fase.

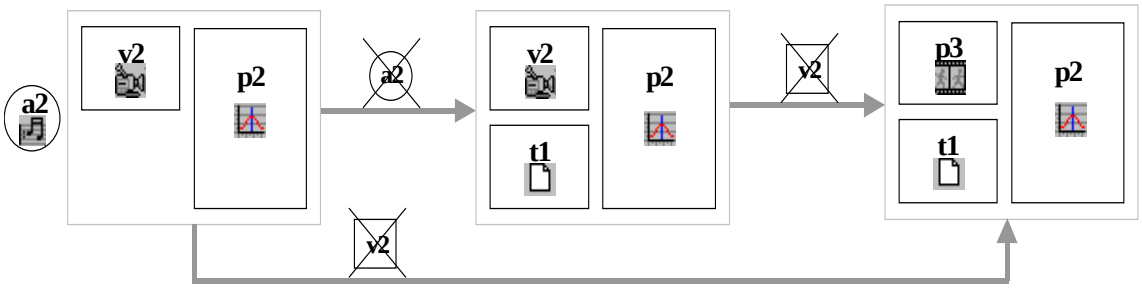


Figure 4.7 - Processo de adaptação (nível 2) na fase de treinamento

Neste trabalho, considera-se que em um ambiente de ensino a distância, onde as apresentações são na maioria das vezes relativamente longas, o usuário não será incomodado devido ao atraso adicional causado pelo mecanismo de adaptação, visto que a interpretação correta do documento é mais crítica do que o atraso. O aluno preferirá assistir uma apresentação coerente com algum atraso casual do que uma apresentação contínua com uma qualidade insatisfatória que gere uma interpretação equivocada.

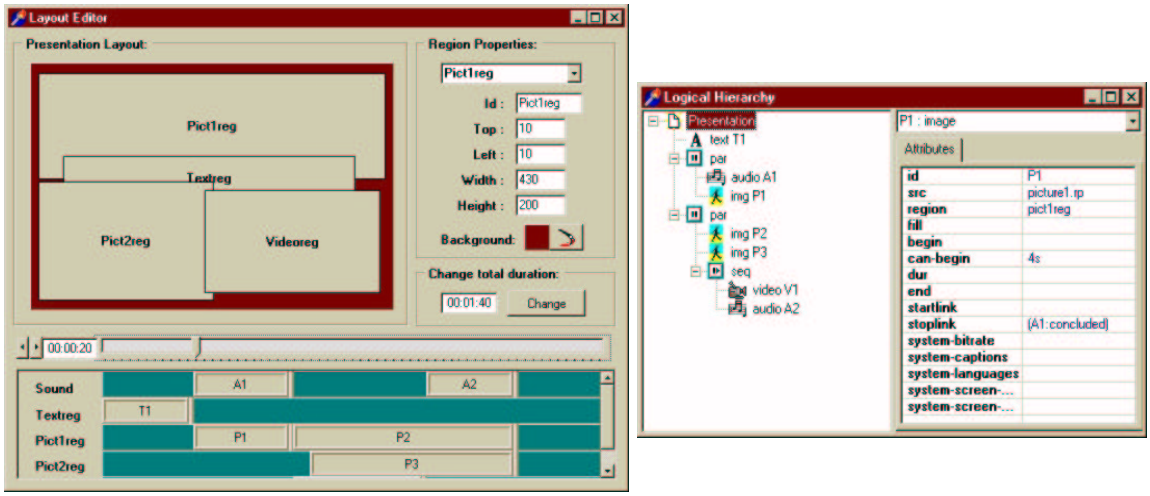
```
<smal>
<controls>
  <description about=”a2”>
    <minRequirements bw=”8kbps”>
    <maxRequirements bw=”16kbps”>
    <stoplink expr=”(v2:stopped)”>
  </description>
  <description about=”v2”>
    <minRequirements width=”220” height=”176” bpp=”8” tx=”15fr/s”/>
    <maxRequirements width=”352” height=”288” bpp=”24” tx=”25fr/s”/>
  </description>
  <description about=”p2”>
    <minRequirements width=”300” height=”400” bpp=”8” color=”bw”/>
    <maxRequirements width=”300” height=”400” bpp=”24” color=”color”/>
  </description>
  <description about=”p3”>
    <minRequirements width=”240” height=”240” bpp=”8” color=”color” tx=”2fr/s”/>
  </description>
</controls>
<alternatives>
  <replace target=”a2”>
    <startlink expr=”(a2:stopped)”/>
    <regionToAdd>
      <![CDATA[  <region id=”rt1” title=”rt1” ... />    ]]>
    </regionToAdd>
    <resourceToSubstitute>
      <![CDATA[  <text id=”t1” region=”rt1”   ... />    ]]>
    </resourceToSubstitute>
  </replace>
  <replace target=”v2”>
    <startlink expr=”(v2:stopped)”>
    <regionToAdd>
      <![CDATA[  <region id=”rp3” title=”rp3” ... />    ]]>
    </regionToAdd>
    <resourceToSubstitute>
      <![CDATA[  <animation id=”p3” region=”rp3” ... /> ]]>
    </resourceToSubstitute>
  </replace>
</alternatives>
</smal>
```

Figura 4.8 - Arquivo de anotação da fase de treinamento

4.5 Uma ferramenta de autoria

Uma visão geral da ferramenta de autoria que foi desenvolvida neste projeto é apresentada nesta seção. Inicialmente, esta ferramenta foi criada para implementar a estratégia proposta utilizando a linguagem SMIL incrementada com as extensões que foram definidas em [42]. Esta ferramenta foi desenvolvida utilizando o ambiente de desenvolvimento de aplicações Delphi 4.0 da Inprise™ para a plataforma Windows98®.

A figura 4.9a ilustra uma das interfaces da ferramenta de autoria. Através desta interface, o autor especifica o *layout* da apresentação definindo as regiões de *playback* onde os *clips* devem ser apresentados. Para cada nova região é adicionada uma linha de tempo onde podem ser vistos os objetos (*clips*). Uma única linha temporal para a trilha sonora da apresentação está sempre presente, podendo, entretanto, permanecer vazia. Através desta ferramenta, é possível visualizar o andamento da apresentação através dos eixos temporais tendo uma idéia da duração de cada objeto.



(a) (b)

Figura 4.9 - (a) interface do editor de *layout*; (b) interface do editor da apresentação

A autoria de documentos mais complexos onde o *layout* mude com frequência, por exemplo, entre capítulos e seções de um documento maior, pode ser alcançada através da edição de um sub-documento (com seu respectivo *layout*) para cada seção e a ligação destes através de um documento principal.

A estrutura lógica para a apresentação está baseada no conceito de grupos de *clips*, onde os *clips* são os objetos das mídias que compõem o documento. Estes grupos são representados em uma árvore hierárquica, figura 4.10, semelhante a uma árvore de diretórios, onde cada grupo pode conter *clips* ou outros grupos, assim como um diretório pode conter arquivos ou outros diretórios. Os grupos de *clips* podem ser de dois tipos: *grupo paralelo*, onde os *clips* devem ser apresentados em paralelo; e *grupo sequencial*, onde os *clips* devem ser apresentados em sequência. Na realidade, o próprio documento é considerado um grupo sequencial, pois os elementos no nível logo abaixo dele devem ser apresentados em sequência.

Nesta ferramenta de autoria, em uma outra interface, mostrada na figura 4.9b, o autor especifica a estrutura lógica da apresentação. Nesta interface é possível criar novos grupos (par, seq e switch), novos *clips*, organizá-los, e definir todos os seus atributos e propriedades temporais/causais. Todas as alterações feitas na estrutura da apresentação,

nos conjuntos de *clips* e suas propriedades é refletido na visualização do eixo temporal de cada região do *layout* apresentado na figura 4.9a.

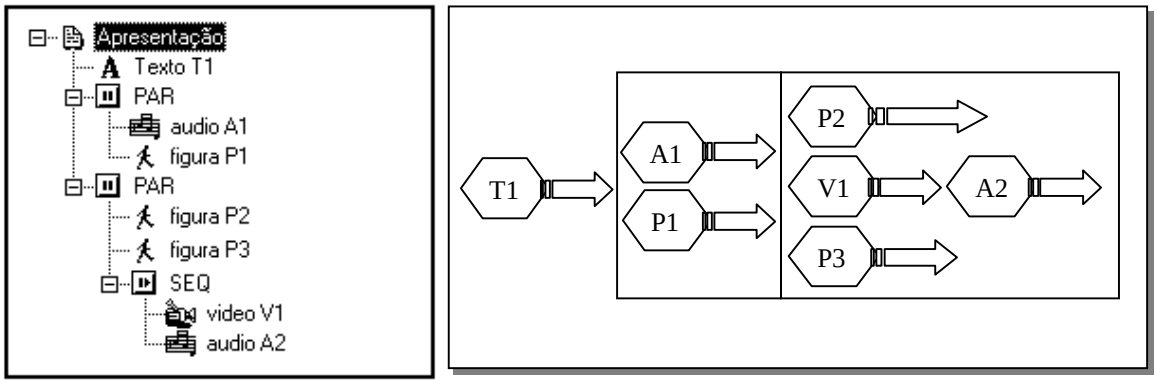


Figura 4.10 - Estrutura lógica da apresentação

4.6 Resumo

Neste capítulo são discutidas as idéias presentes no *framework* [37] apresentado no capítulo anterior, além das características e deficiências da linguagem SMIL no que diz respeito à adaptação de conteúdo. Uma estratégia de autoria é apresentada buscando alcançar uma adaptação dos conteúdos multimídia em dois níveis. Primeiramente, de uma forma suave ao longo das diferentes resoluções disponíveis para uma mídia e, posteriormente, de uma forma estrutural, alterando os tipos de mídias e a estrutura do documento baseado em relacionamentos semânticos entre as alternativas.

Também é apresentada a linguagem de anotação (SMAL) definida para se implementar a estratégia de autoria. Através da SMAL pode-se especificar um arquivo de anotação que descreve as alternativas e os relacionamentos entre elas de maneira a realizar uma adaptação coerente da apresentação.

Capítulo 5

Avaliação da Estratégia de Adaptação

Com o intuito de verificar o comportamento geral da estratégia proposta neste trabalho e, em particular, o desempenho do mecanismo de adaptação da estrutura de um documento multimídia, foram realizados testes práticos em um ambiente de rede corporativa. O comportamento do mecanismo de adaptação foi avaliado através da implementação de uma aplicação servidora de vídeo e de uma aplicação cliente para solicitar os arquivos de vídeo ao servidor. O sistema foi todo implementado em Java utilizando a API *"The Java Media Framework 2.0 (JMF2.0)"*. A seção 5.1 descreve a arquitetura de comunicação desenvolvida para o projeto Servimídia. Na seção 5.2 são apresentadas algumas características básicas da JMF2.0 que são importantes para o entendimento e discussão dos resultados obtidos. Na seção 5.3 é apresentada a arquitetura do sistema implementado. Os resultados obtidos são apresentados na seção 5.4.

5.1 Arquitetura de comunicação

A arquitetura de comunicação do projeto ServiMídia [43, 44] é composta pelos componentes participantes do processo de apresentação e adaptação de um documento multimídia a partir de servidores remotos:

- O Servidor de documentos (*DocServer*) é a entidade que fornece a especificação do documento multimídia. Um agente localizado no servidor de documentos recebe os pedidos de restituição e, baseado nas informações obtidas a partir de uma base de informação de distribuição das mídias (*Distribution Information Base - DIB*), ele decide quais servidores de mídias devem ser sinalizados para iniciar a transmissão dos fluxos pertencentes ao documento solicitado pelo cliente. Feito isso, ele compõe automaticamente a descrição da apresentação inserindo as RTSP URL's dos fluxos para os servidores selecionados.

- A base de dados (*Distribution Information Base - DIB*) armazena informações referentes a localização das mídias nos diversos servidores de mídias, bem como informações sobre a topologia e sobre os recursos disponíveis da rede. Estas informações contribuem para uma escolha dos servidores de mídias que leve a um melhor balanceamento da carga dos dados fluindo na rede.

- Os Servidores de mídias (*MediaServers*) armazenam as informações digitais das mídias (fluxos multimídia). Os servidores de mídias podem existir em qualquer número e recebem as solicitações RTSP feitas pelo cliente para a transmissão dos fluxos.

- O Cliente (*MediaClient*) é o responsável por dar a partida no processamento de um documento ao solicitar sua restituição ao *DocServer*. O endereço do *DocServer* deve ser previamente registrado em todos os clientes deste ambiente. Após ter recebido a descrição da apresentação, a aplicação localizada no cliente inicia a comunicação com os *MediaServers*. Para gerenciar a recuperação dos documentos multimídia e preservar os relacionamentos e requisitos dos fluxos, a transferência dos fluxos das mídias é realizada utilizando a família de protocolos de tempo real, composta pelo RTP/RTCP e pelo RTSP.

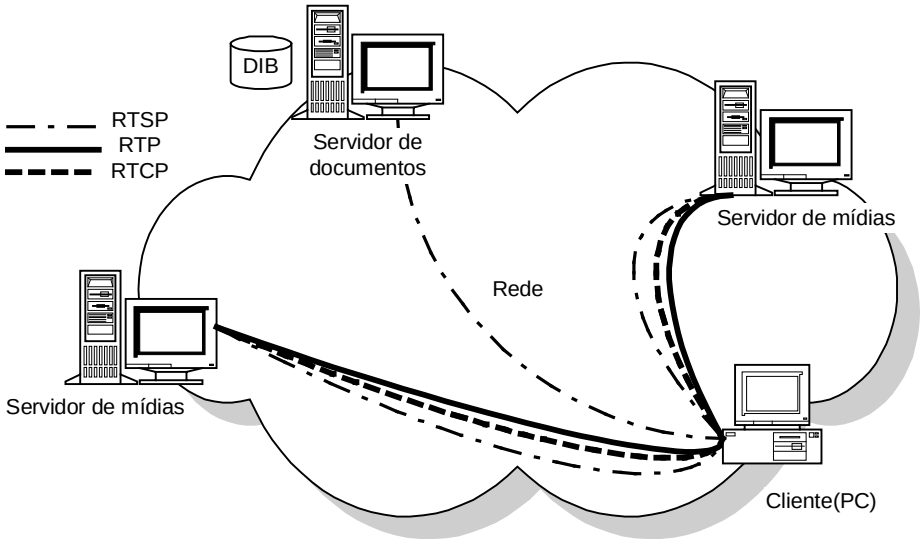


Figura 5.1 - Arquitetura de Comunicação

Os mecanismos de adaptação são baseados em um monitoramento, em tempo real, dos requisitos de QoS dos fluxos com 3 objetivos básicos: (1) gerenciar a transmissão dos fluxos; (2) verificar se os requisitos de QoS estão sendo respeitados e adaptar a qualidade dos fluxos de acordo com a condição da rede (adaptação de nível 1); (3) adaptar a estrutura do documento com base nas relações de dependência entre os fluxos (adaptação de nível 2).

Assim, foram definidos o Monitor do Descritor de Qualidade (Quality Descriptor Monitor – QDM) e o Monitor dos Relacionamentos do Fluxo (Stream Relationships Monitor – SRM). O QDM é responsável pelo monitoramento das mensagens de *feedback* fornecidas pelo protocolo RTCP com o objetivo de informar os parâmetros de qualidade instantânea percebidos na rede (porcentagems de perdas, atrasos e jitter). Com base nestes parâmetros, a adaptação de nível 1 é realizada ao longo da faixa de fidelidade estabelecida pelo descritor de qualidade. O SRM é responsável por monitorar os relacionamentos dos fluxos que tenham ultrapassado os limites do descritor de qualidade e por realizar a adaptação da estrutura do documento (nível 2) seguindo os *links* condicionais.

Existem pelo menos três razões para se incluir o SRM no sistema de apresentação implementado no lado cliente.

- O sistema de apresentação possui as informações (contidas na descrição da apresentação) sobre as interdependências dos fluxos e, portanto, sabe quais relacionamentos condicionais devem ser monitorados.
- Uma vez obtida a descrição da apresentação, as sessões RTSP são estabelecidas entre o cliente e os servidores. Assim, o cliente pode monitorar a transferência dos fluxos através das mensagens RTCP e enviar as mensagens de requisição RTSP para parar, suspender ou iniciar os fluxos. Se o cliente é responsável por iniciar a transmissão de um fluxo, ele deve ser capaz de controlar essa transmissão.
- Uma configuração descentralizada de controle é adotada ao se realizar o monitoramento no sistema de apresentação, onde cada cliente é responsável por monitorar as sessões (fluxos) que estejam estabelecidas entre ele e os servidores.

5.2 Características da JMF2.0

A *Java Media Framework (JMF)* é uma API que permite aos programadores desenvolver aplicações em Java para capturar, processar e apresentar dados multimídia, além de prover suporte à transmissão e à recepção destes dados através da rede com o uso do protocolo RTP. A JMF constitui a base da implementação realizada neste trabalho e consiste de uma arquitetura de alto nível com funcionalidades importantes no que se refere à manipulação de dados multimídia.

5.2.1 Apresentação de dados multimídia

Na JMF, o processo de apresentação é modelado pela interface *Controller* (controlador). A interface *Controller* define o estado básico e o mecanismo de controle para um objeto que processa, apresenta ou captura uma mídia. Esta interface define as fases que um objeto *Controller* deve percorrer e provê um mecanismo para gerenciar a transição entre elas.

Um objeto *Controller* sinaliza uma variedade de eventos específicos (*MediaEvents*) para notificar as mudanças no seu estado. Para receber notificações a partir de um objeto *Controller*, a interface *ControllerListener* deve ser implementada. A JMF define dois tipos de objetos C*ontrollers*: *Players* e *Processors*. Ambos são construídos a partir de uma fonte de dados (*DataSource*) particular.

Apresentadores (*Players*)

Um objeto *Player* processa um fluxo de entrada de dados e o apresenta em um instante preciso. Um objeto *DataSource* é utilizado para fornecer o fluxo de entrada ao objeto *Player*, como mostrado na figura 5.1, e o destino dos dados processados depende do tipo de mídia sendo apresentada.

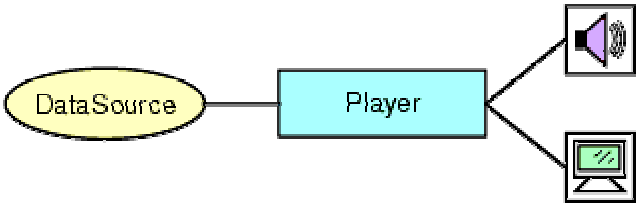


Figura 5.1 - Modelo de um objeto *player* JMF

Um objeto *Player* pode se encontrar em um dentre cinco estados possíveis, como mostrado na figura 5.2. A interface *Clock* define os dois estados primários: *Stopped* e *Started*. Para facilitar o gerenciamento de recursos, a interface *Controller* quebra o estado *Stopped* em cinco estados de preparação: *Unrealized*, *Realizing*, *Realized*, *Prefetching* e *Prefetched*.

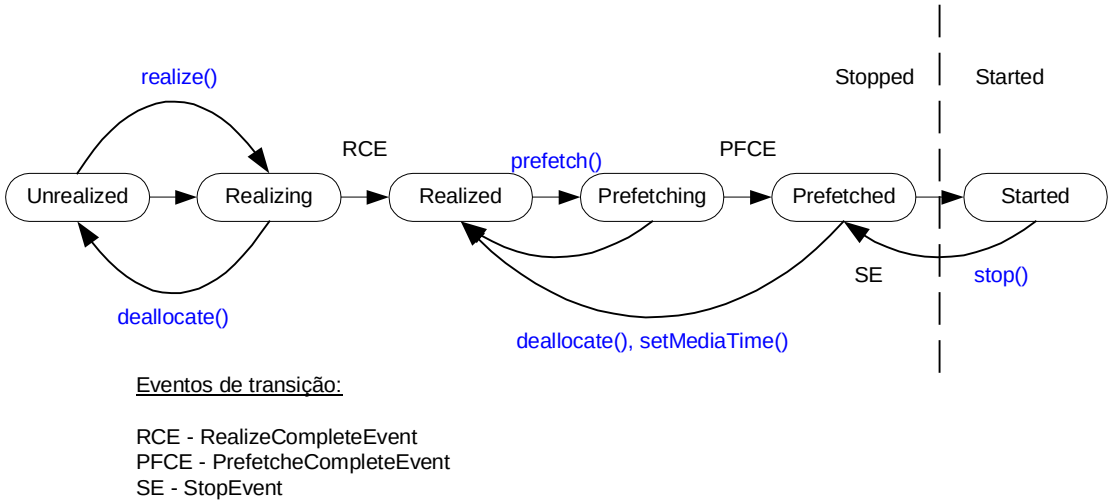


Figura 5.2 - Estados de um objeto *Player*

Em uma operação normal, um objeto *Player* caminha através de cada estado até atingir o estado *Started*:

- Um objeto *Player* no estado *Unrealized* acabou de ser instanciado, mas ainda não possui informação alguma sobre a sua mídia.

- Quando o método *realize()* é chamado, o objeto *Player* sai do estado *Unrealized* e se move para o estado *Realizing*. Neste estado, o objeto *Player* está em processo de determinar os seus requisitos de recursos.

- Quando o objeto *Player* finaliza o estado *Realizing*, ele se move para o estado *Realized*. Neste estado, o objeto *Player* conhece os recursos de que precisa e as informações sobre o tipo de mídia que ele deve apresentar. Sabendo como processar os dados que lhe são entregues, o objeto *Player* já pode fornecer componentes e controles visuais.

- Quando o método *prefetch()* é chamado, o objeto *Player* sai do estado *Realized* e se move para o estado *Prefetching*. Neste estado, o objeto *Player* está se preparando para apresentar a sua mídia. Durante esta fase, o objeto *Player* carrega os primeiros dados da mídia e faz tudo que for necessário para se preparar para a apresentação.

- Quando o objeto *Player* finaliza o estado *Prefetching*, ele se move para o estado *Prefetched* e está pronto para iniciar a apresentação.

- Uma chamanda ao método *start()* coloca o objeto *Player* no estado *Started*. Neste estado, o tempo da mídia é mapeado e o seu relógio é disparado, embora o objeto *Player* possa estar esperando por um instante específico para iniciar a apresentação dos dados.

Processadores (*Processors*)

Um objeto *Processor* é uma especialização de um objeto *Player* que toma uma fonte de dados (*DataSource*) como entrada, realiza algum processamento definido pelo programador sobre os dados e, então, fornece os dados processados. Enquanto o processamento realizado por um objeto *Player* é predefinido pela implementação da JMF, um objeto *Processor* permite ao programador definir o tipo de processamento que é aplicado aos dados. Um objeto *Processor* pode enviar os dados para um dispositivo de apresentação, fazendo o papel de um objeto *Player*, ou para um objeto *DataSource*. Se os dados são enviados para um objeto *DataSource*, este último pode ser usado como fonte de entrada para outro objeto *Player* ou *Processor*, formando uma cadeia de objetos *Controller*.

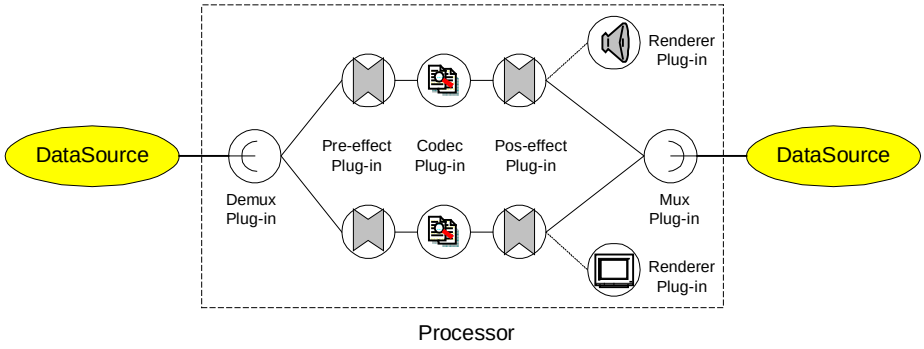


Figura 5.3 - Modelo de um objeto *processor* JMF

Um objeto *Processor* possui dois estados de preparação adicionais aos estados de um objeto *Player*: *Configuring* e *Configured.* Estes estados ocorrem antes do objeto *Processor* entrar no estado *Realizing*, como mostrado na figura 5.4.

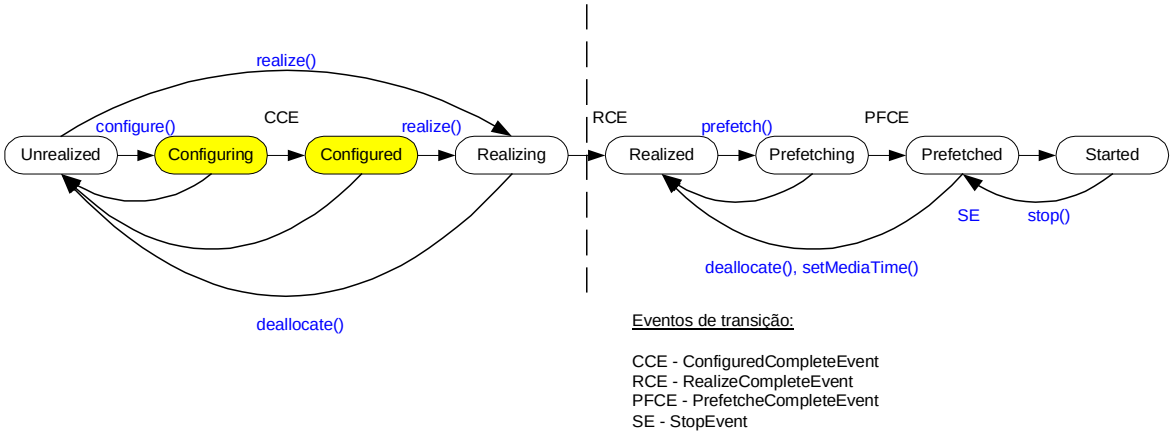


Figura 5.4 - Estados de um objeto *Processor*

- Um objeto *Processor* entra no estado *Configuring* quando o método *configure()* é chamado. Neste estado, ele se conecta ao objeto *DataSource*, demultiplexa o fluxo de entrada e acessa as informações sobre o formato dos dados de entrada.

- O objeto *Processor* se move para o estado *Configured* quando ele se conectou ao objeto *DataSource* e o formato dos dados foi deteminado. Neste estado, podem ser especificadas as operações que devem ser realizadas nas trilhas presentes no fluxo de entrada.

- Quando o método *realize()* é chamado, o objeto *Processor* é transferido para o estado *Realized* e está completamente construído.

Um objeto *Player*, assim como um objeto *Processor*, sinaliza os eventos de transição (*TransitionEvents*) conforme ele se move de um estado para outro. A interface *ControllerListener* provê uma forma de determinar em que estado um objeto *Player* se encontra. Por exemplo, quando um programa chama um método assíncrono em um objeto *Player* ou *Processor*, ele precisa escutar a sinalização do evento apropriado para determinar quando a operação é completada. O apêndice C contém a relação de eventos sinalizados por um objeto *Controller*.

5.2.2 Transmissão de dados multimídia

A JMF 2.0 torna possível a transmissão e recepção de fluxos RTP. Os objetos *Player* e *Processor* são utilizados para apresentar e processar os dados enviados pelos fluxos RTP. Na JMF, um gerente de sessão (*SessionManager*) é utilizado para coordenar uma sessão RTP, como mostrado na figura 5.5. Este gerente da sessão mantém controle sobre os participantes da sessão e sobre os fluxos que são transmitidos e recebidos na sessão RTP.

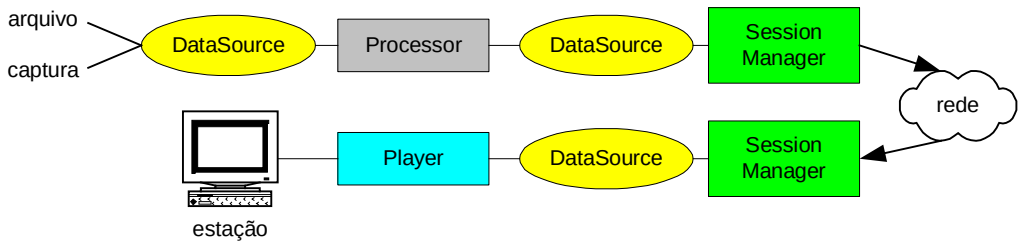


Figura 5.5 - Transmissão e recepção de fluxos RTP na JMF 2.0

Gerentes de sessão *(SessionManagers*)

O gerente de sessão (*SessionManager*) é a entidade responsável pelo gerenciamento de uma sessão RTP aberta para realizar uma comunicação de dados multimídia. Ele mantém o estado da sessão assim como visto pelo participante local, isto é, o gerente de sessão (*SessionManager*) é a representação local de uma entidade distribuída, a sessão RTP.

No momento da criação e inicialização de um objeto *SessionManager*, os endereços local e remoto da sessão RTP devem ser definidos. O endereço local pode ser fornecido pelo usuário ou ser escolhido aleatoriamente pelo próprio objeto *SessionManager*. Este endereço local da sessão corresponde ao par de portas RTP/RTCP utilizados para transmissão de dados se o *host* local for um emissor. O endereço remoto da sessão corresponde ao par de portas RTP/RTCP na máquina remota para onde os pacotes de dados e controle são enviados, e também, ao par de portas na máquina local onde o objeto *SessionManager* fica "escutando" por pacotes vindos dos outros participantes da sessão RTP. O gerente de sessão (*SessionManager*) suporta o estabelecimento de sessões unicast ou multicast.

No momento da inicialização do objeto *SessionManager*, também é fornecido um nome canônico (CNAME), que identifica o usuário local que está criando o objeto *SessionManager*, e um identificador de fonte de sincronização (SSRC), que identifica o fluxo transmitido pelo objeto *SessionManager* se o *host* local for um emissor.

O objeto *SessionManager* controla todos os participantes da sessão, criando um novo participante sempre que um pacote RTCP chegar contendo uma mensagem SDES com um CNAME que ainda não tenha sido visto na sessão, identificando assim um novo participante remoto na sessão. Os participantes podem ser passivos, enviando apenas pacotes RTCP, ou ativos, enviando também um ou mais fluxos de dados RTP.

Cada participante pode ser a origem de um ou mais fluxos RTP, cada fluxo sendo identificado por um identificador de fonte de sincronização (SSRC) usado pela fonte do fluxo. Como o SSRC é associado ao objeto *SessionManager* no momento da inicialização, um participante que queira enviar dois fluxos simultâneos na mesma sessão RTP deve criar e inicializar dois objetos *SessionManager*s distintos, porém, com o mesmo endereço da sessão RTP. Além disso, esta sessão deve ser uma sessão multicast, visto que uma sessão unicast pode conter apenas um fluxo. A figura 5.6 esclarece as características do objeto *SessionManager*.

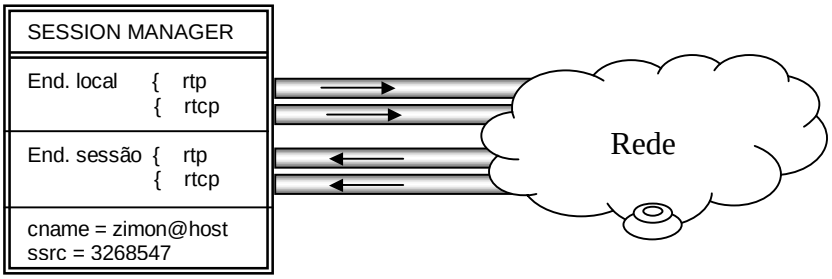


Figura 5.6 - Modelo de um objeto *Session Manager* JMF

O objeto *SessionManager* mantém um objeto *RTPStream* para cada fluxo de dados RTP que ele percebe na sessão. Existem dois tipos de objetos *RTPStream*: o objeto *ReceiveStream* representa o fluxo que está sendo recebido de um participante remoto; e o objeto *SendStream* representa o fluxo de dados vindo de um objeto *Processor* ou *DataSource* e que está sendo enviado para a rede se o *host* local for um emissor.

Vários eventos relacionados à comunicação RTP são utilizados para reportar o estado da sessão RTP e dos seus fluxos. Para receber as notificações destes eventos (*RTPEvents*), a interface *RTPListener* apropriada deve ser implementada e registrada no gerente da sessão (*SessionManager*). O apêndice C contém a relação dos eventos (*RTPEvents*) sinalizados pelo objeto *SessionManager* e as respectivas interfaces *RTPListener* que devem ser implementadas para capturá-los.

5.3 Implementação e testes

Esta seção descreve a arquitetura do sistema implementado para avaliar o desempenho da estratégia de adaptação proposta. Embora toda a arquitetura de comunicação do ambiente Servimídia tenha sido definida neste trabalho, a implementação e os testes realizados se concentraram na avaliação do tempo de resposta do sistema a uma solicitação de adaptação de conteúdo (chaveamento de um fluxo). Assim, foram implementadas uma aplicação servidora, responsável pela transmissão dos fluxos de vídeo, e uma aplicação cliente, responsável pela apresentação dos vídeos e pela solicitação da adaptação das mídias. Esta solicitação corresponde, neste contexto de simulações e testes, tanto a uma solicitação para adaptação de fidelidade (nível 1) quanto a uma solicitação

para adaptação da estrutura do documento (nível 2). Porém, na definição da arquitetura de comunicação do ambiente, o servidor é o responsável por decidir quando realizar a adaptação de nível 1, enquanto o cliente é responsável por solicitar apenas a adaptação de nível 2. A decisão de simular o sistema com o cliente solicitando os dois níveis de adaptação foi tomada para que os intervalos e tempos de resposta do sistema pudessem ser registrados adequadamente e para que a simulação ocorresse de forma controlada. O sistema foi desenvolvido utilizando o ambiente de desenvolvimento de aplicações JBuilder 3.0 em conjunto com a plataforma Java 2 e com a API JMF 2.0.

5.3.1 O servidor

A aplicação servidora é composta por 4 classes principais.

VideoServer – é a primeira classe a ser instanciada, e entra em execução criando um objeto *ServerSocket* na porta 1554 (TCP) e aguardando por pedidos de conexão vindos dos clientes. Quando um cliente solicita uma conexão ao servidor, este último aceita a conexão e cria um novo *socket* para se comunicar com o cliente. Esta conexão é utilizada como um canal de controle através do qual o cliente solicita uma mídia, ou a adaptação de uma mídia, ao servidor. Este canal de controle corresponde a uma sessão RTSP, embora o protocolo utilizado não seja exatamente o RTSP, mas sim, uma simplificação deste.

RTPTransmitter – é instanciada como uma nova linha de execução (*thread*) a cada vez que o *VideoServer* aceitar uma conexão com um novo cliente. Seu construtor recebe o *socket* criado pelo *VideoServer* para conectar o canal de controle ao cliente, e o *RTPTransmitter* fica sendo o responsável por receber e tratar as mensagens de controle vindas do cliente em questão.

SMWrapper – é instanciada por um objeto *RTPTransmitter* sempre que há a necessidade de estabelecer uma sessão RTP entre o servidor e um cliente. A classe *SMWrapper* encapsula o gerente de sessão (*SessionManager*) e implementa as interfaces *ReceiveStreamListener* e *SendStreamListener* para receber as sinalizações dos eventos relacionados à sessão RTP criada pelo seu gerente (*SessionManager*).

VideoTransmit – é instanciada por um objeto *RTPTransmitter* para transmitir um fluxo de vídeo em uma sessão RTP. O localizador para o arquivo de vídeo a ser transmitido e o gerente da sessão RTP (*SessionManager*) são passados para o construtor da classe *VideoTransmit*. Esta classe cria um objeto *Processor* para receber e processar o arquivo de vídeo antes de enviar os dados para o objeto *SessionManager* através da criação de um objeto *SendStream* na sessão RTP. Além disso, a classe *VideoTransmit* implementa a interface *ControllerListener* para receber as notificações de transição e gerenciar os estados do objeto *Processor* criado.

5.3.2 O cliente

A aplicação cliente também é composta por 4 classes principais.

RTPPlayer – é a primeira classe a ser instanciada e entra em execução. Quando o usuário decide receber um vídeo a partir do servidor, a classe *RTPPlayer* cria um *socket* e solicita a conexão do canal de controle à porta 1554 (TCP) do servidor.

SMWrapper – é instanciada por um objeto *RTPlayer* sempre que o cliente quiser estabelecer uma sessão RTP com o servidor. A classe *SMWrapper* encapsula o gerente de sessão (*SessionManager*) e implementa as interfaces *ReceiveStreamListener* e *SendStreamListener* para receber as sinalizações dos eventos relacionados à sessão RTP criada pelo seu gerente (*SessionManager*). O objeto *SessionManager* no cliente deve possuir o mesmo endereço de sessão usado pelo objeto *SessionManager* no servidor para que estes possam participar da mesma sessão.

PlayerWindow – é instanciada por um objeto *SMWrapper* quando este recebe uma sinalização do evento *NewReceiveStreamEvent*, indicando que um novo fluxo foi detectado na sessão RTP. Quando o objeto *SMWrapper* detecta este novo fluxo, ele cria um objeto *Player* para processar e apresentar este novo fluxo. O objeto *Player* criado, o respectivo fluxo, e o gerente (*SessionManager*) em questão são passados para o construtor da classe *PlayerWindow*. A classe *PlayerWindow* corresponde a uma janela onde os componentes visuais do objeto *Player* são adicionados e onde a apresentação do vídeo

acontece. Além disso, a classe *PayerWindow* implementa a interface *ControllerListener* para receber as notificações dos eventos relacionados ao objeto *Player* em questão.

QDM – é instanciada por um objeto *PlayerWindow* para monitorar os pacotes de *feedback* RTCP e os parâmetros de qualidade da transmissão do fluxo associado ao objeto *PlayerWindow*. Um gráfico com os valores de *jitter* e da porcentagem de perda de pacotes vai sendo atualizado continuamente. Além disso, os valores dos campos das mensagens *RTCP ReceiverReports* relacionadas ao fluxo em questão vão sendo registrados em um *log*. Este *log* é identificado pelo SSRC usado pela fonte do fluxo.

5.3.3 Características dos testes

Os testes foram realizados com o objetivo de avaliar o tempo de resposta do sistema a uma solicitação de adaptação de conteúdo (chaveamento de um fluxo). Vários instantes de tempo são registrados ao longo da execução dos testes para marcar os instantes de ocorrência de certos eventos, i.e., as aplicações (cliente e servidor) vão registrando o valor do relógio do sistema no instante de ocorrência de cada evento. Este log é mantido em memória para que não haja nenhuma interferência (atraso) no andamento das aplicações causada por vários acessos sucessivos ao hardware de disco. Os logs são transferidos para o disco apenas no momento em que a respectiva aplicação é encerrada. Em seguida, os logs gerados pelo servidor e pelo cliente são combinados em uma planilha para se avaliar os intervalos de tempo.

Os equipamentos utilizados foram dois PCs com processador Pentium-II 250MHz, 128MB de memória, e sistema operacional Windows NT Workstation 4.0 SP6. Estes dois computadores estavam conectados a um barramento Ethernet (100Mbits) compartilhado por mais três outros computadores, sendo um deles o gateway para outra subrede IP (fig.5.7). Os relógios dos dois PCs (servidor e cliente) foram mantidos sincronizados utilizando-se uma ferramenta de sincronização do relógio do sistema. Esta ferramenta foi instalada nos dois PCs, tendo sido configurada no cliente para sincronizar o relógio do sistema, a cada 5 minutos, com base no valor obtido do relógio do servidor. A cada nova sincronização dos relógios, passados 5 minutos após a sincronização anterior, os relógios

dos dois PCs apresentaram uma defasagem máxima de 0.2 ms, o que representa uma margem de erro bastante aceitável no contexto deste experimento.

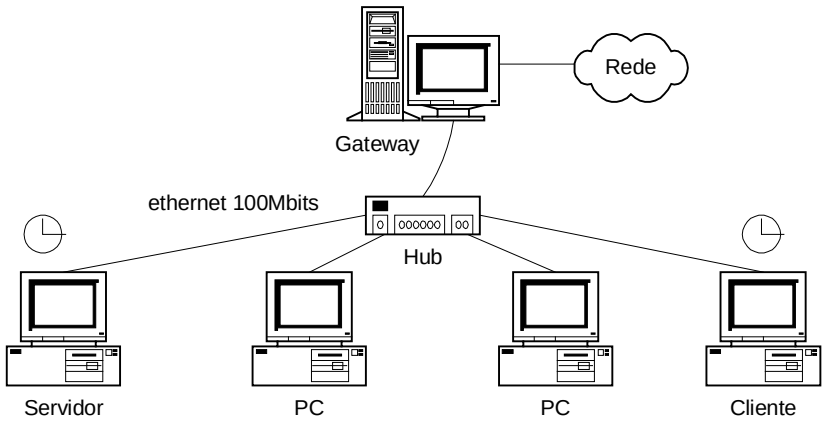
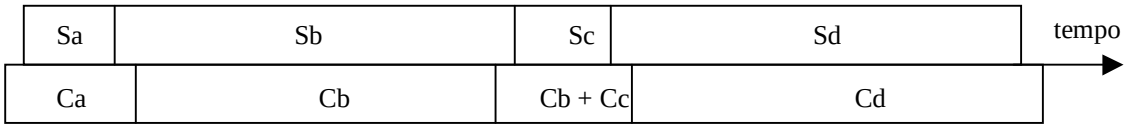


Figura 5.7 - Ambiente de simulação

Os testes se baseiam na avaliação dos intervalos de tempo entre cada um dos diversos eventos registrados no log para uma rodada de apresentação. Como mostrado na figura 5.8, uma rodada de apresentação compreende todo o processamento realizado desde a solicitação de um vídeo pelo cliente, o atendimento deste pedido pelo servidor, a apresentação deste vídeo pelo cliente, o pedido de chaveamento do vídeo inicial pelo cliente, o chaveamento propriamente dito feito pelo servidor, a apresentação do novo vídeo pelo cliente, até o encerramento da aplicação cliente.



Servidor :
Sa – processa pedido de *start* do video inicial
Sb – envia dados do vídeo inicial
Sc – processa pedido de *switch* para o vídeo final
Sd – envia dados do vídeo final

Cliente :
Ca – envia pedido de *start* e aguarda
Cb – recebe e apresenta dados do vídeo inicial
Cc – envia pedido de switch e aguarda
Cd – recebe e apresenta dados do vídeo final

Figura 5.8 - Etapas de uma rodada de apresentação

Durante uma rodada de apresentação, as aplicações cliente e servidora utilizam um protocolo de comunicação para controlar o andamento da comunicação. Este protocolo é baseado em memsagens simples de requisição no formato texto (tabela 5.1) que transportam os comandos do cliente para o servidor através do canal de controle. Este protocolo apresenta funcionalidades limitadas quando comparado ao RTSP, porém,

substitui algumas das funções do protocolo RTSP de forma suficiente para este experimento. As versões futuras da API JMF incluirão suporte ao RTSP, que poderá ser adicionado ao sistema sem muita dificuldade. Entretanto, a versão 2.0 da JMF ainda não possui este suporte, tendo sido necessária a implementação simplificada de um protocolo de controle.

Mensagem	Descrição
START (sp) session_IP (sp) session_PORT (sp) media_LOCATOR (sp) media_TIME	Solicita o início da transmissão do arquivo de vídeo identificado por *media_locator* na sessão RTP especificada.
SWITCH (sp) session_IP (sp) session_PORT (sp) flow_SSRC (sp) media_LOCATOR (sp) media_TIME	Solicita a substituição do fluxo identificado por *flow_SSRC* pelo arquivo de vídeo identificado por *media_locator* na sessão RTP especificada.
STOP (sp) SSRC (sp) media_TIME	Solicita a interrupção da transmissão do fluxo identificado por *flow_SSRC*.

(sp) = espaço

Tabela 5.1 - Mensagens de controle da transmissão

Os eventos que vão sendo registrados nos *logs* são aqueles relacionados tanto com as sessões RTP (*SessionManagers*) quanto com os objetos *Controllers* (*Players* e *Processors*) envolvidos. Os eventos relacionados com as sessões RTP são monitorados, no servidor e no cliente, pela classe *SMWrapper* implementada em cada um. Os eventos relacionados aos objetos *Controllers* são monitorados no servidor pela classe *VideoTransmit* e no cliente pela classe *PlayerWindow*, pois são elas que implementam a interface *ControllerListener*. A tabela 5.2 identifica cada um dos eventos que são registrados nos *logs* e os seus significados.

Evento registrado	Descrição do evento	Local
Creating Manager… / Manager created.	Instantes de início e conclusão da criação de um *SessionManager*. A diferença entre estes dois eventos é o tempo gasto pelo sistema para criar uma nova sessão RTP.	Servidor e Cliente
NewSendStreamEvent	Este evento indica o início da transmissão de um novo fluxo na sessão RTP.	Servidor
NewReceiveStreamEvent	Este evento indica o início do recebimento de um novo fluxo na sessão RTP.	Cliente
TimeoutEvent	Este evento indica que um participante deixou de participar da sessão RTP.	Servidor e Cliente
Creating VideoTransmit.../ VideoTransmit created.	Instantes de início e conclusão da criação de um *VideoTransmit*. A diferença entre estes dois eventos é o tempo gasto pelo sistema para criar um novo *VideoTransmit*.	Servidor

Creating Player… / Player create.	Instantes de início e conclusão da criação de um *Player*. A diferença entre estes dois eventos é o tempo gasto pelo sistema para criar um novo *Player*.	Cliente
Realizing Player...	Indica o início do processo de realização do *Player*.	Cliente
RealizeCompleteEvent	Indica a conclusão do processo de realização do *Player* e o início do processo de *prefetch*.	Cliente
PrefetchCompleteEvent	Indica a conclusão do processo de prefetch do *Player* e o instante de chamada do método *start()* do *Player* para que a apresentação do fluxo seja iniciada.	Cliente
StartEvent	Este evento indica que o *Player* iniciou com sucesso a apresentação do fluxo recebido na sessão RTP.	Cliente
KillOldPlayer	Indica o instante em que, durante o chaveamento, é iniciado o processo de destruição do antigo *Player*.	Cliente
SetNewPlayer	Indica o instante em que, durante o chaveamento, é iniciado o processo de substituição do antigo *Player*.	Cliente
StopEvent	Indica que a atividade do *Controller* foi paralizada.	Servidor e Cliente
ClosedEvent	Indica que o *Controller* foi fechado e não pode mais ser utilizado.	Servidor e Cliente
START,STOP e SWITCH	Indica o instante de recebimento (servidor) ou envio (cliente) de uma mensagem de controle.	Servidor e Cliente

Tabela 5.2 - Eventos registrados pelas aplicações

Nas transmissões de mídias sob demanda, as sessões entre servidores e clientes devem ser sessões unicast para que cada cliente possa ter o controle sobre os fluxos que ele solicitou aos servidores. O chaveamento de dois vídeos neste cenário pode se dar na mesma sessão RTP ou em duas sessões RTP distintas, isto é, o servidor pode interromper a transmissão do primeiro vídeo e iniciar a transmissão de um segundo fluxo na mesma sessão RTP ou em uma sessão RTP diferente.

Pode-se notar que essas duas opções estão diretamente relacionadas aos dois níveis de adaptação definidos neste trabalho e apresentados no capítulo 4. No primeiro nível de adaptação o servidor adapta o fluxo sendo transmitido ao longo do eixo das suas fidelidades e é interessante que a transmissão do novo fluxo ocorra na mesma sessão RTP usada pelo fluxo anterior. Já no segundo nível, a adaptação de um fluxo pode gerar a transmissão de mais de um fluxo simultaneamente. Neste caso, os fluxos simultâneos devem trafegar, obrigatoriamente, em sessões RTP distintas.

Para simular o comportamente do sistema no primeiro nível de adaptação, o cliente envia uma mensagem *SWITCH* para o servidor solicitando que ele troque o vídeo corrente por outro, como se o novo vídeo representasse uma variante de fidelidade do primeiro. A sessão RTP especificada na mensagem deve ser a mesma do fluxo que está sendo substituído. O servidor, então, faz a troca dos fluxos na mesma sessão RTP.

Para simular o comportamento do sistema no segundo nível de adaptação, o cliente envia uma mensagem *SWITCH* para o servidor solicitando que ele inicie a transmissão de um outro vídeo, como se o novo vídeo fosse uma alternativa ao primeiro. A sessão RTP especificada na mensagem deve ser diferente da sessão em que o primeiro fluxo está sendo transmitido.

Nos dois tipos de simulação acima, o fluxo sendo substituído é identificado por *flow_SSRC* enquanto o novo fluxo é identificado por *media_LOCATOR*. O campo *media_TIME* pode ser usado para fornecer o valor para o qual o relógio do novo fluxo deve ser ajustado antes que a transmissão inicie. Da mesma forma, o primeiro fluxo deve ser interrompido quando o seu relógio atingir o valor de *media_TIME*. Se o valor de *media_TIME* não é fornecido, o servidor utiliza o tempo corrente do relógio e faz o chaveamento imediatamente.

Uma característica importante do objeto *SessionManager* deve ser lembrada: um SSRC (identificador do fluxo) é associado ao objeto *SessionManager* no momento da inicialização. Assim, as duas opções acima devem ser implementadas utilizando, respectivamente, o mesmo objeto *SessionManager* ou objetos *SessionManagers* distintos.

5.3.4 Resultados dos testes

Nas tabelas 5.3 e 5.4 é apresentado o conteúdo de dois arquivos de *log* gerados a partir da simulação dos dois níveis de adaptação citados acima. Para cada um dos casos, os *logs* gerados no servidor e no cliente foram combinados para ordenar os eventos em uma linha temporal que apresentasse os eventos do servidor e do cliente em uma mesma planilha. A tabela 5.3 mostra os eventos do primeiro caso em que é utilizado o mesmo objeto *SessionManager* para adaptar o fluxo na mesma sessão RTP. Já a tabela 5.4 mostra

os eventos do segundo caso em que são utilizados dois objetos *SessionManager* para adaptar o fluxo em sessões RTP diferentes.

Note que nestas simulações, a avaliação do tempo de resposta do sistema está baseada, principalmente, na percepção do chaveamento dos fluxos por parte do usuário. Ou seja, considera-se como tempo de resposta do sistema o intervalo de tempo entre a paralização do primeiro vídeo na interface gráfica e o início da apresentação do segundo vídeo, pois estes eventos são percebidos pelo usuário.

Além dos arquivos de log gerados pelas próprias aplicações java, foi utilizado uma ferramenta de análise do tráfego da rede. Esta ferramenta não era capaz de identificar os pacotes dos protocolos RTP e RTCP, mas permitia a extensão dos seus arquivos de configuração para que novos protocolos pudessem ser analisados. Assim, os arquivos de configuração da ferramenta foram modificados utilizando a linguagem definida pelo frabricante, permitindo a captura e a análise dos pacotes dos protocolos RTP/RTCP.

Desta forma, foi possível analisar todos os campos dos protocolos envolvidos, além de acompanhar os instantes de recepção dos pacotes de dados RTP e identificar o tempo de chaveamento dos fluxos. Este procedimento também permitiu identificar quais eventos, registrados nos logs das aplicações, correspondiam à parada no recebimento dos pacotes RTP do primeiro fluxo e ao início da recepção dos pacotes RTP do segundo fluxo. As áreas sombreadas das tabelas identificam estes intervalos de chaveamento para os dois níveis de adaptação.

Caso 1: adaptação de nível 1 (mesma sessão)

No primeiro caso, ilustrado na figura 5.9, quando o chaveamento é solicitado pelo cliente (mensagem *SWITCH*), o servidor cria um objeto *VideoTransmit* para processar e transmitir o novo vídeo na mesma sessão RTP. Este novo vídeo é enviado para o cliente utilizando o mesmo identificador da fonte (SSRC), já que ele está sendo enviado pelo mesmo objeto *SessionManager*. Para o cliente, tudo se passa como se o fluxo que ele está recebendo fosse o mesmo. O cliente não é capaz de detectar a chegada de um novo fluxo, pois o SSRC é mantido e, portanto, o evento *NewReceiveStream* não é sinalizado. O objeto

Player que estava apresentando os dados do primeiro vídeo passa a apresentar os dados do novo vídeo, pois a sua fonte de dados está associada ao fluxo identificado pelo SSRC, o qual foi mantido. Não ocorre a criação de um novo objeto *Player* para apresentar o novo vídeo, o que reduz, notadamente, o tempo de chaveamento.

O gráfico apresentado na figura 5.10 mostra as curvas do tempo de chaveamento para 5 rodadas de simulação do primeiro nível de adaptação. Neste gráfico são registrados os tempos de ocorrência dos eventos a partir do evento de recepção da mensagem SWITCH.

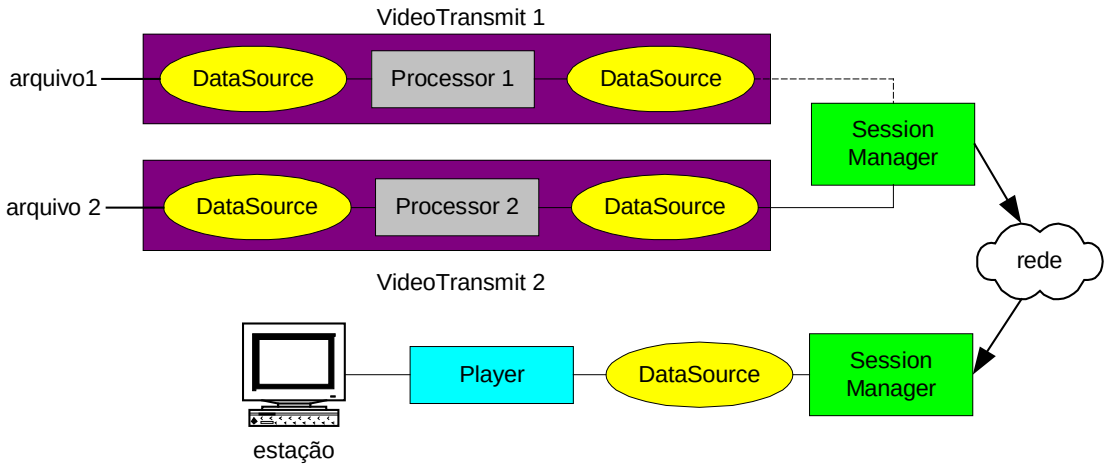


Figura 5.9 - Chaveamento usando o mesmo objeto *SessionManager*

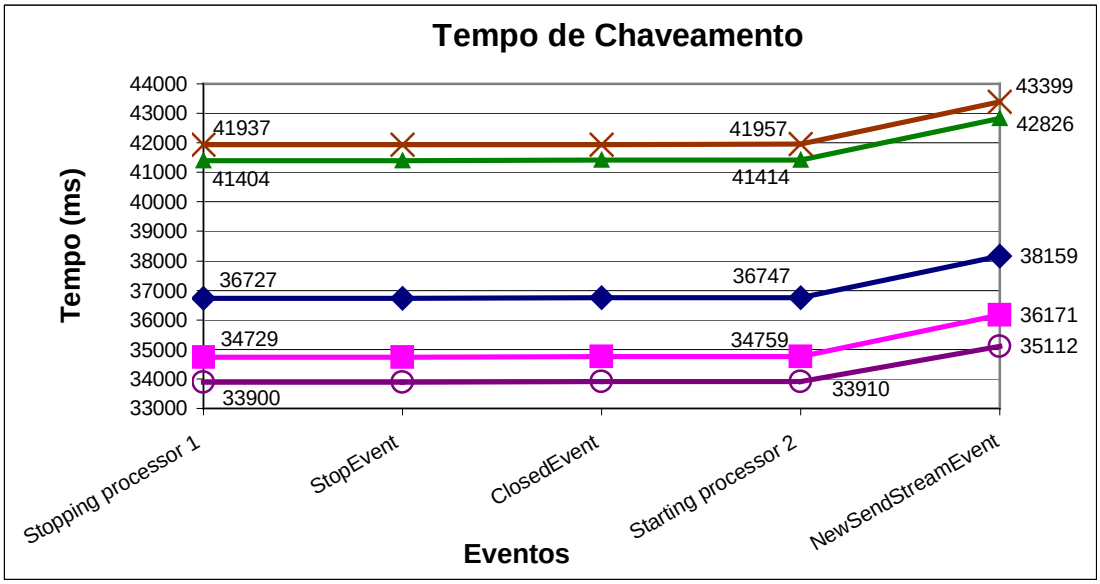


Figura 5.10 - Tempo de chaveamento (nível 1) para 5 rodadas distintas

EVENTOS NO CLIENTE	TEMPO(ms)	EVENTOS NO SERVIDOR
>>> CREATING Session Manager...	0	
SessionManager created	4877	
---START---	4887	
	5009	----START----
	5009	>>> CREATING SessionManager 1 ...
	9886	SessionManager created
	9886	>>> CREATING VideoTransmit 1 ...
	9896	VideoTransmit created
	9896	>>> Starting processor 1 ... using default start time
	12670	NewSendStreamEvent Received
NewReceiveStreamEvent Received	13600	
>>> Creating a new player....	13600	
>>> Realizing player....	14932	
RealizeCompleteEvent received...prefetching	15412	
PrefetchCompleteEvent received...starting	15452	
StartEvent received	15482	
---SWITCH---	33779	
	33900	----SWITCH----
	33900	>>> CREATING VideoTransmit 2 ...
	33900	VideoTransmit created
	33900	>>> Stopping processor 1 at... 4.7047047s
	33900	StopEvent Received
	33910	ClosedEvent Received
	33910	>>> Starting processor 2... setting start time:4.7047047s
	35112	NewSendStreamEvent Received
---STOP---	51835	
	51956	---STOP---
	51956	Stopping processor 2 ...
	51956	StopEvent Received
	52016	ClosedEvent Received
TimeOutEvent Received FROM transmitter	53417	
KillThePlayer called...stopping	53417	
StopEvent received...closing	53427	
Leaving KillThePlayer...	53437	
	71184	TimeOutEvent Received FROM receiver

Tabela 5.3 - Log de eventos para uma simulação da adaptação de nível 1

Caso 2: adaptação de nível 2 (sessões diferentes)

No segundo caso, ilustrado na figura 5.11, o servidor cria um segundo objeto *SessionManager* com o endereço da nova sessão RTP fornecido pelo cliente (mensagem *START*). Assim, o servidor começa a participar da nova sessão RTP com um identificador da fonte (SSRC) diferente do primeiro. Em seguida, o servidor cria um objeto *VideoTransmit* para processar e transmitir o novo vídeo na nova sessão RTP utilizando o segundo objeto *SessionManager*. Quando os dados enviados chegam ao cliente, este detecta o novo fluxo identificado pelo SSRC do segundo objeto *SessionManager* e sinaliza

o evento *NewReceiveStream*. Com isso, um novo objeto *Player* é criado para apresentar os dados do segundo vídeo.

Este objeto *Player* tem que passar por todos os estados preparatórios de um objeto *Controller* antes de iniciar a apresentação do vídeo, o que produz um certo atraso. Entretanto, uma forma de minimizar o efeito deste atraso é interromper o primeiro fluxo apenas após o objeto *Player* do segundo fluxo atingir o estado *prefetched*. Assim, apesar do tempo gasto na preparação do segundo objeto *Player* ser maior, a percepção deste atraso por parte do usuário pode ser praticamente cancelada ao se manter a apresentação do primeiro fluxo até o instante em que o segundo objeto *Player* estiver pronto para apresentar o segundo fluxo.

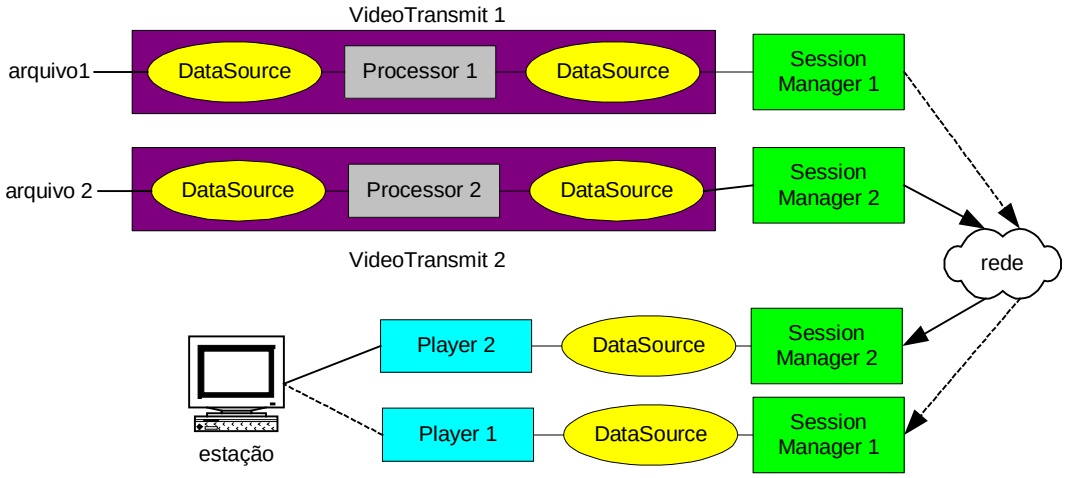


Figura 5.11 - Chaveamento usando dois objetos *SessionManager*

Porém, esta abordagem possui alguns problemas. Uma vez que o chaveamento do primeiro fluxo foi gerado por uma violação dos limites de qualidade especificados e, consequentemente, por uma falta de recursos na rede, iniciar o segundo fluxo mantendo o primeiro ativo consumirá ainda mais recursos da rede e prejudicará ainda mais a qualidade de serviço fornecida. Como o primeiro fluxo só é interrompido depois que o novo objeto *Player* atinge o estado *prefetched* (i.e. depois que os pacotes do segundo fluxo começam a chegar no cliente), o segundo fluxo tem que, por alguns instantes, competir pelos recursos da rede com o primeiro fluxo. Alguma informação importante, que seja transmitida durante o intervalo em que os dois fluxos trafegam em paralelo, pode não ser apreendida

pelo usuário devido à qualidade ser menor do que os limites especificados como aceitáveis para aquele fluxo.

Outro problema que surge é o da sincronização entre os dois fluxos. Uma solução seria o servidor descontar o tempo de preparação do objeto *Player* e iniciar a transmissão do segundo fluxo com um avanço no relógio em relação ao relógio do primeiro fluxo. Este avanço deve corresponder ao tempo de preparação do objeto *Player* para que, quando o segundo fluxo começar a ser apresentado, os relógios dos dois fluxos estejam sincronizados. Por exemplo, considerando o tempo de preparação igual a TP segundos, se o servidor recebe uma solicitação de chaveamento e o relógio do primeiro fluxo marca T segundos, o relógio do segundo fluxo deve ser ajustado para (T+TP) segundos antes de iniciar a transmissão. Da mesma forma, o primeiro fluxo deve ser interrompido quando o seu relógio atingir (T+TP) segundos. Além desta solução poder sofrer com errors na sincronização (variações no tempo de preparação), as informações do primeiro fluxo continuam sendo apresentadas com qualidade inferior à aceitável enquanto o novo objeto *Player* não atinge o estado *prefetched*.

Outra solução seria utilizar técnicas de predição para avaliar, segundo uma linha de tendência, o instante em que o limite de qualidade especificado seria violado. Assim, antes que ocorra uma violação do limite de qualidade de serviço, uma solicitação de chaveamento é enviada. O tempo de preparação, como no caso anterior, também deve ser descontado pelo servidor para que os fluxos sejam sincronizados. Esta solução só é melhor que a anterior pois evita que informações continuem sendo transmitidas após a violação da qualidade de serviço, porém, continua vulnerável às mesmas variações no tempo de preparação. Além disso, a previsão do instante de violação da qualidade pode ser alterada pela própria transmissão do segundo fluxo em paralelo com o primeiro.

Assim, com o objetivo de preservar a qualidade de percepção e simplificar o mecanismo, é melhor iniciar o chaveamento pela interrupção do primeiro fluxo e imediatamente começar a transmitir o segundo fluxo na nova sessão. Embora o tempo de chaveamento percebido pelo usuário seja maior, ocorre uma liberação imediata dos recursos da rede, evita-se que os dois fluxos disputem os mesmos recursos da rede e que haja transmissão de informações com nível de qualidade de serviço inaceitável pela

aplicação. Também, não ocorrem problemas de sincronização pois o relógio do segundo fluxo é ajustado, antes de ser transmitido, para o instante exato em que o primeiro fluxo é interrompido. Os resultados da simulação são apresentados para este tipo de abordagem.

EVENTOS NO CLIENTE	TEMPO(ms)	EVENTOS NO SERVIDOR
>>> CREATING Session Manager...	0	
SessionManager created	4937	
---START---	4957	
	5103	----START----
	5113	>>> CREATING SessionManager 1 ...
	9990	SessionManager created
	9990	>>> CREATING VideoTransmit 1 ...
	10000	VideoTransmit created
	10000	>>> Starting processor 1 ... using default start time
	12774	NewSendStreamEvent Received
NewReceiveStreamEvent Received	13620	
>>> Creating a new player....	13620	
>>> Realizing player....	14982	
RealizeCompleteEvent received...prefetching	15412	
PrefetchCompleteEvent received...starting	15482	
StartEvent received	15512	
---SWITCH---	39657	
	39823	----SWITCH----
	39823	>>> CREATING SessionManager 2 ...
	44379	SessionManager created
	44379	>>> CREATING VideoTransmit 2 ...
	44379	VideoTransmit created
	44379	>>> Stopping processor 1... at 7.607607600
	44379	StopEvent Received
	44449	ClosedEvent Received
	44460	>>> Starting processor 2... at 7.607607600
	45852	NewSendStreamEvent Received
NewReceiveStreamEvent Received	46117	
>>> Creating a new player....	46117	
KillOldPlayer called...stopping	46267	
StopEvent received...closing	46267	
SetNewPlayer called...	46287	
>>> Realizing player....	46330	
RealizeCompleteEvent received...prefetching	46440	
PrefetchCompleteEvent received...starting	46450	
StartEvent received	46461	
ClosedEvent received...	49852	
TimeOutEvent Received FROM:653950482	49882	
---STOP---	64222	
	64398	---STOP---
	64398	Stopping processor 2 ...
	64408	StopEvent Received
	64418	ClosedEvent Received
TimeOutEvent Received FROM:653985022	66746	
KillOldPlayer called... stopping	66746	
StopEvent received... closing	66746	

Tabela 5.4 - Log de eventos para uma simulação da adaptação de nível 2

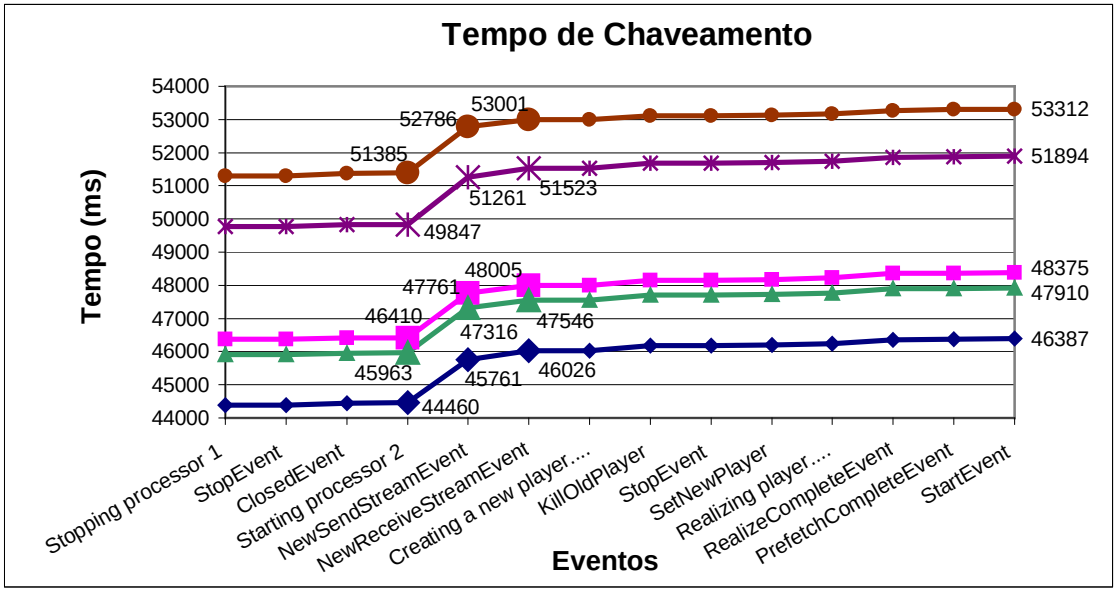


Figura 5.12 - Tempo de chaveamento (nível 2) para 5 rodadas distintas

Os resultados mostram que todas as rodadas apresentaram um comportamento bastante semelhante, tanto no primeiro caso (mesma sessão – figura 5.11) quanto no segundo (sessões diferentes – figura 5.12).

No primeiro caso, o tempo de chaveamento está bem marcado entre os eventos *"Starting Processor 2"* e *"NewSendStreamEvent"*, e corresponde ao tempo gasto pelo servidor para dar a partida na transmissão do fluxo. O tempo médio de chaveamento para o primeiro caso foi de 1394ms.

No segundo caso, nota-se que, até o quinto evento (*"NewSendStreamEvent"*), as curvas apresentam as mesmas características das curvas do primeiro caso. O tempo gasto até o quinto evento responde por 2/3 do tempo total de chaveamento. A outra parcela (1/3) do tempo total fica distribuída ao longo dos eventos registrados pelo cliente durante a criação e preparação do novo objeto *Player*. Pode-se observar que, nesta parcela, os principais incrementos no tempo de chaveamento ocorrem em dois momento: entre os eventos *"Creating a new player..."* e *"KillOldPlayer"* (tempo gasto para criar o objeto *Player*); e entre os eventos *"Realizing player..."* e *"PrefetchCompleteEvent"* (tempo gasto para o objeto *Player* atingir o estado *prefetched*). O tempo médio de chaveamento para o segundo caso foi de 2044ms.

Conclusões

A estratégia de autoria proposta neste trabalho promove uma forte integração entre as diversas mídias que compõem o documento. O estabelecimento de uma malha de relacionamentos condicionais em um documento fornece subsídios para que os sistemas de comunicação e apresentação decidam quando há a necessidade de adaptar alguma informação (conteúdo) e como esta adpatação deve ser realizada com o objetivo de preservar a semântica do documento. Esta característica se mostra de grande valor em um ambiente integrado de ensino à distância.

Neste trabalho foram apresentados dois mecanismos combinados de adaptação para a apresentação de documentos multimídia em sistemas distribuídos e a respectiva arquitetura de implementação: (i) um mecanismo "suave" (nível 1), que efetua uma adaptação a nível de codificação (fidelidades) de mídias independentes de forma transparente ao usuário; e (ii) um mecanismo "forte" (nível 2), que leva o conceito de adaptabilidade ao nível do documento, gerando uma nova estrutura lógica e temporal em tempo de apresentação com base na avaliação dos relacionamentos condicionais entre as mídias.

Para garantir a interoperabilidade com sistemas multimídia já existentes, as informações de controle de adaptabilidade (linguagem SMAL), compostas dos descritores de QoS e dos *links* condicionais, foram mantidas externas ao documento multimídia original (linguagem SMIL 1.0), em um arquivo de anotação. Este tipo de abordagem simplifica o desenvolvimento de uma aplicação (*player)* para apresentar os documentos adaptativos e implica em três grandes vantagens: (i) a apresentação do documento multimídia pode ser efetuada, na sua forma básica, por um aplicação puramente SMIL que não seja capaz de considerar as informações de controle (SMAL); (ii) não existe a necessidade do desenvolvimento integral de uma nova ferramenta de apresentação, uma vez que diversos módulos já desenvolvidos para processar XML (*parsers, geradores, etc*)

podem ser aproveitados no desenvolvimento da ferramenta de apresentação; e (iii) torna o sistema independente de novas versões do SMIL.

O emprego dos protocolos RTSP e RTP/RTCP foi um grande facilitador no desenvolvimento desta arquitetura adaptável pois permite que se realize o monitoramento em tempo-real da comunicação através dos parâmetros de qualidade de entrega das mídias existentes nas unidades de controle destes protocolos. Por ser o padrão adotado pelo IETF, tem sido utilizado pela grande maioria das aplicações multimídias desenvolvidas nos últimos anos. Um mecanismo de monitoramento em tempo-real (QDM + SRM) foi definido neste trabalho para permitir às aplicações clientes gerenciar os requisitos de QoS, bem como preservar os relacionamentos condicionais especificados pelo autor.

Implementações foram realizadas com o objetivo de simular o comportamento do sistema com relação ao tempo de resposta a solicitações de adaptação dos fluxos multimídia para os dois níveis de adaptação propostos. A plataforma *Java Media Framework 2.0* desenvolvida pela *SUN* foi utilizada nas implementações e se mostrou de grande utilidade no desenvolvimento das aplicações cliente e servidora.

Como mostrado pelos resultados das simulações, o primeiro nível de adaptação possui uma latência menor que o segundo nível, pois o chaveamento dos fluxos pode ser realizado na mesma sessão RTP. Neste caso, o tempo de chaveamento é todo gerado pelo servidor ao processar a troca dos fluxos na sessão RTP. No segundo nível de adaptação, quando o chaveamento é feito entre sessões RTP diferentes, o cliente também contribui para o tempo de chaveamento, pois, além do tempo gasto pelo servidor, o cliente também gasta tempo criando e preparando um novo objeto *Player* para apresentar o novo fluxo.

Assim, a realização de uma adaptação em dois níveis permite que a maioria das aplicações não apresentem tempo de chaveamento significativo ao realizar, preferencialmente, as adaptações de primeiro nível. A adaptação de segundo nível deve ser ativada nas situações em que o primeiro mecanismo não conseguir manter os requisitos de QoS especificados. Apesar do tempo de resposta da adaptação “forte” ser implicitamente superior ao tempo de adaptação do mecanismo “suave”, em um ambiente de ensino à distância este atraso é bastante aceitável pois as dimensões temporais das mídias acessadas

neste tipo de aplicação são grandes (muitas vezes superior ao atraso). Além disso, quando lidamos com aplicações voltadas ao ensino é melhor a convivência com um pequeno atraso de chaveamento do que a possibilidade de uma interpretação equivocada das informações.

Como trabalho futuro, pretende-se estudar o uso de mecanismos para o monitoramento dos recursos locais, como taxa de utilização de processador e memória. Desta forma, novas opções para a determinação dos parâmetros de QoS podem ser definidas. No que diz respeito aos mecanismos de adaptação, uma proposta de refinamento na sincronização do reinício do documento adaptado também está sendo estudada. Uma consequência direta deste trabalho será a posterior incorporação na ferramenta de autoria desenvolvida no projeto Servimídia [Cunh99a, Cunh99b,Cunh99c] de toda a metodologia de criação de documentos adaptativos que está sendo gradualmente amadurecida. O objetivo é tornar esta difícil tarefa do ponto de vista do autor o mais transparente possível.

Apêndice A

Formatos dos pacotes dos protocolos RTP/RTCP

A.1 Campos do cabeçalho fixo do RTP

O cabeçalho de um pacote RTP possui o seguinte formato:

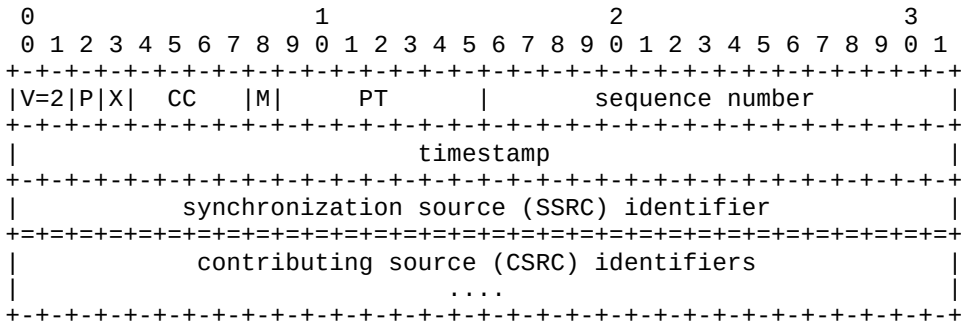


Figura A.1 - Cabeçalho do pacote RTP

Os oito primeiros octetos estão presentes em todos os pacotes RTP, enquanto a lista de identificadores CSRC (*contributing source*) está presente apenas quando inserida por um *mixer*. Os campos do cabeçalho possuem os seguintes significados:

***version* (V): 2 bits**. Versão do RTP. A versão mais recente é a 2.

***padding* (P): 1 bit**. Se ativado, o pacote contém um ou mais octetos de enchimento no fim do pacote que não fazem parte do conteúdo da mídia. O último octeto de enchimento contém a quantidade de octetos de enchimento que devem ser ignorados.

***extension* (X): 1 bit**. Se ativado, o cabeçalho fixo é seguido por exatamente um cabeçalho de extensão.

CSRC *count* (CC): 4 bits. O número de identificadores CSRC que seguem o cabeçalho fixo. Este número é maior do que 1 se o conteúdo do pacote RTP possui dados de várias fontes.

***marker* (M): 1 bit**. Definido por um *profile*, este campo busca permitir que eventos significantes como o limite de um quadro possam ser marcados em um fluxo de pacotes.

***payload type* (PT): 7 bits**. Identifica o formato do conteúdo de um pacote RTP e determina sua interpretação pela aplicação.

***sequence number*: 16 bits**. Incrementado para cada pacote RTP enviado, pode ser usado pelo receptor para detectar perda de pacotes e para recuperar a sequencia dos mesmos. O valor inicial é definido randomicamente.

***timestamp*: 32 bits**. O instante de amostragem do primeiro octeto presente em um pacote RTP. Pode ser usado para sincronização e cálculos de *jitter*. O valor inicial é definido aleatoriamente.

SSRC: 32 bits. Um número escolhido aleatoriamente para distinguir fontes de sincronização dentro de uma mesma sessão RTP. Ele indica onde os dados foram combinados, ou a fonte dos dados se existir apenas uma fonte.

CSRC list: 0 a 15 itens, 32 bits cada. Fontes que contribuem para o conteúdo deste pacote. O número de identificadores é dado pelo campo CC.

A.2 Formato dos Pacotes RTCP *Sender Report* e *Receiver Reports*

Um pacote RTCP deve começar por um cabeçalho fixo que contém:

***version* (V): 2 bits**. Versão do RTP. A versão mais recente é a 2.

***padding* (P): 1 bit**. Se ativado, o pacote contém um ou mais octetos de enchimento no fim do pacote que não fazem parte do conteúdo da mídia. O último octeto de enchimento contém a quantidade de octetos de enchimento que devem ser ignorados.

***reception report count* (RC): 5 bits**. O número de *report blocks* incluidos no pacote.

***payload type* (PT): 7 bits**. Identifica o formato do conteúdo de um pacote RTCP e determina sua interpretação pela aplicação. Igual a 200 no caso de um *Sender Report* ou 201 no caso de um *Receiver Report*.

***length*: 16 bits.** O comprimento do pacote RTCP em palavras de 32-bits após o campo length.

SSRC: 32 bits. O identificador da fonte de sincronização para o originador do pacote RTCP.

Os pacotes RTCP *Receiver Reports* possuem um certo número de *report blocks* seguindo o cabeçalho fixo, como na figura A.2. O número de *report blocks* é definido pelo campo RC do cabeçalho fixo, e pode ser igual a zero.

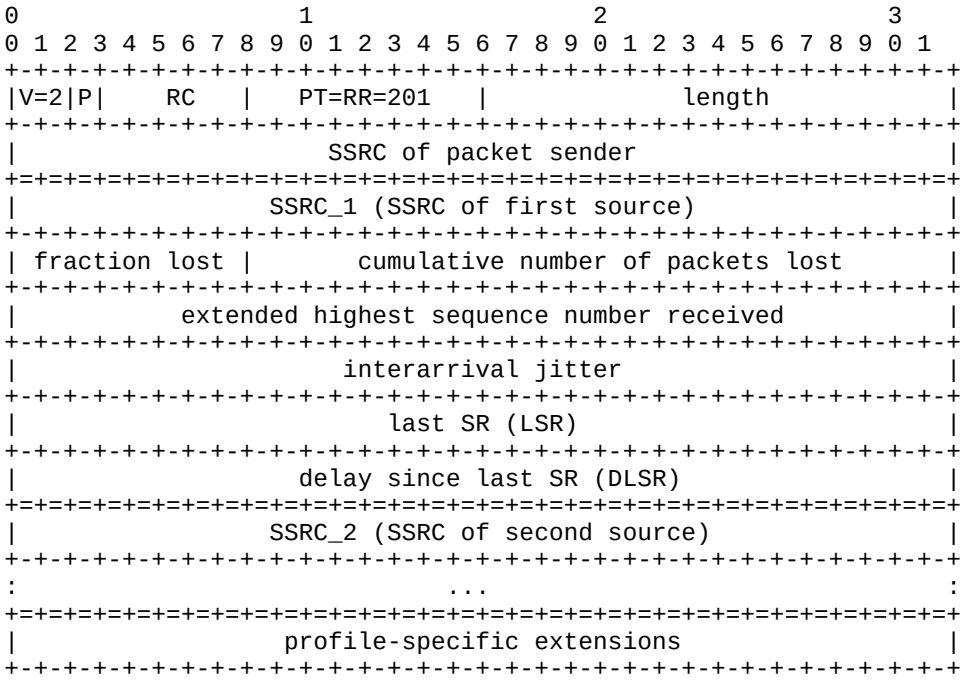


Figura A.2 - Formato do pacote RTCP *Receiver Report*

Após o cabeçalho, os pacotes RTCP *Sender Report* possuem informações fornecidas pelo emissor dos dados RTP (*sender info*). Em seguida, os *Sender Reports* incluem um *report block* para cada fluxo que o emissor estiver recebendo de outros emissores. Ou seja, se o emissor também for um receptor de fluxos RTP, os *Sender Reports* que ele envia possuem todas as informações de um *Receiver Report*, além das informações do emissor. A figura A.3 apresenta o formato do pacote RTCP *Sender Report*.

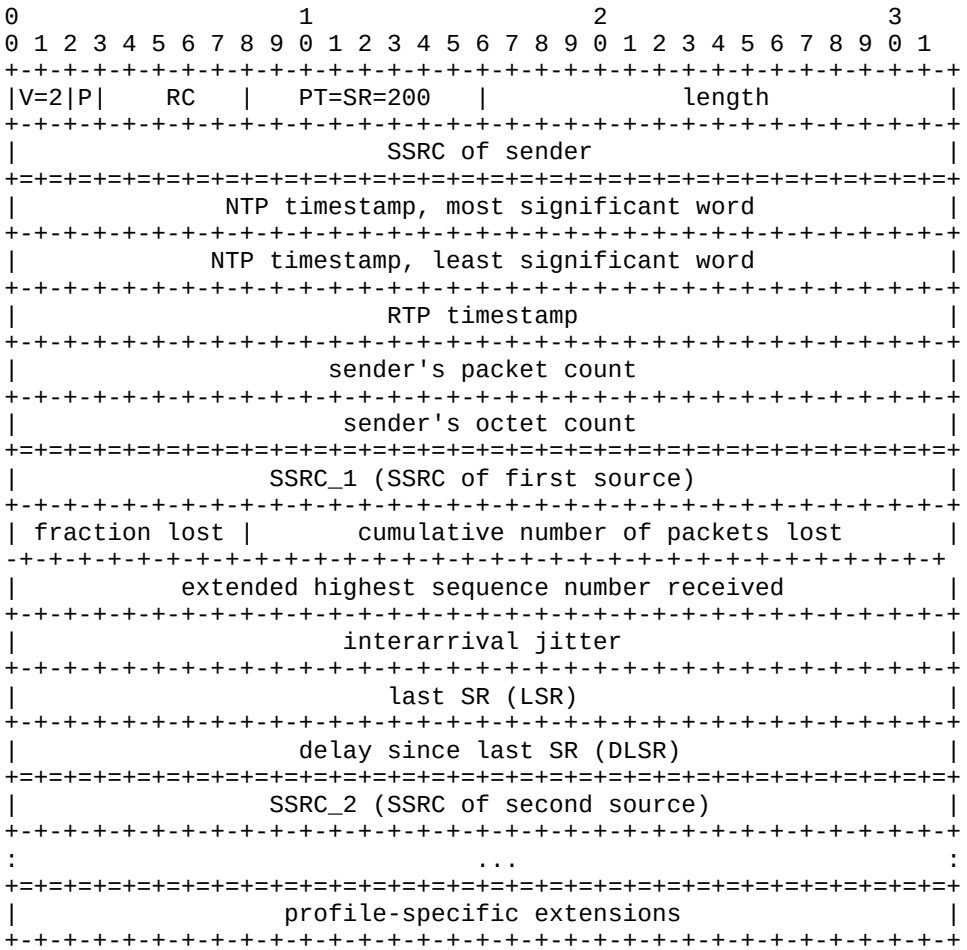


Figura A.3 - Formato do pacote RTCP *Sender Report*

Apêndice B

Formatos das mensagens do protocolo RTSP

Request	=	Request-Line *(general-hearder | request-header | entity-header) CRLF [message-body]	
Request-Line	=	Method (sp) Request-URI (sp) RTSP-Version CRLF	
Method	=	"DESCRIBE"	get low-level desciption of media object
	|	"ANNOUCE"	change description of media object
	|	"GET_PARAMETER"	get presentation/stream parameter
	|	"OPTIONS"	get available methods
	|	"PAUSE"	halt delivery, but keep state
	|	"PLAY"	start *playback*, reposition
	|	"RECORD"	start recording
	|	"REDIRECT"	redirect client to new server
	|	"SETUP"	establish transport
	|	"SET_PARAMETER"	device or encondig control
	|	"TEARDOWN"	remove state
	|	extension-method	

Tabela B.1 - Formato das mensagens de requisição ("request messages")

Response	=	Status-Line *(general-hearder | response-header | entity-header) CRLF [message-body]
Status-Line	=	RTSP-Version (sp) Status-Code (sp) Reason-Phrase CRLF
Status-Code sp **Reason-Phrase**	=	"100" Continue
	|	"200" OK
	|	"201" Created
	|	"250" Low on Storage Space
	|	"300" Multiple Choices
	|	... ver listagem completa em [22]

Tabela B.2 - Formato das mensagens de resposta ("response messages")

Referências

1. CARMO, L.F.R.C., PIRMEZ, L. “ServiMídia: An Integrated System for Multimedia Document Creation and Retrieval with Adaptive QoS Control”. **In: 2nd France-Brazil Symposium on Distributed Computer Systems**, 1997, Recife, Brazil.

2. VOGEL, L., KERHERVÉ, B., BOCHMANN, G., GECSE, J. "Distributed Multimedia and QoS: a Survey". **IEEE Multimedia**, p. 10-19, summer, 1995.

3. WOLF, L. C., GRIODZ, C., STEINMETZ, R. "Multimedia Communications”. **Proceedings of IEEE**, vol.85, , n.12, p. 1915-1933, dec 1997.

4. V. O. K. LI, W. LIAO. “Distributed Multimedia Systems”. **Proceedings of IEEE**, p. 1061-1108, july 1997.

5. FERGUSON, P., HUSTON, G. “Quality of Service on the Internet: Fact, Fiction or Compromise ?”. **In: Internet Conference**, 1998, Genebra, Suiça.

6. AURRECOECHEA, C., CAMPBELL, A.T., HAUW, L. “A survey of QoS architectures”. **ACM Multimedia Systems**, n.6, p.138-151, 1998.

7. STEINMETZ, R. “Human Perception of Jitter and Media Synchronisation”. **IEEE Journal on Selected Areas in Communications**, vol.14, n.1, p.61-72, 1996.

8. GHINEA, G., THOMAS, J.P. “QoS Impact on User Perception and Understanding of Multimedia Video Clips”. **In: ACM Multimedia Conference**, 1998, Bristol, United Kingdom.

9. GHINEA, G., THOMAS, J.P., FISH, R.S. Quality of Perception to Quality of Service Mapping Using a Dynamically Reconfigurable Communication System. **In: IEEE Global Telecommunications Conference**, 1999, Rio de Janeiro, Brazil, p. 2061-2065.

10. CAMPBELL, A., COULSON, G., HUTCHISON, D. "A Quality of Service Architecture". **ACM Computer Communications Review**, 1994.

11. ISO. **Quality of Service Framework**. ISO/IEC JTC1/SC21/WG1 N9680, 1995.

12. HONG, J., DIM, J., PARK, J. "A CORBA-Based Quality of Service Management Framework for Distributed Multimedia Services and Applicatons". **IEEE Network**, p. 70-79, march/april 1999.

13. BERSON, S., LINDELL, R., BRADEN, R. "An Architecture for Advance Reservations in the Internet". **UCS Informations Science Institute**, 1998. Disponível na Internet: http://www.

14. BUSSE, I., DEFFNER, B., SCHULZRINNE, H. “Dynamic QoS Control of Multimedia Applications based on RTP”. **Computer Communications**, vol.19, p. 49-58, jan. 1996.

15. RENINGER, D., RAYCHAUDHURI, D., OTT, M. "A Dynamic Quality of Service Framework for Video in Broadband Networks". **IEEE Network**, p. 22-34, november 1998.

16. HERMANN, A. **Especificação de Qualidade de Serviço e Adaptação de Aplicações Multimídia Distribuídas**. Rio Grande do Sul: UFRGS, 1999. Dissertação. (Mestrado em Informática)

17. GOMES, A., COLCHER, S., SOARES, L. "Um Framework para Provisão de QoS em Ambientes Genéricos de Processamento e Comunicação". **In: 17º Simpósio Brasileiro de Redes de Computadores**, 1999, Salvador, Brasil, p. 307-322.

18. BRADEN, R., CLARK, D., SHENKER, S. “Integrated Services in the Internet Architecture: an Overview**”. IETF RFC 1633**, june 1994.

19. BLAKE, S. et all. “An Architecture for Differentiated Services”. **IETF RFC 2475**, dec. 1998.

20. BRADEN, R. et all. “Resource ReSerVation Protocol (RSVP)”. **IETF RFC 2205**, oct. 1997.

21. SCHULZRINNE, H. et all. “RTP: A Transport Protocol for Realtime Applications”. **IETF RFC 1889**, jan. 1996.

22. SCHULZRINNE, H., RAO, A., LANPHIER, R. “Real Time Streaming Protocol (RTSP)”. **IETF RFC 2326**, april 1998.

23. SCHULZRINNE, H. “RTP Profile for Audio and Video Conferences with Minimal Control”. **IETF RFC 1890**, jan. 1996.

24. CLARK, D., TENNENHOUSE, D. “Architecture considerations for a new generation of protocols”. **In: ACM Computer Communications Conference**, 1990, p. 200-208.

25. CHRISMENT, I., KAPLAN, D., DIOT, C. “An ALF Communication Architecture: Desing and Automated Implementation”. **IEEE Journal on Selected Areas in Communications**, vol.16, n.3, p.332-344, april 1998.

26. ISO. **Information Technology Coding of Multimedia and Hypermedia Information, Part 5: Support for Base-Level Interactive Applications, MHEG-5 IS Document Pre-release 5**. ISO/IEC DIS 13522-5, 1996.

27. CANDAN, K.S., PRABHAKARAN, B., SUBRAHMANIAN, V.S. “Retrieval schedules based on resource availability and flexible presentation specifications”, **ACM Multimedia Systems**, n.6, p. 232-250, 1998.

28. W3C. **Synchronized Multimedia Integration Language (SMIL) 1.0 Specification**. W3C Recommendation, june 1998. Disponível na Internet: http://www.w3.org/TR/REC-smil.

29. W3C. **Extensible Markup Language (XML) 1.0 Specification.** W3C Recommendation, february 1998. Disponível na Internet: http://www.w3.org/TR/REC-xml.

30. ISO. **MPEG 7 Requirements document V.7**. ISO/IEC JTC1/SC29/WG11/N2461. October 98, Atlantic City, USA.

31. ISO. **MPEG 7 Applications document V.7**. ISO/IEC JTC1/SC29/WG11/N2462. October 98, Atlantic City, USA.

32. BULTERMAN, D.C.A. “User-Centered Abstractions for Adaptive Hypermedia Presentations.” In: **ACM Multimedia Conference**, 1998, Bristol, UK.

33. LI, X., PAUL, S., AND AMMAR, M.H. “Layered Video Multicast with Retransmission (LVMR): Evaluation of Hierarchical Rate Control. In: **IEEE INFOCOMM**, 1998, San Francisco, USA, p.1062-1072.

34. MCCANNE, S., JACOBSON, V., AND VETTERLI, M. “Receiver-driven Layered Multicast.” In: **ACM Computer Communications Conference**, 1996, Stanford, USA, p.117-130.

35. SHACHAM, N. “Multipoint Communication by Hierarchically Encoded Data.” In: **IEEE INFOCOMM**, 1992, Florence, Italy, vol.3, p.2107-2114

36. SANTOS, M.T.P., VIEIRA, M.T.P. “Sistema de Recuperação de Informações em um Servidor de Objetos Multimídia”. In: **IV Simpósio Brasileiro de Sistemas Multimídia e Hipermídia**, 1998, Rio de Janeiro, Brazil.

37. MOHAN, R., SMITH, J.R., LI, C.-S. “Content Adaptation Framework: Bringing the Internet to Information Appliances.” In: **IEEE Global Telecommunications Conference,** December 1999, Rio de Janeiro, Brazil, p. 2015-2021.

38. W3C. **Composite Capability/Preference Profile (CC/PP): Requirements and Architecture**. W3C Working Draft, February 2000. Disponível na Internet: http://www.w3.org/TR/CCPP-ra/

39. W3C. **Resource Description Framework (RDF) Model and Syntax Specification**. W3C Recommendation, February 1999. Disponível na Internet: http://www.w3.org/TR/REC-rdf-syntax/

40. RUTLEDGE, L., OSSENBRUGGEN, J.V., HARDMAN, L., BULTERMAN, D.C.A. “A Framework for Generating Adaptable Hypermedia Documents”. In: **ACM Multimedia Conference**, 1997, Seatle, USA.

41. COURTIAT, J.P., CARMO, L.F.R.C., OLIVEIRA, R.C. de. "A General-purpose Multimedia Synchronization Mechanism Based on Causal Relations". **IEEE Journal on Selected Areas in Communications**, vol.14, n.1, p. 185-195, January 1996.

42. CUNHA, E.C. da, CARMO, L.F.R.C., PIRMEZ, L. A Multimedia Document Authoring Strategy for Adaptive Retrieval in a Distributed System. In: **Distributed Multimedia Systems Conference**, July 1999, Aizu, Japan, p. 81-88.

43. CUNHA, E.C. da, CARMO, L.F.R.C., PIRMEZ, L. Design of an Integrated Environment for Adaptive Multimedia Document Presentation Through Real Time Monitoring. In: **Interactive Distributed Multimedia Systems and Telecommunications Services,** October 1999, Toulouse, France, p. 191-201.

44. CUNHA, E.C. da, CARMO, L.F.R.C., PIRMEZ, L. A Stream Relationship Monitor for Adaptive Multimedia Document Retrieval, In**: IEEE Global Telecommunications Conference**, December 1999, Rio de Janeiro, Brazil, p. 2071-2075.