Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica
Departamento de Telemática

Um Sistema para Processamento Distribuído Multimídia

Autor: Marcelo Perazolo
Orientador: Prof.Dr. Edson Moschim

Tese apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas – FEE–UNICAMP, como parte dos requisitos exigidos para a obtenção do título de MESTRE EM ENGENHARIA ELÉTRICA.

ABRIL – 1994

A meus pais.

Agradecimentos

Ao Prof. Edson Moschim pela sua orientação e amizade.

A todo o corpo docente da Faculdade de Engenharia Elétrica da UNICAMP por sua alta capacitação tecnológica e seu ideal de ensino.

Finalmente, a todos que de uma forma ou outra estiveram envolvidos em discussões e suporte a este trabalho.

Sumário

Atualmente os sistemas computacionais são cada vez mais personalizados e integrados na vida das pessoas. Com isso surge a necessidade de diversificar as formas de processamento de informações existentes. As necessidades correntes da sociedade implicam em um grande e diversificado volume de informações de Mídias diferentes, tais como áudio, vídeo, gráficos, textos, animação, etc. Por outro lado, computadores são cada vez mais interligados, visando facilitar a comunicação. Redes de computadores surgem como uma solução para a distribuição de recursos, mas com isso surgem dificuldades decorrentes da grande heterogeneidade de máquinas e programas. Existem diversas soluções envolvendo resursos tecnológicos de grande complexidade. Sistemas verdadeiramente distribuídos são aplicados atualmente em diversos setores da informática. Este trabalho propõe a aplicação de tecnologias de sistemas distribuídos na solução de problemas relacionados ao processamento de diversos tipos de informação, ou seja, processamento Multimídia. Para tanto serão aplicados conceitos já existentes de tecnologia de redes e desenvolvidos conceitos que proporcionem isso, tais como Protocolos de comunicação, Servidor e Cliente Multimídia, registro de Serviços Multimídia, entre outros. Finalmente serão apresentadas conclusões sobre a utilidade do sistema baseados em resultados práticos.

Conteúdo

Lista de Figuras

Considerações Gerais

Este documento apresenta a descrição de uma arquitetura distribuída para processamento Multimídia e, para tanto, realiza-se numerosas definições de conceitos relacionados ao sistema. Algumas convenções devem ser observadas para o entendimento destes conceitos e são colocadas a seguir:

- A implementação de todos os conceitos descritos foi realizada em um ambiente específico e uma linguagem de programação bastante difundida, que é o C++. Procurando manter a coerência com publicações da àrea os elementos de programação e implementação citados no decorrer do texto utilizam uma nomenclatura no idioma Inglês nos seguintes casos: objetos da implementação do sistema, primitivas de comunicação, algumas figuras representativas destes elementos e algumas primitivas internas também representativas de funções da implementação do ambiente. Exemplos comuns no decorrer do texto seriam *MediaManager*, *mmConnect*, *openConnection*, *connectionDescriptor*, etc.

- Existe uma relação entre os nomes conceituais do sistema e seu correspondente na implementação utilizando o paradigma de orientação a objetos. Por exemplo: Gerente de Mídia implementa-se no objeto *MediaManager*.

- As primitivas de comunicação do Protocolo desenvolvido e tipos de dados internos associados seguem uma nomenclatura especial utilizando o sufixo *mm*. Por exemplo: *mmConnect*, *mmObject*, etc.

- Alguns identificadores de eventos e erros, além de constantes globais do ambiente seguem uma nomenclatura especial utilizando letras maiúsculas. Por exemplo: `mmMEDIA_INVALID`, `mmSERVER_ERROR`, etc.

- Os métodos das interfaces de Mídia também devem seguir um padrão recomendado, que é a nomenclatura relacionada à função dos serviços e a utilização da primeira letra de cada palavra em letras maiúsculas. Por exemplo: `Text::ReadDocument`, `Video::SendFrame`, etc.

- Os exemplos contendo trechos de código fonte no decorrer do documento sao todos escritos baseados na sintaxe da linguagem escolhida na implementação e não devem ser interpretados como um código inteiramente funcional, sendo apenas considerados a título explanativo.

- A palavra processo (*process*) tem dois sentidos distintos, sendo primeiramente utilizada para designar um elemento realiza uma transformação, mas tambem pode designar um programa executável do ambiente operacional do sistema. Este documento utiliza indistintamente a palavra, sendo que em casos de sentido dúbio utiliza-se a palavra tarefa (*task*) para designar o elemento executável.

- A palavra cliente (*client*) refere-se a um processo que se utiliza das chamadas do protocolo Multimídia para se comunicar com o Servidor. Designa, então um programa executável nos moldes do ambiente. Mais detalhes sobre a arquitetura são fornecidos no primeiro capítulo.

Introdução Geral

Este trabalho relata uma nova arquitetura distribuída aplicada a processos Multimídia. Os sistemas de computação da atualidade estão cada vez mais dependentes da tecnologia de redes e uma consequência disso é o surgimento de sistemas distribuídos, onde vários processos cooperam entre si para atingir determinado propósito. Desta tecnologia resultaram várias outras visando prover ferramentas para se atingir esta cooperação. Entre elas, Protocolos de comunicação entre processos, conceitos de cliente-servidor, multitarefa, processamento escalar, etc. Ambientes Multimídia existentes não se preocupam com aspectos relacionados com a distribuição, entre eles heterogeneidade de equipamentos, portabilidade, flexibilidade de execução e cooperação, entre outros. O sistema proposto por este trabalho visa a criação de uma nova arquitetura de distribuição aplicada exclusivamente a processos Multimídia e procura atacar os pontos descritos acima [22] de modo a apresentar um direcionamento na busca de sua solução. Os objetivos principais são apresentar um modelo completo que permita a distribuição e também a fácil adaptação e expansibilidade em redes de computadores. No primeiro capítulo o modelo cliente-servidor é descrito visando uma adequação ao ambiente Multimídia proposto. Um protocolo de comunicação entre cliente, servidor e processos Multimídia também

é desenvolvido neste capítulo. No capítulo seguinte o modelamento do servidor Multimídia, a peça-chave do ambiente proposto, é definido em camadas operacionais, cada uma com funções bem determinadas. A seguir, no proxímo capítulo, desenvolvem-se aspectos relacionados com a flexibilidade e extensibilidade do sistema. Define-se o conceito de Projeto de Mídia, ou seja, como o sistema pode ser composto de Tarefas e serviços Multimídia, através de registro e definição de processos. Finalmente, o capítulo seguinte mostra detalhes da implementação e alguns exemplos de registro de serviços e construção de clientes, seguido de alguns resultados práticos. As conclusões gerais a respeito das características do sistema, baseadas em resultados experimentais, são mostradas a seguir, junto com um breve sumário das direções futuras que devem ser tomadas nessa área e planejamentos de trabalhos futuros envolvendo o ambiente desenvolvido.

Capítulo I

Arquitetura Cliente-Servidor Multimídia

Resumo – A arquitetura Cliente-Servidor é a escolha perfeita para se implementar Serviços Distribuídos. Ela e baseada na definição de um Cliente Multimídia comunicando-se com um Servidor Multimídia através de um protocolo especial, chamado Protocolo Multimídia. Esta arquitetura permite que processos e recursos Multimídia sejam controlados através de uma rede e compartilhados entre vários Clientes. Outras vantagens incluem fácil portabilidade, expansibilidade e eficiência.

I.1 Introdução

Este capítulo tem por objetivo apresentar a arquitetura Cliente-Servidor e direcionar o seu modelo e até extende-lo de modo que os fundamentos sejam adequados para as novas definições apresentadas nos capítulos seguintes. Conceitos relativos aos componentes do ambiente são definidos, tais como o Cliente, o Servidor, as Tarefas e o Protocolo, todos aplicados a uma terminologia dita Multimídia que significa antes enquadrá-los como componentes de um sistema direcionado ao processamento Multimídia do que realizarem tarefas específicas de controle da interação entre diversas formas de informação, tal como tarefas de sincronização, por exemplo. Neste contexto definem-se a seguir os componentes da arquitetura.

A arquitetura Cliente-Servidor caracteriza-se pela existência de um componente, chamado Cliente, cuja função é requisitar serviços para outro componente, chamado Servidor, cuja função é prover serviços. Num ambiente distribuído em rede, onde recursos podem ser compartilhados através de protocolos de comunicação, torna-se importante o conceito da arquitetura Cliente-Servidor e de serviços que possam prover flexibilidade, cooperação e portabilidade [21].

No caso de recursos Multimídia, ou seja, recursos que envolvam processamento, comunicação ou tratamento de entrada e saída de dados relacionados a um tipo específico de informação, sendo que este tipo de informação define-se como Mídia, podemos ter como um exemplo dispositivos de video, audio, texto, animação e voz, entre outros. O processamento requerido para alguns destes recursos, como por exemplo, certas transformações usuais em informação de video, podem exigir grande esforço computacional. Esta é uma das motivações principais para o compartilhamento de recursos através da distribuição numa rede de computadores [4]. Além disso, estes recursos na maioria das vezes envolvem conceitos de execução em Tempo-Real [14] [6] e por isso, uma distribuição em rede pode colaborar para a distribuição no carregamento em tarefas de processamento Multimídia elevado. Recursos que sejam caros e exijam características de hardware específicas podem ser situados em pontos estratégicos da rede e mesmo assim são acessíveis a muitos usuários. Outra questão importante refere-se à diversidade de recursos que manipulam informações Multimídia. Cada um deles exige uma forma própria de programação, controle e até de apresentação, muitas vezes envolvendo técnicas complexas de acionamento [1]. Com isso tanto programador como usuário ficam desnecessariamente comprometidos e necessitam conhecer muito bem o recurso a ser utilizado. Este problema pode ser amenizado se pensarmos que o acesso a cada um destes recursos é feito através de um Servidor moldado de forma a fornecer para Clientes da rede uma gama

de serviços padronizados e que permitam controlar o dispositivo de forma transparente. Com a distribuição e a padronização dos serviços podemos atingir a flexibilidade necessária para um ambiente altamente diversificado e compartilhado. Outras requisitos básicos de Sistemas Distribuídos [8], tais como portabilidade e expansibilidade podem ser facilmente suportados por serem inerentes à arquitetura de redes de computadores.

Para entendermos melhor a diferença entre Cliente e Servidor imaginemos que um dado recurso computacional encontra-se disponível em rede e num outro ponto deseja-se utilizar este recurso. Em uma arquitetura convencional isso seria possível se o usuário local utilizasse serviços de conexão remota com a máquina portadora do recurso, operasse diretamente sobre o recurso, perfazendo todo o processamento necessário, sendo que, para tanto, precisa-se conhecer intimamente como controlá-lo e programá-lo. Com a distribuição do recurso somente a máquina portadora consegue controlar o mesmo. Adaptando esse problema para uma arquitetura mais flexível torna-se necessário distribuir também o processo controlador que deve ficar localizado na máquina portadora do recurso. Este processo distribuído encarregado de gerenciar e controlar recursos é chamado comumente de Servidor. Para tornar estes recursos disponíveis em rede definem-se serviços que devem ser honrados pelos Servidores, relacionados com a possibilidade de controle remoto dos recursos. Outros processos remotos que desejam acessar um dado recurso comunicam-se com o Servidor específico através de primitivas de um protocolo bem definido de modo a acionar o serviço desejado. Estes processos remotos são chamados de Clientes. O relacionamento entre estes componentes é mostrado em detalhe na figura I.1.

Nas seções seguintes veremos mais detalhes sobre aspectos internos relacionados a comunicação e aos serviços de controle envolvendo Cliente e Servidor.

I.2 Descrição da Arquitetura

Como visto anteriormente, os componentes principais da arquitetura são o processo Cliente, o processo Servidor e o Protocolo de comunição utilizado nas interações entre estes componentes. Além destes, o modelo aqui descrito propoe um outro tipo de componente, controlado exclusivamente pelo Servidor e necessario na realizacao de servicos requeridos pelo Cliente. Estes componentes são as Tarefas, que são inicializadas para suprir a demanda de processamento específico. O processamento lógico e de controle fica então exclusivo do Servidor. Procurando aplicar este modelo ao ambiente introduzem-se os seguintes termos:

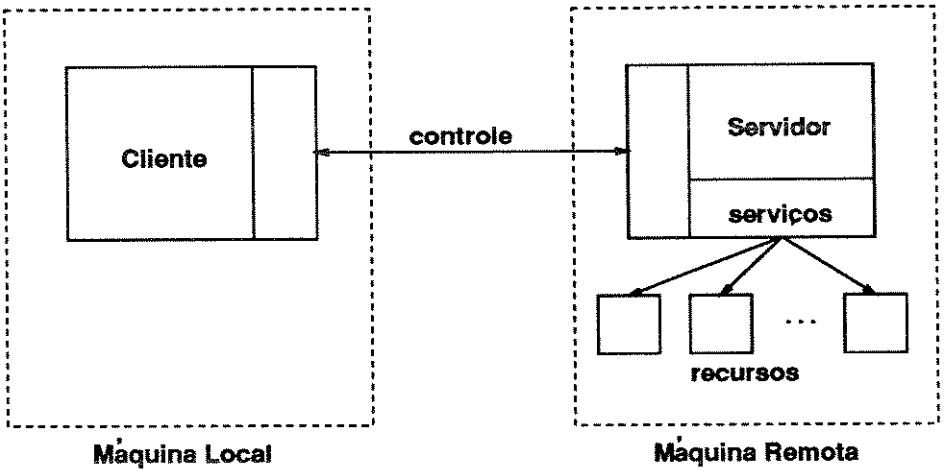

Figura I.1: Relação Cliente-Servidor

- Protocolo Multimídia (*MProtocol*)
- Cliente Multimídia (*MClient*)
- Servidor Multimídia (*MServer*)
- Tarefa Multimídia (*MTask*)

Estes termos referem-se apenas ao fato de serem processos integrados num ambiente Multimídia, e não necessariamente ao fato de manipularem diversos tipos de Mídia concorrentemente. A manipulação concorrente de informações de Mídias diversas é possivel através da implementação dos Serviços Multimídia, gerenciados pelo Servidor. A flexibilidade provida pelo ambiente como um todo é que permite contruç oes verdadeiramente Multimídia. A descrição funcional de cada um destes componentes é mostrada a seguir.

I.2.1 Protocolo Multimídia

Para que um processo possa se encaixar na definição de Cliente Multimídia no sentido de participar de um ambiente Multimídia, o mesmo deve ser capaz de acessar concorrentemente diversos Servidores e, consequentemente, seus serviços [23]. Para ser capaz de acessar esses serviços o Cliente deve se comunicar com o Servidor através de um Protocolo bem definido. Tarefas também se utilizam deste protocolo para servicos básicos de controle e transferência de dados.

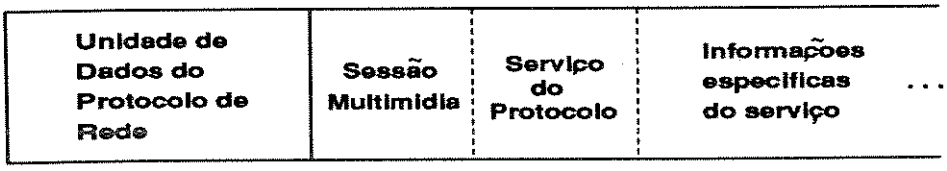

Figura I.2: Unidade de Dados do Protocolo Multimídia

O Protocolo Multimídia situa-se a nível de arquitetura de redes de informação a nível de aplicação [18], ou seja, é construído sobre outros protocolos inerentes à rede utilizada. Para estabelecer uma conexão Multimídia o usuário do Protocolo precisa apenas conhecer o nome da máquina com a qual deseja estabelecer comunicação, deixando as tarefas de roteamento e controle de pacotes para as camadas inferiores da estrutura de comunicação. Uma chamada do Protocolo Multimídia é composta basicamente de um identificador da sessão, um identificador da chamada e informações específicas de cada chamada. Na figura I.2 podemos observar a representação de um pacote do protocolo.

O Protocolo Multimídia utiliza-se de abstrações chamadas de Objetos Multimídia, que são representações de recursos Multimídia. Um Objeto Multimídia é composto de informações específicas e de serviços associados. Então a tarefa do Cliente resume-se em controlar e gerenciar Objetos Multimídia. Cada Cliente pode controlar concorrentemente vários Objetos e sincronizá-los. Essa iteração entre os Objetos define-se como uma Sessão Multimídia. Em uma Sessão Multimídia as ações definidas em objetos fazem parte de uma máquina de estados que pode ter componentes síncronos ou assíncronos, que são estados associados a estes objetos. Estes estados são os elementos finais na execução do que se define como Serviços Multimídia, que por sua vez nada mais são do que processos relacionados com Tarefas de processamento, comunicação e de entrada e saída de dados associados a uma Mídia. A sinalização de controle entre Cliente, Servidor e Tarefas é realizada através de Eventos Multimídia. Os Clientes e Tarefas podem registrar ações a serem disparadas na ocasião da sinalização de eventos selecionados.

As chamadas do Protocolo Multimídia são citadas e classificadas como síncrona, assíncrona e interna. Chamadas síncronas são aquelas que suspendem o processo até a chegada de confirmaçao ou de informações remotas. Chamadas assíncronas não esperam confirmação remota. Finalmente, existem chamadas internas que não fazem parte realmente do protocolo, mas tão somente auxiliam no controle do fluxo de chamadas de comunicação entre as partes envolvidas.

A seguir as chamadas do Protocolo Multimídia são descritas brevemente e apresentadas.

mmConnect Realiza a conexão entre um Cliente e um Servidor Multimídia. É responsável pela criação de Sessões Multimídia. Chamada síncrona.

```
mmSession mmConnect
          (
            mmServerID serverID
          );
```

serverID: Identificador do Servidor a ser conectado

retorno: Descritor da sessão conectada

mmDisconnect Desconecta um Cliente de um dado Servidor Multimídia. Realiza a destruição da Sessão Multimídia anteriormente aberta. Chamada assíncrona.

```
void mmDisconnect
     (
       mmSession session
     );
```

session: Descritor da sessão a ser desconectada

mmAttach Realiza a conexão entre uma Tarefa e um Servidor Multimídia. É responsável pela criação de Sessões especiais, associadas à Tarefa. Chamada síncrona.

```
mmEvent mmAttach
        (
        );
```

retorno: Evento sinalizando sucesso ou falha na operação

mmDetach Desconecta um Cliente de um dado Servidor Multimídia. Realiza a destruição da Sessão Multimídia anteriormente aberta. Chamada assíncrona.

```
mmEvent mmDetach
        (
        );
```

retorno: Evento sinalizando sucesso ou falha na operação

mmQuery Realiza a criação de Objetos Mídia associados a uma dada Sessão Multimídia. Chamada síncrona.

```
mmObject mmQuery
         (
           mmSession session,
           mmObjectID objectID
         );
```

session: Descritor da sessão associada
objectID: Identificador do Objeto Mídia a ser criado
retorno: Descritor do Objeto Mídia criado

mmCreate Realiza a criação de instâncias de Serviços Multimídia associados a uma dada Sessão Multimídia. Chamada síncrona.

```
mmState mmCreate
        (
          mmSession session,
          mmObject media,
          mmStateID stateID
        );
```

session: Descritor da sessão associada
media: Descritor do Objeto Mídia associado
stateID: Identificador do estado Multimídia correspondente
retorno: Descritor do estado Multimídia correspondente

mmDestroy Realiza a destruição de instâncias de Serviços Multimídia associados a dada Sessão Multimídia. Chamada assíncrona.

```
void mmDestroy
     (
       mmSession session,
       mmState state
     );
```

session: Descritor da sessão associada
state: Descritor do estado a ser destruido

mmLink Conecta estados relacionados a serviços e associa transições de estados relacionados a uma máquina de estados através da especificação de uma máscara de Eventos Multimídia. Chamada assíncrona.

```
void mmLink
     (
       mmSession session,
       mmState state1,
       mmState state2,
       mmEvent event
     );
```

session: Descritor da sessão associada
state1: Descritor do estado inicial a ser ligado
state2: Descritor do estado final a ser ligado
event: Evento que direciona a ligação

mmUnlink Desconecta estados relacionados a serviços anteriormente conectados e associados a uma máquina de estados. Chamada assíncrona.

```
void mmUnlink
     (
       mmSession session,
       mmState state1,
       mmState state2,
       mmEvent event
     );
```

session: Descritor da sessão associada
state1: Descritor do estado inicial a ser desligado
state2: Descritor do estado final a ser desligado
event: Evento que direciona a ligação

mmStart Aciona uma máquina de estados relacionada a uma dada Sessão Multimídia. Chamada assíncrona.

```
void mmStart
     (
       mmSession session
     );
```

session: Descritor da sessão associada

mmStop Interrompe uma máquina de estados relacionada a uma dada Sessão Multimídia. Chamada assíncrona.

```
void mmStop
     (
       mmSession session
     );
```

session: Descritor da sessão associada

mmResume Reaciona uma máquina de estados relacionada a uma dada Sessão Multimídia. Chamada assíncrona.

```
void mmResume
     (
       mmSession session
     );
```

session: Descritor da sessão associada

mmRegister Associa ações a serem tomadas na ocasião da chegada de algum Evento Multimídia desejado. Chamada assíncrona.

```
void mmRegister
     (
       mmSession session,
       mmEvent event,
       mmHandler handler
     );

void mmRegister
     (
       mmSession session,
       mmEvent event,
       mmTransitionHandler handler
     );
```

session: Descritor da sessão associada
event: Evento que direciona a acionamento da função
handler: Função a ser registrada

mmUnregister Desassocia ações de acontecimentos relacionados com a chegada de Eventos Multimídia. Chamada assíncrona.

```
void mmUnregister
     (
       mmSession session,
       mmEvent event,
       mmHandler handler
     );

void mmUnregister
     (
       mmSession session,
       mmEvent event,
       mmTransitionHandler handler
     );
```

session: Descritor da sessão associada
event: Evento que direciona a acionamento da função
handler: Função a ser desregistrada

mmLoop Suspende o Processo Multimídia e aguarda o acontecimento de Eventos e Exceções previamente registradas. Chamada interna.

```
void mmLoop
     (
     );
```

Os identificadores utilizados na descrição das chamadas são associados aos tipos de dados manipulados pela implementação do protocolo. Suas características são citadas a seguir.

mmServerID Identifica univocamente um Servidor Multimídia. Em um ambiente distribuído pode-se associar com o endereço da máquina onde o Servidor se encontra mais um número de ordem do mesmo.

mmSession Identifica univocamente uma Sessão Multimídia. Com a sessão associa-se também uma máquina de estados a ser construída posteriormente com instâncias de Objetos e Serviços Multimídia conectados entre si.

mmObjectID Identifica univocamente um Objeto Multimídia. Estes objetos são anteriormente registrados no Servidor Multimídia e associados a processos e informações que servem para formar instâncias de objetos passíveis de serem controladas pelos Clientes.

mmObject Identifica instâncias de Objetos Multimídia utilizados em uma máquina de estados e associados a uma dada Sessão Multimídia.

mmStateID Identifica univocamente um Serviço Multimídia. Estes serviços são anteriormente registrados no Objeto Multimídia e associados a processos que servem processar informações associadas aos objetos.

mmState Identifica instâncias de Serviços Multimídia presentes em uma máquina de estados e associados a uma dada Sessão Multimídia.

mmEvent Identifica uma máscara de Eventos Multimídia específica que pode ser composta de eventos de transição de estados e de eventos configuráveis, pertinentes a cada tipo de Objeto Multimídia.

mmHandler Identifica uma ação a ser tomada na ocasião do acontecimento de certos Eventos Multimídia selecionados.

mmTransitionHandler Identifica uma ação a ser tomada na ocasião do acontecimento de uma transição de estados Multimídia. Esta transição é sinalizada pelo Servidor baseando-se na interação entre Tarefas e repassada aos Clientes registrados.

I.2.2 Cliente Multimídia

Um Cliente Multimídia deve ter acesso a Objetos Multimídia através de comunicação com Servidores Multimídia. Essa comunicação e realizada através do Protocolo Multimídia, descrito anteriormente.

Tipicamente, um Cliente Multimídia deve realizar os seguintes passos e na ordem estabelecida:

- Realizar a conexão com um Servidor e abertura de uma Sessão Multimídia.
  (*mmConnect*)
- Criar as instâncias de Objetos Multimídia necessárias, de acordo com o registro no Servidor conectado.
  (*mmQuery*)
- Criar as instâncias de Serviços Multimídia necessárias, de acordo com o registro nos Objetos Multimídia criado. Cada instância constitui-se então do que é chamado de um estado da sessão.
  (*mmCreate*)
- Conectar as instâncias de Objetos anteriormente criadas, formando assim máquinas de estados que sejam responsáveis pelo controle e sincronização das ações desejadas.
  (*mmLink*)
- Registrar ações a serem tomadas na ocasião do acontecimento de Eventos Multimídia.
  (*mmRegister*)
- Acionar as máquinas de estados associadas com as Sessões Multimídia anteriormente criadas.
  (*mmStart*)
- Processar eventos Multimídia e tomar ações associadas. Esse processamento é implementado através de um Loop infinito que perfaz verificações nas filas de entrada de eventos.
  (*mmLoop*)
- Destruir os recursos anteriormente alocados, na ocasião da chegada de eventos de finalização ou erros irrecuperáveis.
  (*mmStop, mmUnregister, mmUnlink, mmDestroy, mmDisconnect*)

Um exemplo da utilização das chamadas de protocolo seria a criação de um Objeto Multimídia remoto e sua monitoração através do registro de manipuladores de eventos.

Pensemos numa situação onde uma máquina remota tem capacidades de processamento da Mídia Vídeo com facilidades tais como processadores específicos para tal fim. O Servidor Multimídia desta máquina exporta Objetos Multimídia relacionados com sua capacidade especial e os coloca à disposição dos Clientes na Rede. Qualquer cliente que estabelecer uma Sessão Multimídia e se conectar com este servidor em questão esta apto a acessar estes objetos e disparar métodos de processamento de Vídeo. Seguindo os passos anteriormente descritos um Cliente Multimídia tem a estrutura de chamadas como no exemplo mostrado na figura I.3.

As Tarefas correspondentes também se utilizam das chamadas do Protocolo Multimídia para se comunicar com o Servidor. Os exemplos de Tarefas Multimídia utilizados no Cliente anterior poderiam ser codificados como nas figuras I.4 e I.5 mostradas a seguir.

Neste exemplo, primeiramente identifica-se o servidor, através de seu endereçamento na rede, depois pede-se uma conexão que deve retornar um identificador da sessão aberta. Este identificador de sessão passa então a ser utilizado em todas as chamadas de protocolo subsequentes para aquele dado servidor conectado. Entre os objetos oferecidos identificamos um objeto chamado VIDEO_OBJECT que interessa ao Cliente. Este, por sua vez cria uma instância do objeto Video através do serviço mmQuery. Entre os serviços oferecidos por este objeto são selecionados VIDEO_DECODE e VIDEO_DITHER, que são respectivamente associados com tarefas de decodificação de sinais de Video e de transformação dos sinais decodificados. Cada instância dos serviços de Video compõe o que é chamado de um estado. O conjunto de todos os estados de uma dada sessão irá compor uma máquina de estados. Para a criação dos estados utilizou-se a chamada mmCreate. Estes estados são a seguir conectados sequencialmente através da chamada mmLink. Eventuais eventos tais como final de decodificação, final de transformação e eventos de erros, previamente definidos pelo objeto Video, são registrados a acionarem manipuladores de eventos no processo do Cliente através das chamadas mmRegister. O Cliente agora se torna apto a disparar a execução da sessão previamente definida e esperar pela ocorrência de eventos através das chamadas mmStart e mmLoop. Na ocasião da ocorrencia de um evento especial, chamado de evento de finalização ou de um evento de erro irrecuperável, os quais provocam finalização do processo Cliente, os recursos anteriormente alocados no Servidor devem ser liberados. As chamadas mmUnregister cancelam o processamento dos demais eventos, mmUnlink desconecta a máquina de estados, chamadas mmDestroy liberam os estados associados aos Serviços Multimídia e mmDisconnect destroy a sessão e a comunicação com o Servidor.

```
main()
{
  /* Conecta o servidor */
  mmServerID serverID = "videoServer";
  mmSession session = mmConnect(serverID);
  /* Cria um objeto Video */
  mmObjectID videoID = "VIDEO_OBJECT";
  mmObject video = mmQuery(session, videoID);
  /* Cria os estados referentes ao objeto Video */
  mmStateID decodeID = "VIDEO_DECODE";
  mmState decode = mmCreate(session, video, decodeID);
  mmStateID ditherID = "VIDEO_DITHER";
  mmState dither = mmCreate(session, video, ditherID);
  /* Conecta os estados criados */
  mmLink(session, decode, dither, VIDEO_DECODE_READY);
  /* Registra os manipuladores de eventos */
  mmRegister(session, VIDEO_DECODE_READY, decodeHandler);
  mmRegister(session, VIDEO_DITHER_READY, ditherHandler);
  mmRegister(session, VIDEO_ERROR, errorHandler);
  /* Inicia o cliente */
  mmStart(session);
  /* Processa eventos */
  mmLoop();
  /* Libera recursos anteriormente alocados */
  mmUnregister(session, VIDEO_DECODE_READY, decodeHandler);
  mmUnregister(session, VIDEO_DITHER_READY, ditherHandler);
  mmUnregister(session, VIDEO_ERROR, errorHandler);
  mmUnlink(session, decode, dither, VIDEO_DECODE_READY);
  mmDestroy(session, decode);
  mmDestroy(session, dither);
  mmDisconnect(session);
}
```

Figura I.3: Exemplo de Cliente Multimídia

```
mmEvent decodeVideo
        (
          mmEvent mask,
          void *videoEncoded,
          void *videoDecoded
        )
{
  /* Verifica identificador de evento */
  if (mask != VIDEO_DECODE_REQUEST) return;
  ...
  /* Decodifica entrada de video */
  videoDecoded = decode(videoEncoded);

  /* Sinaliza fim da tarefa */
  return (VIDEO_DECODE_READY);
}

main()
{
  /* Conecta-se no servidor */
  mmEvent status = mmAttach();
  if (status != NOERROR) exit(1);

  /* Registra manipulador de transicoes */
  mmRegister(VIDEO_DECODE_REQUEST, decodeVideo);

  /* Processa eventos */
  mmLoop();

  /* Libera registro do manipulador */
  mmUnregister(VIDEO_DECODE_REQUEST, decodeVideo);

  /* Desconecta-se do servidor */
  mmDetach();
}
```

Figura I.4: Tarefa videoDecode

```
mmEvent ditherVideo
        (
          mmEvent mask,
          void *videoNormal,
          void *videoDithered
        )
{
  /* Verifica identificador de evento */
  if (mask != VIDEO_DITHER_REQUEST) return;
  ...
  /* Processa entrada de video */
  videoDithered = dither(videoNormal);

  /* Sinaliza fim da tarefa */
  return (VIDEO_DITHER_READY);
}

main()
{
  /* Conecta-se no servidor */
  mmEvent status = mmAttach();
  if (status != NOERROR) exit(1);

  /* Registra manipulador de transicoes */
  mmRegister(VIDEO_DITHER_REQUEST, ditherVideo);

  /* Processa eventos */
  mmLoop();

  /* Libera registro do manipulador */
  mmUnregister(VIDEO_DITHER_REQUEST, ditherVideo);

  /* Desconecta-se do servidor */
  mmDetach();
}
```

Figura I.5: Tarefa videoDither

I.2.3 Servidor Multimídia

Um Servidor Multimídia deve primeiramente ser capaz de controlar os recursos oferecidos pela máquina em que se situa. Deve também ser capaz de honrar pedidos de serviços provenientes dos Clientes Multimídia, de acordo com o Protocolo Multimídia definido anteriormente. Além disso, deve também ser capaz de controlar dinamicamente o fluxo de informações e de eventos, sendo responsável pela comunicação entre os diversos processos envolvidos. Nesta seção será mostrada uma breve descrição do funcionamento de um Servidor Multimídia, levando em conta sua interação com os outros componentes da arquitetura, que são os Clientes e os processos associados a Serviços Multimídia (*MServices*), chamados de Tarefas Multimídia (*MTasks*). Em um outro nível de abstração podemos imaginar os recursos disponíveis ao Servidor como Objetos, sendo estes objetos passíveis de serem controlados por processos Clientes remotos. Então define-se um Objeto Multimídia (MObject) como sendo uma representação de um recurso Multimídia disponível em rede e passível de controle através de um Servidor Multimídia. Um Objeto Multimídia deve ser composto por informações relativas à Mídia ou ao recurso representado e por Serviços Multimídia associados. Uma arquitetura capaz de prover controle sobre Objetos Multimídia e sobre aspectos de fluxo de informações entre processos deve conter os seguintes componentes:

- Módulo de controle de Conexões, capaz de interpretar as chamadas de protocolo provenientes de Clientes e Tarefas Multimídia.
  Camada de Sessão (*Session Layer*)

- Módulo de controle de Mídia, responsável pelo controle e manipulação de dados pertinentes aos componentes de uma Mídia. Organiza logicamente as máquinas de estado constituídas e controla o fluxo de dados e transições entre os estados destas máquinas.
  Camada de Mídia (*Media Layer*)

- Módulo de controle de Tarefas Multimídia, relacionadas a Serviços e registrados em um Objeto Multimídia. Realiza a coordenação e a associação entre Objetos representando Tarefas e os processos correspondentes.
  Camada de Tarefas (*Task Layer*)

- Módulo de controle de Eventos Multimídia. É responsável pelo roteamento do fluxo de eventos entre o Servidor, os Clientes e as Tarefas. Possui informações sobre interesse ou não de componentes relativo a um determinado evento.
  Camada de Eventos (*Event Layer*)

- Módulo de registro de Objetos e Serviços Multimídia, responsável pelo controle de características de uma determinada Mídia, de estados e Tarefas associados a esta Mídia, de eventos e tipos de dados disponíveis e nomes, entre outras coisas.
  Camada de Registro (*Registration Layer*)

O relacionamento entre os diversos módulos é dinâmico e interativo, ou seja, a qualquer momento um dos módulos pode causar modificações na estrutura construída por outro módulo. Por exemplo, Clientes podem dinamicamente acionar chamadas de protocolo que são interpretadas pelo módulo de conexão e que podem influir diretamente em objetos ou eventos, manipulados respectivamente pelos módulos de controle de objetos e de eventos. Outro exemplo seria a ocorrencia de um erro em uma Tarefa Multimídia, diretamente controlada pelo módulo de controle de Tarefas, que pode gerar um evento registrado no módulo de controle de eventos e que pode ser repassada para os Clientes interessados, através do módulo de conexão. O módulo de registro de Objetos e Serviços influencia diretamente os módulos de controle de Objetos e de Tarefas, que necessitam da informação de disponibilidade ou não dos objetos ou serviços controlados. Como vemos, esta estrutura é dinâmica e flexível de modo a facilitar as interações entre seus diversos componentes.

Cada módulo descrito deve ter associado um componente na estrutura do Servidor para prover as funções correspondentes ao módulo. Estes componentes internos são agora definidos como Gerentes específicos, ou seja, exercem funções de gerenciamento para componentes do Servidor. Estes gerentes são listados a seguir, cada qual correspondendo ao módulo de mesmo nome. Mais detalhes sobre cada componente serão apresentados no capítulo subsequente.

- Gerente de Sessões (*Session Manager*)
- Gerente de Mídia (*Media Manager*)
- Gerente de Tarefas (*Task Manager*)
- Gerente de Eventos (*Event Manager*)
- Gerente de Registros (*Registration Manager*)

Na figura I.6 podemos ver a representação de um Servidor Multimídia e de seus diversos componentes.

As mensagens trocadas entre os processos são controladas pelo Servidor e podem ser classificadas em diversos tipos, dependendo dos elementos envolvidos na comunicação. A seguir veremos quais são os possiveis fluxos de mensagens definidos para esta arquitetura.

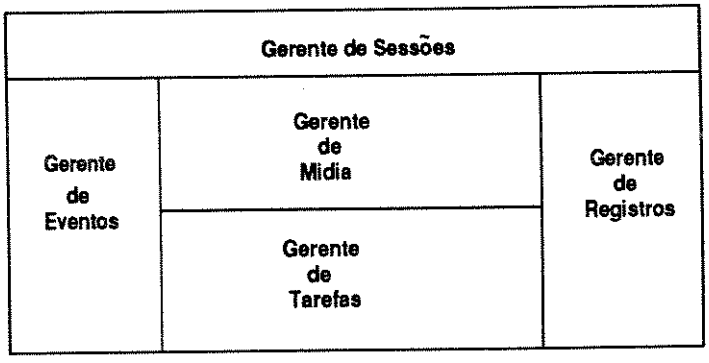

Figura I.6: Estrutura de um Servidor Multimídia

Cliente-Servidor São as mensagens do Protocolo Multimídia das quais o Cliente se utiliza para comandar os Recursos Multimídia controlados pelo Servidor.

Servidor-Cliente São diversos tipos de notificações, entre as quais encontram-se respostas a mensagens do Protocolo Multimídia, notificações do acontecimento de eventos ou exceções ou transferência de informações realizadas pela requisição de algum serviço.

Servidor-Tarefa São mensagens de controle e notificações provenientes de algum acontecimento ou do Cliente originador das tarefas.

Tarefa-Servidor São mensagens de notificação ou transferência de informações processadas pelos serviços associados.

Tarefa-Tarefa São mensagens de transferência de informações tão somente, deixando a cargo apenas do servidor o controle destas mesmas informações para que se ajustem nas Tarefas envolvidas.

Na figura I.7 observamos uma representação dos diversos tipos de mensagens trocadas entre os componentes. Observe-se que as mensagens são divididas entre fluxo de controle e fluxo de informação e que existe um caminho virtual da informação das tarefas diretamente ao Cliente, através do Servidor.

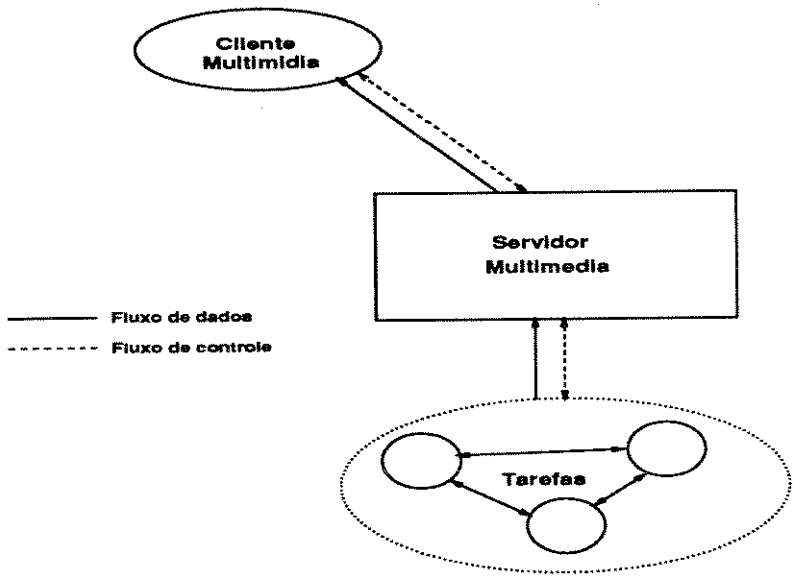

Figura I.7: Fluxo de mensagens

I.3 Comentários

A arquitetura apresentada possibilita a distribuição de Recursos Multimídia em rede e sua utilização através do modelo Cliente-Servidor. Este esquema fornece também certa segurança no que se refere a tolerância a falhas. Cada serviço a ser executado constitui um processo diferente, e estes serviços são disparados por um outro processo que representa o Servidor e são comandados por ainda outro que representa o Cliente. O fluxo de informações é grande e existe um gerenciamento de eventos no ambiente, controlado pelo Servidor, possibilitando assim uma fácil recuperação dos demais processos na ocasião de uma falha espersa em algum componente. O elemento crítico na arquitetura é o Servidor, que realiza grande parte do controle. Entretanto, um esquema de recuperação baseado nos dados de controle que devem ser salvos dinamicamente pode ser proposto e facilmente implementado.

Outro ponto forte é a flexibilidade e expansibilidade proporcionadas. Cada Objeto Multimídia e seus respectivos Serviços pode ser dinamicamente configurados, adicionados ou excluídos através do módulo de registro do Servidor. E, uma vez que os recursos estão devidamente registrados, os mesmos passam a estar disponíveis para

qualquer Cliente que consiga acessar o Servidor utilizando-se do Protocolo Multimídia.

A portabilidade proporcionada por uma arquitetura flexível e uma definição concisa das chamados do protocolo, aliada com a possibilidade de interoperabilidade em ambientes heterogêneos faz com que aplicações Multimídia possam ser construídas baseadas na utilização de diversos tipos de plataformas e otimizadas a ponto de aproveitar ao máximo os recursos distribuídos. Extensões do Protocolo Multimedia proposto podem também proporcionar um compromisso maior do sistema em atender detalhes de uma arquitetura distribuída e conseguir, com isso, maior eficiência e outras vantagens inerentes destes sistemas [24].

Um ambiente construído de modo a atender as especificações desta arquitetura e que forneça todas as vantagens já descritas pode ser classificado verdadeiramente como um Sistema Distribuído Multimídia.

Capítulo II

Modelo Distribuído Multimídia

Resumo – O modelo de um Servidor Multimídia que ofereça serviços distribuídos permite um ambiente flexível e dinâmico para o desenvolvimento de aplicações Multimídia. As opções de registro de serviços permitem que os processos cooperem entre si de modo a distribuir em rede o trabalho demandado por estes serviços e também que os recursos de processamento, comunicação e de entrada e saída de dados possam ser compartilhados por todas as máquinas do sistema.

II.1 Introdução

Este capítulo tem por objetivo apresentar o modelo do ambiente proposto e entrar em mais detalhes internos sobre o funcionamento dos diversos componentes, com ênfase no Servidor Multimídia e as diversas interações dos elementos internos para realizar serviços requisitados através do Protocolo Multimídia. Novos conceitos, tal como uma Máquina de Estados interna que regulamenta e controla o encadeamento dos diversas tarefas realizadas pelo servidor também serão apresentados. Para entende-los deve-se levar em conta a grande abstração do modelo que tenta relacionar-se com elementos globais do ambiente, como serviços, tarefas, eventos, transições, etc. A seguir estes conceitos são descritos e deixam em aberto um ambiente altamente flexível. Definições sobre como se relacionam os serviços externos e característicos de uma Mídia específica serão apresentados posteriormente.

O conceito de servidor Multimídia no ambiente proposto define um processo controlador de recursos e comunicação, que por meio de eventos definidos pode redirecionar o fluxo de informações entre Tarefas Multimídia e entre Clientes Multimídia. Um dado servidor é responsável pelo controle do que chamamos de Recursos Multimídia, através do fornecimento de uma série de chamadas definidas como Serviços Multimídia, que, em ultima instância vão se traduzir em processos chamados de Tarefas Multimídia. Arquiteturas deste tipo devem considerar aspectos de Tempo-Real, visto que processos Multimídia têm esta característica [12]. É por isso que esta arquitetura se preocupa com a divisão entre controle e da execução do processamento em Tarefas externas [25].

Baseando-se nas definições anteriores podemos construir um modelo que tenha as características desejadas de um servidor Multimídia. Podemos identificar as seguintes camadas na arquitetura:

- Camada de Aplicação Multimídia (*Application Layer*)
- Camada de Sessão Multimídia (*Session Layer*)
- Camada de Controle de Mídia (*Media Layer*)
- Camada de Controle de Tarefas Multimídia (*Task Layer*)

A camada de Aplicação é representada pelos clientes Multimídia. Cada cliente define seu próprio comportamento através de chamadas do Protocolo Multimídia que devem ser honradas pelos servidores. As demais camadas são de controle exclusivo do servidor.

A camada de Sessão representa a conexão entre os clientes e os serviços Multimídia providos pelo servidor. Nesta camada será definido

o comportamento de cada chamada do Protocolo Multimídia e a interação com os níveis inferiores. Para cada sessão aberta um objeto é criado. Com isso eliminam-se interações entre as sessões Multimídia. Cada cliente tem a transparência de se utilizar de um gerenciador de sessão único e exclusivo, constituído por este objeto.

O objetivo principal das aplicações Multimídia consiste na utilização de serviços específicos de uma determinada Mídia. Isso se dá no servidor através da criação de objetos Mídia e de máquinas de estados que controlam a execução dos serviços. Os objetos pertinentes são todos gerenciados a partir da camada de controle de Mídia. Por definição, entenda-se que os objetos controlados neste nível sejam tanto objetos referentes a uma Mídia, como tipos de dados manipulados, tarefas associadas e estados e ligações que representam uma máquina de estados.

As tarefas a serem executadas por uma máquina de estados traduzem-se em processos pertinentes ao sistema operacional. Com o objetivo de mapear tarefas em processos e efetuar seu gerenciamento necessita-se de uma camada que irá interagir com o Sistema Operacional. Esta é a camada de controle de Tarefas Multimídia.

Um esquema da interação entre as diversas camadas pode ser visto na figura II.1.

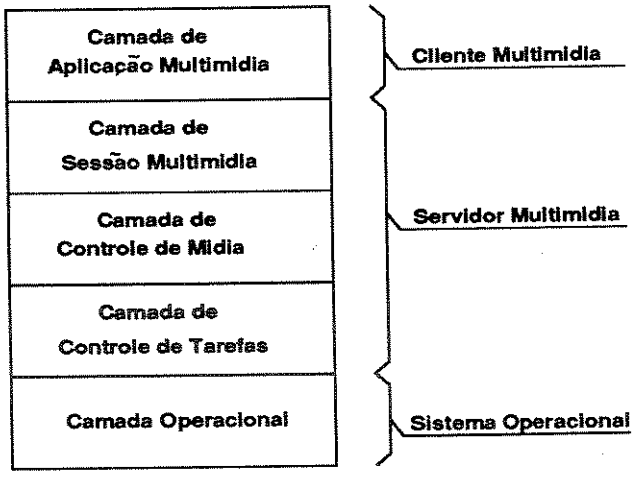

Figura II.1: Definição das Camadas do Ambiente

Com respeito ao Servidor Multimídia, sua funcionalidade começa a nível de sessão, ou seja, cada servidor é responsável pelo controle das comunicações com as aplicações e por prover os serviços necessários às mesmas. Podemos modularizar as camadas pertinentes ao servidor em

elementos de gerência de entidades. As entidades que necessitam de gerenciamento são Sessões, Mídias e Tarefas. Cada uma das camadas associadas a estas entidades serão módulos do servidor, que serão detalhados a seguir.

O primeiro módulo é responsável pela comunicação bidirecional entre cliente e servidor. A cada Sessão Multimídia requisitada por algum cliente, o Servidor abre uma conexão que resultará num novo objeto dedicado que passa a gerenciar o cliente requisitante. Todo o fluxo de controle envolvendo esta conexão é agora gerenciado por este objeto. Interações entre sessões são efetuadas através de eventos, que também são filtrados por este objeto.

A partir da existência de uma conexão, o papel do servidor passa a ser de controle de objetos e tarefas requisitadas pelo cliente. A disponibilidade dos recursos Multimídia relacionados verifica-se pelo Módulo de Controle de Registros ou Gerente de Registros. Sua tarefa é a de gerenciar dinamicamente um banco de registros Multimídia que deve conter entradas referentes aos objetos Mídia disponíveis, tarefas suportadas por estes objetos e respectivas definições de formatos de dados utilizados por estas tarefas. As entidades que correspondem a recursos são, portanto, objetos Mídia, tarefas e tipos de dados e de eventos.

Uma vez definidos os possíveis recursos a serem inicializados, o servidor deve criar uma representação na forma de objetos em sua estrutura interna. Os objetos Mídia são definidos como grupamentos de informação, ou tipos de dados e de processos, ou tarefas. O Módulo de Controle de Mídia ou Gerente de Mídia é o responsável pelo gerenciamento lógico destas entidades. A partir da criação das Mídias, deve-se definir máquinas de estados, síncronas ou assíncronas, para a realização de tarefas Multimídia. Cada estado tem associada uma tarefa e os tipos de dados requisitados na entrada e na saída da tarefa. Existem então objetos específicos para representação destas entidades, que também são controlados por este módulo.

Existe também a necessidade de um módulo específico para controle dos eventos trocados entre os diversos processos controlados por um servidor, entre os quais encontram-se tarefas e clientes. Cada objeto Mídia tem definidos os eventos associados a suas respectivas tarefas que servem para prover um fluxo de controle da máquina de estados criada pelas camadas anteriores. Cada evento pode ou não ser associado com ações ou exceções e o acionamento das mesmas se dá pela atuação do Módulo de Controle de eventos ou Gerente de Eventos.

Depois de iniciadas as operações de uma máquina de estados, existe a necessidade de se disparar processos associados aos serviços Multimídia. Estes processos são controlados e monitorados pela camada de controle de tarefas, através do Módulo de Controle de Tarefas ou Gerente de Tarefas.

Todo o controle relativo aos processos de Tarefas é realizado por este módulo.

Como foi visto anteriormente, a relação entre os diversos módulos é simples e aberta. Os tipos de dados controlados, bem como as tarefas, os eventos e a própria interação entre as entidades é perfeitamente configurável. Tudo baseia-se nas definições dos objetos Mídia e seus aspectos internos. Os elementos identificados na estrutura do Servidor Multimídia são listados a seguir:

- Gerente de Sessões (*SessionManager*)
- Gerente de Mídia (*MediaManager*)
- Gerente de Registros (*RegistrationManager*)
- Gerente de Eventos (*EventManager*)
- Gerente de Tarefas (*TaskManager*)

Uma visão geral dos mecanismos e a troca de informações internas a um servidor podem ser observadas na figura II.2.

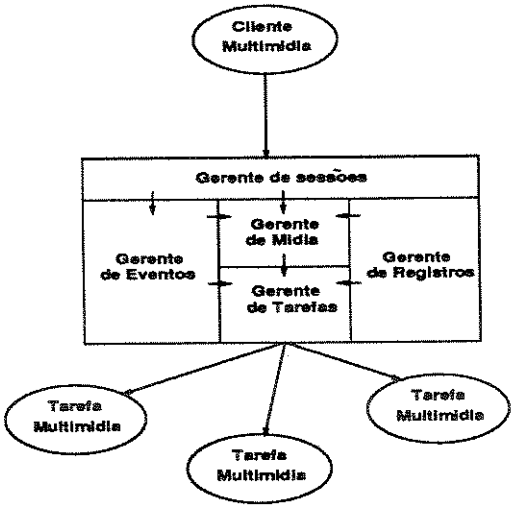

Figura II.2: Servidor Multimídia - Aspectos Internos

Estes elementos da estrutura de um Servidor Multimídia são detalhados na seção a seguir.

II.2 Descrição Funcional

A especificação dos componentes do Servidor Multimídia é baseada na definição de Objeto. Alguns termos necessitam de definição para que se possa entender melhor a descrição orientada a objetos. Define-se Classe de um objeto a sua descrição comportamental e informativa. Por descrição comportamental define-se um grupamento de procedimentos ou funções e por descrição informativa um grupamento de características, que juntos, moldam um objeto. Instancias diferentes de classes de objetos formam objetos com diferentes rótulos, mas com comportamento semelhante. Existe ainda a possibilidade de hierarquizar um objeto através de sua classe. Quando um objeto tem com outro uma relação de especialização diz-se que uma Classe é derivada da outra. Na Classe derivada especificam-se apenas as diferenças para a Classe ancestral. Tal técnica oferece suficiente clareza e flexibilidade na especificação dos componentes internos do Servidor Multimídia. Outro ponto importante, principalmente para efeito de implementação, é a possibilidade de se referenciar genericamente Classes de objetos especializados por meio de um ancestral em comum. Com isso generalizam-se tratamentos para um mesmo ramo de uma hierarquia de objetos. Essa técnica é comumente chamada de Polimorfismo.

A tecnologia de objetos foi escolhida na definição do ambiente por constituir uma ferramenta altamente difundida, por permitir flexibilidade e fácil entendimento e manutenção, características estas também desejáveis na arquitetura do Sistema em especificação. As definições a seguir empregarão a terminologia de objetos indistintamente e na especificação de módulos e componentes.

II.2.1 Camada de Sessão Multimídia

A primeira função encontrada no servidor Multimídia refere-se a um módulo responsável pelo gerenciamento das sessões e conexões. Quando um cliente deseja uma conexão, a chamada a ser efetuada é mmConnect. O servidor, ao receber tal chamada, cria um Manipulador de Sessão, também chamado de Gerente de Sessão, que fica responsável pelo tratamento das demais chamadas do protocolo para aquele cliente específico. Sua estrutura é semelhante a um disparador de serviços que fica infinitamente em uma rotina de verificação da existência de chamadas e do tratamento das mesmas. Este módulo tem a função de rotear as chamadas para os demais módulos do servidor e assegurar que a integridade dos serviços providos seja conservada. Os módulos a sofrerem uma interação direta são o Gerente de Mídia, que cuida das chamadas que afetam objetos Mídia ou estados e o Gerente de Eventos, que cuida de chamadas que afetam registros de eventos. Transições também são controladas a nível de sessão Multimídia. Sincronismo entre processos Multimídia é um tópico de relevante importância em sistemas distribuídos aplicados a Multimídia [13]. Por isso o controle destas transições deve ser realizado com extrema eficiência.

A seguir segue-se uma breve descrição destas ações tomadas por parte do disparador, incluindo esquemas que mostram as interações entre os diversos módulos do servidor para cada chamada.

mmConnect

Esta chamada do Protocolo Multimídia tem por função realizar a conexão entre um cliente e um servidor. Ao receber uma mensagem contendo esta chamada, o servidor abre um manipulador de sessão específico para o cliente requisitante, devolvendo ao mesmo um identificador desta sessão. Não existe interação com nenhum outro módulo, pois a abertura de conexão realiza-se apenas a nível superficial, sem nenhuma verificação de segurança ou validação.
Possíveis erros são a não existência do Servidor Multimídia referenciado na chamada ou erros no estabelecimento da conexão. Os códigos de erro são, respectivamente,
`mmSERVER_INVALID` e `mmSERVER_ERROR`.
A figura II.3 mostra esquematicamente as interações existentes em uma chamada mmConnect.

Figura II.3: Aspectos internos da chamada mmConnect

mmAttach

Esta chamada do Protocolo Multimídia tem por função realizar a conexão entre uma Tarefa e um servidor. Ao receber uma mensagem contendo esta chamada, o servidor abre um manipulador de sessão específico para a Tarefa requisitante, devolvendo à mesma um identificador desta sessão. Não existe interação com nenhum outro módulo, pois a abertura de conexão realiza-se apenas a nível superficial, sem nenhuma verificação de segurança ou validação.
Possíveis erros são a não existência do Servidor Multimídia referenciado na chamada ou erros no estabelecimento da conexão. Os códigos de erro são, respectivamente,
`mmSERVER_INVALID` e `mmSERVER_ERROR`.
A figura II.4 mostra esquematicamente as interações existentes em uma chamada mmAttach.

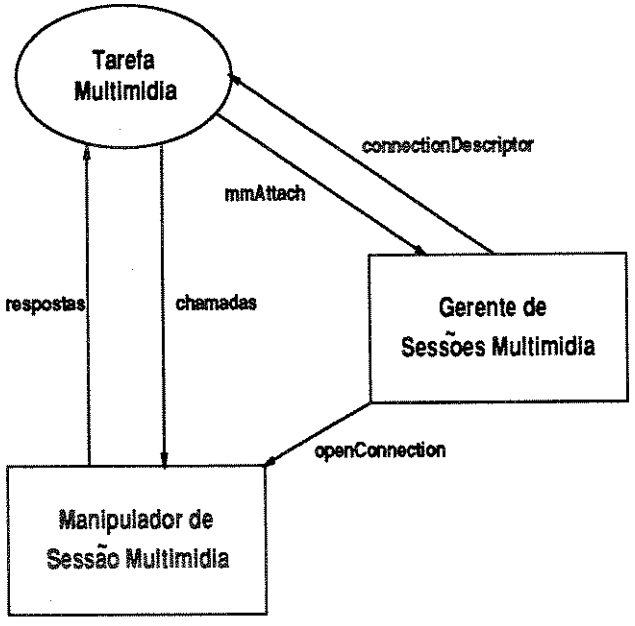

Figura II.4: Aspectos internos da chamada mmAttach

mmQuery

A chamada mmQuery é utilizada para inicializar um objeto Mídia em uma sessão. O objeto a ser inicializado deve estar registrado entre os recursos disponíveis do servidor conectado. Ao receber esta mensagem o manipulador de sessão sinaliza o gerente de Mídia que, por sua vez, irá realizar a criação do objeto, mas não sem antes consultar o gerente de registro sobre a disponibilidade do recurso requerido. Caso não exista disponibilidade do mesmo, a chamada mmCreate, que é síncrona, sinaliza o cliente sobre o insucesso.

Possíveis erros relacionados a esta chamada são a não existência de registro referente ao objeto Mídia ou a indisponibilidade para criação do mesmo. Os códigos de erro são, respectivamente, `mmMEDIA_NOT_REGISTERED` e `mmSERVER_ERROR`.

A figura II.5 mostra esquematicamente as interações existentes em uma chamada mmQuery.

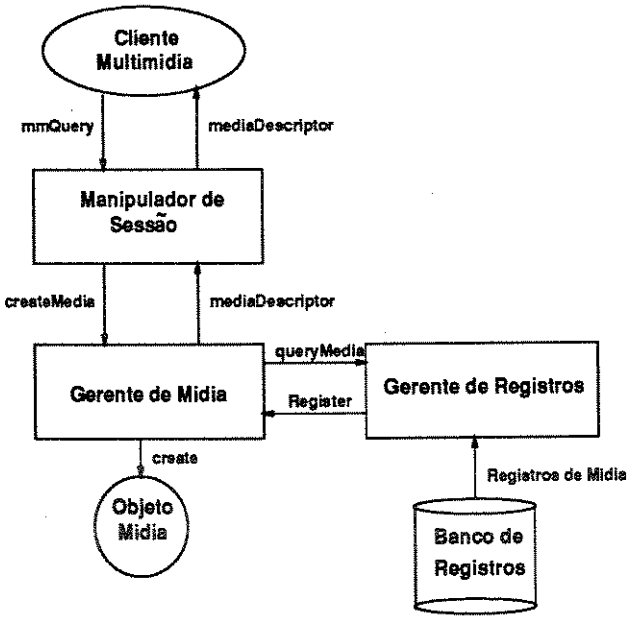

Figura II.5: Aspectos internos da chamada mmQuery

mmCreate

A criação da máquina de estados que irá reger o mecanismo da aplicação Multimídia é o próximo passo. Para a criação dos estados utiliza-se a chamada mmCreate, que deve conter a identificação de um Serviço Multimídia a ser relacionado com o estado. A chamada chega primeiramente ao manipulador de sessão que a redireciona ao gerente de Mídia. Este gerente faz a verificação primeiramente do objeto Mídia que deve estar associado com o estado e a seguir a consulta no Gerente de Registros sobre a disponibilidade e definição do serviço requerido e, caso exista o recurso, realiza a inicialização de um objeto interno representando o estado. Sendo esta chamada uma chamada síncrona, o cliente recebe um identificador do estado criado ou códigos de erro, que podem significar a não existência do objeto Mídia ou a falha na pesquisa sobre o serviço nos registros.
Possíveis erros relacionados a esta chamada são a não existência de registros referentes ao objeto a ser criado, ou a referência a um objeto Mídia inválido. Os códigos de erro são, respectivamente,
`mmSTATE_NOT_REGISTERED` e `mmMEDIA_INVALID`.
A figura II.6 mostra esquematicamente as interações existentes em uma chamada mmCreate.

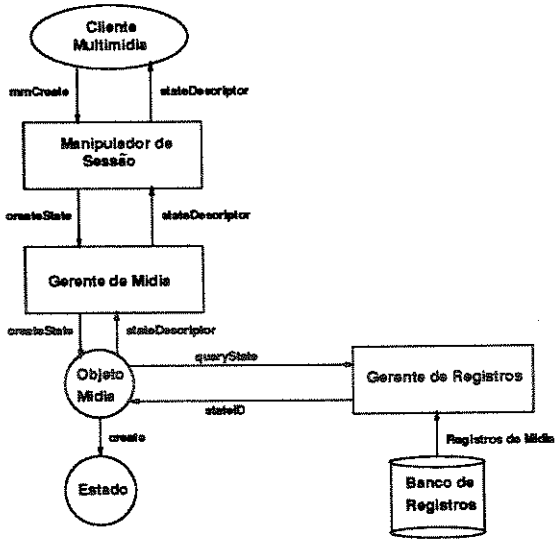

Figura II.6: Aspectos internos da chamada mmCreate

mmLink

Com a ligação entre os estados a máquina é finalmente formada logicamente e as transições entre os mesmos tornam-se possíveis. Existe neste nível uma verificação da possibilidade da ligação. O formato de dados da saída de um estado deve ser o mesmo da entrada do outro estado para que a ligação seja possível. A característica dessa ligação influi também na classificação das tarefas associadas, como de processamento, de entrada/saída ou de comunicação. Mais detalhes podem ser vistos no capítulo seguinte. A chamada chega no manipulador de sessão, que por sua vez identifica o objeto Mídia controlador dos estados a serem ligados, e o sinaliza sobre a chamada. O objeto então verifica a existência dos estados a serem ligados e realiza a ligação, caso possível.

Possíveis erros relacionados a esta chamada são a não existência do objeto Mídia associado, ou a não existência dos estados associados ou a impossibilidade de se realizar a ligação entre os estados devido a incompatibilidade dos tipos de dados tratados. Os códigos de erros são, respectivamente,

`mmMEDIA_INVALID`, `mmSTATE_INVALID` ou `mmLINK_INVALID`.

A figura II.7 mostra esquematicamente as interações existentes em uma chamada mmLink.

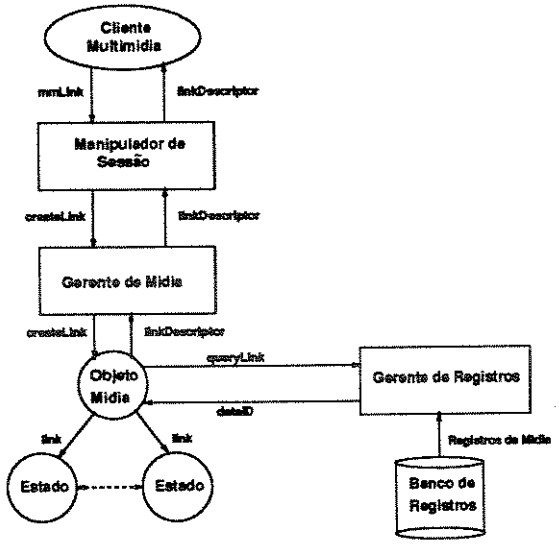

Figura II.7: Aspectos internos da chamada mmLink

mmRegister

Depois de criados os estados e efetuadas as correspondentes ligações, o cliente deve registrar eventos de seu interesse para posterior tratamento. A partir do momento que a máquina de estados é disparada, as únicas informações sobre o fluxo de controle na mesma são enviadas na forma de eventos. Existem também, além de eventos relacionados ao fluxo de controle outros característicos de cada tipo de Mídia. Esta chamada é recebida através do manipulador de sessão, que repassa a mesma ao Gerente de Eventos. Então, o Gerente de Eventos associa o cliente requisitante como interessado no recebimento do evento em questão e confirma este registro. Assim que o cliente recebe a confirmação, associa um manipulador para o evento e passa a despachar estas ocorrências.

Possíveis erros relacionados a esta chamada são a não existência do objeto Mídia associado ou a impossibilidade de se realizar o registro devido à não existência de registro referente ao evento desejado. Os códigos de erros são, respectivamente,
`mmMEDIA_INVALID` e `mmEVENT_NOT_REGISTERED`.

A figura II.8 mostra esquematicamente as interações existentes em uma chamada mmRegister.

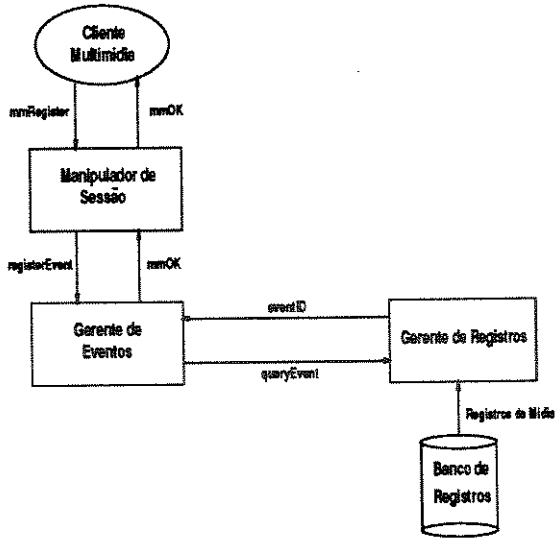

Figura II.8: Aspectos internos da chamada mmRegister

mmStart

Esta chamada dispara a máquina de estados de uma sessão, que deve ter sido previamente construída, ligada e com os eventos de interesse registrados. O manipulador de sessão repassa o comando ao Gerente de Mídia que determina quais os objetos Mídia passíveis de execução. Estes objetos, por sua vez determinam qual o estado inicial a ser disparado e sinalizam o mesmo. O controle de execução a nível de estados é realizado apenas para sinalização das Tarefas Multimídia correspondentes. A partir daí o controle fica automatizado e baseado na troca de eventos entre as entidades.

Possíveis erros relacionados a esta chamada são a não existência de qualquer máquina de estados passível de ser inicializada ou a não disponibilidade de recursos para a execução. Os códigos de erros são, respectivamente,

`mmSESSION_NOT_READY` e `mmSERVER_ERROR`.

A figura II.9 mostra esquematicamente as interações existentes em uma chamada mmStart.

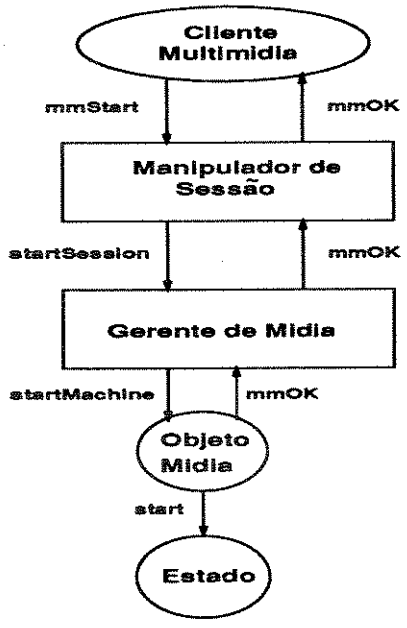

Figura II.9: Aspectos internos da chamada mmStart

mmStop

Esta chamada suspende a execução de uma máquina de estados de uma sessão, que deve ter sido previamente disparada. O manipulador de sessão repassa o comando ao Gerente de Mídia que determina quais os objetos Mídia correntemente controlando a execução de suas respectivas máquinas de estado. Estes objetos, por sua vez, suspendem a execução do estado correntemente ativo através de sinalização de parada. O controle a nível de Tarefas Multimídia se dá por intermédio do estado afetado que deve ser responsável pela suspensão dos processos correntes.

Possíveis erros relacionados a esta chamada são a não existência de qualquer máquina de estados sendo executada ou a incapacidade de suspensão devido a erros internos. Os códigos de erros são, respectivamente,

`mmSESSION_NOT_RUNNING` e `mmSERVER_ERROR`.

A figura II.10 mostra esquematicamente as interações existentes em uma chamada mmStop.

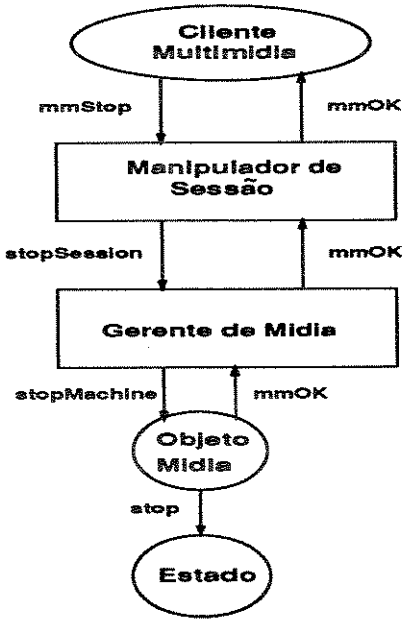

Figura II.10: Aspectos internos da chamada mmStop

mmResume

Esta chamada reassume a execução de uma máquina de estados de uma sessão, que deve ter sido previamente suspensa. O manipulador de sessão repassa o comando ao Gerente de Mídia que determina quais os objetos Mídia suspensos. Estes objetos, por sua vez, retomam a execução do estado anteriormente ativo através de sinalização de reexecução. O controle a nível de Tarefas Multimídia se dá por intermédio do estado afetado que deve ser responsável pela reexecução dos processos correntemente suspensos.

Possíveis erros relacionados a esta chamada são a não existência de qualquer máquina de estados suspensa ou a incapacidade de reexecução devido a erros internos. Os códigos de erros são, respectivamente,

`mmSESSION_ALREADY_RUNNING` e `mmSERVER_ERROR`.

A figura II.11 mostra esquematicamente as interações existentes em uma chamada mmResume.

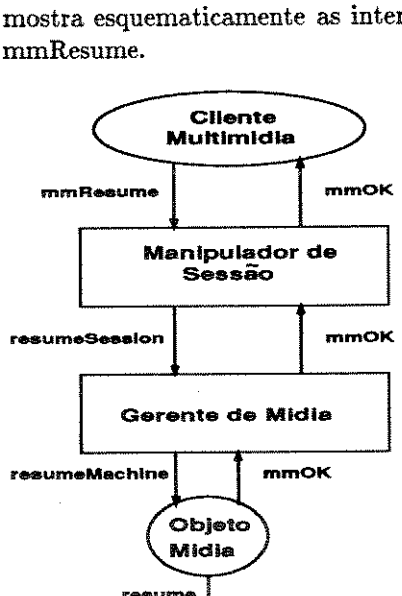

Figura II.11: Aspectos internos da chamada mmResume

mmUnregister

Os eventos anteriormente registrados em um cliente podem não ser mais necessários durante a execução de uma máquina de estados. Esta chamada sinaliza sobre o não interesse do evento para o servidor. Quando o manipulador de sessão recebe a mensagem contendo esta chamada a repassa ao Gerente de Eventos que verifica a existência ou não do registro do evento. O sucesso na destruição do registro é sinalizado ao cliente que toma as precauções cabíveis.
Possíveis erros relacionados a esta chamada são a não existência do registro referente ao evento referenciado ou erros na referência do evento. Os códigos de erros são, respectivamente,
`mmEVENT_NOT_REGISTERED` e `mmSERVER_ERROR`.
A figura II.12 mostra esquematicamente as interações existentes em uma chamada
mmUnregister.

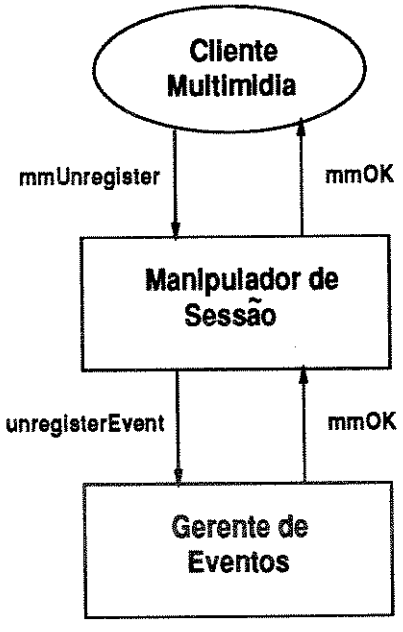

Figura II.12: Aspectos internos da chamada mmUnregister

mmUnlink

As ligações entre estados podem também ser dinamicamente alteradas. Esta chamada ocasiona a destruição da ligação entre dois estados. O manipulador de sessão recebe a mensagem referente a chamada e a repassa para o Gerente de Mídia que é então encarregado de encontrar o objeto Mídia que controla a ligação. O objeto, uma vez encontrado, realiza a desconexão dos estados. A máquina deve estar suspensa para que esta operação seja efetuada com sucesso e quando a máquina for redisparada, uma nova verificação da viabilidade desta reexecução e de sua integridade é efetuada.
Possíveis erros relacionados a esta chamada são a não existência de algum dos estados a serem desconectados, a não existência de uma conexão entre os estados ou a impossibilidade da desconexão devido ao fato da máquina correspondente se encontrar em estado de execução. Os códigos de erros são, respectivamente,
`mmSTATE_INVALID`, `mmLINK_INVALID` e `mmSESSION_ALREADY_RUNNING`.
A figura II.13 mostra esquematicamente as interações existentes em uma chamada
mmUnlink.

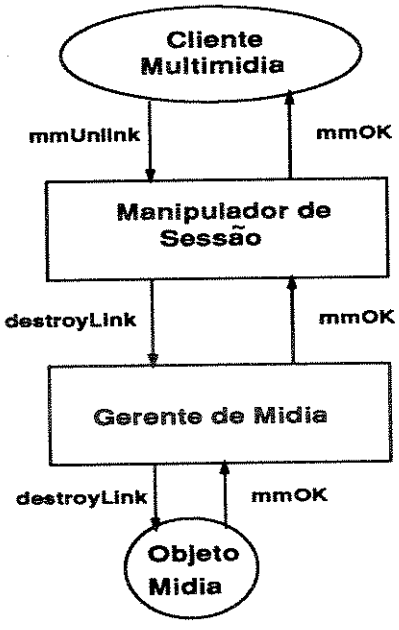

Figura II.13: Aspectos internos da chamada mmUnlink

mmDestroy

Estados relacionados a serviços Multimídia podem ser também dinamicamente destruídos tão logo não sejam mais necessários. Esta chamada ocasiona a destruição de um estado. Quando o manipulador de sessão recebe uma mensagem referente a destruição de um estado, repassa a mesma ao Gerente de Mídia que tem a função de identificar o objeto Mídia que controla aquele estado. Este objeto, após identificado tem a função de realizar a destruição do estado, desde que a máquina correspondente não esteja correntemente em estado de execução.

Possíveis erros relacionados a esta chamada são a não existência do estado a ser destruído, a impossibilidade da destruição devido ao fato da máquina correspondente se encontrar em execução, ou devido a erros internos que inviabilizem a destruição. Os códigos de erros são, respectivamente,
`mmSTATE_INVALID`, `mmSESSION_ALREADY_RUNNING` e `mmSERVER_ERROR`.
A figura II.14 mostra esquematicamente as interações existentes em uma chamada mmDestroy.

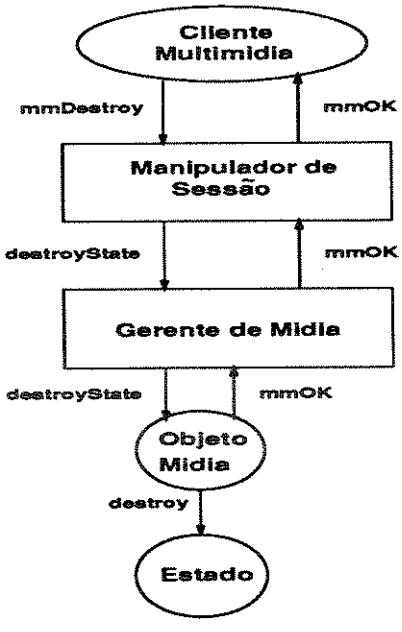

Figura II.14: Aspectos internos da chamada mmDestroy

mmDetach

Quando uma tarefa particular de uma sessão Multimídia estiver cumprida a sessão deve ser desconectada, de modo que os recursos alocados pela mesma sejam liberados. Com a desconexão automaticamente são destruídos todos os registros de eventos e os vínculos com máquinas de estados de outras sessões. O manipulador de sessão se encarrega de rotear as mensagens de destruição ao Gerente de Eventos.

Possíveis erros relacionados a esta chamada são a não existência da conexão referenciada ou algum erro interno que impossibilite a desconexão. Os códigos de erros são, respectivamente, `mmSESSION_INVALID` e `mmSERVER_ERROR`.

A figura II.15 mostra esquematicamente as interações existentes em uma chamada mmDetach.

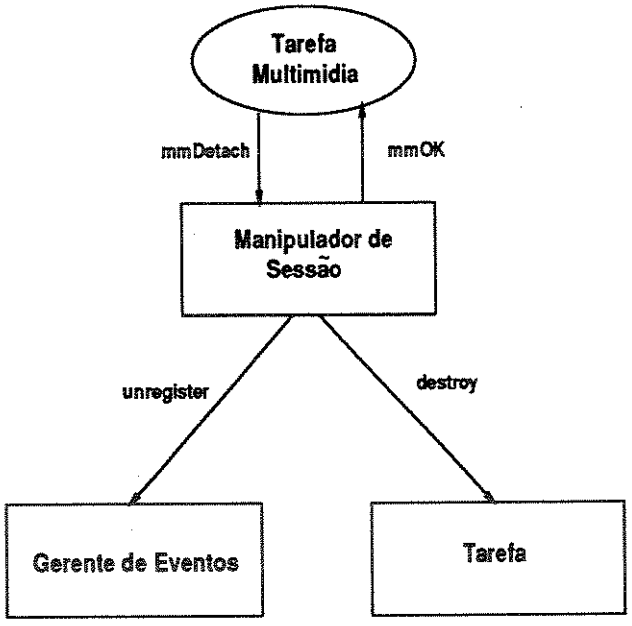

Figura II.15: Aspectos internos da chamada mmDetach

mmDisconnect

Quando as tarefas desejadas de uma sessão Multimídia estiverem cumpridas a sessão deve ser desconectada, de modo que os recursos alocados pela mesma sejam liberados. Com a desconexão automaticamente são destruídos todos os registros de eventos, as ligações e os estados correspondentes das máquinas de estado da sessão. O manipulador de sessão se encarrega de rotear as mensagens de destruição ao Gerente de Eventos ou ao Gerente de Mídia.
Possíveis erros relacionados a esta chamada são a não existência da conexão referenciada ou algum erro interno que impossibilite a desconexão. Os códigos de erros são, respectivamente,
`mmSESSION_INVALID` e `mmSERVER_ERROR`.
A figura II.16 mostra esquematicamente as interações existentes em uma chamada
mmDisconnect.

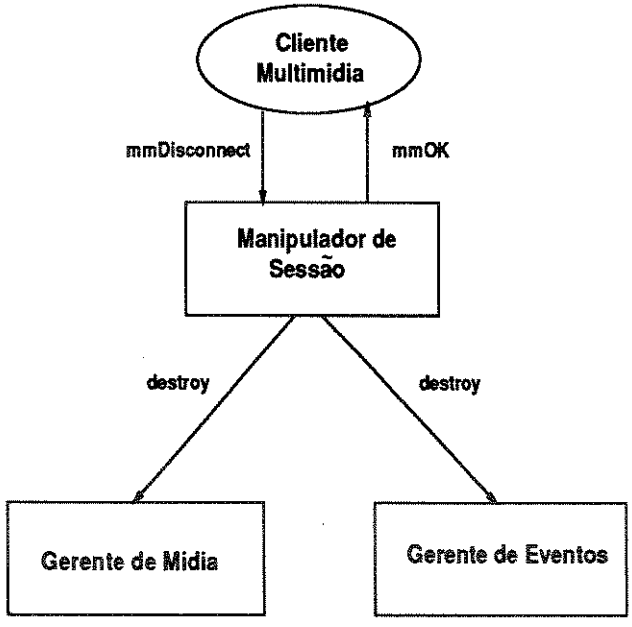

Figura II.16: Aspectos internos da chamada mmDisconnect

II.2.2 Camada de Controle de Mídia

Como foi visto anteriormente as diversas chamadas do Protocolo Multimídia acarretam uma comunicação entre as camadas de sessão e de controle de Mídia. A camada de controle de Mídia realiza o controle lógico dos serviços requisitados, através da criação e gerenciamento de objetos Multimídia. Por objetos Multimídia entenda-se objetos internos ao servidor que façam parte do modelamento dos serviços Multimídia, como por exemplo, objetos Mídia, Registros de Mídia, Eventos, Estados componentes de máquinas de estado, transições entre estes estados, exceções e Tarefas. A camada de controle de Mídia é subdividida entre três componentes principais, cada um cuidando de aspectos específicos de controle. Os componentes são:

- Gerente de Mídia
- Gerente de Registros
- Gerente de Eventos

O módulo de gerenciamento de Mídia encarrega-se da estrutura lógica das máquinas de estados. É encarregado de criar, controlar e destruir objetos que representem Mídias, Estados e suas Ligações e Exceções. Para configurar estes objetos e definir quais as suas características existe o módulo de gerenciamento de registros de Mídia, que é encarregado de gerenciar um banco de registros onde cada tipo de Mídia é definida, seus tipos de dados e tipos de eventos, bem como seus serviços traduzidos em tarefas Multimídia. Outro fator de controle importante é a dinâmica de uma máquina de estados, depois de disparada. Tal controle é efetuado por meio de eventos e o módulo responsável pelo gerenciamento destes eventos é o gerenciador de eventos Multimídia. Na figura II.17 encontramos um esquema das interações entre os diversos componentes responsáveis pelo gerenciamento de Mídia e de estados Multimídia.

A seguir segue-se um detalhamento destes módulos e suas relações com os demais componentes.

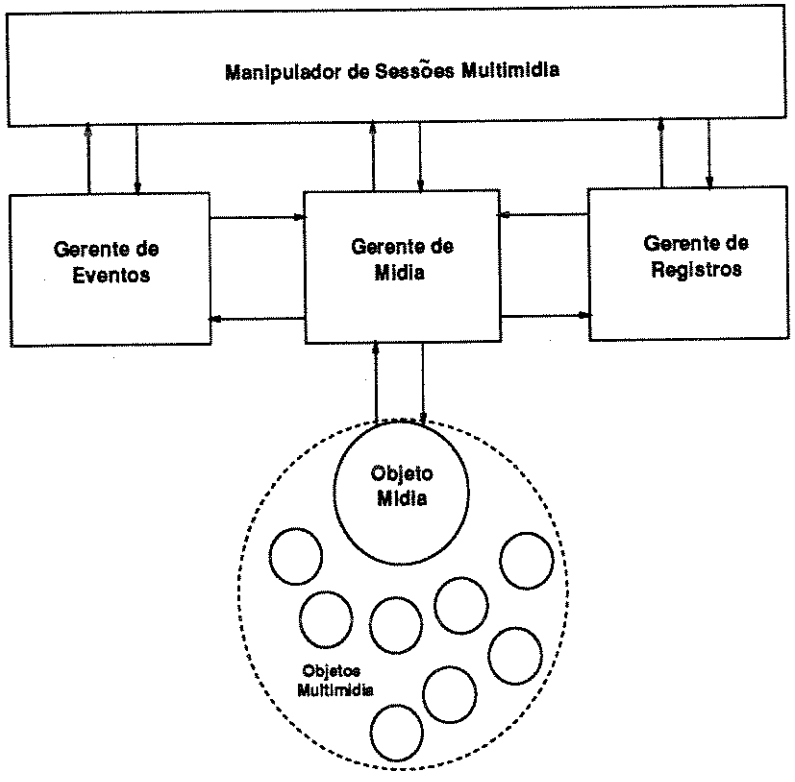

Figura II.17: Módulos de Gerenciamento de Mídia e de Estados Multimídia

Módulo de Gerenciamento de Mídia

O objetivo principal deste módulo é a criação e controle dos objetos internos do modelo de uma Mídia. Entre os serviços oferecidos podemos citar:

- Criação de Mídias, Estados, Transições, Células de Dados e Tarefas
- Destruição de objetos
- Controle do fluxo de informações entre os objetos
- Controle de Transições numa máquina de estados
- Geração de Exceções numa máquina de estados

A criação de objetos é realizada mídiante prévia consulta ao Gerente de Registros para validação do objeto a ser criado. Depois desta validação uma instância da classe que corresponde ao objeto pedido é criada e um identificador da mesma é retornado para referência.

A destruição dos objetos criados depende da existência dos mesmos. O objeto a ser destruído deve ser referenciado por meio de seu descritor.

Objetos que representam estados de uma máquina necessitam realizar um intercambio de informações entre si, para que dados processados em um estado possam ser reprocessados em estados posteriores. O controle da passagem de informações entre estados é realizado através da criação e acesso a objetos especiais, chamados de células de dados. Seu comportamento é tal que o intercambio de informações seja possível inclusive entre processos distintos e também entre máquinas distintas. A distribuição de descritores destes objetos torna possível o controle do fluxo de informações na máquina de estados.

O fluxo de controle de uma máquina de estados através de objetos especiais, chamados transições, também se torna possível através da consulta ao Gerenciador de Eventos. Cada transição tem associada um evento específico, e assim que for detetado pelo Gerenciador de Eventos, sinaliza-se o acontecimento de uma transição internamente ao Gerente de Objetos. Este módulo tem então a incumbência de manipular corretamente as suas máquinas de estados de modo que o estado anterior seja desativado e o próximo estado seja ativado. Mais detalhadamente, verifica-se que um apontador que referencia os estados corretamente em execução é renovado e os estados afetados são sinalizados para suspensão, no caso do estado que originou a transição e para ativação, no caso do estado de destino da transição.

Por fim, situações anormais ou de erro devem gerar exceções para que os clientes interessados possam tomar as devidas medidas. A geração destas exceções se realiza através da sinalização de um evento correspondente ao tipo da exceção para o Gerente de Eventos.

Módulo de Gerenciamento de Registros

O objetivo principal deste módulo é a criação e controle de objetos especiais, chamados de Registros de Mídia, objetos estes que servem para definir parâmetros e comportamentos de uma Mídia. Entre os serviços oferecidos por este módulo podemos citar:

- Criação Registros de Mídia
- Destruição de Registros de Mídia

- Pesquisa e manutenção de um Banco de Registros de Mídia local
- Distribuição das informações registradas
- Testes e procuras nos Bancos de Registros

Primeiramente este módulo deve ser capaz de acessar um banco de dados especial, local a cada máquina, chamado de Banco de Registros de Mídia. Este banco de dados contém a configuração do servidor Multimídia para cada máquina, através da disponibilidade de registros de Mídia. Um registro de Mídia é composto dos seguintes elementos:

- Identificador de Mídia
- Descrições de Tipos de Dados
  - Identificador do Tipo de Dados
  - Tamanho da Célula de Dados
- Descrições de Tipos de Eventos
  - Identificador do Evento
  - Identificador da Classe do Evento
- Descrições de Tipos de Estados
  - Identificador do Estado
  - Identificador do Serviço Multimídia
  - Lista de Identificadores de Dados aceitos para processamento
  - Lista de Identificadores de Dados retornados após processamento
  - Identificador do processo correspondente ao Estado

Podemos descrever a sintaxe das definições acima utilizando-se de uma forma canônica simples, comumente utilizada em descrições de linguagens. Esta representação considera definições abstratas representadas entre [], identificadores ou elementos de linguagem representados entre ¡¿ e palavras reservadas em texto simples. Além disso, definições compostas agrupam outros elementos entre e indicadores de multiplicidade ... podem ser utilizados para especificar mais de um elemento do mesmo tipo. Observe a especificação completa dos registros de Mídia na figura II.18.

O exemplo da figura II.19 mostra a definição de um Registro de Mídia destacando alguns de seus componentes e a sintaxe utilizada.

Como visto no exemplo, através de uma linguagem simples e da definição de identificadores é possível criar um Registro de Mídia como um objeto interno no servidor, associado ao objeto Mídia correspondente, de modo a configurar o mesmo, bem como seus possíveis estados, tipos de dados e eventos associados.

Uma vez criados os objetos de Registro de Mídia e associados aos objetos Mídia, sua destruição é vinculada ao mesmo. Quando um objeto Mídia é destruído, a instância correspondente ao registro também é automaticamente descartada.

Outros serviços pertinentes a este módulo referem-se a manutenção do banco de dados externo que define os registros de Mídia, além de serviços de procura e localização de registros no mesmo através de identificadores. Como exemplo, uma chamada que localize todos os registros que contenham a definição de um determinado evento Multimídia. A distribuição destas informações para outros módulos, como o Gerente de Eventos ou o Gerente de Objetos, para que realizem seus próprios serviços também deve ser possível.

Módulo de Gerenciamento de Eventos

O objetivo principal deste módulo é a criação e controle de objetos especiais, chamados Eventos. Cada objeto define situações ou acontecimentos relacionados ao ambiente que são utilizados para fins de controle ou sincronização. É este módulo também o responsável pela identificação destes acontecimentos e pela associação do mesmo a um Evento previamente definido. Entre serviços oferecidos podemos citar:

- Criação e customização de Eventos
- Destruição de Eventos
- Controle do fluxo de eventos entre os objetos
- Recepção de eventos externos ao servidor
- Envio de eventos externos ao servidor

A criação e customização de objetos Eventos se dá pela associação de um Evento Multimídia, que deve ser previamente validado através dos serviços de procura do Gerente de Registros, e de uma lista de entidades interessadas no acontecimento do evento. Entre estas entidades podem se encontrar módulos internos, como no caso do Gerente de Objetos ou do Gerente de

```
[MEDIA_DEFS]:       Media <MEDIA_ID>
                    {
                      [DATA_DEFS]
                      [EVENT_DEFS]
                      [STATE_DEFS]
                    }
[DATA_DEFS]:        Data <DATA_ID>
                    {
                      [DATA_SIZE]
                      [DATA_PERMS]
                    }
[EVENT_DEFS]:       Event <EVENT_ID>
                    {
                      [EVENT_CLASS]
                    }
[STATE_DEFS]:       State <STATE_ID>
                    {
                      [SERVICE_DEFS]
                      [INPUT_DEFS]
                      [OUTPUT_DEFS]
                      [PROCESS_DEFS]
                    }
[DATA_SIZE]:        Size <DATA_SIZE>;
[EVENT_CLASS]:      Class <EVENT_CLASS>;
[SERVICE_DEFS]:     Service <SERVICE_ID>, ...;
[INPUT_DEFS]:       Input <DATA_ID>, ...;
[OUTPUT_DEFS]:      Output <DATA_ID>, ...;
[PROCESS_DEFS]:     Task <PROCESS_ID>, ...;
<MEDIA_ID>,
<DATA_ID>,
<EVENT_ID>,
<STATE_ID>,
<SERVICE_ID>,
<PROCESS_ID>:       <ID>
<EVENT_CLASS>:      <CLASS>
<DATA_SIZE>:        INTEGER
<CLASS>,
<ID>:               STRING
```

Figura II.18: Sintaxe do Registro de uma Mídia

```
/* Identificador do objeto Video */
Media "VIDEO_OBJECT"
{
    /* Identificador de Video codificado */
    Data "VIDEO_CODED" {
      Size 512KB; /* Tamanho maximo */
    }
    /* Identificador de Video decodificado */
    Data "VIDEO_DECODED" {
      Size 2MB; /* Tamanho maximo */
    }
    /* Identificador do evento de decodificacao */
    Event "VIDEODECODING" {
      Class "VIDEO_PROCESS"; /* Classe do evento */
    }
    /* Identificador do evento de transformacao */
    Event "VIDEODITHERING" {
      Class "VIDEO_PROCESS"; /* Classe do evento */
    }
    /* Identificador do evento de erro */
    Event "VIDEOERROR" {
      Class "ERROR"; /* Classe do evento - erro */
    }
    /* Identificador do estado de decodificacao */
    State "VIDEO_DECODE" {
      Service "DECODE";        /* Servico do estado  */
      Input   "VIDEO_CODED";   /* Dados de entrada   */
      Output  "VIDEO_DECODED"; /* Dados de saida     */
      Task    "video_decode";  /* Processo do estado */
    }
    /* Identificador do estado de transformacao */
    State "VIDEO_DITHER" {
      Service "DITHER";        /* Servico do estado  */
      Input   "VIDEO_DECODED"; /* Dados de entrada   */
      Output  "VIDEO_DECODED"; /* Dados de saida     */
      Task    "video_dither";  /* Processo do estado */
    }
}
```

Figura II.19: Exemplo de um Registro de Mídia

Tarefas ou mesmo processos externos ao servidor, como clientes ou tarefas Multimídia. Quando o evento correspondente é processado pelo Gerente de Eventos, as entidades interessadas são notificadas, seja apenas através de sinalização interna, através da chamada de serviços ou de sinalização externa, através do uso de técnicas de comunicação entre processos.

A destruição destes objetos especiais de registro de eventos se dá através da desistência de todas as entidades registradas da monitoração do evento. Esta desistência realiza-se por meio de um serviço do Gerente de Eventos. Outras funções deste módulo são o correto roteamento de sinalizações de eventos entre os objetos internos e o envio e recepção de eventos externos ao servidor através de primitivas de comunicação.

II.2.3 Camada de Controle de Tarefas

Depois de definida a lógica e as ações de uma determinada máquina de estados associada a um objeto Mídia, necessita-se realizar o acionamento e gerenciamento dos serviços Multimídia relacionadas com cada estado. Este controle é realizado no nível da camada de controle de tarefas, através do Módulo de Controle de Tarefas ou Gerente de Tarefas. Sua função é a de utilizar serviços específicos da máquina e sistema operacional hospedeiro de modo a acionar e controlar processos referentes a serviços, definidos como Tarefas Multimídia, além de ser responsável pela representação destes processos de modo que seja possível referenciá-los nos moldes já definidos logicamente, ou seja, nas máquinas de estados. Para isso surge a necessidade de mais um objeto especial, chamada de Objeto Tarefa e que faz este papel de representação.

Entre os serviços oferecidos por este módulo podemos identificar:

- Criação de Objetos Tarefa
- Destruição de Objetos Tarefa
- Controle de Processos externos relacionados a Objetos Tarefa

Internamente a um objeto estado, são referenciados estes objetos especiais, chamados objetos Tarefa. Um objeto Tarefa é criado pelo Gerente de Tarefas e automaticamente associado ao objeto Estado requisitante do serviço de criação. A partir daí o controle do processo relacionado àquela tarefa fica sob controle do objeto Tarefa e, por consequência, sob o controle do Gerente de Tarefas.

A interação entre estes objetos e o processo de criação de uma tarefa pode ser visto na figura II.20.

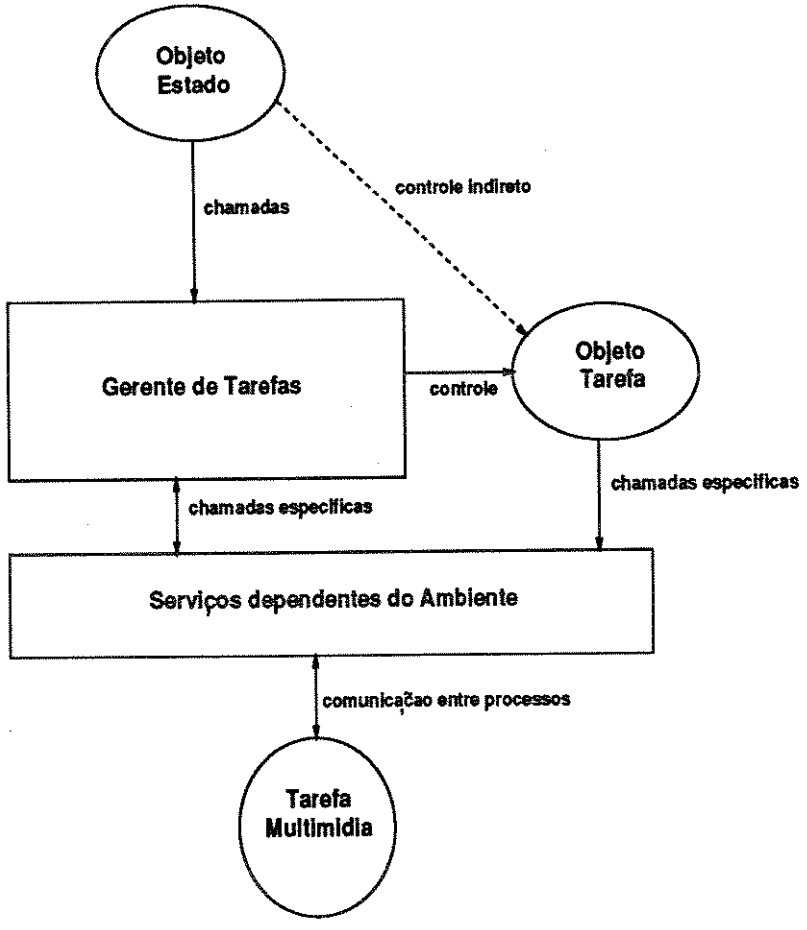

Figura II.20: Interações envolvendo o Gerente de Tarefas

A destruição de um Objeto Tarefa pode ser ordenada pelo Objeto Estado correspondente ou é automatica na ocasião da destruição do Objeto Estado correspondente. O controle dos processos é realizado por intermédio de uma camada intermediéria entre o servidor e o ambiente operacional hospedeiro que contém definições de serviços básicos necessários ao controle. Estes serviços são:

- Criação de um processo
- Destruição de um processo
- Acionamento de um processo
- Suspensão de um processo
- Retomada de execução de um processo
- Sinalização entre processos
- Troca de mensagens entre processos

Estes serviços básicos são posteriormente utilizados para o correto andamento das máquinas de estados definidas logicamente nas camadas superiores. São providos para os estados serviços de criação, destruição, acionamento, parada e retomada de execução de tarefas. Logicamente, estes serviços são chamados por intermédio de um descritor de tarefa que o Objeto Estado deve receber na criação de um Objeto Tarefa e utilizar em outros serviços.

II.3 Modelamento dos Objetos do Ambiente Multimídia

Os modelos do servidor Multimídia e de objetos Mídia valem-se da definição de toda uma arquitetura de objetos que representem as suas estruturas e que forneçam meios de controle por parte dos processos envolvidos, notoriamente os clientes e os servidores Multimídia. Existem três tipos de objetos envolvidos nesta arquitetura. Os primeiros são objetos básicos, para auxílio na representação e na especialização de objetos mais complexos. Existem os objetos internos, que representam o modelo da máquina de estados e o modelo de uma Mídia, ditos objetos Multimídia. Outro tipo de objeto seriam os objetos de controle que irão formar a funcionalidade do servidor propriamente dito, que correspondem aos gerentes e controladores internos.

II.3.1 Objetos Genéricos

Entre os objetos genéricos, podemos citar as seguintes classes básicas que serão utilizadas no topo das demais hierarquias de objetos:

Classe Object
Ascendente: Nenhum
Definição de um objeto genérico simples, ou seja, que não seja composto de outros objetos. É o princípio da hierarquia de objetos. Contém características básicas como nome e identificador de um objeto.

Classe List
Ascendente: Nenhum
Definição de um objeto genérico composto, ou seja, que contenha referências a outros objetos. É o princípio da hierarquia de objetos compostos. Armazena informações diversas num formato de lista dinâmica.

Classe ListIterator
Ascendente: Nenhum
Definição de um objeto genérico auxiliar para acesso e controle de um objeto List. Origina vários tipos de iteradores, um para cada objeto List utilizado.

Os objetos genéricos podem ser vistos na figura II.21.

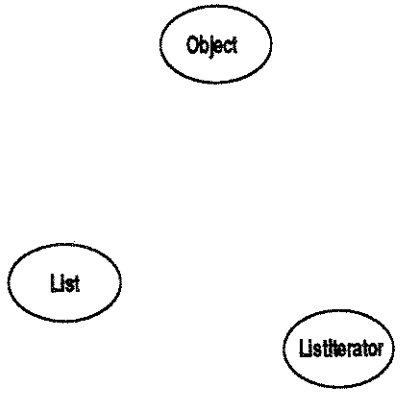

Figura II.21: Objetos Genéricos do Ambiente

II.3.2 Objetos Internos

Entre os objetos internos, podemos citar as seguintes classes básicas que serão utilizadas no modelamento das Mídias e das correspondentes máquinas de estados:

Classe Entity

Ascendente: *Object*

Esta classe define uma especialização de um objeto simples que servirá como base para a elaboração de classes que representem entidades internas ao servidor. Esta classe pode ser referenciada polimorficamente para designar seus descendentes.

Classe Session

Ascendente: *Entity*

Representa a especialização de uma entidade que contém características de uma Sessão Multimídia. É inicializada sempre que uma chamada de conexão é recebida. Sua função é direcionar as chamadas subsequentes para aquela seção de modo que sejam corretamente executadas pelos demais módulos.

Classe ClientSession

Ascendente: *Session*

Representa a especialização de um objeto Session. É inicializada sempre que a sessão Multimídia relacionada for ocasionada por um processo Cliente. Sua função é controlar as interações entre o processo Cliente e o Servidor.

Classe TaskSession

Ascendente: *Session*

Representa a especialização de um objeto Session. É inicializada sempre que a sessão Multimídia relacionada for ocasionada por um processo Tarefa. Sua função é controlar as interações entre o processo Tarefa e o Servidor.

Classe Mídia

Ascendente: *Entity*

Representa a especialização de uma entidade que contém características de uma Mídia. É inicializada sempre que uma chamada de criação de Mídia é recebida. Sua função é configurar operações que referenciem serviços relacionados a uma determinada Mídia.

Classe State

Ascendente: *Entity*

Representa a especialização de uma entidade que contém características de um Estado. É inicializada sempre que uma chamada de criação de Estado é recebida. Sua função é representar um componente de uma máquina de estados e prover funcionalidade de processamento através de comunicação com as camadas inferiores.

Classe Process

Ascendente: *Entity*

Representa a especialização de uma entidade que contém características de um Processo externo ao Servidor. É inicializada sempre que um pedido de conexão é recebido. Sua função é interagir com o processo externo através de sinais e troca de informações.

Classe Client

Ascendente: *Process*

Representa a especialização de um objeto Process e caracteriza um Cliente conectado ao Servidor, responsável pela criação da máquina de estados.

Classe Task

Ascendente: *Process*

Representa a especialização de um objeto Process e caracteriza uma Tarefa conectada ao Servidor, responsável pela execução dos Serviços Multimídia requeridos pela máquina de estados.

Classe Action

Ascendente: *Object*

Esta classe define uma especialização de um objeto simples que servirá como base para a elaboração de classes que representem ações internas às máquinas de estados. Esta classe pode ser referenciada polimorficamente para designar seus descendentes.

Classe Transition

Ascendente: *Action*

Representa a especialização de uma ação que contém características de uma Transição de estados. É inicializada sempre que uma chamada de ligação entre estados é recebida. Sua função é representar uma transição de estados em uma máquina e armazenar referências para que a transição seja executada corretamente.

Classe Exception

Ascendente: *Action*

Representa a especialização de uma ação que contém características de uma Exceção ou erro. É inicializada sempre que eventos de exceções são definidos. Sua função é representar e armazenar referências a eventos e ações a serem tomadas na ocorrência de acontecimentos ditos exceções.

Classe Signal

Ascendente: *Action*

Representa a especialização de uma ação que contém características de um Sinal de comunicação entre os processos envolvidos na máquina de estados. Sua função é assegurar que estes sinais de controle sejam corretamente recebidos.

Classe Information

Ascendente: *Object*

Esta classe define uma especialização de um objeto simples que servirá como base para a elaboração de classes que representem informações e dados internos ao servidor. Esta classe pode ser referenciada polimorficamente para designar seus descendentes.

Classe Data

Ascendente: *Information*

Representa a especialização de uma informação que contém características de Dados associados a entrada ou saída no processamento efetuado internamente a um estado. É inicializada sempre que um Estado necessite de uma área de dados para utilizar como entrada de uma tarefa ou como saída após processamento. Sua função é assegurar um formato de dados específico às necessidades do Estado ao qual pertence.

Classe Event

Ascendente: *Information*

Representa a especialização de uma informação que contém características de um Evento ou acontecimento do ambiente. É inicializada sempre que um evento seja registrado ou definido. Sua função é assegurar que os eventos representados sejam corretamente sinalizados às entidades que estejam interessadam.

Classe Register

Ascendente: *Information*

Representa a especialização de uma informação que contém características de um registro ou configuração de elementos Multimídia. É inicializada sempre que uma chamada que acarrete uma busca de configuração seja efetuada. Sua função é armazenar referências de configuração para que possam ser facilmente acessadas por outros objetos.

Classe MediaRegister

Ascendente: *Register*

Representa a especialização de um registro que contém características de gerais relacionados a uma Mídia. É inicializada sempre que uma chamada que acarrete uma busca de configuração de Mídia seja efetuada. Sua função é armazenar referências de configuração para que possam ser facilmente acessadas por outros objetos.

Classe DataRegister

Ascendente: *Register*

Representa a especialização de um registro que contém características de dados relacionados a uma Mídia. É inicializada sempre que uma chamada que acarrete uma busca de configuração de Dados seja efetuada. Sua função é armazenar referências de configuração para que possam ser facilmente acessadas por outros objetos.

Classe StateRegister

Ascendente: *Register*

Representa a especialização de um registro que contém características de um estado relacionado a uma Mídia. É inicializada sempre que uma chamada que acarrete uma busca de configuração de um Estado seja efetuada. Sua função é armazenar referências de configuração para que possam ser facilmente acessadas por outros objetos.

A hierarquia de objetos internos mostrando sua relação de especialização pode ser vista na figura II.22.

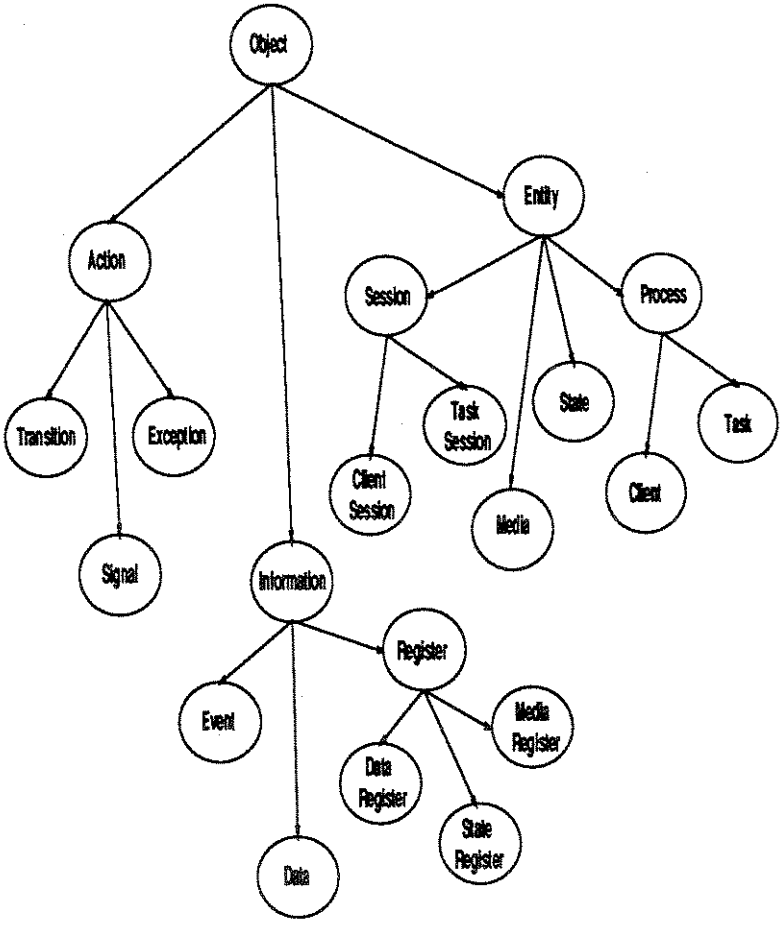

Figura II.22: Hierarquia de Objetos Internos

Existem também vários objetos compostos que representam listas dos objetos simples descritos acima. Estes objetos são todos descendentes do objeto Lista (*List*) e são utilizados sempre que uma classe necessite possuir referências de outras classes. Os objetos que possuem versões compostas são os seguintes:

- EntityList
- SessionList
- MediaList
- StateList
- ProcessList
- ActionList
- TransitionList
- ExceptionList
- SignalList
- InformationList
- EventList
- DataList
- RegisterList

Nas seções posteriores será mostrada a relação de contenção entre os objetos. Esta relação necessariamente utiliza-se dos objetos compostos para representar objetos contidos, pois um grupamento de objetos simples é conseguido através destes objetos compostos. Por exemplo, dizemos que um estado está contido em uma Mídia, que possui um objeto composto do tipo Lista de Estados (*StateList*). Com isso o objeto Mídia (*Media*) pode guardar referências para vários objetos Estado (*State*).

A hierarquia de objetos compostos mostrando sua relação de especialização pode ser vista na figura II.23.

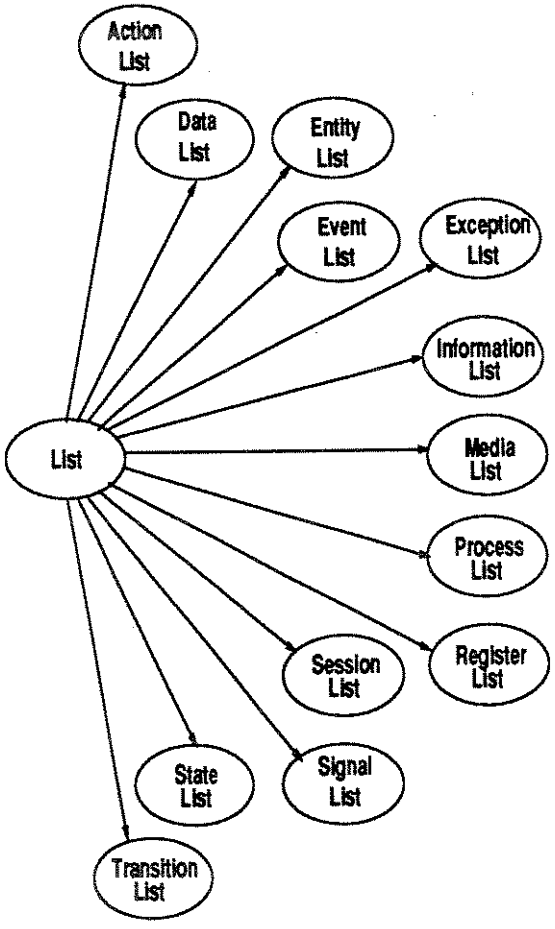

Figura II.23: Hierarquia de Objetos Compostos

II.3.3 Objetos de Controle

Entre os objetos de controle, podemos citar as seguintes classes básicas que serão como módulos do servidor e representarão as camadas do modelo definido:

Classe Server
Ascendente: *Object*
Esta classe representa a definição completa de um servidor Multimídia, relacionando todos os seus módulos internos e controlando a relação entre os mesmos.

Classe Manager
Ascendente: *Object*
Esta classe representa a abstração de um objeto de gerenciamento interno. Como todos os módulos do servidor foram projetados de modo a se comportarem como gerentes de objetos especiais, os objetos internos, existe a necessidade de se definir uma classe comum a estes gerentes, que pode referenciar polimorficamente seus descendentes.

Classe SessionManager
Ascendente: *Manager*
Representação de um Gerente de Sessões que realiza todos os serviços atribuídos à camada de sessão no modelo proposto. É inicializada sempre que uma conexão válida é recebida e passa a gerenciar as demais chamadas destinadas à sessão.

Classe ObjectManager
Ascendente: *Manager*
Representação de um Gerente de Objetos que realiza os serviços genéricos correspondentes a características comuns dos objetos internos ao Servidor, como controle de identificação.

Classe MídiaManager
Ascendente: *Manager*
Representação de um Gerente de Mídia que realiza os serviços atribuídos ao módulo de controle de objetos no modelo proposto. Realiza o controle de objetos Mídia e de máquinas de estados e seus objetos componentes, Estados, Transições, Exceções e Dados. Faz parte da camada de controle de Mídia.

Classe RegistrationManager

Ascendente: *Manager*

Representação de um Gerente de Registros que realiza os serviços atribuídos ao módulo de controle de registros no modelo proposto. Realiza o controle de objetos de Registro de Mídia e os atribui através de informações requisitadas aos outros gerentes componentes da camada de controle de Mídia, da qual também é componente.

Classe EventManager

Ascendente: *Manager*

Representação de um Gerente de Eventos que realiza os serviços atribuídos ao módulo de controle de eventos no modelo proposto. Realiza o controle de objetos de Eventos e sinaliza processos interessados nos acontecimentos representados por eventos. Faz parte da camada de controle de Mídia.

Classe TaskManager

Ascendente: *Manager*

Representação de um Gerente de Tarefas que realiza os serviços atribuídos ao módulo de controle de tarefas no modelo proposto. Realiza o controle de objetos Tarefa e também dos processos representados por estes objetos. Utiliza funções específicas do ambiente operacional para realizar este controle. Faz parte da camada de controle de Tarefas.

A hierarquia de objetos mostrando sua relação de especialização pode ser vista na figura II.24.

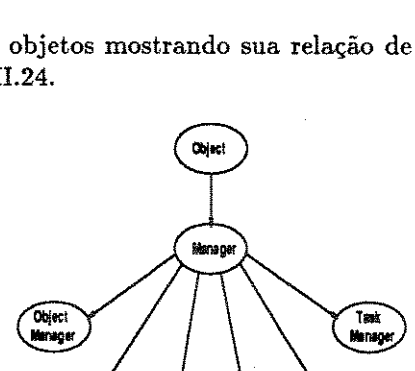

Figura II.24: Hierarquia de Objetos de Controle

II.3.4 Relacionamento entre os Objetos

Existe também uma relação de contenção que determina como os objetos se relacionam através de referências e trocas de mensagens. Dizemos que uma mensagem acionada em um objeto é o mesmo que o acionamento de um método do objeto. Então existe uma relação de acionamento de métodos entre os módulos e os objetos internos do servidor que só pode ser efetuada através da referência de um método de um objeto em outro. Isso é conseguido através da referência do próprio objeto em outro. Dizemos então que um objeto está contido em outro, ou seja, há uma relação de contenção.

O objeto Server, depois do acionamento do processo do servidor, inicializa um objeto *SessionManager* que é incumbido de monitorar pedidos de conexão. Quando um chamada de um processo cliente atinge o servidor, o objeto SessionManager valida o pedido de conexão e inicializa o objeto Session, que a partir de agora representa uma Sessão Multimídia e passa a monitorar as chamadas daquela sessão. São então inicializados os demais gerentes, *ObjectManager*, *RegistrationManager*, *EventManager* e *TaskManager*, que irão ser incumbidos do tratamento de chamadas que afetem respectivamente, controle de objetos *Media* e de máquinas de estado, controle de registros, controle de eventos e controle de tarefas. Estes gerentes são responsáveis pela criação, destruição e gerenciamento de objetos *Media* (representando definições de Mídia), *Register* (representando registros de Mídia), *Event* (representando Eventos) e *Task* (representando Tarefas Multimídia). Objetos *Media* são objetos especiais que são incumbidos de controlar objetos *State*, que por sua vez representam Estados da máquina Multimídia e que possuem referência e controle de objetos *Exception* (representando Exceções), *Transition* (representando Transições) e *Data* (representando Dados e informações pertinentes). A comunicação entre estes objetos se dá através do acionamento de métodos de objetos contidos. A contenção se realiza pela inicialização de objetos compostos, no caso dos objetos internos, e da inicialização de objetos de controle, no caso dos gerentes internos ao servidor.

A relação de contenção dos objetos do Servidor é mostrada graficamente na figura II.25.

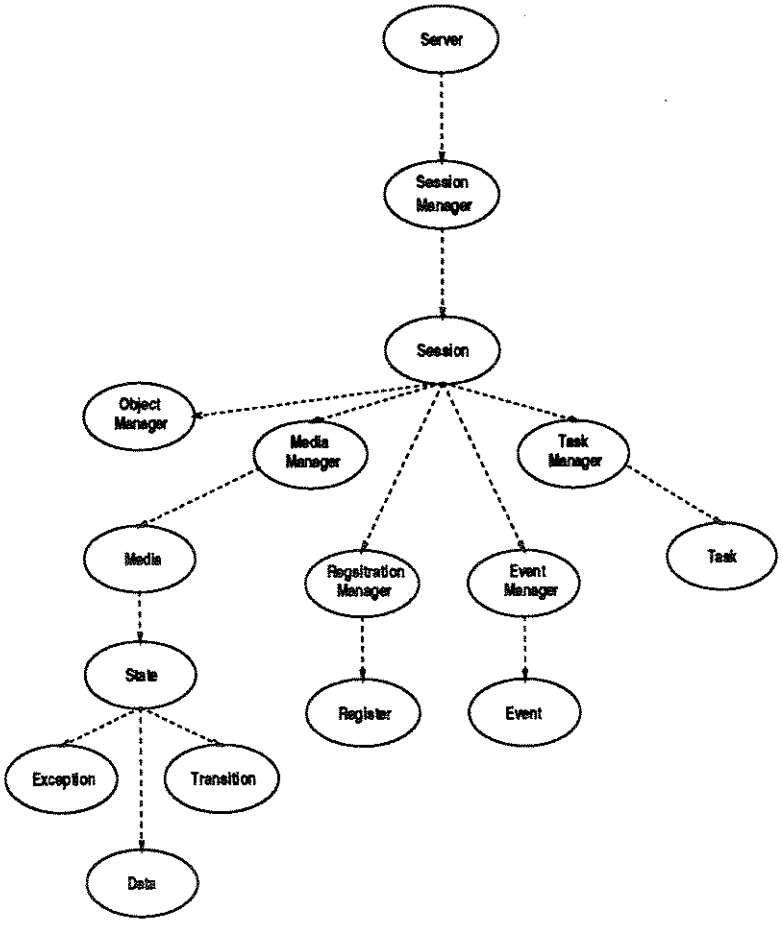

Figura II.25: Relação de Contenção

II.4 Comentários

O modelo proposto divide a funcionalidade do ambiente em camadas. A primeira camada representa as aplicações Multimídia e são relacionadas aos processos clientes. A camada de sessão representa as conexões entre clientes e servidores através do protocolo Multimídia. As demais camadas são encapsuladas no servidor Multimídia e são subdivididas de modo a modelar estruturas lógicas representadas por objetos internos e máquinas de estados que se encarregarão da execução dos serviços Multimídia. A definição deste modelo mostrou-se bastante flexível e robusta, de modo que os objetivos deste ambiente sejam plenamente atingidos. A flexibilidade, portabilidade, interoperabilidade e eficiência são atingidas através da partição do ambiente em processos distintos baseados na tecnologia de objetos.

Capítulo III

Modelamento de uma Mídia

Resumo – O modelo de um Objeto Mídia deve seguir regras definidas para sua utilização pelo servidor Multimídia. Entre estas regras encontra-se a construção da biblioteca de programação que será utilizada pelos clientes Multimídia, a definição de tipos de dados a serem manipulados pela Mídia, tipos de tarefas passíveis de serem acionadas para realização de algum serviço Multimídia e eventos associados a acontecimentos de relevância referentes a Mídia. Além disso é necessário o registro em Servidores Multimídia para disponibilizar o seu uso.

III.1 Introdução

Este capítulo tem por objetivo utilizar os conceitos definidos anteriormente para mostrar como se extende o ambiente no sentido da definição de serviços relacionados a uma Mídia específica, através do modelamento da mesma em funções de processamento, de entrada e saída e de comunicação.

A definição de uma Mídia é algo muito difícil de se fazer, ainda mais se pensarmos que suas características devem ser genéricas o suficiente para se adaptar ao modelo proposto da arquitetura Cliente/Servidor. Sendo assim necessita-se da definição de um elemento que possa representar uma Mídia em uma dada circunstância, dependendo dos serviços relacionados que este elemento possa oferecer [20]. Em outras palavras, definiremos aqui o conceito de Objeto Mídia, que nada mais é que uma representação de uma Mídia em termos de definições de serviços e dos tipos de dados manipulados.

Para se adaptar ao modelo proposto um objeto Mídia deve conter os seguintes componentes:

- Tipos de dados manipulados
- Tipos de eventos associados
- Tipos de tarefas
  - Tarefas de Entrada/Saída
  - Tarefas de Processamento
  - Tarefas de Comunicação

A definição dos componentes acima é primordial e deve ser feita em um processo que é chamado de Projeto de Mídia. Esta é a primeira fase do modelamento da Mídia. A fase seguinte consiste na implementação das tarefas anteriormente definidas utilizando-se de facilidades e mecanismos de controle e interação com o Servidor Multimídia. Para isso será definido um objeto especial, chamado de Tarefa de Mídia (MediaTask). Este objeto não deve ser confundido com a representação interna ao servidor de uma Tarefa de Mídia, que foi anteriormente definida como objeto Tarefa. Finalmente, após definições e implementações a Mídia deve ser colocada em disposição para ser utilizada por clientes Multimídia. Para isso deve-se construir uma biblioteca que represente serviços de Mídia e que se utilize do protocolo Multimídia para acionar as tarefas anteriormente implementadas e registradas.

III.2 Projeto de Mídia

O projetista de um Objeto Mídia deve identificar os elementos componentes da Mídia em questão e qual o escopo de execução deste objeto [19]. Serviços que requeiram hardware específico devem ser implementados e registrados apenas onde exista disponibilidade física ou dispositivos que possibilitem sua execução. Os principais tipos de elementos a serem identificados são:

Elementos de Entrada/Saída
Entre os elementos de entrada e saída existem os tipos de dados associados a dispositivos de entrada e saída. Dispositivos de entrada utilizam-se de algum mecanismo de comunicação com o usuário, como por exemplo, teclado, vídeo, microfones, mesas de digitalização, cameras, impressoras, para adquirir ou exteriorizar dados. Cada um dos dispositivos existentes define um tipo de dados específico para representação de informações. Estes tipos de dados, pertinentes à Mídia a ser definida, representa um elemento de entrada e saída. Eventos associados à aquisição ou exteriorização, bem como situações de erro são possíveis definições de elementos interessantes. Além disso, necessita-se de um processo específico na aquisição ou exteriorização destes dados. São estes processos que se transformarão em tarefas de Mídia e também em elementos de entrada e saída.

Elementos de Processamento
Depois de adquiridos os dados referentes a uma Mídia pode-se realizar diversos tipos de transformações aos mesmos. Estas transformações podem originar outros tipos de dados ou manter o mesmo tipo, mas com uma diferente instância. Ocorre aqui o que chama-se de processamento dos dados. Caso existirem tipos de dados diferentes gerados pelo processamento, estes tipos de dados, bem como as tarefas correspondentes aos processos e tipos de eventos associados a estes acontecimentos serão cadastrados como elementos de processamento da Mídia. Cada tipo de Mídia conta com uma infinidade de tipos de processamento diferente. Conversões de tipos de dados para tarefas de entrada/saída ou comunicação também são consideradas como tarefas de processamento, ditas tarefas de conversão ou então de codificação e decodificação.

Elementos de Comunicação
Elementos de comunicação podem ser pensados também como elementos de entrada e saída, mas a transferência de informações neste caso é realizada entre dois sistemas de computação, ao passo que elementos de entrada/saída realizam esta transferência de dados

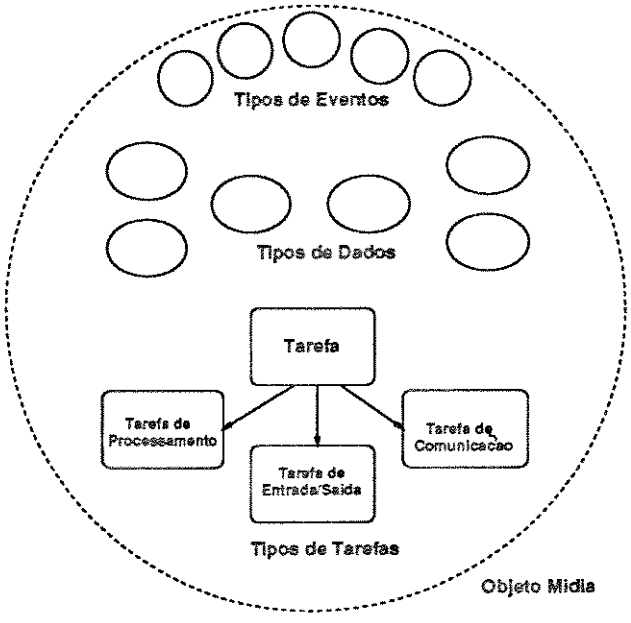

Figura III.1: Relação entre os Componentes de Mídia

entre um sistema de computação e um dispositivo externo, com intenção de comunicação com o mundo físico ou um usuário. Cada tipo de comunicação utilizado entre dois sistemas define os tipos de dados que devem ser transferidos e como é realizada esta transferência, o que chamamos de protocolos de comunicação. Cada Mídia tem associados vários protocolos de comunicação e tipos de dados necessários para a comunicação. Cada um destes tipos de dados e os processos responáveis polo controle da comunicação, que implementam o protocolo utilizado são possíveis definições de elementos de comunicação. Eventos associados aos processos, bem como erros que possam acontecer, são também elementos da Mídia.

Uma vez identificados os elementos de Mídia, os componentes são automaticamente identificados. A relação entre os diversos componentes pode ser vista na figura III.1.

A seguir definem-se pormenores e regras a serem seguidas nas definições dos componentes de Mídia.

III.2.1 Definição de Dados

Para a identificação dos tipos de dados componentes de um Objeto Mídia, o Projetista de Mídia deve seguir os seguintes passos:

- Identificar os tipos de dispositivos de entrada/saída associados à Mídia e os tipos de dados correspondentes a cada dispositivo.
- Identificar os protocolos de comunicação a serem aplicados à Mídia e os tipos de dados correspondentes a cada protocolo.
- Identificar todos os tipos de codificação e decodificação possíveis e transformações entre os tipos de dados já existentes visando a criação de tipos de dados intermediários.

Cada tipo de dado identificado deve então ser caracterizado para posterior registro no Servidor Multimídia. Esta caracterização se dá através do cálculo da quantidade de memória necessária para inicialização do que se chama de células de dados correspondentes ao tipo em questão.

Estas células de dados conterão a informação correspondente aos tipos de dados que serão utilizadas nas tarefas de Mídia. Além disso deve-se definir se o tipo de dado é estático ou dinâmico, ou seja, se a quantidade de informação a ser armazenada numa célula de dados correspondente é variável ou não.

Para tipos de dados estátivos a quantidade de informação existente é sempre a mesma, correspondente ao tamanho da célula definido anteriormente. Por outro lado, para tipos de dados dinâmicos a quantidade de informação pode variar com o tempo ou dependendo da situação onde é utilizada, e o tamanho da célula definido anteriormente corresponde ao máximo de informação possível de ser representada por células de dados. Finalmente, para cada tipo de dados definido deve ser associado um identificador para referência no ambiente.

III.2.2 Definição de Eventos

Para a identificação dos tipos de eventos componentes de um Objeto Mídia, o Projetista de Mídia deve seguir os seguintes passos:

- Identificar as ações de entrada/saída associados à Mídia e os acontecimentos e sinalizações possíveis de serem realizadas, bem como os tipos de erros que possam existir.
- Identificar as ações de comunicação e os acontecimentos e sinalizações possíveis de serem realizadas, bem como os tipos de erro que possam existir decorrentes de falhas de comunicação.

- Identificar os serviços de processamento e os acontecimentos e sinalizações possíveis de serem realizadas, bem como os tipos de erro que possam existir nos processos.

Cada tipo de evento identificado deve então ser caracterizado para posterior registro no Servidor Multimídia. Esta caracterização se dá através classificação do evento em uma das seguintes categorias:

- Erros fatais (Classe *FATAL*)
- Erros não fatais (Classe *ERROR*)
- Advertências (Classe *WARNING*)
- Eventos internos (Classe *INTERNAL*)
- Eventos de controle (Classe *CONTROL*)
- Eventos de Mídia (Classe *MEDIA*)

As classes correspondentes a erros e advertências, bem como eventos de Mídia são as mais comumentes utilizadas pelo Projetista de Mídia, sendo que os eventos internos e de controle são definidos no registro do Servidor Multimídia. Existe porém a possibilidade da definicão destas classes de eventos para uma Mídia, mas deve ser feita com cuidado, pois pode influenciar o comportamento do Servidor e o controle do mecanismo de execução dos serviços requisitados por clientes.

Erros fatais são aqueles ocasionados por falhas graves e que podem provocar situações irrecuperáveis. Erros não fatais são associados a falhas de serviços que podem ser recuperadas através de ações de controle ou de comando. Advertências são sinalizações de incoerências na realização de serviços ou de ações de controle que podem provocar erros no decorrer dos acontecimentos. Eventos internos são sinalizações de acontecimentos internos a uma Mídia ou ao Servidor Multimídia, que não devem ser exteriorizados do processo de origem. Eventos de controle são sinalizações de acontecimentos internos a uma Mídia, Cliente Multimídia ou Servidor Multimídia, que devem ser repassados a outro processo para a realização de ações de controle. Finalmente, eventos de Mídia são apenas sinalizações de acontecimentos úteis no decorrer de serviços associados a Tarefas de Mídia, que podem ser compartilhados com Servidores ou Clientes.

III.2.3 Definição de Tarefas

Para a identificação dos tipos de tarefas componentes de um Objeto Mídia, o Projetista de Mídia deve seguir os seguintes passos:

- Identificar os serviços necessários para adquirir ou exteriorizar os dados referentes a dispositivos de entrada e saída e os processos correspondentes a cada serviço.
- Identificar os serviços correspondentes a ações dos protocolos de comunicação associados à Mídia e os processos correspondentes a cada serviço.
- Identificar os serviços de processamento correspondentes a ações de transformação, codificação ou decodificação possíveis de serem aplicadas à Mídia e os processos correspondentes a cada serviço.

Cada tipo de tarefa identificada deve então ser caracterizado para posterior registro no Servidor Multimídia. Esta caracterização se dá através da definição de um estado Multimídia associado à Tarefa. Cada estado Multimídia tem os seguintes componentes:

- Identificador dos tipos de dados da entrada do Estado
- Identificador dos tipos de dados da saída do Estado
- Identificafor das Tarefas de Mídia associadas ao Estado
- Identificador do Serviço de Mídia correspondente ao Estado

Os tipos de dados registrados como entrada de um estado devem ser compatíveis com o tarefas de entrada/saída, processamento ou comunicação, dependendo do tipo de tarefa registrada para execução do serviço Multimídia relacionado. Como decorrência, os tipos de dados registrados como saída devem ser os mesmos possíveis da saída do tipo de tarefa registrada. Deve existir também um identificador unívoco para registro do estado no ambiente. A relação entre um estado e as Tarefas de Mídia será melhor detalhada nas seções seguintes.

III.3 Projeto de uma Tarefa de Mídia

Cada Tarefa de Mídia corresponde a um processo à parte que deve ser capaz de se comunicar com um Servidor Multimídia e de ser controlado por ele. Cada um destes processos deve seguir certas regras em sua implementação para que isso seja possível. Uma Tarefa de Mídia deve realizar os seguintes passos:

- Inicializar os canais de comunicação com o Servidor.
- Receber a célula de dados contendo as informações a serem processadas na Tarefa, quando for o caso.
- Requisitar ao Servidor a criação de uma célula de dados para receber a resposta do processamento, quando for o caso.
- Realizar o serviço Multimídia correspondente.
- Sinalizar o Servidor de acontecimentos de eventos registrados, inclusive Eventos de erro e de controle.
- Sinalizar término de ciclo no final do serviço.

Para realizar a comunicação de Tarefas de Mídia com o Servidor deve-se definir um novo objeto, chamado de Objeto Tarefa de Mídia (MediaTask) que apresenta facilidades e características comuns de controle, tais como receber e requisitar células de dados e receber e enviar eventos.

Este objeto Tarefa de Mídia oferece os seguintes serciços a um processo representante de uma Tarefa de Mídia:

Inicialização (*MediaTask::Initialize*)
Inicializa canais de comunicação com o Servidor. Este serviço deve ser obrigatoriamente chamado quando uma Tarefa de Mídia for criada.

Entrada de dados (*MediaTask::GetInput*)
Recebe informações da célula de dados definida como entrada da Tarefa de Mídia. Estas informações são correspondentes a um tipo de dados devidamente registrado como entrada da Tarefa. Esta chamada é pré-requisito do processamento do serviço Multimídia definido para a Tarefa, salvo o caso em que este serviço não necessita de uma entrada de dados.

Saída de dados (*MediaTask::SetOutput*)
Requisita ao Servidor a criação de uma célula de dados e preenche a mesma com dados provenientes de processamento executado pelo serviço Multimídia. Esta chamada é pré-requisito da transferência de informações entre tarefas, após o término do ciclo, salvo o caso em que este serviço não define uma saída de dados.

Envio de eventos (*MediaTask::SendEvent*)
Envia um evento Multimídia ao Servidor. Este evento deve ser propriamente registrado como um Evento de controle, de erro, de advertência ou definido pela Mídia. Esta chamada deve ser acionada sempre que algum acontecimento interno seja identificado com algum evento pré-definido.

Registro de manipuladores (*MediaTask::RegisterHandler*)
Registra um manipulador para eventos específicos de controle recebidos do Servidor. Quando algum acontecimento seja identificado com um evento de controle as tarefas interessadas naquele acontecimento recebem uma notificação do evento e acionam os manipuladores registrados.

Finalização (*MediaTask::Finalize*)
Sinaliza a destruição da Tarefa de Mídia e rompe os canais de comunicação com o Servidor.

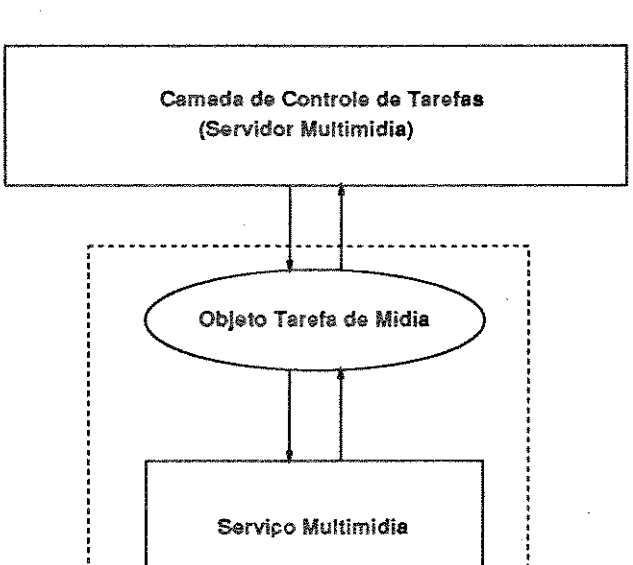

Figura III.2: Aspectos Internos de uma Tarefa de Mídia

A figura III.2 mostra um esquema da comunicação entre uma Tarefa de Mídia e o Servidor Multimídia utilizando-se de um objeto Tarefa de Mídia.

As Tarefas de Mídia podem ser classificadas quanto ao seu propósito, como vimos anteriormente na subdivisão entre Tarefas de entrada/saída, processamento ou comunicação, que serão comentadas nas seções seguintes. Outro tipo de classificação seria quanto ao fluxo de informações através de uma Tarefa. Existem aquelas que requerem um tipo de informação específico na entrada, ou aquelas flexíveis quanto a alguns tipos de informação aceitos. Existem ainda aquelas Tarefas que não necessitam de informações de entrada. Por outro lado, existem Tarefas que produzem apenas um determinado tipo de informação como saída, ou um tipo de informação específico, ou ainda aquelas que não produzem saída de dados.

Uma tarefa pode, então, com respeito à entrada ou saída de dados ser classificada em:

- simples
- composta
- inexistente

Diz-se que a Tarefa pode ser de entrada simples, composta ou inexistente e, igualmente, pode ser de saída simples, composta ou inexistente. A relação entre entre estes tipos de tarefas e os serviços executados do objeto Tarefa de Mídia é clara. Tarefas de entrada inexistente não necessitam executar o serviço *GetInput*. Quando a entrada é simples o serviço é acionado para um tipo de dado específico e quando a entrada é composta necessita-se especificar o tipo de dados requisitado no serviço. Igualmente para tipos de saídas simples, composta ou inexistente o serviço *SetOutput* pode ou não ser acionado.

III.3.1 Tarefas de Entrada/Saída

Tarefas de entrada/saída são aquelas que se relacionam com dispositivos de aquisição ou exteriorização de dados para o mundo externo, de uma forma, na maioria das vezes, passível de representação para um usuário. A entrada deste tipo de tarefas, geralmente ocasiona entrada inexistente e saída simples para o caso de Tarefas de entrada. Por outro lado a entrada é simples e a saída é inexistente para Tarefas de saída. Não se deve aqui confundir o propósito da Tarefa, se de entrada (aquisição de dados do mundo externo), ou de saída (exteriorização de dados para o mundo externo), com a classificação da Tarefa referente a tipos de entrada de dados ou saída de dados. Um exemplo de uma Tarefa de Entrada/Saída pode ser visto na figura III.3.

O exemplo apresentado refere-se a uma Tarefa de aquisição de quadros de vídeo de um dispositivo. O programa principal referente à Tarefa apenas define o tipo de dados associado com o registro da célula correspondente ao identificador `VIDEO_CODED`, inicializa um objeto *MediaTask* e uma variável para receber os dados. Depois disso a Tarefa entra em um ciclo infinito esperando um evento de ativação, previamente definido, que aciona o manipulador. Este manipulador realiza a aquisição de um quadro de vídeo do dispositivo e testa a ocorrência de erro. Em acontecendo alguma falha, o evento `VIDEO_ERROR` é sinalizado, caso contrário a saída da Tarefa é registrada com as informações adquiridas.

```
mmEvent getVideo(mmEvent event, void *, void *output)
{
  /* Recebe pacotes de video do dispositivo */
  if (read(videoDevice, output) == -1)
  {
    /* Sinaliza evento de erro */
    task -> SendEvent(VIDEO_ERROR);
  }
  else
  {
    /* Atualiza a saida da Tarefa */
    task -> SetOutput(VIDEO_CODED, output);
  }
}

main()
{
  /* Inicializa a Tarefa */
  task = new MediaTask();
  task -> Initialize();
  /* Registra manipulador para evento de transicao */
  task -> RegisterHandler(mmTRANSITION, getVideo);
  /* Processa eventos */
  task -> Loop();
  /* Desregistra manipulador para evento de transicao */
  task -> UnregisterHandler(mmTRANSITION, getVideo);
  /* Finaliza a Tarefa */
  task -> Finalize();
  delete task;
}
```

Figura III.3: Exemplo de Tarefa de Entrada/Saída

III.3.2 Tarefas de Processamento

Tarefas de processamento são aquelas que transformam tipos de dados ou realizam codificação e decodificação de dados. Este tipo de tarefas geralmente apresenta entradas e saídas dos tipos simples ou composta. Entradas e saídas inexistentes por definição não são relacionadas com Tarefas de processamento, visto que estas necessitam obrigatoriamente de pelo menos uma entrada e criam obrigatoriamente pelo menos uma saída. Um exemplo de Tarefa de Processamento pode ser visto na figura III.4.

O exemplo apresentado refere-se a uma Tarefa de decodificação de quadros de vídeo, de um formato `VIDEO_CODED` para outro formato, `VIDEO_DECODED`. O programa começa com a definição dos tipos de dados associados a estes formatos e com o registro do manipulador do evento de acionamento (previamente definido). Este manipulador recebe a célula de dados correspondente à entrada da Tarefa, realiza a decodificação dos dados e testa a ocorrência de erros neste processo. Em acontecendo alguma falha o evento `VIDEO_ERROR` é sinalizado, caso contrário a célula de dados de saída é requisitada e preenchida com o resultado da decodificação. Em conjunção com a decodificação, o evento `VIDEO_DECODING` foi sinalizado ao Servidor, que se encarrega de verificar se existem outros processos interessados neste tipo de evento.

III.3.3 Tarefas de Comunicação

Tarefas de comunicação são aquelas que realizam transferência de informações entre dois sistemas de computação. Esta transferência de informações sempre se realiza através de protocolos de comunicação, que são definições comuns aos pares comunicantes envolvidos. Cada tipo de Mídia pode possuir vários protocolos de comunicação, cada um dos quais pode definir diversas Tarefas de Comunicação. Este tipo de tarefa geralmente se caracteriza por apresentar uma grande diversidade de situações de controle. Existem tarefas com entrada simples e saída inexistente, no caso de envio de dados; outras com entrada inexistente e saída simples, no caso de recebimento de dados; outras ainda com entradas e saídas simples ou compostas ou entradas e saídas inexistentes, dependendo se for o caso de tarefas de controle e sinalização ou de entidades de comunicação independentes. Um exemplo de uma Tarefa de Comunicação pode ser visto na figura III.5.

O exemplo apresentado refere-se a uma Tarefa de envio de quadros de vídeo, de um formato `VIDEO_CODED` para um sistema de computação remoto. O programa começa com a definição do tipo de dados associado aos dados a serem enviados e registra um manipulador do evento de

```
mmEvent decodeVideo(mmEvent event, void * input,
                    void * output)
{
  /* Recebe os dados da entrada */
  task -> GetInput(VIDEO_CODED, input);
  /* Realiza a decodificacao dos pacotes de video */
  if (decode(input, output) == -1)
  {
    /* Sinaliza evento de erro */
    task -> SendEvent(VIDEO_ERROR);
  }
  else
  {
    /* Atualiza a saida da Tarefa */
    task -> SetOutput(VIDEO_DECODED, output);
    /* Sinaliza evento */
    task -> SendEvent(VIDEO_DECODING);
  }
}

main()
{
  /* Inicializa a Tarefa */
  task = new MediaTask();
  task -> Initialize();
  /* Registra manipulador para evento de transicao */
  task -> RegisterHandler(mmTRANSITION, decodeVideo);
  /* Processa eventos */
  task -> Loop();
  /* Desregistra manipulador para evento de transicao */
  task -> UnregisterHandler(mmTRANSITION, decodeVideo);
  /* Finaliza a Tarefa */
  task -> Finalize();
  delete task;
}
```

Figura III.4: Exemplo de Tarefa de Processamento

```
mmEvent sendVideo(mmEvent event, void * input, void *)
{
  /* Recebe os dados da entrada */
  task -> GetInput(VIDEO_DECODED, input);
  /* Envia os pacotes de video */
  if (send(input) == -1)
  {
    /* Sinaliza evento de erro */
    task -> SendEvent(VIDEO_ERROR);
  }
  else
  {
    /* Sinaliza evento */
    task -> SendEvent(VIDEO_SENDING);
  }
}

main()
{
  /* Inicializa a Tarefa */
  task = new MediaTask();
  task -> Initialize();
  /* Registra manipulador para evento de transicao */
  task -> RegisterHandler(mmTRANSITION, decodeVideo);
  /* Processa eventos */
  task -> Loop();
  /* Desregistra manipulador para evento de transicao */
  task -> UnregisterHandler(mmTRANSITION, decodeVideo);
  /* Finaliza a Tarefa */
  task -> Finalize();
  delete task;
}
```

Figura III.5: Exemplo de Tarefa de Comunicação

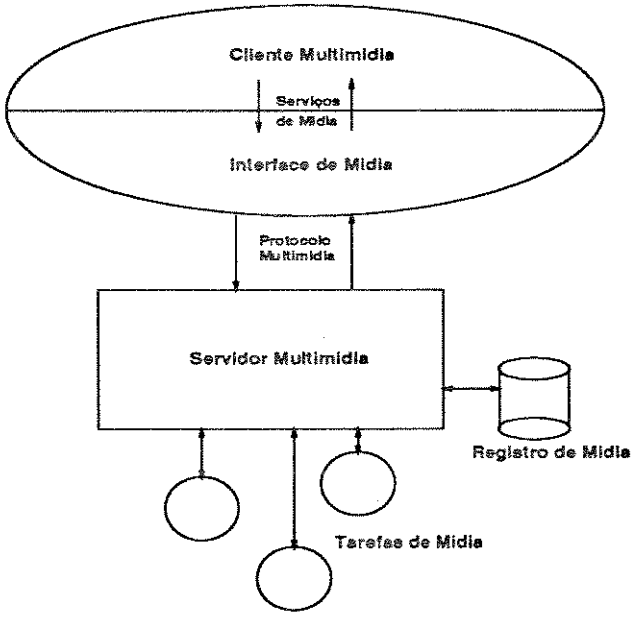

Figura III.6: Acionamento de um Serviço de Mídia

acionamento (previamente definido). Este manipulador recebe a célula de dados correspondente à entrada da Tarefa, realiza o envio dos dados e testa a ocorrência de erros neste processo. Em acontecendo alguma falha o evento `VIDEO_ERROR` é sinalizado, caso contrário o evento `VIDEO_SENDING` é sinalizado e o Servidor se encarrega de verificar se existem outros processos interessados neste tipo de evento.

III.4 Definição dos Serviços de Mídia

O passo final da definição de um Objeto Mídia é a construção dos serviços de Mídia associados ao objeto. Cada um dos serviços relacionados constituirá uma chamada possível de ser utilizada por clientes Multimídia. Estas chamadas utilizam-se do protocolo Multimídia e das Tarefas definidas para realizarem o serviço Multimídia. A figura III.6 mostra a relação entre uma chamada da interface de Mídia e os serviços correspondentes.

Os serviços a serem definidos podem ou não serem encapsulados em um objeto. Se a tecnologia de objetos for utilizada na contrução da biblioteca de Mídia diz-se que o cliente utiliza-se de um objeto derivado de uma Mídia.

Se a biblioteca não for orientada a objetos, diz-se que o cliente utiliza-se de serviços de uma Mídia. Em todos os casos, devem existir sempre os seguintes componentes relacionados a uma Mídia na interface oferecida:

- Inicialização da Mídia
- Realização de um Serviço Multimídia
- Registro de Manipuladores de Eventos Multimídia

As chamadas de protocolo referentes a abertura de Sessões devem fazer parte de uma biblioteca padrão, pois não se referem a aspectos específicos de uma Mídia. Na função de inicialização da Mídia a chamada mmQuery deve ser acionada para validar o registro da Mídia referente à biblioteca. Acionamento de serviços Multimídia causam chamadas referentes a criação de estados, ligação de estados, registro de eventos e o correspondente controle das mesmas para que o serviço seja concluído (*mmCreate, mmLink, mmRegister*). Deve ainda existir a possibilidade do cliente registrar manipuladores próprios para a ocasião do tratamento particular de algum evento de interesse (*mmRegister*).

III.5 Comentários

O modelamento de uma Mídia como uma coleção de tarefas e registros, onde deve-se especificar exatamente os tipos de dados manipulados, eventos que possam ocorrer e os estados associados aos serviços existentes, torna o mecanismo de controle de Mídia por parte de um Servidor Multimídia flexível e eficiente. Com a definição de bibliotecas de Mídia que façam a correspondência de serviços Multimídia em chamadas do Protocolo Multimídia a programação dos clientes Multimídia torna-se mais coerente e fácil, pois a tarefa de criação e controle de objetos Multimídia internamente ao Servidor fica totalmente escondida. O ambiente de programação fica então completo com a definição dos serviços de Mídia.

Capítulo IV

Desenvolvimento do Ambiente Multimídia

Resumo – A implementação de um Servidor Multimídia que proporcione serviços verdadeiramente distribuídos em Rede deve seguir algumas regras básicas que permitem a flexibilidade de criação de novas Mídias ou novos Serviços dinamicamente. O ambiente de desenvolvimento e também alguns exemplos práticos de extensibilidade serão mostrados através do projeto de uma Interface de Mídia simples.

IV.1 Introdução

Este capítulo tem por objetivo apresentar o ambiente de desenvolvimento do sistema e apresentar resultados provenientes de sua implementação. Além disso, um exemplo relativo á sua utilização é apresentado através de diretivas para a confecção de uma interface de Mídia e de seus respectivos métodos, que representam Tarefas. Em linhas gerais este esquema demonstra a facilidade e a flexibilidade da utilização do ambiente.

A arquitetura proposta é flexível o suficiente para permitir configurações dinâmicas em cada Servidor com respeito aos serviços oferecidos. Para tanto o próprio Servidor deve prover elementos que possibilitem esta flexibilidade e sua implementação deve seguir também esta linha. O ambiente deve ser tal que permita facilmente mudanças dinâmicas de configuração e a adição ou subtração de recursos de maneira simples e concisa.

Como ambiente inicial de implementação foi escolhido um Sistema Operacional derivado do padrão POSIX [9] e a linguagem C++ [7]. O sistema operacional é altamente difundido nos meios acadêmicos e industriais e bastante flexivel em tarefas de desenvolvimento de sistemas, tendo inclusive facilidades na área de redes distribuídas [10] sendo estes os motivos de sua escolha. A linguagem C++ permite a programação orientada a objetos [15] que simplifica enormemente a definição e manutenção de grandes sistemas e torna mais inteligível o código final, além de possibilitar o uso de diversas técnicas avançdas de codificação [17]. Além disso foram necessárias várias técnicas de análise e projeto estruturado [2] [3] e de programação concorrente para a implememtação do sistema [5].

Na seção seguinte serão descritos fatores essenciais no processo de implementação deste Sistema relacionados a sua otimização e organização.

Finalmente serão apresentadas as diretivas para confecção de uma Interface de Mídia e sobre os seus serviços que vão se traduzir em métodos dos objetos da interface. Um exemplo destas definições é mostrado no projeto da Mídia Texto.

IV.2 Ambiente de Desenvolvimento

Os componentes essenciais do ambiente de desenvolvimento são definidos baseando-se na estrutura interna do sistema desenvolvido. Com referência ao Servidor e o suporte aos Clientes e Tarefas podem ser identificatos os seguintes componentes:

Objetos genéricos
Corresponde aos objetos genéricos utilizados na estrutura do sistema. Compoẽ-se basicamente de elementos básicos da árvore de objetos e objetos auxiliares, tais como listas e iteradores.

Máquina de Estados
Corresponde aos objetos componentes da estrutura da máquina de estados implementada internamente ao Servidor Multimídia. Esta máquina de estados compõe-se de objetos simples e compostos e de listas de objetos. Entidades, tipos de dados e ações são os seus principais representantes.

Gerentes de Camadas
Corresponde aos objetos componentes da estrutura de camadas implementada internamente ao Servidor Multimídia. Esta estrutura corresponde a objetos com caracteísticas de gerenciamento e de controle dos objetos componentes da máquina de estados ou de eventos gerados pelo ambiente.

Protocolo
Corresponde aos objetos componentes da Interface de Programação do ambiente, que acessa diretamente chamadas de protocolo na comunicação com o Servidor e controla eventos e transições de estados, assim como o fluxo de chamadas de Serviços Multimídia.

Servidor
Corresponde aos objetos e definições globais do Servidor Multimídia. Apenas faz papel de iniciador e controlador dos demais componentes do Servidor.

Cliente
Corresponde aos objetos e definições globais do Cliente Multimídia. Apenas faz papel de iniciador e controlador dos demais componentes do Cliente e efetivamente define a sequencia de chamadas Multimídia.

Tarefas
Corresponde aos Objetos e definições globais de Tarefas Multimídia. Este componente engloba vários subcomponentes, cada um correspondendo a uma determinada Tarefa Multimídia.

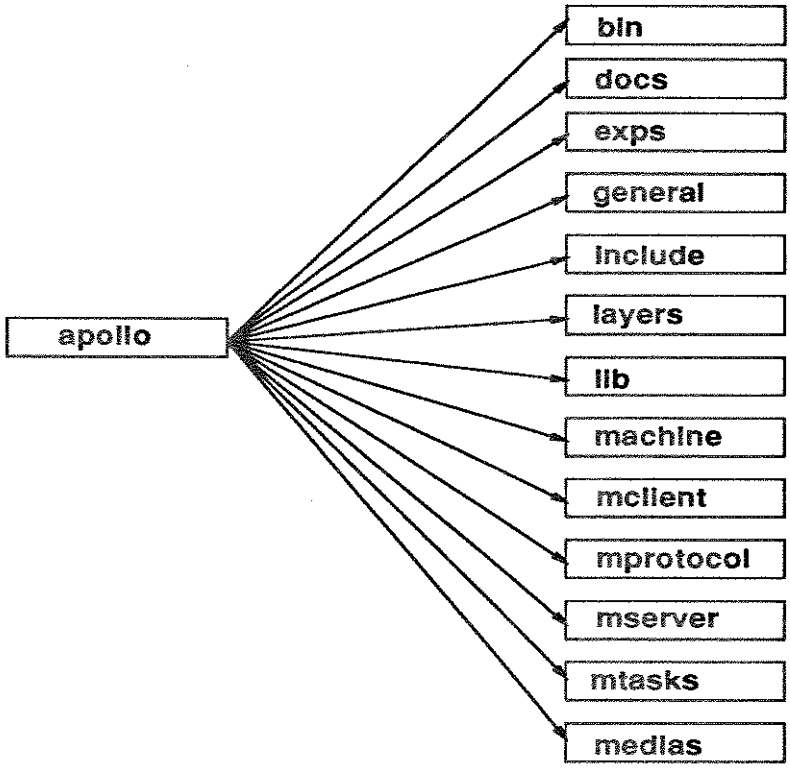

Figura IV.1: Sistema de Arquivos

Cada componente tem associado uma área correspondente no Sistema de Arquivos final. Alguns dados estatísticos são interessantes de serem relatados, tais como numero de arquivos gerados na implementação do ambiente, quantidade de linhas de código trabalhadas, facilidade de uso do ambiente, entre outros. Veja o apêndice para detalhes relacionados à implementação do sistema.

A figura IV.1 mostra o Sistema de Arquivos do ambiente, sendo que os componentes descritos acima encontram-se espalhados neste ambiente de desenvolvimento, do mesmo modo que outros arquivos auxiliares, tais como bibliotecas de ligação (*lib*), código executável (*bin*), entre outros.

Com os objetos implementados pelo ambiente os processos correspondentes ao Servidor Multimídia e aos Clientes Multimídia são gerados. Os clientes necessitam apenas do componente de comunicação, ou seja, do Protocolo Multimídia, para se comunicarem com o Servidor. Neste

componente também são definidos objetos básicos utilizados na construção de outros Objetos Mídia. Entre os objetos básicos encontram-se o objeto *MClient*, que controla o fluxo de mensagens com o Servidor e objetos correspondentes a Handlers de Transições e Exceções. Existem também objetos de interface correspondentes a cada Mídia criada. Estes objetos utilizam-se de referências ao objeto *MClient* para criar criar instâncias de estados específicos e facilitar a programação por parte do usuário. Estes objetos adicionais estão localizados no componente de Mídias e procuram extender, por assim dizer, os serviços implementados pelos objetos básicos que estes referenciam.

IV.2.1 Interface de Mídia

A definição da Interface de uma Mídia deve seguir certos guias. As regras principais que devem ser aplicadas a construção de novos tipos de Mídia são descritas a seguir:

- Novas interfaces de Mídia devem ser encapsuladas em um objeto correspondente
- Novos objetos de interface de Mídia devem referenciar serviços providos pelos objetos básicos do Protocolo
- Definições correspondentes a Mídias devem se limitar ao componente de Mídias
- Clientes que desejem se utilizar das definições da nova Mídia devem incluir as definições de interfaces da mesma, bem como utilizar o componente de Mídias

A figura IV.2 mostra a relação entre os serviços providos em uma Mídia e os serviços básicos providos ao Cliente a nível de interface. Estes serviços evidenciam-se na utilização das bibliotecas de ligação genérica (*libMproto*) e específica da Mídia utilizada (*libMedias*).

Observa-se que os processos Clientes devem acessar os serviços básicos e os serviços específicos de cada Mídia. Os serviços básicos servem para controle global do fluxo de execução, construção e manutenção das máquinas de estados e para registro de manipuladores de transições e exceções. Os serviços específicos das Mídias referenciam tarefas de processamento, comunicação e entrada/saída de dados.

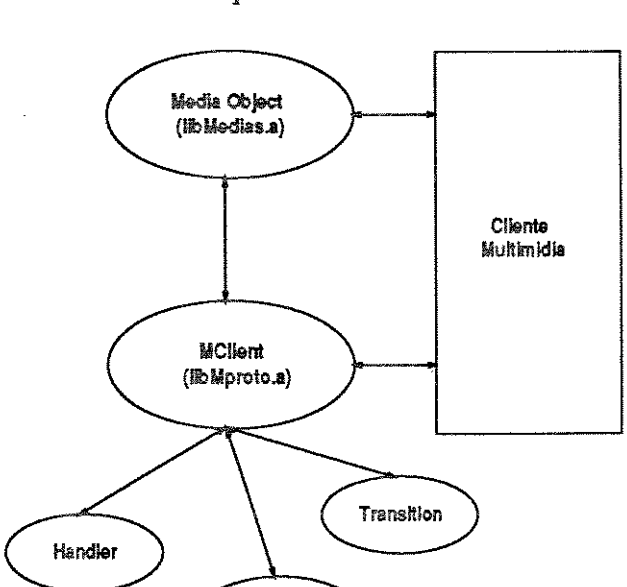

Figura IV.2: Interface de uma Mídia

Exemplificando melhor este processo, tomemos por exemplo a Mídia Texto, onde precisamos criar estados para ler um arquivo, escrever resultados em um arquivo, processar texto, etc. Para isso utilizamos as chamadas do protocolo *mmCreate* e *mmLink*. A função da interface de Mídia, neste caso, é a de facilitar estes serviços comuns que são necessários em praticamente todas os clientes. Para tanto, define-se na interface funções especiais que realizem estas funções, como por exemplo, métodos de um objeto tais como *Text::Read* para realizar uma leitura de arquivo e *Text::Write* para escrever no arquivo. É claro que estes são exemplos muito simples que podem ser implementados com apenas poucas chamadas do Protocolo Multimídia, mas existirão outros tipos de serviços comuns muito mais complexos, envolvendo mesmo pequenas máquinas de estado que podem ser encapsuladas em métodos do objeto da interface.

IV.3 Estudo de Caso – Mídia Texto

Neste estudo de caso definiremos uma Mídia simples, referente a serviços de manipulação de Texto, com os serviços para leitura de Texto de um arquivo e para escrita de Texto no console. A seguir desenvolveremos um exemplo simples de cliente que se utiliza destes dois serviços para ler um arquivo do sistema de arquivos e escrever seu conteúdo na tela. Para isso o cliente se comunica com o servidor Multimídia e pede a criação de uma Mídia Texto, através de sua interface apropriada, e também a criação de estados de Leitura (*Read*) e Escrita (*Write*). A seguir estes estados são conectados em uma máquina de estados simples e ligados. A máquina de estados é acionada e ocasiona a leitura de um arquivo específico por uma Tarefa Multimídia, a passagem dos dados através do Protocolo Multimídia, e a escrita destes dados no console por intermédio de outra Tarefa Multimídia. Todo este processo é controlado pelo Servidor Multimídia. A utilização das interfaces para acesso dos serviços de Texto fica então bastante simplificada.

A seguir mostraremos um exemplo de um cliente Multimídia que se utiliza dos serviços de Texto descritos. Observe como a utilização é simples, através do acionamento dos métodos dos objetos da interface. A figura IV.3 ilustra o código e a figura IV.4 ilustra o formato de um manipulador de eventos registrado pelo cliente.

Após a execução deste exemplo, observa-se que o Servidor Multimídia abre dois novos processos: *mmTextRead*, que corresponde ao serviço de leitura e *mmTextWrite*, que corresponde ao serviço de escrita de Texto. Estes processos fazem parte da máquina de estados associada ao cliente. Assim que esta máquina de estados é ativada a sequência de eventos resultante pode ser acompanhada na figura IV.5.

A saída de dados associada às atividades das Tarefas resume-se a listagem do conteúdo de um determinado arquivo no console.

```
main(int argc, char ** argv)
{
  MClient    * client;
  Text       * text;
  mmSession    session;
  mmState      state1, state2;
  mmServerID   node;

  /* Identifica o endereco do servidor */
  if (argc > 1) node = argv[1];
  else node = "9.179.3.25";

  /* Inicializacao do cliente e da maquina de estados */
  client  = new MClient();
  session = client->Connect(node);
  text = new Text(client, session);
  state1 = text->Read();
  state2 = text->Write();
  client->Link(session, state1, state2, mmTEXT_READY);

  /* Registro de manipuladores */
  client->Register(session, mmTEXT_READY, textHandler);

  /* Acionamento e manutencao da maquina de estados */
  client->Start(session);
  client->Loop();
  client->Stop(session);

  /* Finalizacao */
  client->Unregister(session, mmTEXT_READY, textHandler);
  client->Unlink(session, state1, state2, mmTEXT_READY);
  client->Destroy(session, state1);
  client->Destroy(session, state2);
  client->Disconnect(session);
  delete (text);
  delete (client);
}
```

Figura IV.3: Exemplo de Cliente Multimídia

```
void textHandler(mmEvent event)
{
  /* Apenas sinaliza o recebimento do evento */
  fprintf(stderr,
          "textHandler activation - mmEvent = %u\n",
          event);
  fflush(stderr);
}
```

Figura IV.4: Cliente Multimídia – Manipulador de eventos

IV.4 Comentários

Como no exemplo da Mídia Texto descrita nesta seção este ambiente permite grande flexibilidade na definição de serviços distribuídos Multimídia. Existem certas interfaces de programação que devem ser respeitadas, através da utilização de objetos e da arquitetura cliente-servidor, mas ainda assim os clientes provenientes desta arquitetura estão muito mais sujeitos a uma maior maleabilidade e extensibilidade no que se diz respeito a reutilização e redefinição de serviços de Mídia, pois isso é feito a nível de Tarefas Multimídia. O controle também é feito de forma tolerante a falhas, pois tudo é baseado em um controle distribuído, realizado pelo servidor. O único ponto fraco é o próprio processo do servidor, que pode ser resguardado utilizando-se esquemas de replicação num futuro próximo. Novos tipos de Mídia podem ser facilmente definidas e mescladas num ambiente totalmente heterogêneo, sendo que as definições de serviços e suas interfaces são o único elo de ligação dos clientes com o ambiente.

Os resultados obtidos com o exemplo da Mídia Texto também foram satisfatórios, pois provou-se a grande facilidade da definição de novos serviços e a grande facilidade na utilização dos mesmos por parte dos clientes. Os resultados referentes a performance também foram satisfatórios, levando-se em conta que os atrasos referentes a processamento e mensagens cliente-servidor-tarefas são mínimos comparados com tempos de processamento de arquivos externos e processamentos multimídia, por exemplo.

```
Relato de Atividades do Ambiente:
=================================

mmCONNECT:    retorno: session = 24991
mmQUERY:      session          = 24991
              mediaID          = Text
mmQUERY:      session          = 24991
              retorno: media   = 1
mmCREATE:     session          = 24991
              media            = 1
              stateID          = mmTextRead
              retorno: state   = 1
mmCREATE:     session          = 24991
              media            = 1
              stateID          = mmTextWrite
              retorno: state   = 2
mmLINK:       session          = 24991
              states           = 1 to 2
              mask             = 1001
mmSTART:      session          = 24991
              state            = 1

... Atividade das Tarefas ...

mmSTART:      session          = 24991
mmUNLINK:     session          = 24991
              states           = 1 to 2
              mask             = 1001
mmDESTROY:    session          = 24991
              media            = 1
              state            = 1
mmDESTROY:    session          = 24991
              media            = 1
              state            = 2
mmDISCONNECT: session          = 24991
```

Figura IV.5: Teste do Ambiente Multimídia

Conclusão

O trabalho apresentado descreveu uma nova arquitetura de processamento Multimídia distribuída e seus componentes, clientes, servidor e tarefas. Um ambiente para definição e registro de processos Multimídia foi apresentado, tomando-se por base as máquinas e seus respectivos servidores, que são os pontos-chave de controle do ambiente. Além disso foram apresentados resultados experimentais na definição de alguns serviços simples relacionados com a Mídia Texto e sua utilização em um cliente distribuído. Com isso mostrou-se que o sistema supre características desejáveis em um sistema verdadeiramente distribuído, tais como flexibilidade, portabilidade (necessário devido a heterogeneidade das redes de computação), cooperatividade, possibilidade de divisão de resursos e carregamento, entre outras. Além disso mostrou-se a grande simplicidade e facilidade de programação atingidas.

Diversos melhoramentos e expansões podem ser feitas baseando-se neste ambiente. Algumas são citadas na lista abaixo e recomenda-se fortemente que sejam realizadas a fim de viabilizar a disseminação do sistema:

- Utilização de novas tecnologias na infra-estrutura do ambiente, tais como Threads, ORBs (Object Request Brokers) [16] e Servidores concorrentes [11].
- Implementação de Serviços das principais Mídias, tais como Voz, Audio, Texto, Gráficos, Imagem, Vídeo, Telefonia e FAX.
- Implementação de Ferramentas de Programação e Definição voltadas para o ambiente, tais como compiladores, utilitários gráficos para projeto e testes.
- Conformação com padrões de sistemas abertos e de Multimídia que deverão surgir em breve.

Tomando por base este trabalho como o núcleo de um ambiente completo de execução e programação Multimídia e realizando as extensões necessárias podemos modificar os conceitos existentes referentes ao processamento de informações na atualidade. A evolução dos sistemas computacionais faz com que o processamento distribuído Multimídia seja um fator importante na vida das pessoas e este sistema procura fornecer meios para tais realizações.

Apêndice A

Informações Gerais

Resumo – Neste apêndice serão apresentados os arquivos e resultados obtidos da implementação do ambiente distribuído Multimídia, assim como fatores numéricos que evidenciam a extensão do projeto e sua funcionalidade.

Arquivos do Ambiente

Os principais arquivos no desenvolvimento do ambiente são apresentados em tabelas a seguir, classificados pelo componente que representam e correspondem a arquivos no formato estabelecido pela linguagem C++.

A tabela A.1 mostra os arquivos correspondentes do componente genérico (arquivos de suporte) no ambiente, tais como listas e referências globais.

A tabela A.2 mostra os arquivos correspondentes do componente do Servidor Multimídia no ambiente, com suas várias camadas evidenciadas em objetos.

A tabela A.3 mostra os arquivos correspondentes do protocolo Multimídia que auxiliam na comunicação entre clientes, servidores e tarefas.

A tabela A.4 mostra os arquivos correspondentes da máquina de estados implementada pelo ambiente que é responsável pelo controle e mapeamento interno dos objetos Multimídia.

A tabela A.5 mostra os arquivos correspondentes de uma mídia típica, com serviços de processamento de texto simples e sua respectiva interface.

Resultados

Os principais resultados que representam o esforço de de desenvolvimento do sistema são resumidos a seguir.

- 146 arquivos implementados no total
- 22026 linhas de código C++ inteiramente operacionais
- 53 objetos implementados no ambiente
- grande facilidade de inclusão de novos objetos
- facilidade de utilização do ambiente
- interoperabilidade entre sistemas

Serviços Genéricos	
	apollo/include/ObjectID.h
	apollo/include/Events.h
	apollo/include/Globals.h
	apollo/include/Protocol.h
	apollo/include/Primitives.h
apollo/general/Object.C	apollo/general/Object.h
apollo/general/List.C	apollo/general/List.h
apollo/general/ListIterator.C	apollo/general/ListIterator.h

Tabela A.1: Arquivos Genéricos

Servidor Multimídia	
apollo/mserver/MServer.C	
apollo/layers/EventManager.C	apollo/layers/EventManager.h
apollo/layers/Manager.C	apollo/layers/Manager.h
apollo/layers/ObjectManager.C	apollo/layers/ObjectManager.h
apollo/layers/RegistrationManager.C	apollo/layers/RegistrationManager.h
apollo/layers/TaskManager.C	apollo/layers/TaskManager.h
apollo/layers/SessionManager.C	apollo/layers/SessionManager.h
apollo/layers/MediaManager.C	apollo/layers/MediaManager.h

Tabela A.2: Arquivos do Componente Servidor

Protocolo Multimídia	
apollo/mprotocol/MTask.C	apollo/mprotocol/MTask.h
apollo/mprotocol/MClient.C	apollo/mprotocol/MClient.h
apollo/mprotocol/Connection.C	apollo/mprotocol/Connection.h
apollo/mprotocol/Handler.C	apollo/mprotocol/Handler.h
apollo/mprotocol/TransitionHandler.C	apollo/mprotocol/TransitionHandler.h
apollo/mprotocol/ConnectionList.C	apollo/mprotocol/ConnectionList.h
apollo/mprotocol/HandlerList.C	apollo/mprotocol/HandlerList.h

Tabela A.3: Arquivos do Protocolo Multimídia

Máquina de Estados	
apollo/machine/Action.C	apollo/machine/Action.h
apollo/machine/Transition.C	apollo/machine/Transition.h
apollo/machine/Exception.C	apollo/machine/Exception.h
apollo/machine/Signal.C	apollo/machine/Signal.h
apollo/machine/Information.C	apollo/machine/Information.h
apollo/machine/Event.C	apollo/machine/Event.h
apollo/machine/Data.C	apollo/machine/Data.h
apollo/machine/Register.C	apollo/machine/Register.h
apollo/machine/MediaRegister.C	apollo/machine/MediaRegister.h
apollo/machine/DataRegister.C	apollo/machine/DataRegister.h
apollo/machine/StateRegister.C	apollo/machine/StateRegister.h
apollo/machine/Entity.C	apollo/machine/Entity.h
apollo/machine/Session.C	apollo/machine/Session.h
apollo/machine/ClientSession.C	apollo/machine/ClientSession.h
apollo/machine/TaskSession.C	apollo/machine/TaskSession.h
apollo/machine/Media.C	apollo/machine/Media.h
apollo/machine/State.C	apollo/machine/State.h
apollo/machine/Process.C	apollo/machine/Process.h
apollo/machine/Client.C	apollo/machine/Client.h
apollo/machine/Task.C	apollo/machine/Task.h
apollo/machine/ActionList.C	apollo/machine/ActionList.h
apollo/machine/TransitionList.C	apollo/machine/TransitionList.h
apollo/machine/ExceptionList.C	apollo/machine/ExceptionList.h
apollo/machine/SignalList.C	apollo/machine/SignalList.h
apollo/machine/InformationList.C	apollo/machine/InformationList.h
apollo/machine/EventList.C	apollo/machine/EventList.h
apollo/machine/DataList.C	apollo/machine/DataList.h
apollo/machine/RegisterList.C	apollo/machine/RegisterList.h
apollo/machine/EntityList.C	apollo/machine/EntityList.h
apollo/machine/SessionList.C	apollo/machine/SessionList.h
apollo/machine/MediaList.C	apollo/machine/MediaList.h
apollo/machine/StateList.C	apollo/machine/StateList.h
apollo/machine/ProcessList.C	apollo/machine/ProcessList.h
apollo/machine/ClientList.C	apollo/machine/ClientList.h
apollo/machine/TaskList.C	apollo/machine/TaskList.h

Tabela A.4: Arquivos da Máquina de Estados

Tarefas de Mídia	
apollo/mtasks/textRead/TextRead.C apollo/mtasks/textWrite/TextWrite.C	
Interface de Mídia	
apollo/medias/Text.C	apollo/medias/Text.h

Tabela A.5: Arquivos da Mídia Texto

Bibliografia

[1] C.L.Liu, J.W.Layland,
"Scheduling Algorithms for Multiprogramming in a Hard–Real–Time Environment",
Journal of the Association for Computing Machinery, pp.46-61,
January 1973.

[2] T.DeMarco,
"Structured Analysis and System Specification",
Yourdon Press, 1978.

[3] E.Yourdon and L.L.Constantine,
"Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design",
Prentice-Hall, 1979.

[4] B.W.Lampson, M.Paul and H.J.Siegert,
"Distributed Systems Architecture and Implementation",
Springer-Verlag, 1981.

[5] M.Ben-Ari,
"Principles of Concurrent Programming",
Prentice-Hall, 1982.

[6] H.Gomaa,
"Software Development of Real–Time Systems",
Communications of the ACM,
July 1986.

[7] B.Stroustrup
"The C++ Programming Language",
Addinson–Wesley, 1986.

[8] G.F.Coulouris, J.Dollimore,
"Distributed Systems Concepts and Design",
Addison-Wesley, 1988.

[9] Institute of Electronic and Electrical Engineers,
"IEEE Standard 1003.1-1990 – Portable Operating System for Computer Environment",
IEEE, 1990.

[10] W.R.Stevens,
"UNIX Network Programming",
Prentice-Hall, 1990.

[11] J.E.Faust, H.M.Levy,
"The Performance of an Object–Oriented Threads Package",
OOPSLA'90 Proceedings - ACM Press, pp.278-288,
October 1990.

[12] C.Nicolaou,
"An Architecture for Real–Time Multimedia Communication Systems",
IEEE Journal on Selected Areas in Communications, pp.391-400,
April 1990.

[13] R.Steinmetz,
"Synchronization Properties in Multimedia Systems",
IEEE Journal on Selected Areas in Communications, pp.401-412,
April 1990.

[14] D.Ferrari,
"Client Requirements for Real–Time Communication Services",
IEEE Communications, pp.65-72,
November 1990.

[15] G.Booch,
"Object Oriented Design",
Benjamin/Cummings, 1991.

[16] Object Management Group,
"The Common Object Request Broker: Architecture and Specification",
OMG Document Number 91.12.1, Revision 1.1, 1991.

[17] J.O.Coplien,
"Advanced C++ Programming Styles and Idioms",
Addison-Wesley, 1992.

[18] N.Natarajan, G.M.Slawsky,
"A Framework Architecture for Multimedia Information Networks",
IEEE Communications, pp.97-104,
February 1992.

[19] K.H.Smith,
"Accessing Multimedia Network Services",
IEEE Communications, pp.72-80,
May 1992.

[20] P.V.Rangan, H.M.Vin, S.Ranathan,
"Designing an On-Demand Multimedia Service",
IEEE Communications, pp.56-64,
July 1992.

[21] J.Daniels, S.Cook,
"Strategies for Sharing Objects in Distributed Systems",
Journal of Object-Oriented Programming,
January 1993.

[22] M.Perazolo, E.Moschim,
"PROJECT APOLLO: Multimedia Distributed System",
XI Brazilian Telecomunications Symposium,
September 1993.

[23] M.Perazolo, E.Moschim,
"A Multimedia System Protocol Implementation",
International Telecommunication Symposium,
August 1994.

[24] M.Perazolo, E.Moschim,
"A Protocol Proposal for Distributed Multimedia Services",
Second International Workshop on Advanced Teleservices and High Speed Communication Architectures,
September 1994.

[25] M.Perazolo, E.Moschim,
"The Architecture of a Distributed Multimedia Server",
East-West International Conference on Multimedia, Hypermedia and Virtual Reality,
September 1994.

```
##############################################################################*
##############################################################################*
##############################################################################*
##############################################################################*



******************************************************************************
******************************************************************************


ITLE:                   apollo.ps

IME PRINTED:            Thu Dec 01 12:04:00 1994

IME QUEUED:             Thu Dec 01 12:03:59 1994

RINTED AT:              laser (4019) @ everest.sumare.ibm.com

UBMITTED BY:            map@everest.sumare.ibm.com

ELIVER TO:              =====> map@everest.sumare.ibm.com <=====

******************************************************************************
******************************************************************************


LAG VALUES:
=0, d=s, f=, j=+, u=1, A=1, B=gn, C=!, H=everest.sumare.ibm.com,
=1, P=ps1:laser, X=IBM-850


******************************************************************************
******************************************************************************
```