UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Sincronização de Fluxos de Dados em Aplicações Multimídia

Neil Paiva Tizzo

Orientador: **Prof. Dr. Maurício Ferreira Magalhães**

Dissertação de Mestrado

Este exemplar corresponde à redação final da tese defendida por Neil Paiva Tizzo e aprovada pela comissão Julgadora em 28/02/97

Orientador

Campinas - SP Brasil

1997

UNICAMP
BIBLIOTECA CENTRAL

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Sincronização de Fluxos de Dados em Aplicações Multimídia

Neil Paiva Tizzo

Orientador: **Prof. Dr. Maurício Ferreira Magalhães**

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Elétrica, Área de concentração: Automação Industrial

Campinas - SP Brasil

1997

UNIDADE BC
N.º CHAMADA:
UNICAMP
T545s
V. Ex
TOMBO BC 34084
PROC. 395/98
C [ ] D [X]
PREÇO R$ 11,00
DATA 30/05/98
N.º CPD

CM-00112430-5

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Tizzo, Neil Paiva
T545s Sincronização de fluxos de dados em aplicações multimídia / Neil Paiva Tizzo.--Campinas, SP: [s.n.], 1997.

Orientador: Maurício Ferreira Magalhães.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Sincronização. 2. Sistemas multimídia. 3. Fluxo de dados (Computação). I. Magalhães, Maurício Ferreira. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Não há como agradecer àquelas pessoas que nos deram a vida e insistem em afirmar o contrário.

Aos meus pais Bento e Zeneide
e à minha irmã Neila

Gostamos de pensar que somos fortes. E quando não somos capazes há sempre alguém que nos faz sentir que somos sim.

À minha noiva Evaine

Há várias coisas na vida que valem a pena lutar.
Uma delas é a amizade.
No momento não me recordo das outras.

"Assim como são as coisas são as criaturas"
Sílvio

"Nada impede"
Marcone

"Senti mal"
Douglas

"Eu acho"
Claudio

[À entonação vocal cesariana]
César

AGRADECIMENTOS

À CAPES pelo apoio financeiro.

Ao Prof. Dr. Maurício Ferreira Magalhães pela orientação e incentivo.

Ao Adilson, Afonso, Maezi e Alencar pela amizade e discussões técnicas.

Ao Bauer, Carmelita e família por me emprestarem a família de vez em quando.

Ao casal Celso e Neila e a Tatiana pela amizade e alegria.

Aos amigos da engenharia química, em especial Jotinha, Garcia, Ana Paula e Betina.

À Adriana e Kelen pela alegria e amizade.

Àqueles que passaram pela república: Claudio, Marcone, Douglas, Tetsu, César, Ronaldo, Capitão, Arturo, Chang e ao agregado Sílvio.

E, finalmente, à Buba por ter existido.

Porque o mais importante na vida é humildade e amor.

Reginaldo Rossi

SUMÁRIO

Sumário VII
Lista de Figuras X
Lista de Gráficos XI
Lista de Tabelas XII
Lista de Abreviações XIV
Resumo XVI
Abstract XVII

Capítulo 1 - INTRODUÇÃO 1

1.1 Objetivos 3
1.2 Motivação 4
1.3 Trabalhos Correlatos 5
1.4 Organização do Trabalho 8

Capítulo 2 - UMA ABORDAGEM SOBRE SISTEMAS MULTIMÍDIA E SINCRONIZAÇÃO 9

2.1 Introdução 9
2.2 O que é Multimídia 10
2.3 Aplicações dos Futuros Sistemas Multimídia 10
2.3.1 Automação de Escritório 11
2.3.2 Aplicações na Indústria de Serviços 11
2.3.2.1 Educação 11
2.3.2.2 Finanças 11
2.3.2.3 Saúde 12
2.3.3 Comércio 12
2.3.3.1 Editoração 12
2.3.3.2 Turismo 12
2.3.4 Aplicações Domésticas 13
2.3.5 Aplicações nas Ciências e Engenharia 13
2.3.6 Trabalho Cooperativo Suportado por Computador 14
2.3.6.1 Sistemas de Mensagens 14
2.3.6.2 Conferência por Computador 14
2.3.6.3 Salas de Encontro 14
2.3.6.4 Sistemas de Co-Autoria 14
2.4 Tipos de Mídias 15
2.4.1 Mídias Contínuas 15
2.5 Requisitos de Tempo-Real de Mídia Contínua para Processamento Distribuído 16
2.5.1 A QoS e a Sincronização 17

2.6 Sincronização 18
2.7 Técnicas de Sincronização 19
2.7.1 Sincronização Sintética e Viva 19
2.7.2 Sincronização Condicional, Serial e Paralela 20
2.7.3 Sincronização através de Carimbos de Tempo, de Canais, de Multiplexação e de *Buffers* 20
2.7.4 Utilização ou não de Relógio Global 22
2.7.5 Localização Física das Ações Corretivas 23
2.7.6 Métodos Utilizados para a Correção do Desalinhamento 23
2.8 Modelos de Classificação 23

Capítulo 3 - DESENVOLVENDO UMA PLATAFORMA DE SERVIÇOS DE SINCRONIZAÇÃO PARA APLICAÇÕES DISTRIBUÍDAS MULTIMÍDIA 26

3.1 Introdução 26
3.2 Orquestação (*Orchestration*) 27
3.3 IMA/PREMO/MSS 29
3.3.1 Interface Group 31
3.3.2 Interface Stream 31
3.3.2.1 A Interface ControledStream 33
3.3.2.2 A Interface SyncStream 33
3.4 ACME 35
3.4.1 Casamento da Taxa 37
3.5 Plataforma de Sincronização para Aplicações Distribuídas Multimídia 39

Capítulo 4 - O SERVIDOR DE SINCRONIZAÇÃO MULTIMÍDIA (SESIM) 43

4.1 Introdução 43
4.2 O *Hardware* Utilizado 44
4.3 A Placa *ActionMedia II* e o *Software* MMPM/2 44
4.4 Implementação 46
4.4.1 Algoritmos de Sincronização 49
4.4.1.1 Sincronismo com Fluxo Mestre 51
4.4.1.2 Sincronismo sem Fluxo Mestre 53
4.5 Resultados Obtidos 55
4.5.1 Sincronização sem Fluxo Mestre 56
4.5.2 Sincronização com Fluxo Mestre 63
4.5.3 FDDI com Tráfego e Ethernet 67

Capítulo 5 - CONCLUSÕES E TRABALHOS FUTUROS **70**

5.1 Conclusões e Trabalhos Futuros 70

REFERÊNCIAS BIBLIOGRÁFICAS **73**

Apêndice A - O SISTEMA OPERACIONAL OS/2

A.1 Introdução
A.2 Multitarefa no OS/2
A.3 Prioridades no OS/2

Apêndice B - MULTIMEDIA PRESENTATION MANAGER/2

B.1 Introdução
B.2 O Ambiente *Presentation Manager (PM)*
B.3 O *Software Multimedia Presentation Manager/2 (MMPM/2)*

LISTA DE FIGURAS

Figura 1.1	Apresentação de um *slide*.	7
Figura 1.2	Apresentação do *slide* em Rede de Petri.	7
Figura 2.1	Modelos de localização dos dados multimídia.	22
Figura 2.2	Classificação visual de alguns algoritmos de sincronização.	24
Figura 2.3	O modelo de referência de quatro camadas de Gerold e Ralf.	24
Figura 3.1	Arquitetura de camadas: Gerenciador de Sincronização, Orquestrador de Alto Nível (OAN) e Oquestrador de Baixo Nível (OBN).	27
Figura 3.2	Interação entre o OAN e o OBN.	28
Figura 3.3	Grafo do fluxo de dados multimídia do MSS.	30
Figura 3.4	Diagrama de herança de interface.	30
Figura 3.5	Hierarquia de classe da interface Stream.	32
Figura 3.6	Casamento de unidades entre fluxos de taxas diferentes.	34
Figura 3.7	Arquitetura do MSS sob o gerenciamento do OAN.	35
Figura 3.8	Os objetos que compõem o servidor ACME.	36
Figura 3.9	Diagrama de passo escada para dispositivo.	37
Figura 3.10	Representação gráfica do algoritmo skip()/pause().	38
Figura 3.11	Arquitetura em camadas da plataforma de sincronzação.	40
Figura 3.12	Arquitetura do SESIM mostrando as classes que o compõem.	42
Figura 4.1	Configuração fisica do SESIM.	45
Figura 4.2	Interface homem-máquina do SESIM.	46
Figura 4.3	Os processos, fluxos de dados e fluxos de controle do SESIM.	47
Figura 4.4	Ciclo de execução do Sincronizador.	48
Figura 4.5	Os parâmetros Skew_D e Skew_E como eixos ortogonais.	50
Figura 4.6	Fluxograma da tomada de decisão do Sincronizador com fluxo mestre.	52
Figura 4.7	Fluxograma da tomada de decisão do Sincronizador sem fluxo mestre.	54

LISTA DE GRÁFICOS

Gráfico 4.1	FDDI com pouco tráfego, sem fluxo mestre, sem atraso e sem sincronismo (tabela B1).	57
Gráfico 4.2	FDDI com pouco tráfego, sem mestre, sem atraso e com sincronismo (tabela B2).	57
Gráfico 4.3	FDDI com pouco tráfego, sem fluxo mestre, com atraso do áudio e sem sincronismo (tabela B3).	58
Gráfico 4.4	FDDI com pouco tráfego, sem fluxo mestre, com atraso do áudio, Skew_E=1 e Skew_D=700ms (tabela B5).	59
Gráfico 4.5	FDDI com pouco tráfego, sem fluxo mestre, com atraso do áudio, Skew_E=1 e Skew_D=200ms (tabela B4).	60
Gráfico 4.6	FDDI com pouco tráfego, sem fluxo mestre, com atraso do áudio, Skew_E=3 e Skew_D=5 000ms (tabela B6).	60
Gráfico 4.7	FDDI com pouco tráfego, sem fluxo mestre, com atraso do áudio, Skew_E=5 e Skew_D=200ms (tabela B8).	61
Gráfico 4.8	FDDI com pouco tráfego, sem fluxo mestre, com atraso de texto, Skew_E=1 e Skew_D=700ms (tabela B5).	61
Gráfico 4.9	FDDI com pouco tráfego, sem fluxo mestre, com atraso do texto, Skew_E=5 e Skew_D=200ms (tabela B8).	62
Gráfico 4.10	FDDI com pouco tráfego, sem fluxo mestre, com atraso do áudio e do texto, Skew_E=1 e Skew_D=700ms (tabela B5).	62
Gráfico 4.11	FDDI com pouco tráfego, sem fluxo mestre, com atraso do áudio e do texto, Skew_E=5 e Skew_D=700ms (tabela B7).	63
Gráfico 4.12	FDDI com pouco tráfego, com fluxo mestre, atraso do áudio, Skew_D=1 e Skew_D=200ms (tabela A4).	64
Gráfico 4.13	FDDI com pouco tráfego, com fluxo mestre, com atraso do texto, Skew_D=1 e Skew_E=200ms (tabela A4).	64
Gráfico 4.14	FDDI com pouco tráfego, com fluxo mestre, com atraso do áudio e do texto, Skew_E=1 e Skew_D=200ms (tabela A4).	65
Gráfico 4.15	FDDI com pouco tráfego, com fluxo mestre, com atraso do áudio e texto, Skew_E=5 e Skew_D=700ms (tabela A7).	66
Gráfico 4.16	FDDI com pouco tráfego, sem fluxo mestre, com atraso do áudio, Skew_E=1 e Skew_D=700ms (tabela B7).	66
Gráfico 4.17	FDDI com tráfego, com fluxo mestre, com atraso do áudio e do texto, Skew_E=5 e Skew_D=700ms (tabela C7).	67
Gráfico 4.18	Ethernet, com fluxo meste, com atraso do áudio e texto,	68

Skew_E=5 e Skew_D=700ms (tabela E7).

Gráfico 4.19 FDDI com tráfego, sem fluxo mestre, atraso do áudio, Skew_E=1 e Skew_D=200ms (tabela D4). 68

Gráfico 4.20 Ethernet, sem fluxo mestre, com atraso do áudio, Skew_E=1 e Skew_D=200ms (Tabela F4). 69

LISTA DE TABELAS

Tabela 4.1 Testes realizados. 56

LISTA DE ABREVIAÇÕES

ACME - A Continuous Media I/O Server and Its Synchronization Mechanism

API - Application Programming Interface

AVK - Audio Video Kernel

BER - Bit Error Rate

CORBA - Common Object Request Broker

CPU - Central Processing Unit

CSCW - Computer Supported Cooperative Work

CTR - Carimbos de Tempo Relativos

EA - Exibição do Áudio

ET - Exibição do Texto

HLO - High Level Orchestration Manager

IMA - Interactive Multimedia Association

ISO - International Stardard Organization

LLO - Low Level Orchestration Manager

LTS - Logical Time System

MCI - Media Control Interface

MMPM/2 - Multimedia Presentation Manager/2

MSS - Multimedia System Service

OAN - Orquestrador de Alto Nível

OBN - Orquestrador de Baixo Nível

OCPN - Object Composition Petri Net

ODP - Open Distributed Processing

PER - Packet Error Rate

PREMO - Presentation Enviroment for Multimedia Objetics

QoS - Quality of Service

RAM - Random Access Memory

SESIM - Servidor de Sincronização Multimídia

TA - Topo do Áudio

TCP/IP - Transport Control Protocol - Internet Protocol

TT - Topo do Texto

UM - Unidade de Mídia

RESUMO

Tizzo, N. P., Sincronização de Fluxos de Dados em Aplicações Multimídia. Campinas: DCA/FEE/UNICAMP, 1997. (Dissertação de Mestrado)

Em um futuro próximo, os computadores oferecerão aos usuários a possibilidade de manipularem mídias contínuas através de sistemas totalmente interativos. Para estes sistemas mídias como, por exemplo, áudio e vídeo são especialmente importantes e são chamadas desta forma, mídia contínua, devido à estreita relação com o tempo. Em computação, o mecanismo capaz de manter esta relação é chamado de Sincronizador. Ele é o agente que determina quando cada mídia dever ser capturada/reproduzida de forma a manter não só sua própria relação temporal, assim como a relação existente entre as várias mídias envolvidas em um sistema. Neste trabalho é apresentada uma implementação de um servidor de sincronização multimídia que manipula áudio e texto através de dois computadores ligados em rede. Quando integrado em um sistema multimídia, o sincronizador deve interagir com diversos outros agentes que são responsáveis, por exemplo, pela transmissão das mídias, pelo controle/monitoramento do progresso do fluxo e pela (re)negociação/controle/monitoramento da QoS. Dentro deste contexto, foi desenvolvida também uma plataforma de serviços multimídia com o objetivo de definir uma interface de comunicação e as funções de cada agente.

Palavras chaves: Sincronização, Sistemas Multimídia, Fluxo de Dados.

ABSTRACT

Tizzo, N. P., Sinchronization of Data Stream in Multimedia Application. Campinas: DCA/FEE/UNICAMP, 1997. (Masters)

In a near future, the computers will offer to users the possibility to manipulete continuous media through totally interacted systems. For these systems medias like, for example, audio and video are especially important and are called continuous media because of the closed relation with the time. In computing, the mechanism capable to maintain this relation is called Synchronizator. It is the agent that establishes when each media might be captured/reproduced in a way that maintain not only its own temporal relation, as the relation exists between anothers envolved medias in the system. In this work it is presented a implementation of a multimedia synchronization server that manipulates audio and text across two computers in a net. When the synchronization is integrated in a multimedia system, it should interact with many other agents that are responsable, for example, by the media transmission, by the control/monitoring of the stream progress and for the QoS (re)negotiation/control/monitoring. In this context, it was developed also a plataform of multimedia services with the goal to define a communication interface and the functions of each agent.

Keywords: Synchronization, Multimedia System, Data Stream.

Capítulo 1

INTRODUÇÃO

"Agora que o micro migrou para dentro das casas das pessoas, percebe-se que o nome dado à máquina talvez esteja errado. A prioridade não é mais computar, calcular, orçar, contar. É comunicar. Quem está em casa não quer computar. Quer ter acesso, ou fazer transações. Esse é o mistério da coisa."

James McGroddy, pesquisador da IBM nos EUA.

Há pouco tempo atrás os computadores eram capazes de "entender" somente textos e números. Hoje, estão muito mais poderosos. Trabalham com áudio, vídeo e já reunem em si mesmo vários aparelhos de comunicação: telefone, fax, rádio, TV, etc. Num futuro próximo as mídias também mudarão tentando se adaptar ao mundo moderno: duas horas de filme serão transmitidas em alguns segundos; os jornais em papel terão uma versão para computador com áudio e vídeo e será difícil distinguí-los dos telejornais atuais.

À integração destas diversas mídias como, por exemplo, texto, gráfico, áudio, vídeo, etc, feita através do computador, dá-se o nome de multimídia. Existem no

mercado vários aplicativos que manipulam[1] áudio e vídeo. Entretanto, a maioria é de uso somente local, em estações de trabalho *stand-alone*, que fornecem pouca ou nenhuma possibilidade de interferência do usuário. O desafio, hoje, é a representação digital de qualquer tipo de dado em sistemas que sejam distribuídos e que possam ser controlados pelo usuário, ou seja, o desenvolvimento de sistemas interativos e distribuídos de multimídia.

Em um ambiente distribuído será possível o acesso a base de dados multimídia remotas, ao uso de fontes de mídia localizadas em diferentes regiões, ao uso de vários computadores interligados em rede abrindo uma gama enorme de novas aplicações para a informática.

Mas para que as diversas mídias sejam suportadas por computador e sejam transmitidas através de suas redes, várias tecnologias precisam ser empregadas. Uma delas é a sincronização.

Em qualquer transmissão de dados via rede de computadores são inseridos atrasos que não chegam a causar problemas quando as mídias são independentes do tempo. Entretanto, quando atrasos geram problemas, uma cadência deve ser mantida de forma que haja continuidade na apresentação da mídia. Da mesma forma, deve ser mantida a relação temporal entre as várias mídias envolvidas em um mesmo sistema. A estas mídias como, por exemplo, áudio, vídeo e animação dá-se o nome de mídias contínuas.

Sincronização, em um sistema multimídia, é o termo referente à relação existente entre a mídia e o tempo. Mas é possível encontrar na literatura várias abordagens sobre o assunto: sincronização de conteúdo, de espaço [ver Capítulo 2] e temporal. A sincronização temporal define a relação de tempo existente entre as unidades de mídia contínua e pode ser classificada em sincronização paralela ou serial. A sincronização paralela ocorre entre dois ou mais fluxos, ao passo que a serial tenta restringir a variação de atraso de um único fluxo.

Uma unidade de mídia é a unidade elementar de uma mídia definida pelo sistema, podendo ser uma amostra de áudio, um quadro (*frame*) de vídeo, um gráfico, etc. Desta forma, o objetivo da sincronização temporal é fazer com que as unidades de um fluxo de mídia que foram capturadas por fontes independentes sejam reproduzidas com a mesma taxa de captura e que mantenham a mesma relação temporal existente entre ambas no momento da captura.

Entre o instante de captura de uma imagem por uma camêra de vídeo e o de sua apresentação em um monitor, várias taferas podem ter sido realizadas. A relação temporal entre os fluxos pode ter sido deturpada por uma ou mais destas ações, como por exemplo, a compactação/descompactação das unidades, a transmissão, o escalonamento de tarefas pelo sistema operacional, etc.

---

[1] Entende-se por manipulação, neste trabalho, captura, armazenamento, compactação, descompactação, apresentação e outras eventuais ações requeridas no tratamento digital de áudio, vídeo e outras mídias.

No caso em que a mídia não seja "viva"[1], seu armazenamento deve de alguma forma manter sua relação temporal, ou de forma implícita, ou utilizar-se de métodos que possibilitem a definição explícita desta relação. A partir de sua recuperação ela fica sujeita às mesmas ações descritas no parágrafo anterior que possam causar o atraso de suas unidades.

Assim sendo, a sincronização deve ser considerada como uma tarefa distribuída pelas várias camadas que compõem um sistema multimídia, incluindo o operacional, o de comunicação, o de base de dados e até as aplicações.

Em cada uma destas camadas, a tarefa a ser executa destina-se a alcançar objetivos distintos e específicos. Partindo-se do sistema operacional e indo-se em direção à aplicação, a sincronização vai deixando de se preocupar com eventos extremamentes minuciosos (granulosidade fina) para ocupar-se com fatores de gerenciamento de execeções, de definições de políticas e de qualidade de serviço. Neste trabalho, propõe-se uma plataforma de sincronização que oferece à aplicação serviços de sincronização, de monitoramento e controle do fluxo.

1.1 Objetivos

O objetivo deste projeto consiste no estudo e implementação de técnicas de sincronização paralela (entre dois ou mais fluxos) em aplicações distribuídas multimídia. Em geral, nestas aplicações, há a necessidade de que os fluxos de áudio, vídeo, imagens, dados, etc sejam sincronizados de modo que a apresentação das informações ocorra de forma íntegra e correta. Por exemplo, áudio e vídeo devem estar sincronizados de modo que o movimento labial da imagem de uma pessoa esteja de acordo com o áudio correspondente.

Dentro deste contexto, foi implementado um servidor de sincronização baseado na metodologia utilizada pelo ACME [Anderson91]. O servidor desenvolvido tem a capacidade de realizar a sicronização paralela entre unidades de mídias contínuas através da inserção de carimbos de tempo (*time-stamps*).

Entretanto, devido à complexidade de se desenvolver aplicações multimídia, é desejável que tarefas que vão além da sincronização estejam disponíveis em uma plataforma que ofereça serviços de suporte com o intuito de facilitar o desenvolvimento destas aplicações e que funcione como uma camada situada entre a aplicação e o sistema operacional. É neste sentido que se propõe uma plataforma baseada na arquitetura e nos serviços especificados em três trabalhos que definem métodos para a sincronização de unidades de mídias e para o gerenciamento do fluxo: Orquestração[2] [Blair92] [Campbell], IMA[3]/PREMO[4]/MSS[5] [PREMO] e ACME[6] [Anderson91].

---

[1]Defini-se como "mídia viva" (ou "dado vivo") toda aquela em que sua fonte captura eventos do meio externo convertendo-os em dados para o sistema na mesma taxa em que são requeridos para consumo. Os dados são capturados e apresentados sem passarem por uma fase de armazenamento. Exemplo deste tipo de mídia são as geradas pelo microfone e pela câmera de vídeo.

[2]Orquestração: do termo em inglês *Orchestration*

Entre os serviços oferecidos, pode-se citar como exemplo, o monitoramento do fluxo, o controle de seu progresso, a definição da qualidade de serviço e sua eventual renegociação a posteriori, a obtenção de recursos e a notificação de falhas/exceções. A utilização de qualquer uma destas facilidades pode ser feita individualmente sobre cada fluxo ou sobre um determinado conjunto de fluxos previamente agrupados.

Infelizmente, durante a implementação do servidor proposto empecilhos técnicos impediram que a mídia de vídeo fosse utilizada. O *software* em uso (MMPM/2) não foi capaz de retornar a imagem capturada por uma câmera e posteriormente digitalizada por uma placa *ActionMedia II*. A utilização de um outro *software*, AVK (*Audio Video Kernel*), foi tentado, mas, novamente, sem a obtenção de sucesso.

A solução encontrada foi a substituição do vídeo por texto. A sincronização foi realizada, então, com fluxos de caracteres e áudio. Do ponto de vista da validação, esta substituição em nada influencia os resultados, uma vez que a sincronização deve funcionar independentemente das mídias com as quais se esteja trabalhando.

1.2 Motivação

A possibilidade de comunicação é a grande vantagem de se ligar os computadores em rede. Mas, muito mais do que a troca de simples textos, gráficos ou imagens estáticas, a comunidade de usuários quer e necessita de uma integração muito maior, que possa combinar mídias contínuas como o áudio e o vídeo. Entretanto, a limitação dos recursos[1] e a forte relação destas mídias com o tempo requerem o estudo de novas tecnologias para que efetivamente possam ser manipuladas e transmitidas por redes de computadores e para que os usuários finais tenham o total controle do progresso dos fluxos e da qualidade da apresentação.

O desenvolvimento de sistemas multimídia combinado ao suporte de sistemas para aplicações distribuídas implicará em diversas novas aplicações para os computadores e aumentará bastante a interação entre estes e os usuários finais. Somente como exemplo, pode-se citar aplicações nas áreas hospitalar, comercial, automação de escritório, educação, lazer, etc. Praticamente, todas as atividades humanas serão afetadas pela multimídia, inclusive o comportamento humano.

Em um futuro próximo será possível trabalhar sem sair de casa, realizar reuniões entre pessoas localizadas em cidades diferentes, etc. Os sistemas multimídia terão vários efeitos colaterais como, por exemplo, ajudarão a diminuir o trânsito nas grandes cidades e aumentarão a concorrência de produtos e serviços, uma vez que

[3]IMA: *Interactive Multimedia Association*

[4]PREMO: *Presentation Enviroment for Multimedia Objetics*

[5]MSS: *Multimedia System Service*

[6] ACME: *A Continuous Media I/O Server and Its Synchronization Mechamisn*

[1] Banda passante, CPU entre outros.

será possível realizar pesquisa de preço de maneira fácil, rápida e comprar sem sair de casa.

1.3 Trabalhos Correlatos

Existem vários métodos que abordam o problema da sincronização entre fluxos de mídias contínuas em aplicações distribuídas. As soluções consideram que a sincronização seja suportada por vários componentes do sistema incluindo o sistema operacional, o sistema de comunicação, a base de dados, os documentos e até as aplicações. Deste modo, a sincronização deve ser considerada como uma funcionalidade distribuída em várias camadas do sistema multimídia. Dentro deste contexto, várias técnicas já foram propostas.

Escobar et al. [Escobar91] [Escobar92] sugerem um método de sincronização de relógio (*clock*). As unidades de mídia transmitidas contém carimbos de tempo (*time-stamps*) que permitem ao receptor determinar o tempo presente.

Um outro tipo de sincronização de relógio usado em conjunto com um controle do transmissor é usado por Rangan et al. [Rangan93]. Assume-se que as estações receptoras têm menos poder e retornam o número corrente de unidades de mídia apresentadas. Na transmissora, as ações corretivas em cada fluxo podem ser tomadas, se necessário. Para recuperar a sincronização, unidades de mídia são deletadas ou duplicadas. A suposição básica aqui é a limitação e o conhecimento da variação do atraso (*jitter*) na rede.

Chakrabarti et al. [Chakrabarti94] sugerem uma solução onde o relógio do receptor comanda a sincronização das unidades de mídia. Dependendo do tamanho do *buffer* usado no receptor, a taxa de envio dos quadros é controlada. Assim como outros métodos de retorno, o aumento do atraso na rede diminui a reação e a estabilidade não pode ser provada quando o atraso é indeterminado. Os parâmetros de controle do algoritmo são determinados experimentalmente.

Little et al. [Little93a] [Little93b] propõem uma técnica de sincronização baseada na recepção. Quando o nível da fila de recepção alcança o limite máximo ou mínimo, os quadros são abandonados ou duplicados respectivamente. Como o processo de apresentação preocupa-se unicamente com a sincronização, não é necessário o ajuste do relógio. A escolha dos limites inferior e superior muito próximos um do outro provocará freqüentes ações corretivas e conseqüentemente erros. A função corretora que controla a duplicação ou o abandono dos quadros é aleatória, mas somente uma função baseada em taxa constante foi investigada.

O algoritmo de sincronização do *Xphone* proposto por Eleftheriadis [Eleftheriadis94] atua somente no lado de recepção, sem nenhuma influência no nó de captura, e é baseado em carimbos de tempo e na ocupação do *buffer* de áudio. Utiliza o conceito de fluxo mestre (no caso o áudio). Por conseqüência, as ações do sincronismo somente serão executadas sobre o fluxo escravo: o vídeo. O sincronismo é conseguido através da comparação entre a taxa de ocupação do *buffer* de áudio e de um histórico dos carimbos de tempo de suas unidades para determinar se uma unidade

de vídeo está atrasada ou adiantada em relação ao áudio. Caso esteja atrasada, será abandonada, caso contrário, será colocada no *buffer*.

No trabalho "Definição de um Serviço de Sincronização Contínua para Aplicações Multimídia" [COPPEa] foi desenvolvido um servidor de sincronização centralizado para dados vivos e/ou armazenados que implicitamente carregam consigo as relações temporais. O método não utiliza relógio global. A sincronização é realizada através da colocação de Carimbos de Tempo Relativos (CTR) à geração e à reprodução de cada unidade de mídia. A CTR não é um número, mas sim um intervalo de tempo em que a unidade pode ter sido gerada ou reproduzida levando-se em consideração as probabilidades de menor e maior atraso da rede.

A sincronização serial é realizada através do armazenamento em *buffers* de unidades de mídia (UM) e de unidades de realimentação que são transmitidas pelo dispositivo de mídia mestre quando da exibição de certas UM's. Tal fato tem a finalidade de se evitar que os *buffers* fiquem vazios (*starvation*) ou cheios (*overrun*).

A sincronização intermídia é alcançada através da análise de unidades de mídia do fluxo escravo que poderiam ter sido reproduzidas juntamente com determinadas unidades de mídia do fluxo mestre.

As ações corretivas são executadas somente sobre o fluxo escravo que fica sujeito a saltos (para apressá-lo) ou repetições (para atrasá-lo) quando a assincronia ultrapassar valores pré-determinados.

Little e Ghafoor [Little90] propõem uma especificação formal dos requisitos de composição da mídia para o armazenamento e a recuperação em base de dados. A proposta é baseada em modificações do modelo da Rede de Petri que adicionam as funcionalidades da definição de tempo e da utilização de recursos. O novo modelo é chamado de OCPN (*Object Composition Petri Net*). Os objetos são representados pelo grafo $N=\{T, P, A, D, R, M\}$ onde T, P e A são os conjuntos de transições, de lugares e de arcos, respectivamente; M é o mapeamento do conjunto de lugares em inteiros ($M:P \rightarrow I$, $I = \{0, 1, 2, \ldots\}$), D é o mapeamento do conjunto de lugares em números reais (durações) ($D:P \rightarrow R$) e R, o mapeamento do conjunto de lugares no conjunto de recursos ($R:P \rightarrow \{r_1, r_2 .. r_n\}$).

Nesta nova definição, a transição ocorre instantaneamente e os lugares, ao invés das transições, possuem estados. O lugar permanece ativo durante o intervalo de tempo τ especificado. Para apresentar um *slide*, por exemplo, que consiste em uma sequência de áudio e elementos visuais sincronizados de durações variáveis, faz-se uso de dois eixos: o horizontal indica o tempo e o vertical os recursos (ver Figura 1.1).

Transportanto esta representação para a Rede de Petri, obtem-se a Figura 1.2. Para cada lugar está assinalado o recurso necessário à apresentação e o tempo requerido para a reprodução da unidade de mídia. As transições são indicadas por pontos de sincronização e os lugares por processamento.

Através desta técnica, é possível representar a associação existente entre várias mídias. Entretanto, o método possui algumas restrições: a apresentação espacial na tela não é definida; não há interação dinâmica com o usuário; para mudar uma

apresentação é necessário mudar a base de dados; não prevê os atrasos inseridos com a implementação e não fica clara a utilização dos recursos.

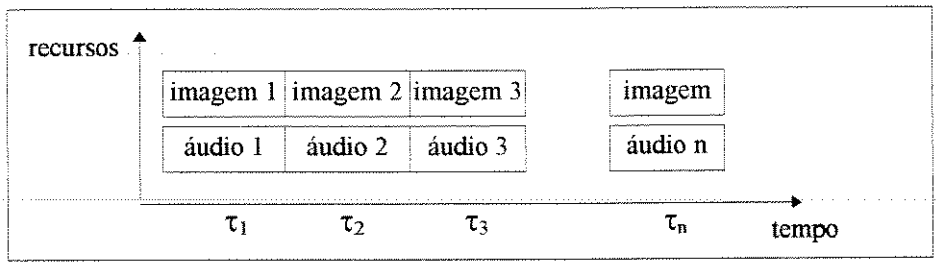

Figura 1.1 - Apresentação de um *slide*.

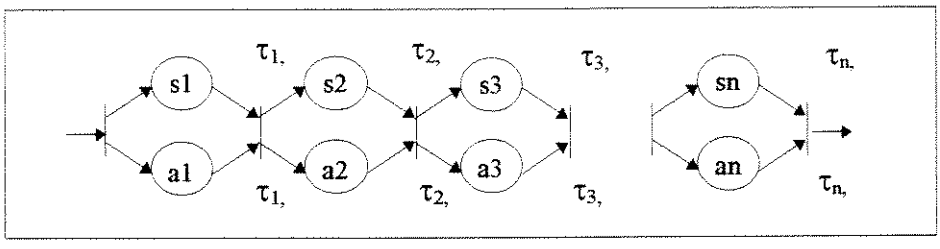

Figura 1.2 - Apresentação do *slide* em Rede de Petri.

No trabalho "Comunicação Sincronizada de Dados Pré-Orquestrados" [COPPEb] é enfocado o problema da sincronização de dados de mídia armazenados e pré-orquestrados. O armazenamento segue o modelo estendido da Rede de Petri temporizada (*Object Composition Petri Net - OCPN* [Little90]) adicionando a este as funcionalidades de obtenção e liberação de recursos e a definição de pontos de sincronização.

Através das informações temporais contidas na OCPN estendida, a recuperação dos objetos é planejada de modo a se definir o tamanho dos *buffers* necessários a uma apresentação contínua da mídia. Entretanto, somente o planejamento da recuperação não é suficiente para garantir que os dados sejam apresentados de forma síncrona e contínua. Para isto, é estabelecido pontos de sincronização na OCPN que são transmitidos para o destino. Ao recebe-los, algoritmos de sincronização, baseados no bloqueio dos fluxos que estejam mais adiantados, são executados.

O ACME (*A Continuous Media I/O Server and Its Synchronization Mechanism*) é um servidor de sincronização para áudio e vídeo desenvolvido por Anderson et al. [Anderson91]. A sincronização é realizada elegendo-se um fluxo mestre (normalmente o áudio), segundo o qual todos os outros (chamados de fluxos escravos) devem acompanhá-lo: caso o mestre se atrase ou se adiante, os escravos também se atrasam ou se adiantam. Para tal, faz-se o uso de carimbos de tempo de um relógio lógico chamado de LTS (*Logical Time System*). O LTS deve andar

aproximadamente com a mesma taxa de um relógio de tempo real e cada unidade de mídia marcada com carimbos de tempo `t` deve ser reproduzida o mais próximo possível de quando o LTS alcançar também o valor `t`.

Em decorrência de fatores como a aplicabilidade para sincronização paralela entre fluxos de mídia viva, a facilidade e simplificações que puderam ser feitas, este último foi o algoritmo escolhido para se fazer a implementação e, devido à sua importância para o trabalho, ele está mais detalhadamente descrito no Capítulo 3.

1.4 Organização do Trabalho

O trabalho está organizado da seguinte forma: o Capítulo 2 apresenta aplicações dos sistemas distribuídos multimídia e aborda o problema da sincronização de mídia contínua. O Capítulo 3 descreve os modelos de sincronização (ACME, IMA/PREMO/MSS e Orquestração) que deram suporte à plataforma proposta deste capítulo. No 4, encontra-se a implementação do servidor de sincronização e os resultados obtidos. E, finalmente, o Capítulo 5 traz as conclusões e os trabalhos futuros.

Para facilitar a legibilidade resolveu-se adotar a seguinte notação:

- uma referência a uma interface específica, ou a um método, ou a uma constante definida para uma interface, aparece com a fonte `Courier`, por exemplo, a interface `Group` descrita na seção 3.3.1;
- referências a termos da língua inglesa que ainda não tenham um termo correspondente em português, ou que o correspondente em português ainda não seja de ampla divulgação está escrito em *itálico*;
- também está em *itálico* as palavras que se queria dar ênfase ou simplesmente ressaltar.

Capítulo 2

UMA ABORDAGEM SOBRE SISTEMAS MULTIMÍDIA E SINCRONIZAÇÃO

2.1 Introdução

As possíveis aplicações dos sistemas multimídia são inúmeras. As potenciais áreas de desenvolvimento vão desde as aplicações domésticas até a medicina, engenharia, comércio, etc. Todas elas se beneficiarão da agregação de diferentes mídias em um sistema integrado.

A utilização de diversas mídias impõe a existência de mecanismos capazes de gerenciá-las. Quando áudio e vídeo são transmitidos por uma rede de computadores, pertubações temporais podem surgir e deverão ser corrigidas através de mecanismos de sincronização capazes de garantir que a qualidade da informação não seja comprometida.

O estudo e a relação existente entre aplicações, tipos de mídias, qualidade de serviço e sincronização serão os temas abordados neste capítulo.

2.2 O que é Multimídia

> *"Multimídia: An application requiring more than two trips to the car to operate."*
>
> *Interactive Multimedia Association (IMA)*

Neste século, o mundo presenciou um aumento significativo na forma e nos meios de comunicação. Primeiro veio a transmissão de som através do rádio e telefone, depois a televisão adicionou imagem a este e, ultimamente, temos a comunicação via fax, computadores, telefones e várias outras.

O desejo de unir todas estas formas de comunicação, ou mídias (voz, texto, vídeo, animação, etc), levou à criação do termo multimídia, que pode, então, ser definido da seguinte forma [Williams91]:

multimída = variedade + integração

É necessário que um sistema multimídia suporte vários tipos de mídia. Isto pode ser modesto como suportar texto e gráficos ou bastante complexo como suportar áudio e vídeo. Entretanto, isto ainda não é suficiente para um ambiente multimídia. É importante que as várias formas de mídias sejam integradas num único contexto de sistema. Desta maneira, um sistema multimídia[1] é aquele que permite aos usuários finais compartilhar, comunicar e processar uma variedade de formas de informação de um modo integrado.

2.3 Aplicações dos Futuros Sistemas Multimídia

As oportunidades abertas pela multimídia são imensas. Existem várias áreas de aplicação onde a multimídia pode melhorar a utilidade dos sistemas existentes. Similarmente, existem muitas áreas novas de aplicação que se tornaram possíveis com a emergência da multimídia. Esta seção examina algumas das potencias áreas de aplicação da multimídia com ênfase nas aplicações distribuídas. A discussão considera cada uma das seguintes áreas [Williams91]:

- automação de escritório;
- aplicações na indústria de serviços;
- comércio;
- aplicações domésticas;
- ciência e engenharia e

[1] Também chamado, neste trabalho, de aplicação multimídia.

- trabalho cooperativo suportado por computador (*CSCW - Computer Supported Cooperative Work*).

2.3.1 Automação de Escritório

Um dos maiores beneficiários da tecnologia multimídia encontra-se na automação de escritório e nos negócios. Os escritórios produzem e consomem grandes quantidades de informação sob várias formas. Fornecer o suporte para o processamento efetivo e comunicação desta informação é um dos objetivos principais de qualquer aplicação de escritório baseada em computação.

2.3.2 Aplicações na Indústria de Serviços

Existe atualmente um grande esforço no sentido de investigar como a computação multimídia pode melhorar os serviços fornecidos na área dos negócios, educação e saúde.

2.3.2.1 Educação

O uso da multimídia na educação tem sido motivo de atenção nos últimos anos. As escolas têm mostrado interesse na utilização de texto, imagem e som para complementar e melhorar os métodos tradicionais de ensino. É interessante observar que, diferentemente das aplicações comerciais onde a multimídia é considerada como mecanismo para processamento de tipos de informação não disponíveis anteriormente, a multimídia na educação tem sido utilizada devido a sua potencialidade em despertar o interesse pelo aprendizado tornando-o uma tarefa mais agradável.

A possibilidade de interação remota entre professores e alunos é uma área de aplicação que se encontra em estudo.

2.3.2.2 Finanças

O sistema financeiro é visto como um dos maiores consumidores de informação. O sucesso de uma organização deste tipo depende da eficiência de coleta de informações e dos seus sistemas de comunicação. A natureza dos mercados financeiros requer que a informação relevante seja filtrada rapidamente e colocada à disposição daquelas pessoas que decidem sobre a realização de investimentos baseados nas condições de mercado. A informação recebida pode ser originada de diversas fontes diferentes (por exemplo, relatórios da empresa, informações políticas, etc), as quais podem incluir uma forma variada de tipos de mídia. É importante, portanto, que o sistema que recebe, comunica e apresenta a informação, seja capaz de manipular as muitas formas potenciais de informação que possam ser encontradas. De modo a suportar este nível sofisticado de processamento da informação, é necessário

o uso dos sistemas distribuídos multimídia. Várias aplicações na área financeira estão sendo estudadas de modo a melhor compreender os seus requisitos de informação.

2.3.2.3 Saúde

O serviço de saúde consiste em um número variado de equipamentos que produzem grandes volumes de informação. *Scanners*, por exemplo, produzem saída de vídeo, enquanto as máquinas de raio-X produzem imagens de alta qualidade. A qualidade da informação disponível à equipe médica e a facilidade de acesso a estas informações são fundamentais para o tratamento de pacientes, especialmente quando estes encontram-se em situações de risco, acidentes, etc. A multimídia distribuída fornece a infraestrutura de suporte para o desenvolvimento de sistemas médicos em condições de manipular a informação de vários tipos, como também, acessá-las rápido e facilmente de qualquer localização.

2.3.3 Comércio

A multimídia fornece oportunidades em várias áreas do comércio onde a tecnologia pode ser empregada para oferecer aos compradores uma informação completa e atualizada sobre o produto. A informação fornecida pode ser uma descrição do produto ou ser o produto propriamente dito. Esta aplicação tenderá a aumentar a concorrência entre as empresas, uma vez que será possível comparar e escolher remotamente um produto.

2.3.3.1 Editoração

A multimídia tem o potencial de afetar a indústria de editoração de várias formas. A edição de jornais, por exemplo, possui requisitos similares às organizações financeiras que necessitam obter e comunicar rapidamente informações de fontes variadas. Outra importante área é a publicação eletrônica, ou seja, a produção de jornais e livros na forma eletrônica permitindo que os editores utilizem o áudio e o vídeo em adição ao texto e às imagens. Publicações *on-line* ofereceram aos leitores informações de forma dinâmica e interativa. Prevê-se que as publicações eletrônicas torna-se-ão parte integrante de sistemas de informação sofisticados que fornecerão acesso a uma variedade de informações disponíveis em uma rede global multimídia. Estas informações, incluindo jornais, livros, relatórios internos de empresas, irão requerer um alto grau de gerenciamento de informação e uma ampla disponibilidade de redes de alta velocidade.

2.3.3.2 Turismo

A indústria de editoração foi apresentada como uma área onde a tecnologia multimídia pode auxiliar na substituição de produtos baseados em papel por alternativas eletrônicas. A informação eletrônica possui a vantagem de ser mais compreensiva, atualizada e interativa. A indústria do turismo também pode se

beneficiar destas formas de apresentação da informação. Os turistas podem ter acesso a pontos de informação com descrição em vídeo e áudio sobre localidades de férias.

2.3.4 Aplicações Domésticas

Durante muitos anos os métodos de comunicação utilizados no acesso às residências têm sido o correio, o telefone e a televisão. Cada um destes sistemas de comunicação possui limitações do ponto de vista da interação e seleção da informação. As empresas telefônicas e de televisão têm tentado reduzir estas limitações através do uso do teletexto, no caso da televisão, e de linhas de informação no caso do telefone. Assim, os sistemas tentam oferecer um conjunto de informações que podem ser selecionadas interativamente. Os fatores limitantes destes sistemas são os tipos restritos de mídia e os métodos de interação.

A introdução de tecnologia de comunicação capaz de transmitir informação multimídia às residências, está criando a infra-estrutura necessária para que as pessoas possam acessar com grande praticidade os serviços de informação. Compras, cursos, manutenção de residência, informações de segurança, consultas médicas, são exemplos de informações e serviços que estarão à disposição. Embora os exemplos sejam voltados especificamente para requisitos domésticos, o fornecimento de tecnologia multimídia nas residências também permitirá aos usuários a posibilidade do trabalho profisssional ser realizado a partir de casa. O potencial das aplicações domésticas da multimídia é um dos principais aspectos associados a esta tecnologia para os próximos anos.

2.3.5 Aplicações nas Ciências e Engenharia

Embora uma grande proporção do esforço de pesquisa das aplicações multimídia tenha se concentrado nas áreas de automação de escritório e nas atividades de negócios, as atividades científicas e de engenharia também são áreas de importantes aplicações. Existe uma diferença substancial entre estas aplicações em função da finalidade da informação que deve ser suportada. A multimídia no contexto de escritório/negócios pode ser vista como um mecanismo de comunicação pessoa a pessoa onde áudio e vídeo podem suportar a tomada de decisão e o trabalho cooperativo via videoconferência. Neste tipo de aplicação mídias como o áudio e o vídeo não requerem processamento. Como o processamento do áudio e do vídeo, neste caso, é realizado pelos ouvidos e olhos dos usuários, é possivel reduzir a qualidade de modo a diminuir a demanda do processamento de comunicação e dos dispositivos de armazenamento.

Aplicações multimídia na área técnica estão mais realcionadas à análise da informação de modo a derivar conclusões sobre a fonte de informação. O uso da multimídia neste caso não é mais pessoa à pessoa, mas sim, da fonte para o analisador, onde este último pode ser uma pessoa ou um dispositivo digital. Desta maneira, a aplicação técnica coloca restrições mais rígidas em relação à qualidade da informação.

2.3.6 Trabalho Cooperativo Suportado por Computador

O termo Trabalho Cooperativo Suportado por Computador (*CSCW - Computer Supported Cooperative Work*) refere-se ao uso do computador para suportar grupos de usuários. Atualmente é reconhecido que CSCW é um tópico de pesquisa eminentemente interdisciplinar e requer o envolvimento de um número variado de áreas como sociologia, ciência organizacional, psicologia e ciência da computação.

Dentro das atividades de CSCW, quatro classes gerais de sistemas cooperativo têm se destacado: sistemas de mensagens, conferência por computador, salas de encontro e sistemas de co-autoria.

2.3.6.1 Sistemas de Mensagens

Estes sistemas são descendentes dos primeiros programas de correio eletrônico que permitiam a um usuário de um computador central enviar mensagens textuais a outros usuários da mesma máquina. Com o desenvolvimento das redes de computadores, os sistemas de correio eletrônico aumentaram em complexidade e funcionalidade.

2.3.6.2 Conferência por Computador

Estes sistemas também são desenvolvidos a partir dos programas de correio eletrônico. Entretanto, neste caso é necessário impor uma estrutura de como as mensagens são agrupadas. Uma conferência por computador típica consiste em números de grupos, cada qual com o seu conjunto de membros. As conferências são organizadas de modo a que, individualmente, abordem um único tópico. O usuário subscreve aquelas conferências nas quais tem interesse.

2.3.6.3 Salas de Encontro

Uma sala de encontro automatizada típica consiste de uma sala de conferência, equipada com uma grande tela de projeção, um computador (ou rede de computadores), um número de terminais individuais para entrada de dados/votação e um terminal de controle. O sistema computacional que suporta a reunião geralmente utiliza programas multi-usuário baseado em alguma técnica de decisão.

2.3.6.4 Sistemas de Co-Autoria

Representam uma classe de sistema cujo objetivo é suportar a negociação e argumentação envolvidas em um trabalho em grupo. A autoria cooperativa de documentos é uma demonstração desta classe de cooperação onde a geração final de um documento representa o produto de um processo de negociação entre os autores. Esta classe de sistemas é caracterizada pelo uso da tecnologia de hipertexto.

2.4 Tipos de Mídias

Os sistemas multimídias não pretendem criar nenhum tipo de mídia diferente. O objetivo principal é a integração entre as várias formas de mídias, inclusive as que, de alguma forma, já eram tratadas isoladamente por aplicações anteriores. A percepção das grandes vantagens advindas com a união de diversas mídias em um único sistema causará uma mudança radical no uso do computador, tornando-o muito mais próximo e útil do usuário final. Vejamos alguns exemplos destas mídias:

- *Texto Codificado:* trata-se de caracteres representados por um determinado código (ex: ASCII ou EBCDIC). Informações adicionais tais como o fonte utilizado, o estilo do caracter (*itálico*, **bold**, etc) e o tamanho podem ser armazenados juntamente com a descrição propriamente dita do caracter;
- *Gráficos Vetoriais:* imagens no formato são armazenadas como uma lista de instruções de como desenhar a imagem ao invés da representação direta da imagem. Estas imagens são mais comuns nos pacotes de projeto auxiliado por computador e nos pacotes de desenho orientados a objetos;
- *Imagens Mapeadas em Bit (bitmap):* estas imagens são usadas pela máquina para armazenar representações da informação a ser apresentada na tela. Deve ser observado que caracteres armazenados na forma *bitmap* são distintos daqueles armazenados na forma de texto codificado;
- *Voz e Áudio de Alta Qualidade:* a distinção entre os tipos voz e áudio deve-se ao fato de que estas mídias possuem características diferentes. Em particular, as cordas vocais dos indivíduos geram sons dentro de um intervalo de freqüência relativamente pequeno e o homem possui a capacidade de corrigir, subconscientemente, erros e ruídos nas mensagens de voz. O áudio de alta qualidade, por outro lado, requer uma representação muito mais acurada se o seu significado não puder ser perdido. Este tipo de mídia é particularmente útil para registro de eventos ligados à vida real para futuras referências ou para inclusão em documentos multimídia;
- *Vídeo:* o vídeo digitalizado refere-se à informação gerada quando um sinal de vídeo analógico é amostrado e convertido na forma digital. O termo também refere-se à imagem em movimento.

De modo a discurtir as questões que surgem quando da tentativa de integrar estes tipos de mídia, é útil agrupá-las em dois conjuntos: as que são contínuas por natureza e as que não o são.

2.4.1 Mídias Contínuas

Dentre estes diversos tipos de mídia, pode-se distinguir um grupo especialmente importante para as aplicações multimídia. É o grupo das mídias contínuas (ou isócronas), denominadas desta forma devido à sua estreita relação com o tempo. Possuem uma dinâmica natural expressa por uma freqüência de captura, comunicação e apresentação que se não for obedecida reduz significativamente a qualidade da

informação associada. Neste grupo encontram-se o áudio e o vídeo, que serão as mídias de estudo, para efeito de sincronização apresentada neste e nos próximos capítulos.

O outro grupo, denominado de mídia não contínua (ou não isócrona), não possue relação com o tempo e, por este motivo, o mais importante é a composição espacial destas no momento da apresentação (por exemplo: texto, gráfico e desenho).

2.5 Requisitos de Tempo-Real de Mídia Contínua para Processamento Distribuído

A característica mais fundamental de sistemas multimídia é que eles incorporam mídias contínuas. O uso de tais mídias em sistemas computacionais distribuídos, implica em capturar, processar, transmitir e apresentar dados continuamente por longos períodos de tempo. Além disso, devido às suas características, requer-se a garantia de que os dados possam ser entregues e apresentados de forma cadenciada no tempo.

A partir de uma perspectiva temporal, aplicações multimídia podem ser consideradas como uma subclasse de uma grande classe de aplicações de tempo real, a qual inclui também computação distribuída de tempo real, sistemas de controle remoto e aplicações interativas.

Porém, apesar de possuírem características temporais, sistemas multimídia não podem ser caracterizados propriamente como sistemas de tempo real crítico (*hard real time*), uma vez que toleram ocasionais atrasos provocados por latências de armazenamento, comunicação e transmissão, sem que ocorram resultados catastróficos ao sistema.

No outro extremo, em determinadas situações, sistemas multimídias são descritos apropriadamente pela semântica *melhor-nunca-do-que-tarde*. Este é o caso em que um quadro de vídeo encontra-se tão atrasado que é melhor não apresentá-lo. Assim, sistemas multimídia não podem ser caracterizados completamente como sistemas de tempo real flexível (*soft real time*).

Em vista disso, assume-se que sistemas multimídia sejam mais adequadamente caracterizados como sistemas de tempo real orientado à Qualidade de Serviço (*Quality of Service - QoS*), onde os seus requisitos temporais de desempenho são descritos a partir de índices de qualidade de serviço, os quais o sistema tem que atender.

Desse modo, a noção de qualidade de serviço torna-se fundamental. Por isto é indispensável caracterizá-la da forma mais adequada possível.

O termo QoS é bastante intuitivo para o usuário de aplicações multimídias, ele expressa basicamente uma noção subjetiva de quão bons estão os serviços fornecidos pelo sistema. Porém, para avaliar objetivamente o desempenho do sistema é necessário que se disponha de padrões mensuráveis. Visando isto, foi realizada uma tentativa de descrever a qualidade de serviço de sistemas multimídia objetivamente, a partir de um conjunto de parâmetros bem definidos, os quais buscam caracterizar

satisfatoriamente a noção subjetiva de qualidade de serviço fornecida por um dado sistema multimídia [Guedes95].

2.5.1 A QoS e a Sincronização

Há vários parâmetros que podem ser utilizados na caracterização de QoS das diversas mídias suportadas por um ambiente de computação multimídia. Eles podem ser agrupados em níveis, desde os mais baixos, como os relacionados aos sistemas operacionais, até os mais altos, como os destinados aos usuários da aplicação. Importante, neste trabalho, são os parâmetros que estão nos níveis inferiores do sistema multimídia.

Estes parâmetros estão mais relacionados com as características requeridas pelo sistema de comunicação, caracterizando-se assim, como um nível mais baixo de abstração. Os parâmetros estabelecidos são [Shepherd92] [Furht94]:

- Vazão (Largura de Banda, *throughput*);
- Atraso fim-a-fim (*end-to-end delay*);
- Variação do atraso fim-a-fim (*jitter*);
- Taxa de pacotes perdidos (*Packet Error Rate - PER*);
- Taxa de *bits* errados (*Bit Error Rate - BER*).

A vazão é a quantidade de *bits* por segundo (b/s) requerida para a transmissão na rede por uma mídia.

O atraso fim-a-fim é o tempo gasto desde a produção de um dado até sua exibição. Por produção entende-se a captura do dado por um periférico (câmera ou microfone, por exemplo) ou sua ativação numa base de dados. Este atraso é devido às latências na captura, ativação, processamento e transmissão dos dados.

A variação do atraso fim-a-fim é medida entre duas exibições sucessivas de uma mídia. Para obter uma boa qualidade na apresentação, mídias contínuas requerem baixos níveis de variação do atraso. Idealmente, a variação deveria ser nula, ou seja, a taxa de apresentação dos dados deveria ser constante, porém, devido ao compartilhamento de recursos de processamento, à variação na taxa de compressão/descompressão dos dados e à natureza estocástica das redes, tem-se que o atraso fim-a-fim não é constante.

A taxa de pacotes perdidos e a taxa de *bits* errados são parâmetros de QoS que visam caracterizar o grau de confiabilidade do sistema de comunicação. Eles são bastantes importantes, uma vez que para a maioria das aplicações distribuídas multimídia o mecanismo de retransmissão de dados não é factível. Estes parâmetros são obtidos de forma estatística, o que confere ao conjunto de parâmetros de QoS uma característica estocástica.

Os parâmetros como a largura de banda, a taxa de pacotes perdidos e a taxa de *bits* errados são determinados exclusivamente pela rede. A largura de banda deve ser garantida, através de mecanismos de reserva de recursos, visando eliminar o máximo

possível as fontes causadoras de perturbações temporais. A sincronização pode, então, focar-se na variação de atraso de apresentação das mídias.

2.6 Sincronização

> *... multimedia I/O is not synonymous with interface bells and whistles. Instead, multimedia should be synonymous with the synchronization of bells and whistles so that application programs can integrate data from a broad spectrum of independent sources (including those with strict timing requirements).*
>
> D.C.A. Bulterman e R. Liere

Sincronização, em um sistema multimídia, é o termo referente à relação existente entre a mídia e o tempo. Seu objetivo é manter a cadência de apresentação das mídias isócronas, ou seja, fazer com que as unidades de um fluxo de mídia que foram capturadas por fontes independentes sejam reproduzidas com a mesma taxa de captura e que mantenham a mesma defazagem temporal existente entre ambas no momento da captura.

Matematicamente, pode-se expressar o objetivo da sincronização da seguinte forma. Seja t_{oi}^{aq} e t_{oi}^{ap} o tempo de aquisição e apresentação da i-ésima unidade do fluxo o. Então, para uma sincronização exata, as seguintes condições devem ser satisfeitas [Eleftheriadis94]:

1) $t_{oi}^{ap} - t_{oi-1}^{ap} = t_{oi}^{aq} - t_{oi-1}^{aq}$ para todo i e o (sincronização serial); e [Equação 2.1]

2) $t_{oi}^{ap} - t_{ki}^{ap} = t_{oi}^{aq} - t_{ki}^{aq}$ para todo i, o e k (sincronização paralela). [Equação 2.2]

Desde que a obtenção de tempo de um computador, especialmente em ambientes multitarefa, não oferece exatidão, tanto o tempo de aquisição quanto o de apresentação podem somente ser aproximados. Estas condições somente podem, então, ser satisfeitas de forma aproximada.

Em vista da incerteza do tempo, a tarefa do algoritmo de sincronização é garantir as seguintes condições:

1) $t_{oi-1}^{ap} \leq t_{kj}^{ap} < t_{oi}^{ap}$ se $t_{oi-1}^{aq} \leq t_{kj}^{aq} < t_{oi}^{aq}$ para todo o, k, i e j; e [Equação 2.3]

2) $t_{oi}^{ap} < t_{oj}^{ap}$ se $t_{oi}^{aq} < t_{oj}^{aq}$ para todo `i` e `j`. [Equação 2.4]

A útima condição é simplesmente uma condição de ordenação. Nota-se que estas condições limitam o desalinhamento da sincronização pelo intervalo de tempo requerido para a aquisição de duas mídias sucessivas. Assim, uma melhoria no desempenho do sistema (por exemplo, o aumento da taxa de amostragem dos quadros de vídeo) deve causar uma maior exatidão da sincronização.

2.7 Técnicas de Sincronização

Para se conseguir realizar uma coordenação ordenada dos eventos no tempo, várias técnicas de sincronização vêm sendo desenvolvidas na tentativa de se abranger todos os requisitos de tempo-real e de qualidade de serviço requeridas por aplicações deste tipo. Estas técnicas podem ser classificadas da seguinte forma [Pronios94] [Blair92] [Köhler94] [Blakowski96]:

1) sincronização sintética e viva;
2) sincronização condicional, serial e paralela;
3) sincronização através de carimbos de tempo (*time-stamps*), de canais, de multiplexação e de *buffers*;
4) utilização ou não de relógio global;
5) quanto à localização física das ações corretivas: na estação transmissora ou na receptora;
6) quanto aos métodos utilizados para a correção do desalinhamento: aumento ou diminuição da velocidade de apresentação das unidades de mídia, ou inserção ou eliminação das mesmas.

A escolha por um determinado critério deve ser feita levando-se em conta o *hardware* disponível, a facilidade de implementação e as demais características do sistema. A princípio, não existe nenhuma relação entre os critérios utilizados e o desempenho do sincronizador.

2.7.1 Sincronização Sintética e Viva

A sincronização sintética é usada na apresentação de mídias que foram previamente armazenadas em base de dados, enquanto que a sincronização viva se caracteriza por reproduzir exatamente, na apresentação, a relação temporal existente durante o processo de captura.

No caso da sincronização sintética a relação temporal deve estar explicitada. Embora não seja uma necessidade das aplicações multimídia, em alguns casos, o algoritmo de sincronização pode utilizar-se de métodos de retransmissão de unidades de mídia, por exemplo, daquelas que foram perdidas.

2.7.2 Sincronização Condicional, Serial e Paralela

A sincronização condicional (ou sincronização de eventos) ocorre quando a apresentação de uma mídia está ligada à satisfação de uma condição. Desta forma, sempre que algum evento ocorrer (por exemplo, quando o usuário der uma pausa na aplicação), a ação correspondente a este evento deve ser executada. A resposta do sistema a esta ação não precisa ser imediata, mas deve ocorrer dentro de um período de tempo que não cause distorções à apresentação.

A sincronização serial (ou sincronização intramídia) tem como objetivo manter a taxa de apresentação de uma sequência de unidades de mídia igual, ou o mais próxima possível, à taxa de captura, ou seja, tornar os efeitos da variação do atraso inócuos. É fortemente dependente do protocolo de comunicação, que devido à própria natureza estocástica da rede, não garante uma taxa de transmissão efetivamente constante. Outros fatores que contribuem para a variação do atraso são: a escassez de recursos (CPU, largura de faixa, etc) e a variação da taxa compressão/descompressão de dados.

Na sincronização paralela (ou sincronização intermídia) há o envolvimento de dois ou mais fluxos aumentando-se, consequentemente, a complexidade dos algoritmos de sincronização devido à necessidade de manter sob controle a posição de uma sequência de unidades de mídia em relação a outra. É importante notar que quando da transmissão das mídias por canais distintos, a variação do atraso de um fluxo poderá ser, e provavelmente será, diferente da variação do atraso de um outro. O objetivo é minimizar o máximo possível a diferença entre estes dois atrasos, tentando manter o alinhamento entre as mídias envolvidas.

A união destas três técnicas permite que a sincronização seja o mais abrangente possível: elas não são mutuamente exclusivas, pelo contrário, são complementares e devem estar presentes em um sistema de sincronização multimídia.

Alguns autores definem, ainda, mais duas formas de sincronização que não estão relacionadas ao tempo, são elas: a de conteúdo e a espacial. A relação de conteúdo define a dependência das mídias em relação a dados comuns. Um exemplo são dois gráficos que mostram interpretações diferentes dos mesmos dados.

A sincronização espacial define o espaço (*layout*) usado para a apresentação da mídia em um determinado momento. Por exemplo, os quadros de um vídeo devem ser apresentados em uma janela que pode ser movimentada ou alterada de tamanho [Blakowski96].

2.7.3 Sincronização através de Carimbos de Tempo, de Canais, de Multiplexação e de *Buffers*

Esta técnica de sincronização diz respeito à forma como a sincronização paralela ou serial pode ser efetivada. Eventualmente pode-se combinar duas ou mais destas, buscando um aumento do desempenho do sistema. As técnicas são:

1) *Carimbos de Tempo*: realiza-se a transferência de informação de tempo através da inserção de carimbos no fluxo de mídia. A inserção na estação transmissora, por

exemplo, e posterior verificação na receptora, pode determinar o atraso total na rede e a sua variação. O monitoramento dos carimbos permite a verificação e renegociação da QoS oferecida pelo sistema.

2) *Canais de Sincronização*: canais especias de sincronização transmitem periodicamente informações de sincronização e carimbos de tempo;

3) *Multiplexação*: os vários fluxos de mídia são multiplexados e transmitidos por um único canal lógico. Esta técnica não pode ser usada no caso de fontes de mídias localizadas em locais distintos. Existe, ainda, alguns argumentos contra a sua utilização [Blair92]:

   - o sobrecarregamento (*overhead*) e a complexidade da multiplexação/demultiplexação é significante, especialmente quando diferentes esquemas de codificação/compressão são usados por diferentes mídias;
   - o fato anterior é especialmente relevante quando for possível interfacear o *hardware* diretamente no sistema de comunicação;
   - a oportunidade de processar canais vituais separados em paralelo é perdida;
   - canais virtuais separados permitem o uso de canais melhores adaptados para cada mídia, conduzindo a uma utilização mais eficiente dos recursos; multiplexação conduz a uma QoS combinada que pode ser dispendiosa e inadequada para alguns tipos de mídia;
   - e ainda, quando a mídia se origina de diferentes fontes, a multiplexação não é uma opção (ver Figura 2.1).

   Entretanto, unidades de mídia fluindo através de canais lógicos diferentes podem chegar ao destino com atrasos diferentes, causando problemas de sincronização no nível de apresentação [Pronios94]. Problemas estes causados principalmente devido à natureza estocástica das redes que, mesmo diante da reserva de recursos, pode-se assumir que o seu atraso total é não determinístico.

   A complexidade adquirida pela separação das mídias por canais lógicos distintos pode parecer desnecessária diante da possibilidade de se fazer uso da multiplexação e o consequente uso de um só canal. Porém, as vantagens advindas da separação e até mesmo, em alguns casos, da impossibilidade de se fazer a multiplexação, justificam este aumento de complexidade.

4) *Buffers de Sincronização*: a ocupação do *buffer* é limitada a um valor máximo e mínimo definido pelo sistema, e através do monitoramento de sua taxa de ocupação, a ordenação dos eventos é definida.

   Como não há meios de se eliminar a variação do atraso, a sincronização deve contorná-la através da utilização de *buffers*. Atrasando a apresentação das mídias por alguns milisegundos, consegue-se fazer com que os *buffers* adquiram uma pré-condição de inicialização do sistema (*start-up*). A partir deste momento, o monitoramento da taxa de ocupação dos *buffers* é feito de forma que os limites estabelecidos não sejam ultrapassados.

O tamanho dos *buffers* deve ser calculado em função da variação do atraso: quanto maior a variação do atraso, maior deve ser o tamanho do *buffer*. Por conseguinte, quanto maior os *buffers*, maior o atraso fim-a-fim. Estes conflitos são inerentes aos sistemas multimídia e cabe ao gerenciador de QoS encontrar um ponto ótimo que esteja de acordo com as especificações da aplicação.

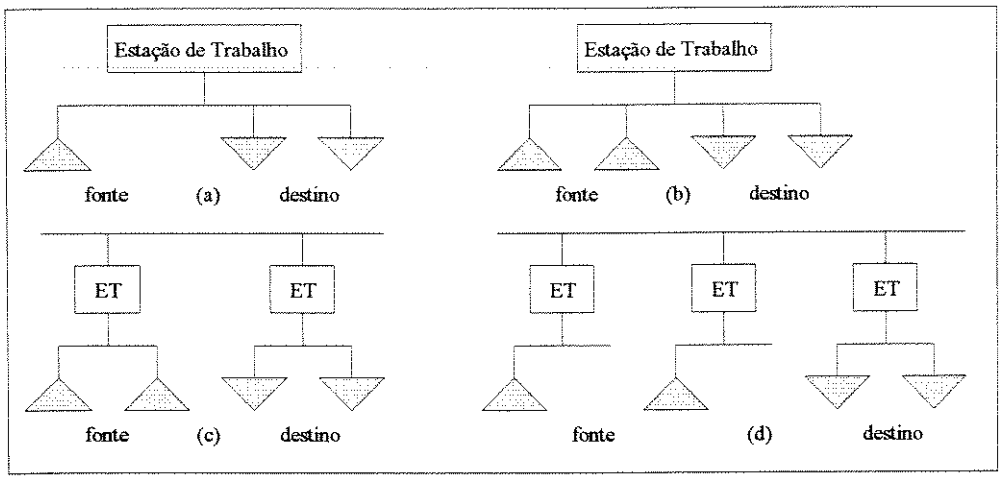

Figura 2.1 - Modelos de localização dos dados multimídia [Bulterman]. Em (a) fonte única; (b) múltiplas fontes locais; (c) fonte única distribuída; e (d) múltiplas fontes distribuídas: este é o modelo mais genérico de localização dos dados. A informação pode ser coletada em várias fontes diferentes em muitas estações de trabalho assim como os destinos podem também estar espalhados em vários locais. Os problemas de sincronização incluem escalamento de E/S através de um conjunto de estações de trabalho, atrasos de transferência através de muitos conectores e atrasos de processamento em um ou mais nós.

2.7.4 Utilização ou não de Relógio Global

O quarto critério refere-se à necessidade, ou não, do algoritmo possuir um relógio de tempo global. Em caso afirmativo, algum tipo de sincronização de relógio é utilizado. Se nenhum relógio global é requerido, a sincronização pode ser alcançada através de estratégias de *buffers* de sincronização:

- *Com Relógio de Tempo Global:* requer que os relógios das estação de trabalho sejam pré-sincronizados e se mantenham desta forma pois as unidades de mídia serão sincronizadas através de carimbos de tempo, relativos ou absolutos, inseridos na captura e, às vezes, na recepção também;
- *Sem Relógio de Tempo Global:* o sistema não requer uma noção única de tempo, desta forma não há necessidade da pré-sincronização dos relógios. A sincronização pode ser realizada através de *buffers* de sincronização ou de sincronizadores centralizados.

2.7.5 Localização Física das Ações Corretivas

As ações corretivas podem ser realizadas tanto na estação receptora quanto na transmissora das unidades de mídia contínua:

- *Receptora:* todas as ações corretivas são executadas nesta estação sem que haja qualquer notificação ou interferência da transmissora;
- *Transmissora:* controle da sincronização nesta estação sempre requer algum mecanismo de retorno da receptora. Pode ser adequado quando estas últimas têm pouco poder de processamento.

Poderia-se pensar que um método eficiente utilizaria uma combinação destes dois critérios da seguinte forma: a sincronização seria realizada na estação receptora, mas toda vez que ocorresse uma sobrecarga da rede, e as unidades de mídia começassem a chegar muito atrasadas uma em relação à outra, um sinal seria enviado à transmissora para que diminuisse a taxa de transmissão. Entretanto, quanto maior o tráfego na rede, menor será a eficiência do mecanismo de retorno.

2.7.6 Métodos Utilizados para a Correção do Desalinhamento

Esta técnica distingüe os métodos que são utilizados para se corrigir o desalinhamento, a saber:

- *Controle da velocidade:* a correção é obtida através do aumento ou da diminuição da velocidade de produção e/ou apresentação das unidades de mídia;
- *Controle das unidades:* este método faz a inserção ou a eliminação de unidades. Neste caso, pode-se inserir dados nulos ou duplicar o último capturado/apresentado. Conhecido também como sicronização *bit* ou *byte*.

2.8 Modelos de Classificação

Para facilitar a classificação e a visualização de alguns algoritmos, Köhler e Müller [Köhler94] fizeram uma representação espacial com três eixos ortogonais. Em cada eixo foi colocado dois critérios. O primeiro eixo representa o critério de localização das ações: na receptora ou na transmissora; no segundo eixo obtém-se a forma de correção: controle da velocidade ou das unidades; e o terceiro, a necessidade, ou não, de um relógio global (ver Figura 2.2).

Para poder classificar, identificar, comparar e definir os vários requisitos de sincronização e tempo-real para multimídia, assim como para definir as interfaces entre as várias estruturas dos mecanismos que suportam estes requisitos, Gerold e Ralf [Blakowski96] propõem um modelo de referência para sincronização multimídia.

O modelo de referência em quatro camadas é mostrado na Figura 2.3. Cada camada implementa mecanismos de sincronização que são fornecidos por uma interface apropriada. Estas interfaces podem ser usadas para especificar e enfocar as relações temporais. Cada interface define serviços, isto é, oferece ao usuário um meio

de definir seus requisitos. Cada interface pode ser usada por uma aplicação diretamente, ou pela próxima camada superior. Camadas superiores oferecem maiores abstrações de programação e de qualidade de serviço.

A camada de mídia opera sobre um único fluxo e é responsável por sua sincronização serial através de mecanismos de controle de fluxo entre os dispositivos produtores e consumidores. A reserva e o gerenciamento de recursos devem ser garantidos.

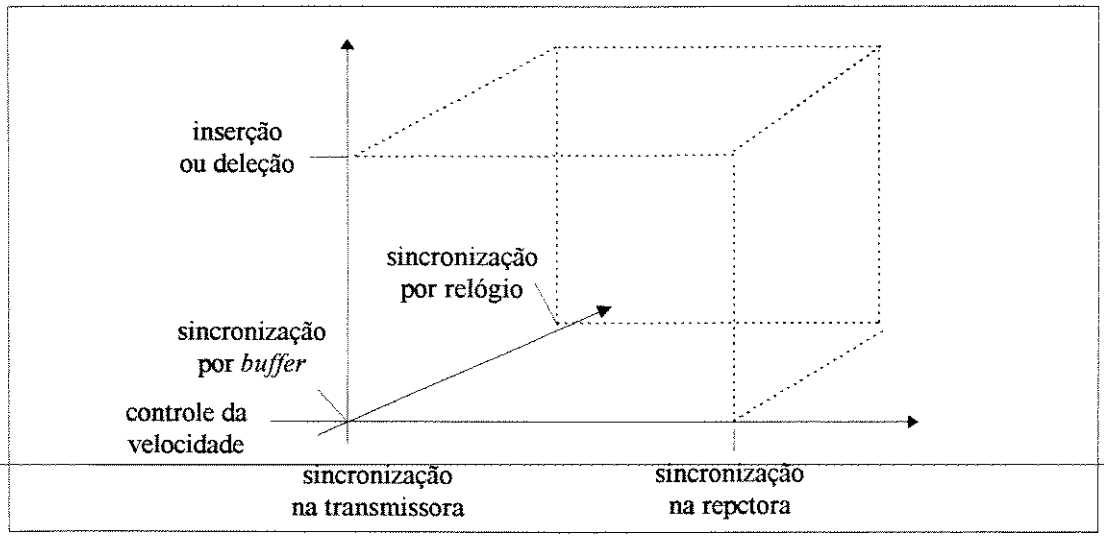

Figura 2.2 - Classificação visual de alguns algoritmos de sincronização.

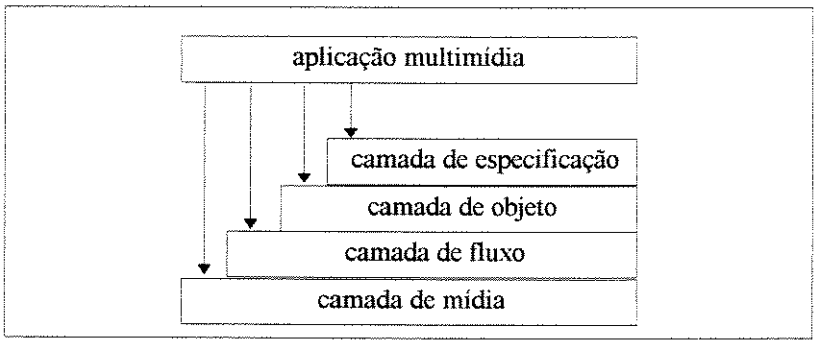

Figura 2.3 - O modelo de referência de quatro camadas de Gerold e Ralf.

Na camada de fluxo é possível a manipulação tanto sobre um único como sobre um grupo de fluxos. A sincronização paralela é, então, efetuada. Abstratações para o controle do fluxo (avanço, parada, pausa) e para a definição dos parâmetros de QoS são fornecidos. A interação com os objetos de mídia independente do tempo é

realizada através da inserção de eventos no fluxo de mídia continua. Tais eventos são enviados à aplicação sempre que o fluxo alcançar o ponto especificado.

A função da camada de objeto é operar sobre todos os tipos de mídia, omitindo da aplicação as diferenças entre as mídia dependentes e indenpendentes do tempo. Através das especificações de sincronização da aplicação as tarefas de apresentação das mídias são corretamente escalonadas.

A camada de especificação é uma camada aberta. Não oferece nenhum serviço de sincronização. Sua função é fornecer ferramentas que permitam a especificação dos parâmetros de sincronização.

É no sentido de possibilitar um ambiente com as características descritas anteriormente que se apresenta, no próximo capítulo, um modelo de plataforma de serviços de sincronização para aplicações distribuídas multimídia.

Capítulo 3

DESENVOLVENDO UMA PLATAFORMA DE SERVIÇOS DE SINCRONIZAÇÃO PARA APLICAÇÕES DISTRIBUÍDAS MULTIMÍDIA

3.1 Introdução

Um dos grandes desafios hoje é fornecer uma infra-estrutura para a construção de uma plataforma de computação multimídia que suporte aplicações interativas com capacidade de realizar sincronização e de tratar mídias dependentes do tempo em um ambiente distribuído heterogêneo. A operação em um ambiente distribuído é importante devido à grande tendência da indústria de computação em direção à arquitetura cliente/servidor e à computação distribuída. Outra grande tendência é a computação multimídia. O resultado inevitável será a intersecção destas duas áreas de forma a produzir o ambiente distribuído multimídia.

A proposta deste capítulo é o desenvolvimento de uma plataforma que procura atender parte destas necessidades. Para isto apresentar-se-á primeiramente três

trabalhos que formaram a base para a definição dos serviços e a integração entre seus objetos. A integração entre estes trabalhos, Orquestração [Blair92] [Campbell], IMA/PREMO/MSS [PREMO] e ACME [Anderson91], é realizada de forma a produzir uma plataforma que possa fornecer serviços de sincronização, controle e monitoramento do fluxo às aplicações distribuídas multimídia.

3.2 Orquestação (*Orchestration*)

Desenvolvido na Universidade de Lancaster, este trabalho [Blair92] [Campbell] faz parte do projeto ESPRIT 95. Parte de seus objetivos é o desenvolvimento de novos serviços de transporte adaptados ao ambiente de redes multiserviços de alta velocidade.

Denominada de Orquestração, esta plataforma fornece à aplicação, através de uma variedade de funções coordenadas que agem sobre múltiplas conexões de transporte, serviços de sincronização em tempo-real para mídia contínua.

Esta plataforma é dividida em múltiplas camadas. Cada uma fornece uma política à camada inferior e um mecanismo à superior. Deste modo, cada camada estabelece os objetivos a serem cumpridos pela inferior e também trata de sua reestruturação. As camadas são três: Gerenciador de Sincronização (*Synchronization Manager*), Orquestrador de Alto Nível (OAN) (*High Level Orchestration Manager - HLO*) e Orquestrador de Baixo Nível (OBN) (*Low Level Orchestration - LLO*).

O Gerenciador de Sincronização é a visão no nível da plataforma dos serviços de orquestração e aparece para a aplicação como uma interface de tipos de dados abstratos. A realização das conexões físicas de transporte é de sua responsabilidade. No nível inferior encontram-se as camadas OAN e OBN que executam sua política de sincronização (ver Figura 3.1).

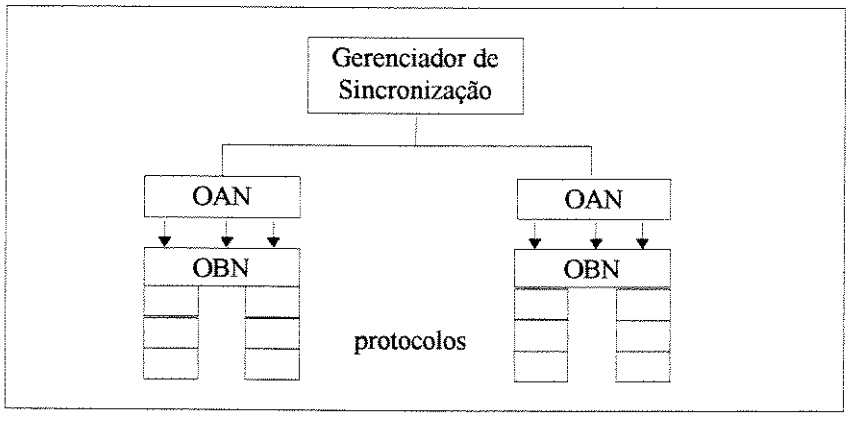

Figura 3.1 - Arquitetura de camadas: Gerenciador de Sincronização, Orquestrador de Alto Nível (OAN) e Oquestrador de Baixo Nível (OBN).

A camada OAN monitora e regula as conexões de transporte via interface da OBN. A interação entre estes dois níveis se dá através de três conjuntos de primitivas. O primeiro é usado para estabelecer e modificar as conexões. O segundo opera sobre as conexões de transporte, fornecendo a habilidade de manipular os fluxos de mídias contínuas de forma atômica. O terceiro grupo controla as taxas dos fluxos em cada conexão individualmente formando, assim, a base para a sincronização contínua através de várias conexões.

A Figura 3.2 mostra um exemplo de interação entre as camadas OAN e OBN. Em cada intervalo pré-definido, a OAN envia uma mensagem à OBN especificando a nova taxa estabelecida. Caso não seja enviada nenhuma mensagem, a anterior será mantida. Ao final de cada intervalo a OBN retorna uma mensagem de sucesso ou falha. Baseado neste retorno, a OAN especifica os novos objetivos para o próximo intervalo a fim de compensar a existência de alguma variação de velocidade. A OBN opera sob o princípio de melhor esforço e é de responsabilidade da OAN tomar as decisões apropriadas (definir novos objetivos ou renegociar a QoS) caso a OBN falhe repetidamente. O tamanho dos intervalos definem a granulosidade da sincronização.

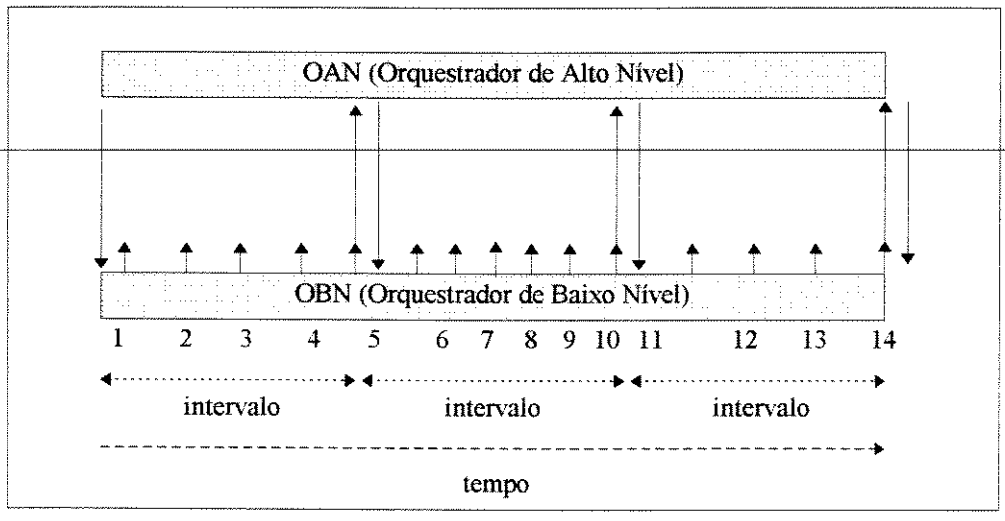

Figura 3.2 - Interação entre o OAN e o OBN.

A OBN realiza a sincronização no protocolo de transporte através do seu monitoramento e eventual redefinição da taxa de transmissão (QoS). Embora a renegociação da QoS com cada um dos dispositivos em separado seja uma vantagem poderosa, requer-se um grande esforço da aplicação que necessita executá-la atomicamente. Neste sentido, o PREMO/IMA/MSS desenvolve duas interfaces que facilitam a definição da QoS e o gerenciamento do progresso do fluxo para todo o grafo de uma aplicação.

3.3 IMA/PREMO/MSS

O IMA (*Interactive Multimedia Association*) é uma associação internacional[1] que representa todo o espectro da indústria multimídia: desenvolvedores de aplicações; fornecedores de *hardware* e *software*; integradores de sistemas, editores e distribuidores; educadores e usuários que formaram um fórum de troca de opiniões e de propostas técnicas para fins de padronização.

Desta união surgiu um documento chamado de MSS (*Multimedia System Service*) que foi submetido à ISO/IEC JTC1 SC24 como Parte 4 do PREMO (*Presentation Enviroment for Multimedia Objetics*). Hoje o PREMO é a maior atividade em andamento dentro do subcomitê JTC1 SC24.

Atualmente[2], o PREMO é composto de quatro partes e tem como objetivo focar-se na padronização dos aspectos de apresentação multimídia, assim como concentrar-se nas ferramentas de programação. Este trabalho restringir-se-á à parte do documento que corresponde ao MSS. Para maiores detalhes das demais partes consulte [PREMO].

O MSS é uma plataforma *middleware* entre o sistema operacional e as aplicações que visa fornecer um conjunto de padronizações de serviços que possa ser usado pelos desenvolvedores de aplicações multimídia em uma variedade de ambientes computacionais. Seu principal objetivo é permitir aplicações multimídia em um ambiente computacional distribuído e heterogêneo. Algumas das principais razões que levaram à sua definição são:

- fornecer abstrações e mecanismos que tornem possível às aplicações obter sucesso ao tratar os problemas de computação distribuída multimídia;
- facilidade de implementação de aplicações complexas, tais como vídeo conferência;
- fornecer abstrações para as aplicações lidarem com dispositivos sem se preocuparem com as características específicas de cada plataforma, rede, etc;
- fornecer uma metodologia padrão, especialmente para a manipulação de "dados vivos";
- considerar a natureza de tempo crítico dos dados;
- facilitar o controle da QoS.

O MSS descreve um grafo (ver Figura 3.3) de fluxo de dados, onde cada nó é um objeto VirtualDevice[3] que fornece facilidades para o gerenciamento de recursos, controle do progresso do fluxo das mídias, abstração de dispositivos e abstração de formato da mídia. Os arcos do grafo, conhecidos como objetos VirtualConnection,

[1]Entre seus componentes estão a IBM, Sun, HP, Digital, Equipment e outros.

[2]Como o documento encontra-se em desenvolvimento, pode haver modificações.

[3]A nomenclatura dos objetos pertencentes ao MSS será mantida na lingua inglesa devido à tendência deste documento tornar-se um padrão.

fornecem uma abstração para o transporte do fluxo entre os dispositivos virtuais. Estes dois tipos são uma subclasse do tipo VirtualResource.

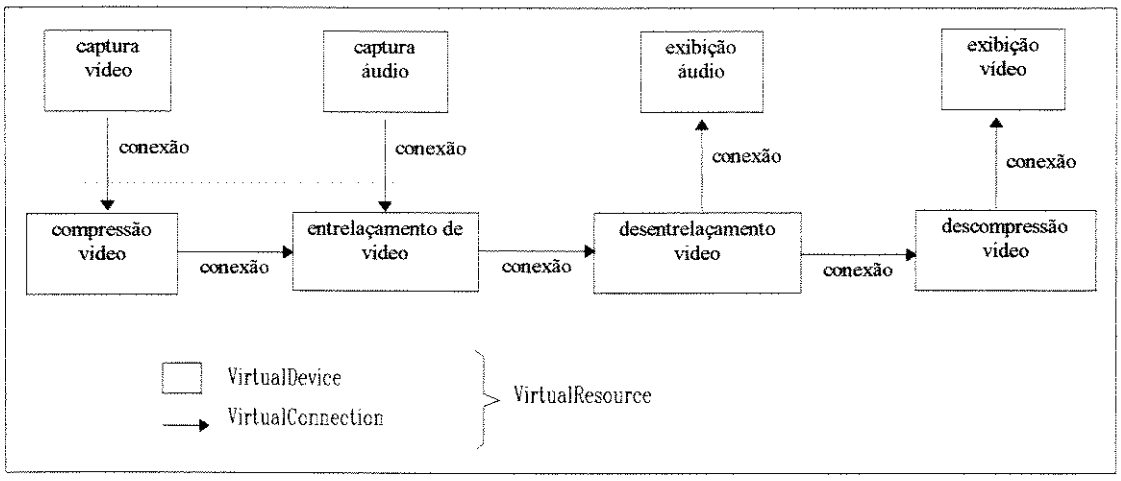

Figura 3.3 - Grafo do fluxo de dados multimídia do MSS.

O VirtualResorce é, portanto, uma abstração de recursos físicos que fornece ao cliente a possibilidade de desenvolver um modelo de programação consistente, independente dos detalhes específicos da implementação. A noção de recurso virtual torna a aplicação mais portátil e facilita a definição dos parâmetros de QoS e o controle dos dispositivos físicos através das interfaces Group e Stream.

A interface Group é também uma subclasse da VirtualResource. Sua função é agrupar os vários recursos virtuais como se fossem um único, a fim de estabelecer a QoS fim-a-fim requerida para a aplicação. Facilita também o controle do fluxo da mídia que esteja passando pelos recursos agrupados através da chamada de uma única interface (método).

A interface Stream permite ao cliente determinar a posição corrente do fluxo e executar funções para controlar o seu progresso. Esta interface fornece um meio genérico de se controlar o fluxo independentemente do tipo de mídia envolvido (ver Figura 3.4).

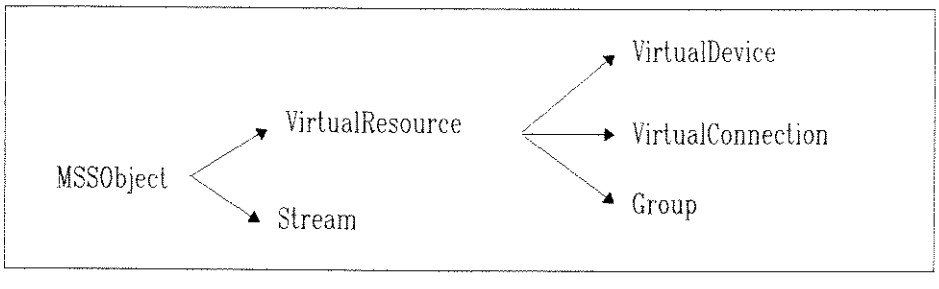

Figura 3.4 - Diagrama de herança das interfaces do MSS.

O cliente (aplicação) pode associar a interface Stream à VirtualResource. É desta forma que a interface Group, que é uma subclasse da VirtualResource, obtém o controle do fluxo que passa pelos seus dispositivos. Nas próximas seções detalha-se um pouco mais estas duas interfaces.

3.3.1 Interface Group

Sempre é desejável manipular vários recursos como um grupo. Quando estiver expressando requisitos de QoS, por exemplo, um cliente irá querer especificar o atraso fim-a-fim de um conjunto de dispositivos conectados ao invés do atraso dos elementos individuais. Quando estiver controlando um fluxo de dados através de um conjunto de dispositivos conectados, é sempre mais conveniente manipular um único objeto, ao invés de manipular os objetos associados a cada um dos recursos. Para resolver este problema a interface Group permite agregar os objetos VirtualDevice e VirtualConnection, através da chamada dos métodos add_resource() e add_resource_graph(). add_resource() adiciona somente um recurso ao grupo, ao passo que add_resource_graph() adiciona todo o grafo.

O objetivo da interface Group é realizar as seguintes operações:

- *aquisição de recursos com QoS fim-a-fim*: numa situação onde o cliente criou e conectou uma série de recursos virtuais para realizar uma função específica, a aquisição de recursos e o procedimento da especificação da QoS podem ser muito complexos. Embora a habilidade para especificar uma (possivelmente diferente) QoS para cada dispositivo virtual e para cada conexão virtual seja uma característica muito poderosa, sempre requer um grande esforço e entendimento do cliente para especificá-la corretamente. Em muitas situações, o cliente gostaria de especificar somente a QoS fim-a-fim e deixar que o sistema determine a QoS para cada nó. A habilidade de agrupar um conjunto de recursos virtuais numa única QoS resolve este problema para o cliente;
- *controle do fluxo*: desde que a interface Group seja uma subclasse da interface VirtualResource, ela pode ter uma interface Stream associada. A interface Stream fornece ao cliente a informação do progresso do fluxo de dados do grupo. Se o objeto Stream for um objeto ControlledStream (ver Seção 3.3.2.1), ele dará ao cliente um único ponto de controle para manipular todo o progresso do fluxo de mídias. Isto permite ao cliente chamar um único método de controle para ter este comando propagado para cada objeto membro do grupo.

3.3.2 Interface Stream

A interface Stream e suas subclasses fornecem um único ponto de foco para todas as consultas e controle do progresso do fluxo de mídia independentemente de seu tipo. Os objetos Streams nunca são criados isoladamente, eles são objetos inclusos dos objetos VirtualResource cuja meta é monitorar e controlar o progresso do fluxo para todos os recursos.

O monitoramento da posição do fluxo deve ser feito respeitando-se os diferentes tipos de objetos: objetos diferentes requerem conceitos de posição diferentes. São eles:

- *objeto de transporte:* entende somente pacotes de transporte;
- *objeto de fluxo:* compreende apenas unidades de mídia;
- *objeto de tempo:* sabe como extrair o tempo das unidades do fluxo.

Estes objetos podem especificar a posição em termos de uma origem e de uma coordenada de espaço. Os três modos de se interpretar a origem são:

- *absoluto*: a origem é o começo do fluxo;
- *relativo*: a origem é a posição onde o objeto recebe a requisição. O cliente pode invocar métodos tais como "para daqui a 100ms";
- *módulo*: a origem é periódica. Esta convenção é útil para eventos. O cliente pode invocar métodos tais como "notifique a cada 100ms".

A coordenada de espaço pode ser especificada em relação ao tempo, ao número de unidades ou ao número de *bytes*. A especificação da coordenada de espaço e da origem define uma lista de chave/valor que pode variar de objeto para objeto. Mas o método `get_position_key_list()` é capaz de identificar quais chaves/valores o objeto é capaz de compreender. Outros métodos da interface `Stream` permitem ao cliente:

- *requerer a precisão da posição do fluxo*: o método `get_position_precision()` retorna a precisão da posição atual do fluxo (ver método `set_sync_precision()`);
- *obter a posição corrente do fluxo*: o método `get_position()` retorna a posição atual do fluxo;
- *ser notificado de forma assíncrona sobre a posição de um fluxo*: método `set_postion_reporting()`. Pode-se, inclusive, gerar eventos periódicos especificando-se uma origem de posição modular;
- *monitorar o progresso do fluxo*: o método `insert_marker()` faz a inserção de marcas no fluxo que são posteriormente detectadas e notificadas por eventos.

Todos os métodos acima têm como argumento uma lista de chaves/valores. Isto permite ao cliente especificar a posição em termos de múltiplas coordenadas de espaço, dependendo da capacidade de cada objeto.

A interface `Stream` possui a subclasse `ControlledStream` com métodos para o controle do progresso do fluxo, que por sua vez, possui a subclasse `SyncStream` para sincronização de fluxo (ver Figura 3.5).

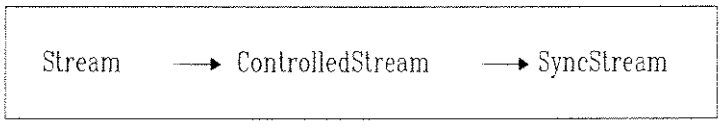

Figura 3.5 - Hierarquia de classe da interface `Stream`.

3.3.2.1 A Interface ControledStream

A interface ControlledStream suporta o controle do fluxo e a transformação de tempo. Os métodos para o controle do fluxo permitem, entre outras funções:

- suspender o progresso do fluxo: pause();
- iniciar ou reiniciar o avanço do fluxo: resume();
- esvaziar os *buffers* internos dos dispositivos lógicos: drain().

Todos estes métodos tomam uma posição (chave/valor) como parâmetro. Isto permite ao cliente especificar uma seqüência de operações de controle do fluxo em termos da sua posição; isto é necessário para permitir o controle preciso do fluxo em um ambiente onde o atraso na rede é uma variável desconhecida.

Esta interface fornece, ainda, métodos para especificar a transformação temporal do fluxo: set_stream_transform(). Alterando-se o carimbo de tempo (*time-stamp*) de um quadro de vídeo, por exemplo, pode-se alterar o momento de sua exibição. Desse modo, há somente a alteração lógica do tempo do fluxo, nenhuma alteração é feita na taxa de amostragem ou na largura de faixa gerada ou consumida.

Ainda que a transformação do fluxo não altere a taxa na qual os dados estão sendo produzidos ou consumidos numa porta, o cliente pode ter a necessidade de alterá-la a fim de sincronizar dois fluxos ou para compensar desalinhamentos da taxa. O método set_rate_scale() realiza tal tarefa.

3.3.2.2 A Interface SyncStream

A interface SyncStream é projetada para permitir a sincronização de múltiplos fluxos de mídia. Com a chamada do método attach_master() o cliente especifica um segundo objeto Stream para fornecer uma referência de posição mestre. O objeto SyncStream tentará então, ficar em sincronia com este fluxo mestre. O MSS não especifica como isto será realizado, pode ser através do monitoramento do objeto Stream para determinar sua posição, ou através do registro de interesse em eventos capazes de notificar a posição do objeto. Ainda que haja um fluxo mestre para fornecer uma referência de posição, é necessário estabelecer limites. Isto é feito através do método set_sync_precision() que define a resolução com a qual a sincronização será computada.

Semelhante ao método set_stream_transform(), o método set_master_transform () realiza a transformação temporal do fluxo mestre. Os parâmetros destes métodos são dois: scale e offset, e servem para alterar (ajustar) as taxas dos fluxos. Novamente, nenhuma alteração real na taxa de amostragem dos dispositivos físicos é efetuada. Antes da comparação entre os fluxos escravo e mestre, a transformação para uma mesma base de tempo deve ser efetivada:

posição_nova = offset + (scale * posição_atual) [Equação 3.1]

Suponha, por exemplo, um fluxo escravo que esteja gerando unidades de mídia nos tempos 0, 125, 250, 375, 500ms e assim por diante. Por sua vez o fluxo mestre gera unidades nos intervalos 0, 100, 200, 300, 400, 500ms... Para igualar as duas taxas, a solução seria fazer scale = 4/5 e offset = 0ms e aplicar a Equação 3-1 ao fluxo mestre (ver Figura 3.6).

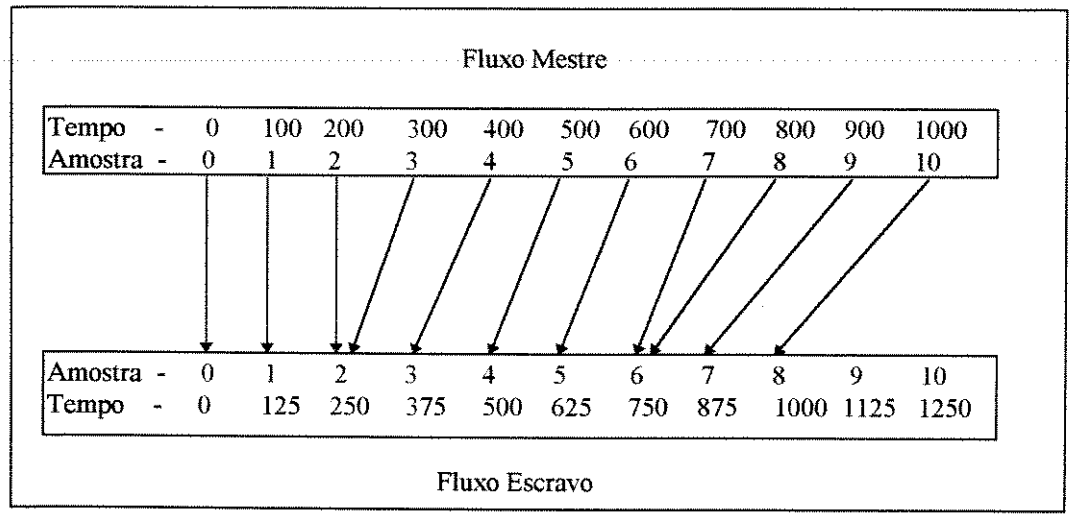

Figura 3.6 - Casamento de unidades entre fluxos de taxas diferentes.

O cliente pode também criar uma lista de *pontos de sincronização* para fins de alinhamento. Isto permite ao cliente registrar interesse em eventos que são gerados quando os fluxos atravessarem estes pontos. Isto pode ser usado pelo cliente para detectar exceções de sincronismo.

Neste ponto, é possível tecer alguns comentários sobre as semelhanças e diferenças entre o MSS e a Orquestração com o intuito de integrá-los para a construção da plataforma de sincronização.

A Orquestração define uma camada, OBN, para monitorar as conexões entre os dispositivos físicos. No MSS esta tarefa é realizada por um objeto VirtualResource associada a uma interface Stream. Todo o gerenciamento da sincronização fica a cargo da camada OAN que a realiza através do envio e recebimento de mensagens em intervalos pré-definidos. Para o MSS, toda comunicação entre os objetos é feita através da interface Stream, que não gera automaticamente notificações para a camada superior. Mas o problema é resolvido com a chamada de métodos que geram eventos periódicos (vistos anteriormente). Quando estes eventos ocorrerem, a camada superior pode tomar conhecimento da vazão atual da rede.

A Figura 3.7 mostra como seria um sistema construído sobre esta filosofia. O gerenciamento da sincronização é feito pela OAN através da interface Stream da VirtualConnection, que para efeito de facilidade de gerenciamento agrupa os recursos do

tipo VirtualResource de uma conexão virtual e também controla/monitora o fluxo através de suas interfaces Stream.

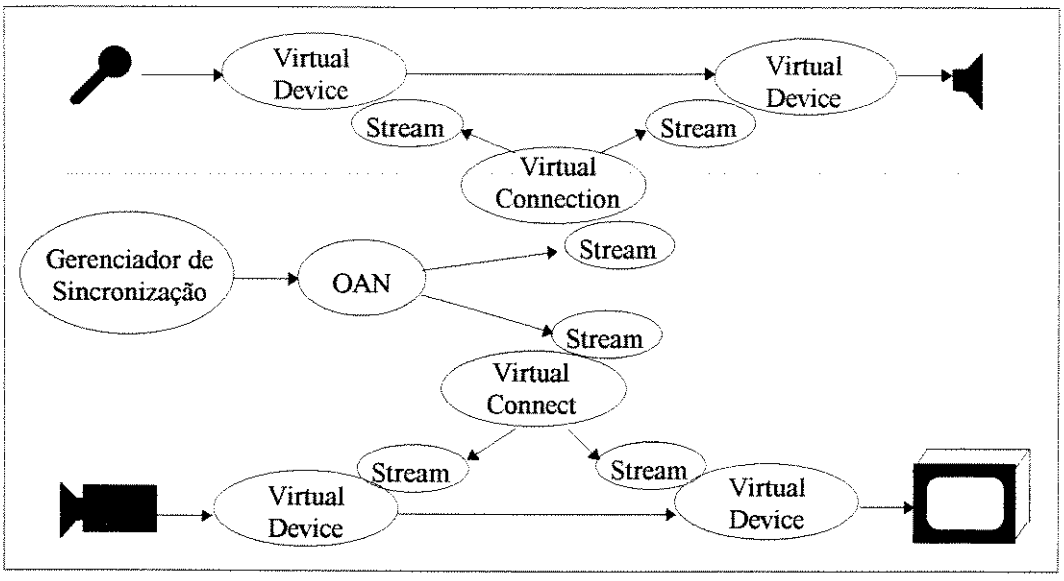

Figura 3.7 - Arquitetura do MSS sob o gerenciamento do OAN.

A sincronização na Orquestração é essencialmente uma sincronização de monitoramento e de alterações nas taxas de transmissão do protocolo de transporte. Esse controle tem por objetivo manter sob determinados limites a variação do atraso, ou seja, realizar a sincronização intramídia.

O IMA/PREMO define interfaces de sincronização. Há a possibilidade de se atrelar um fluxo a outro e também de se realizar transformações de tempo, visando o alinhamento entre eles. Entretanto, a implementação destas interfaces é deixada totalmente em aberto.

É preciso, portanto, definir o método de sincronização que será implementado por estas interfaces. É para realizar tal tarefa que propor-se-á, na próxima seção, o uso do servidor ACME.

3.4 ACME

Desenvolvido na Universidade de Berkeley o ACME *(Abstractions for Continuous Media)* é um servidor de sincronização paralela de granulosidade fina, ou seja, sua função é controlar o momento exato de apresentação de cada unidade de mídia. Suas ações são executadas somente no nó de apresentação através da análise de carimbos de tempo (*time-stamps*).

A arquitetura do sistema é mostrada na Figura 3.8. Múltiplos processos, que representam os dispositivos físicos a eles associados, compartilham o mesmo espaço de endereçamento. Estes processos realizam a E/S de unidades de mídia no servidor.

Os Processos_E/S_Rede fazem a conexão entre a rede e os *buffers* internos, e os Processos_E/S_Dispositivos, por sua vez, interligam os *buffers* aos dispositivos físicos. Os *buffers* são utilizados pelo sincronizador como um local estratégico de armazenamento de unidades de mídia.

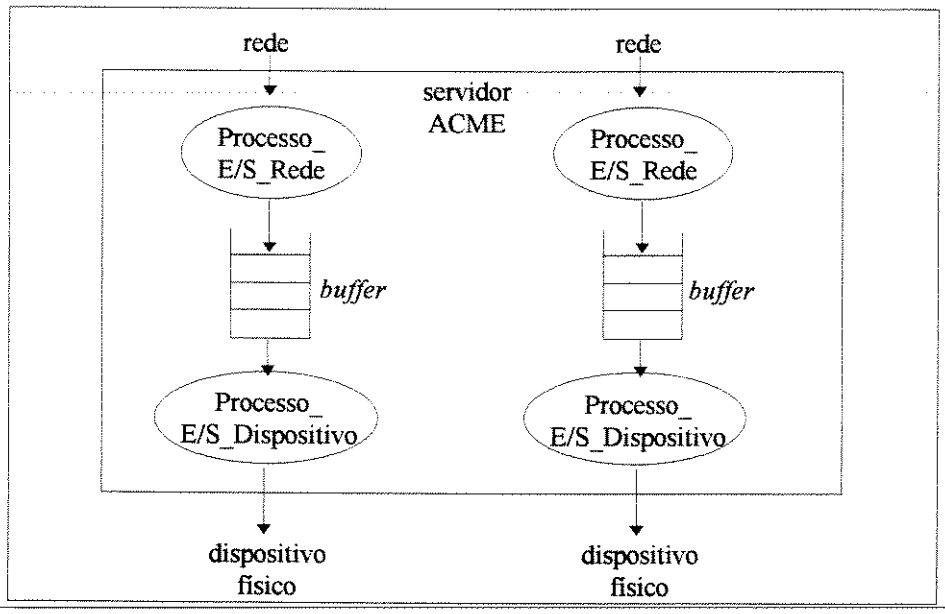

Figura 3.8 - Os objetos que compõem o servidor ACME.

A abstração de sincronização do ACME é chamada de LTS (*Logical Time System*). O LTS permite aos seus clientes especificarem a sincronização de entrada e saída de dados. Para descrever a semântica do LTS, os fluxos de dados das mídias são modelados como uma seqüência de unidades (quadros de vídeos ou unidades de áudio). Cada unidade tem um carimbo de tempo X não-negativo dado pela sua posição temporal relativa ao começo do fluxo (*time-stamps*).

O LTS tem um valor corrente (inicialmente zero) e enquanto estiver rodando, o seu valor corrente aumenta aproximadamente com a mesma taxa do tempo-real. A E/S em cada dispostivo físico é sincronizada com o LTS: a entrada ou a saída de uma unidade de dados com carimbo de tempo X é bem próxima do LTS quando este último alcançar também o valor X.

O avanço do LTS pode ser dirigido segundo um dos dois métodos:

- *dirigido-a-dispositivo:* o LTS avança com a taxa do dispositivo físico mestre D_{mestre}. A E/S neste dispositivo ocorre na sua taxa natural, enquanto que nos outros dispositivos ($D_{escravos}$) pode haver um ajuste. D_{mestre} é tipicamente o dispositivo em que os erros temporais são mais facilmente percebidos (usualmente áudio);

- *dirigido-a-conexão:* o LTS avança de acordo com a taxa de transmissão dos dados de uma conexão especificada. Esta é a escolha apropriada caso se necessite de um baixo atraso, isto porque a velocidade de apresentação acompanha a velocidade de transmissão dos dados, evitando o seu acumulo no *buffer*.

Durante sua operação, o ACME pode se encontrar em duas situações: ou a situação normal, ou a de falta de dados (*starvation*). Durante a operação normal, o casamento das taxas é conseguido com a execução de duas ações: skip() e pause(). Em caso de falta de dados, o avanço do LTS é impedido até que a situação se normalize. Vejamos com maiores detalhes estes dois casos.

3.4.1 Casamento da Taxa

O ajuste das taxas é feito com a execução dos métodos: skip() e pause(). Skip() aumenta a velocidade do dispositivo em relação ao tempo-real, enquanto que pause() a diminui. Para um dispositivo de saída, skip() significa descartar os dados do *buffer*. Para fazer o pause(), não se remove nenhum dado do *buffer*, mas exibe-se dados "nulos" (último quadro no caso de vídeo e nenhum dado no caso de áudio) por um período. Para dispositivo de entrada, o skip() é feito escrevendo-se dados nulos no *buffer*, enquanto o pause() é feito descartando-se a unidade de mídia corrente. Para um LTS dirigido-a-dispositivo, o D_{mestre} não sofre nem skip() nem pause(), estas ações somente são executadas pelos $D_{escravos}$.

O tempo de E/S num dispositivo pode ser representado usando-se um diagrama de passo-escada. Cada caixa representa a E/S de um bloco de dados. As laterais da caixa são o tempo-real no qual a E/S começa e termina. A base e o topo são o começo e o fim do carimbo de tempo. As caixas passo-escada definem se um dispositivo roda na sua taxa nominal ou não (ver Figura 3.9).

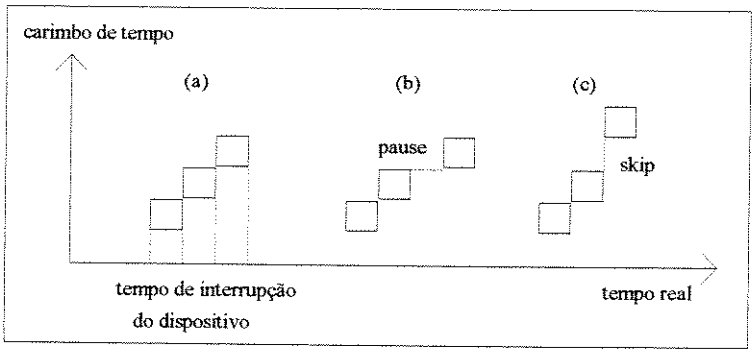

Figura 3.9 - Diagrama de passo-escada para dispositivo: (b) com pause(), (c) skip() ou (a) sem nenhuma ação.

A taxa de um LTS rodando é determinada por um dos métodos descrito anteriormente (dirigido-a-dispositivo ou dirigido-a-conexão), enquanto a taxa de cada dispositivo lógico é determinada pelo dispositivo físico associado. Estas taxas podem

ter uma pequena diferença. Para um LTS dirigido-a-dispositivo (D_{mestre}) o ACME ajusta a taxa realizando skip() ou pause() nos dispositivos escravos, enquanto que a taxa do dispositivo mestre não sofre nenhuma influência.

A cada interrupção de um $D_{escravo}$, o ACME decide se skip(), pause() ou se nenhuma destas ações será necessária. O algoritmo é o seguinte: seja P_{mestre} e $P_{escravo}$ os períodos de interrupções nominais de D_{mestre} e $D_{escravo}$ respectivamente; e T_{mestre} e $T_{escravo}$ os períodos de interrupções real. Seja X a taxa de $D_{escravo}$ relativa a D_{mestre}, isto é:

$$X = (P_{escravo}/T_{escravo}) / (P_{mestre}/T_{mestre}); \qquad \text{[Equação 3.1]}$$

quando $X > 1$, o $D_{escravo}$ está atrasado, e quando $X < 1$ $P_{escravo}$ está adiantado.

Seja $\overline{X}$ o limite superior entre $(X, 1/X)$, então define-se $H = D_{escravo} * \overline{X}$, como sendo uma tolerância de *hardware*. Estendendo-se cada bloco do D_{mestre} por um fator de $H/2$ em cada direção, define-se as regiões R_{exibe}, R_{pause} e R_{skip} (ver Figura 3.10a).

Em cada interrupção do $D_{escravo}$, o ACME verifica qual região contém o canto superior do bloco que está sendo manipulado e reage apropriadamente (ver Figura 3.10b e Figura 3.10c).

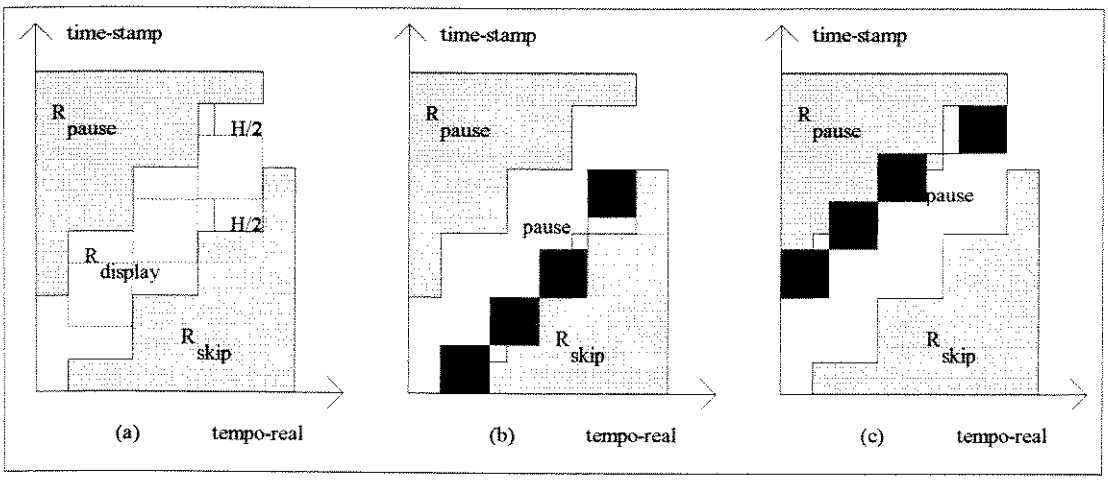

Figura 3.10 - Em (a), representação gráfica do algoritmo skip()/pause() usado para o controle da taxa de um dispositivo físico num LTS dirigido a dispositivo. Os blocos tracejados representam o diagrama de tempo do D_{mestre}. Em (b), $D_{escravo}$ roda mais devagar do que o D_{mestre} e deve eventualmente fazer um skip(); em (c), o $D_{escravo}$ roda mais rápido do que o D_{mestre}, isto levará a um pause().

Se um dispositivo lógico ficar bloqueado devido à falta de dados (*buffer starvation*), ele ficará atrás do LTS. Se esta diferença não exceder a um limite superior especificado pela aplicação (skew_max), procede-se a E/S normalmente; caso

contrário, o LTS fica temporariamente parado: seu valor não avançará e nenhuma E/S será feita no dispositivo lógico associado a ele. Se o dispositivo for de saída, enquanto o LTS ficar parado, os dados serão lidos da conexão até que se tenha recebido unidades suficientes para ajustar o dispositivo lógico ao valor corrente do LTS. Neste ponto, o dispositivo reassume a E/S. Se o dispositivo for de entrada, então dados nulos serão escritos para ajustá-lo ao valor corrente do LTS.

3.5 Plataforma de Sincronização para Aplicações Distribuídas Multimídia

A plataforma de sincronização fornece uma interface única e consistente para o desenvolvimento de aplicações distribuídas multimídia. Os serviços fornecidos por esta interface baseiam-se na integração das funcionalidades dos trabalhos anteriormente apresentados neste capítulo e destinam-se fundamentalmente ao controle, monitoramento e sincronização dos fluxos de mídia contínua envolvidos em uma aplicação multimídia.

A arquitetura proposta está fundamentada nos conceitos de ODP (*Open Distributed Processing*) [Stefani95]. As especificações dos padrões CORBA e COSS [COSS] formam as bases para o suporte aos objetos distribuídos. A plataforma está sendo desenvolvida com o uso do *software* ORBIX que é uma implementação do CORBA.

O assunto abordado aqui terá o intuito de especificar o contexto em que o sincronizador foi desenvolvido, uma vez que maiores detalhes sobre partes específicas podem ser encontrados em outros trabalhos de implementação em desenvolvimento.

Dividindo-se a plataforma em níveis, encontra-se, no mais inferior, as funcionalidades oferecidas pelo ACME e que se destinam à sincronização intermídia de granulosidade fina. Acima deste, estão os serviços desenvolvidos pelo MSS para o controle/monitoramento do fluxo. E no nível superior estão os serviços de Orquestração com a responsabilidade de determinar as conexões físicas e traduzir a QoS especificada pela aplicação em valores significativos para o sistema (ver Figura 3.11).

A aplicação comunica-se com a plataforma a partir do Gerente_Orquestração que aparece para a aplicação como uma interface de tipos de dados abstratos, assim como em [Blair92] e [Campbell]. Este é responsável por estabelecer as conexões físicas entre os nós envolvidos e negociar a QoS entre a aplicação e o sistema.

O Orquestrador_Alto_Nivel (OAN) faz um mapeamento dos requisitos de QoS que foram negociados entre a aplicação e o Gerente_Orquestração para os parâmetros significativos para o sistema [Shepherd92]: variação do atraso, taxa de transmissão, atraso fim-a-fim, taxa de pacotes perdidos e taxa de *bits* errados. Estabelece, também, as políticas de sincronização e monitora o fluxo através da Conexão_Virtual. Este monitoramento é feito com a utilização de um dos métodos descritos na interface Stream. O interesse do OAN é verificar a freqüência com que ocorre o desalinhamento entre os fluxos. Caso este esteja acima dos limites determinados pela aplicação, deve-

se informar ao Gerente_Orquestração para que este volte a entrar em negociação com a aplicação.

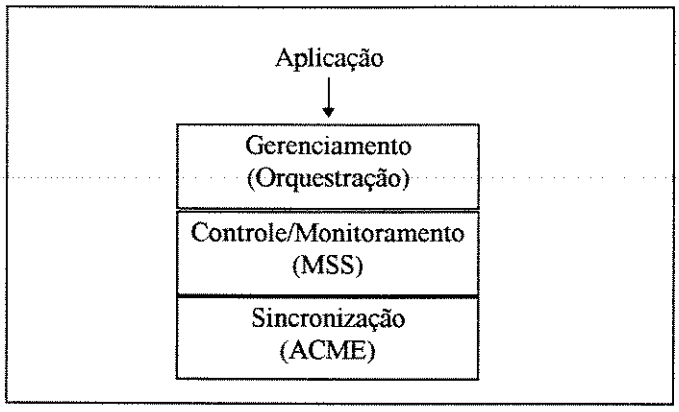

Figura 3.11 - Arquitetura em camadas da plataforma de sincronização.

A Conexão_Virtual realiza o controle e monitoramento do fluxo a um nível mais baixo do que o OAN. Ela agrupa todos os objetos envolvidos pelo sistema desde a captura até a apresentação da mídia. O Grupo permite que, dependendo do tipo de serviços requisitados, suas ações sejam repassadas aos seus componentes. Como exemplo pode-se citar que, uma requisição de parada do progresso do fluxo em um grupo de objetos é feita com a chamada de um método da interface Stream da Conexão_Virtual que imediatamente a repassa para todos os componentes do grupo. A utilização de grupos também proporciona uma maior facilidade quando da definição da QoS para todo um grafo. O acompanhamento do fluxo feito pela Conexão_Virtual é feito pela reportagem de eventos dos objetos que realizam E/S dados na rede (Processo_E/S_Rede).

A Conexão_Virtual também é responsável por negociar e definir um padrão de transmissão de dados que esteja de comum acordo entre os Processo_E/S_Rede de cada nó. Isto porque cada processo pode ser capaz de transmitir dados em vários formatos diferentes.

O Sincronizador é o objeto responsável pela sincronização paralela de granulosidade fina. Pode suportar várias políticas, como por exemplo, carimbos de tempo e *buffers* de sincronização, inserção ou deleção de unidades de dados e aumento ou diminuição da velocidade de apresentação das mídias. A escolha destas políticas é feita pelo OAN e a ele deve ser reportado, na forma de eventos, o sucesso ou a falha na execução de suas tarefas. É no monitoramento destes eventos e dos eventos gerados pela Conexão_Virtual que o OAN deve tomar suas decisões.

Os objetos Processo_E/S_Rede e Processo_E/S_Dispositivo realizam a entrada e saída de dados no sistema e são, portanto, os dispositivos virtuais definidos pelo ACME, adicionando a estes uma interface Stream para se comunicarem com os demais objetos.

Toda a comunicação entre os objetos é realizada através de suas respectivas interfaces Stream assim como está definido no MSS (ver Figura 3.12).

A área branca definida pela Figura 3.12 representa o Servidor de Sincronização Multimídia (SESIM) que foi implementado neste trabalho. O SESIM vem dar suporte à sincronização de granulosidade fina que é prevista mas não definida nem realizada pelo MSS, mas que pode ser controlado através de suas interfaces. No próximo capítulo apresentar-se-á como ele foi implemento e os resultados obtidos.

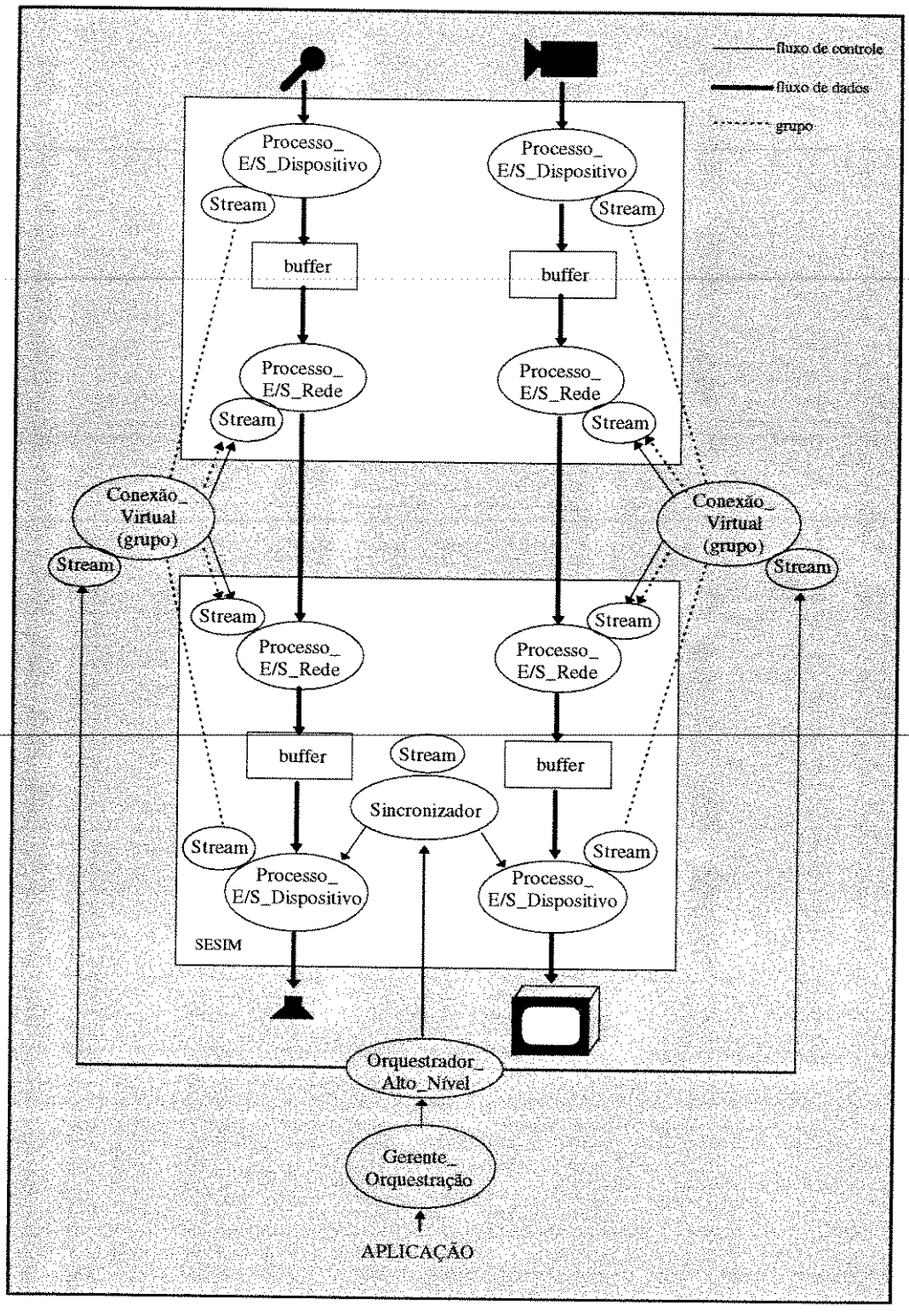

Figura 3.12 - Arquitetura do SESIM mostrando as classes que o compõem.

Capítulo 4

O SERVIDOR DE SINCRONIZAÇÃO MULTIMÍDIA (SESIM)

4.1 Introdução

No capítulo anterior foi apresentada uma plataforma capaz de controlar e monitorar os fluxos de mídia, e de definir e renegociar a QoS entre a aplicação e os recursos do sistema. As tarefas realizadas pela plataforma estão distribuídas em várias camadas de acordo com a funcionalidade de cada uma. A camada mais inferior é a responsável pela sincronização dos fluxos envolvidos em uma aplicação.

Sincronização, no conceito de multimídia, é o termo empregado para designar os mecanismos que controlam a ordenação temporal dos processos no acesso aos dispositivos físicos [Steinmetz90]. Dentro deste contexto, o SESIM (Servidor de Sincronização Multimídia) é o processo encarregado de determinar a ordem e o momento de execução dos processos que estão envolvidos na apresentação das mídias. Seu funcionamento foi baseado no ACME.

As implementações das funcionalidades definidas no MSS estão sendo desenvolvidas em outros trabalhos que pertencem a esse projeto [Lopes96].

O capítulo inicia-se com a exposição do *hardware* utilizado e do MMPM/2, *software* utilizado para o controle dos dispositivos físicos. Em seguida, descrever-se-á a implementação do SESIM, terminando com a apresentação dos resultados obtidos.

4.2 O *Hardware* Utilizado

O SESIM foi implementado em computadores IBM PS/2 (CPU 486, 33MHz, 16Mb de memória RAM) rodando o sistema operacional IBM OS/2 v2.1. Foram usados dois computadores ligados em rede, podendo-se escolher entre a FDDI ou Ethernet (com as duas foram feitos testes). Em ambos os casos o protocolo de comunicação utilizado foi o TCP/IP. Complementam a configuração do *hardware* os seguintes dispositivos:

- uma placa *M-Audio* capaz de capturar/reproduzir áudio no formato .WAV [Microsoft];
- um *kit M-Box* com alto-falantes e microfone que possibilita controlar volume, nível de gravação, nível de agudos e balanço dos canais de som;
- uma câmera de vídeo e uma placa *ActionMedia II* [ActionMediaII], que gera arquivos de vídeo no formato .AVS [Microsoft].

Para o controle de ambas as placas, *M-Audio* e *ActionMedia II*, optou-se por utilizar um *software* que estivesse disponível no mercado e que facilitasse a implementação com interfaces independentes do dispositivo a ser controlado, liberando, assim, a implementação de se preocupar com detalhes específicos de cada *hardware*. Por estas características o *software MMPM/2* (*Multimedia Presentation Manager Toolkit/2 v1.1*) [MMPM/2] foi escolhido. Entretanto, dificuldades surgiram entre este *software* e a placa *ActionMedia II*.

4.3 A Placa *ActionMedia II* e o *Software* MMPM/2

O MMPM/2 é uma ferramenta que permite aos desenvolvedores de aplicações tornarem-se mais produtivos em relação ao desenvolvimento de programas multimídia para o OS/2. Esta facilidade é provida através de uma interface, chamada de MCI (*Media Control Interface*), que fornece às aplicações serviços para o envio de mensagens de controle aos dispositivos físicos multimídia através das funções de uma API (*Application Programming Interface*). A utilização de tais interfaces favorece o desenvolvimento de aplicações independentes do *hardware* e da mídia envolvidos [ver Apêndice B].

O MMPM/2 seria utilizado para controlar as placas *M-Audio* e *ActionMedia II*. A implementação da captura, transmissão e reprodução do áudio foi realizada sem grandes dificuldades. Entretanto, problemas começaram a surgir quando verificou-se que o *software* não seria capaz de trazer a imagem capturada pela placa para o espaço de endereçamento do SESIM. Toda ação de manipulação da placa era realizada por uma API que deixava este *buffer* transparente (sem acesso) para o usuário. Apesar de

existir no manual, a função que deveria realizar tal tarefa (MCL_GETIMAGEBUFFER()) não estava implementada.

Após algum tempo gasto, sem sucesso, na tentativa de se contornar tal problema, contactou-se a IBM para que sugerisse uma solução. A IBM forneceu, então, o *software AVK* (*Audio Video Kernel*) [ActionMediaII]. Mas, novamente, problemas surgiram. O AVK era uma versão em 16-bits e o compilador utilizado, IBM C/C++ Tool v2.01, era uma versão 32-*bits*. Tentou-se, então, adquirir uma nova versão 32-*bits* deste *software*, o que não foi possível uma vez que a IBM interrompeu o seu desenvolvimento. A alternativa restante foi encontrar um compilador que gerasse código em 16-*bits*. Entretanto, esta era uma versão antiga, não mais fornecida pela empresa.

Uma vez concluídas todas as alternativas possíveis, verificou-se, infelizmente, que não seria possível trabalhar com o vídeo. A solução encontrada foi a sua substituição por texto. A sincronização seria feita, então, com fluxos de caracteres (texto) e áudio. Do ponto de vista da validação, esta substituição em nada influencia os resultados, uma vez que a sincronização deve funcionar independentemente dos fluxos e das mídias com as quais esteja trabalhando.

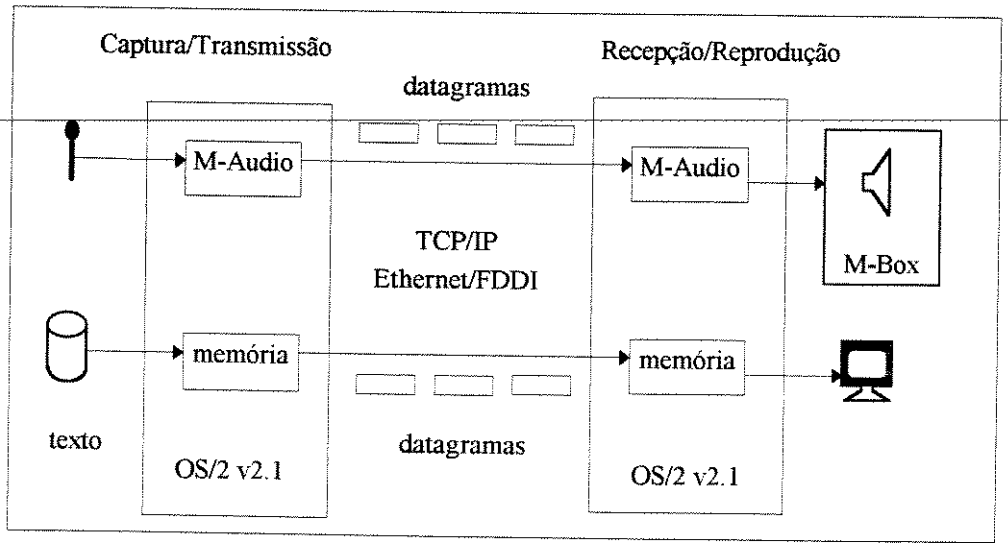

Figura 4.1 - Configuração física do SESIM.

Assim sendo, o sistema ficou da seguinte forma: do lado da captura/transmissão, um microfone gera o sinal de áudio para a placa *M-Audio* a partir de uma voz que lê um texto que é apresentado em uma janela no monitor deste nó. Texto este que se encontra armazenado em um arquivo em disco. O texto é recuperado com uma certa freqüência, ou seja, a cada 93ms apresenta-se um caracter no monitor. Esta freqüência foi determinada de modo que possibilitasse uma leitura tranquila do texto à medida em que estivesse sendo apresentado. Estas duas mídias, caracteres e voz, são transmitidas através de datagramas para um outro computador semelhante. Neste lado, da recepção/reprodução, ocorre o inverso, o som é recebido e enviado à placa *M-Audio* que o reproduz através da *M-Box*; e o texto é exibido no

monitor. Duas redes diferentes foram empregadas para testar o sistema: uma FDDI e uma Ethernet. A transmissão foi realizada utilizando-se datagramas (ver Figura 4.1).

4.4 Implementação

O SESIM foi implementado em linguagem C, compilador IBM C/C++ Tool v2.01, utilizando o ambiente PM (*Presentation Manager*) (ver Apêndice B) onde foi desenvolvida a interface homem-máquina. O controle da execução, início e parada, assim com a exibição, para leitura, dos caracteres de texto no nó de captura/transmissão e a respectiva reprodução no outro nó ficaram disponíveis através da interface mostrada na Figura 4.2.

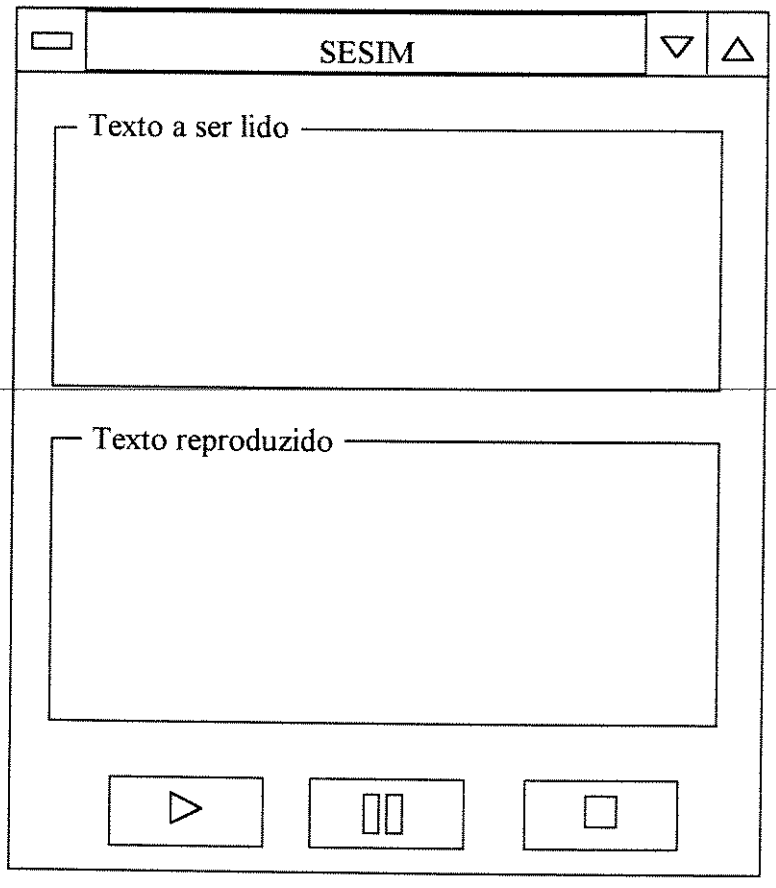

Figura 4.2 - Interface homem-máquina do SESIM.

A arquitetura do SESIM é a seguinte: no nó de apresentação, quatro *threads* realizam a entrada e saída de dados no sistema. Uma quinta *thread*, disponível apenas no nó de recepção/reprodução, realiza o sincronismo.

Uma *thread* pode ser vista como uma função que é executada paralelamente com outras funções. As *threads* são: duas Dispositivo_Virtual_Rede, duas Dispositivo_Virtual_Mídia e a *thread* Sincronizador (ver Figura 4.3). Embora a possibilidade de se trabalhar com *threads* seja uma ferramenta importante fornecida pelo OS/2, isto

não basta. É preciso obter o controle da ordem de execução de cada uma. Para isto, o OS/2 define vários níveis de prioridade que definem uma hierarquia de execução [ver Apêndice A].

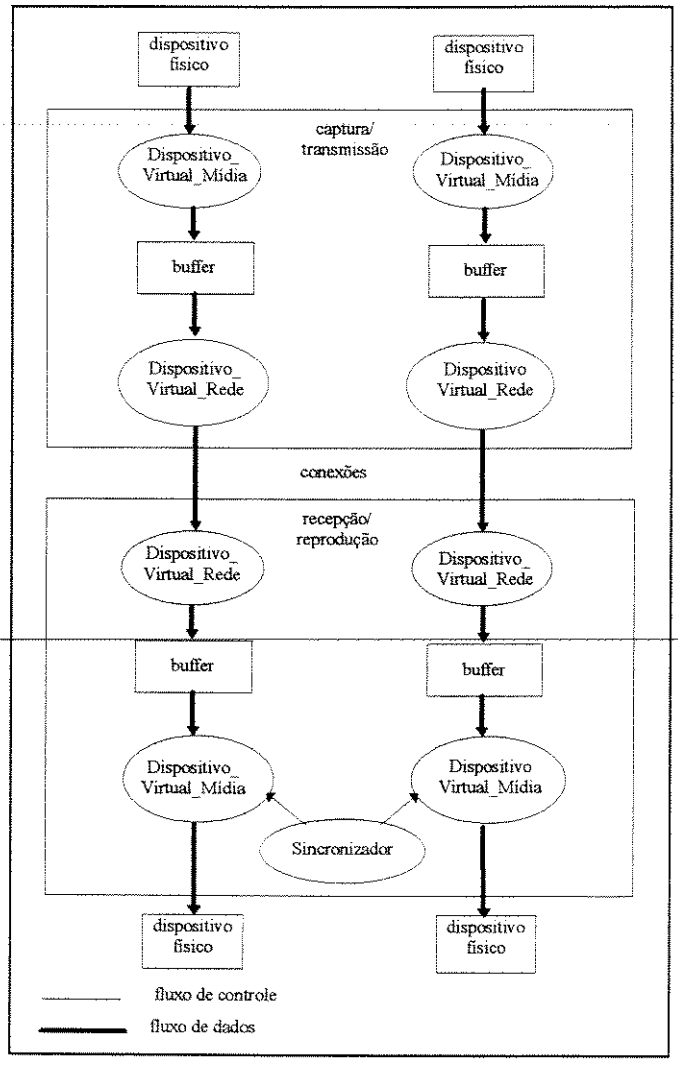

Figura 4.3 - Os processos, fluxos de dados e fluxos de controle do SESIM.

Embora o SESIM seja compreendido tanto pelas *threads* que estão no nó de captura/transmissão quanto pelas que estão no nó de recepção/reprodução, de agora em diante, toda vez que se fizer uma referência ao SESIM (salvo menção contrária), está-se referindo somente às *threads* da recepção/reprodução, pois somente estas sofrem influência do Sincronizador. As *threads* de captura/transmissão têm todas a mesma prioridade e ficam rodando em *loop* infinito competindo livremente pela CPU. Internamente, o funcionamento de cada uma é análogo à sua repectiva *thread* do nó recepção/reprodução, apenas manipulando o fluxo no sentido inverso.

O Dispositivo_Virtual_Rede realiza a E/S de dados da rede. Um processo de saída recebe os dados da rede e os transfere para o *buffer* (nó de captura/reprodução). Um processo de entrada faz o contrário. Estas *threads* ficam rodando em *loop* infinito e possuem prioridade dois.

As *threads* Dispositivo_Virtual_Midia transferem os dados do *buffer* para os dispositivos físicos (nó de recepção/reprodução), ou vice-versa (nó de captura/transmissão). As *threads* do nó de recepção/reprodução permanecem dormindo até serem ativadas pelo Sincronizador, e como possuem prioridade três, a maior do sistema, elas entram imediatamente em execução. Há, desta forma, a garantia de que nunca existirão dados esperando para serem exibidos.

O *buffer* é uma memória compartilhada entre os dispositivos virtuais. Sua funcionalidade é armazenar os dados devido à variação do atraso da rede. Do lado da captura/transmissão armazena-os até que a rede possa transmiti-los e, do lado da recepção/reprodução, até que chegue o momento de reproduzi-los.

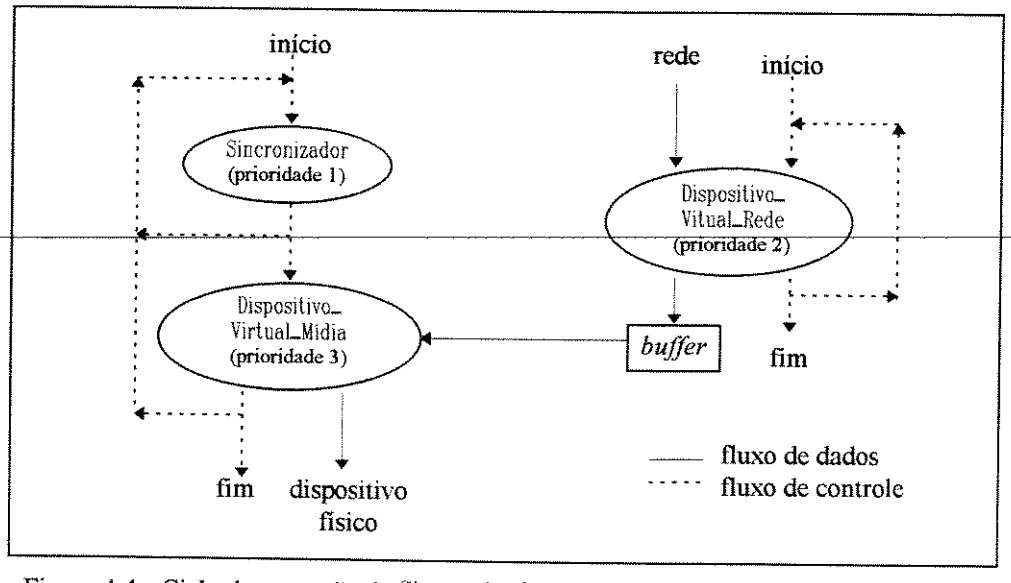

Figura 4.4 - Ciclo de execução do Sincronizador.

O Sincronizador possui a menor prioridade do sistema, um. Entretanto, jamais ficará bloqueado, aliás, este é o processo que mais ocupa a CPU. Isto ocorre porque, ao receber um dado da rede, o Dispositivo_Virtual_Rede envia-o para o *buffer* e volta a dormir. Neste momento, o Sincronizador entra em execução e percebe a chegada do novo dado. Então, verifica se ele pode ser reproduzido. Em caso negativo ele é descartado ou fica aguardando a sua vez. Caso contrário, ativa o Dispositivo_Virtual_Midia, que devido à sua maior prioridade, entra imediatamente em execução e, após terminar a reprodução, envia um aviso ao Sincronizador e volta a dormir. O Sincronizador volta a ocupar a CPU, mas não realiza nenhuma ação até a chegada de um novo dado. E aí, o ciclo recomeça novamente (ver Figura 4.4).

4.4.1 Algoritmos de Sincronização

No ACME, as unidades de mídia[1] são marcadas com um carimbo de tempo e devem ser reproduzidas o mais próximo possível de um relógio lógico global (*Logical Time System - LTS*) que avança, aproximadamente, com a mesma velocidade do tempo real. Este avanço pode ser dirigido-a-conexão ou dirigido-a-dispositivo. As ações corretivas são realizadas através das funções skip() ou pause().

No SESIM, o carimbo de tempo referente à chegada da unidade é armazenado em um vetor cujo índice representa o número da unidade. A implementação do LTS feita desta forma permite o sincronismo tanto do índice quanto do carimbo de tempo. A sincronização é dirigida-a-conexão (os dados são exibidos à medida que chegam no nó destino) e pode haver a eleição de um fluxo mestre ou não. O fluxo mestre não deve sofrer nenhuma influência do Sincronizador, ele é exibido à medida que seu fluxo chega ao receptor. A preocupação de sincronização fica apenas com os fluxos escravos que seguem estritamente a reprodução do mestre.

Sem a utilização de um mestre, a reprodução é mais flexível: qualquer um dos dois pode continuar a ser reproduzido caso o outro se atrase. O fluxo atrasado não é exibido até que consiga novamente alcançar o mais adiantado.

As ações corretivas têm a mesma funcionalidade do ACME e a abstração de sincronização é baseada na definição, pelo cliente, de dois parâmetros:

- *desvio em números de unidades* (Skew_E): duas unidades de mídias diferentes, que chegam juntas ao destino, não podem ser exibidas caso a diferença ordinal entre elas ultrapasse o valor Skew_E. Em outras palavras, mede a defazagem em relação ao número de unidades; e
- *desvio temporal* (Skew_D): duas unidades de mídias diferentes que possuem o mesmo carimbo de tempo não podem ser exibidas caso a defazagem temporal entre elas seja maior do que o valor Skew_D, ou seja, mede a defazagem temporal entre duas unidades.

Estes dois parâmetros são definidos pela aplicação e têm influência significativa no resultado. Utilizando-se uma representação espacial, estes dois parâmetros são colocados em dois eixos ortogonais. O eixo das ordenadas representa a defasagem temporal em *ms*, e o eixo das abcissas, a defazagem em número de unidades (ver Figura 4.5).

No Sincronizador estes parâmetros são mapeados nas seguintes variáveis:

- TT (Topo do Texto): quantidade total de unidades de texto recebidas;
- ET (Exibição do Texto): quantidade total de unidades de texto exibidas;
- TA (Topo do Áudio): quantidade total de unidades de áudio recebidas;
- EA (Exibição do Áudio): quantidade total de unidades de áudio exibidas;

[1]Unidade de mídia é a unidade elementar de uma mídia definida pela sistema, podendo ser uma amostra de áudio, um quadro de vídeo, um gráfico, etc.

- relógio: contém a hora atual do sistema (centésimos de segundo);
- HoraExibiçãoÁudio[ ]: vetor onde é armazenado a hora em que cada unidade de áudio é exibida, ou seja, é o carimbo de tempo das unidades de áudio;
- HoraExibiçãoTexto[ ]: vetor onde é armazenado a hora de exibição de cada unidade de texto, ou seja, é o carimbo de tempo das unidades de texto.

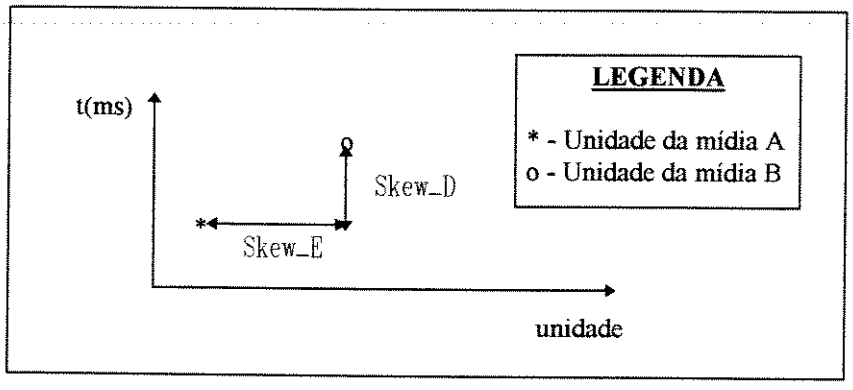

Figura 4.5 - Os parâmetros Skew_D e Skew_E como eixos ortogonais.

Para ser reproduzida, uma unidade de número X da mídia A necessita satisfazer duas condições:

1) a diferença entre seu carimbo de tempo e o da unidade X da mídia B deve ser menor do que Skew_D unidades de tempo (HoraExibiçãoMídiaA[X] - HoraExibiçãoMídiaB[X] < Skew_D);
2) a diferença entre X e Y deve ser menor do que Skew_E, sendo Y a última unidade reproduzida da mídia B (X - Y < Skew_E).

O sistema assume que não há perdas de pacotes na rede[1] e que não há atraso na captura das unidades[2]. Desta forma, as unidades chegam sequencialmente e toda variação temporal se deve aos atrasos na rede. Cada unidade recebe o seu carimbo de tempo no momento em que é recebida[3], e caso não houvesse atraso, o carimbo de tempo da unidade X da mídia A seria igual ao carimbo de tempo da unidade X da mídia B. Como isto nem sempre é verdadeiro, faz-se necessário a interferência do Sincronizador, que poderá casar os fluxos segundo a adoção de uma mídia mestre ou não.

[1]Isto deve ser suportado pelo protocolo de comunicação.

[2]Quando da aquisição de recursos, deve-se ter a garantia da CPU de que poderá atender aos requisitos de tempo da aplicação.

[3]Como é assumido que não há perda e nem alteração na ordem de chegada dos pacotes, estes já possuem, implicitamente, um carimbo de tempo referente ao momento da captura.

4.4.1.1 Sincronismo com Fluxo Mestre

O algoritmo de sincronização com fluxo mestre funciona da seguinte forma: o Sincronizador fica rodando em *loop* até a chegada de uma unidade. Caso seja uma unidade de áudio, as variáveis TA[1] e EA são incrementadas e o Sincronizador sinaliza a liberação da *thread* Dispositivo_Virtual_Mídia do áudio que entra imediatamente em execução devido à sua maior prioridade.

Quando da chegada de uma unidade de texto é necessário verificar qual a sua posição em relação à última unidade de áudio reproduzida. O áudio jamais sofrerá interferência do Sincronizador, entretanto, o texto, por ser o fluxo escravo, pode estar atrasado ou a frente do áudio. Neste último caso, simplesmente a reprodução de sua unidade terá que esperar pela do áudio. Isto é feito igualando-se as variáveis de exibição do texto e de áudio (ET=EA). Como a reprodução do texto está à frente da do áudio, esta atribuição impedirá o progresso da reprodução do texto. Caso esteja atrasado, duas alternativas são possíveis:

1) o texto continuar atrasado: pode ter acontecido a chegada de unidades de texto, mas em quantidade insuficiente para alcançar a reprodução do áudio e, assim sendo, não devem ser reproduzidas;

2) ocorrer a chegada de uma rajada de unidades de texto como consequência de um descongestionamento da rede. Neste caso atribui-se a quantidade total de unidades de texto que já chegaram no nó de reprodução à variável de exibição do texto (ET=TT). Isto permitirá que a reprodução do texto alcance a do áudio. Entretanto, ainda será necessário verificar se a unidade encontra-se dentro dos limites de reprodução, ou seja:

   - a defasagem numérica entre a unidade de texto e a última unidade de áudio reproduzida não deve ser maior do que Skew_E (EA - ET > Skew_E); e
   - a defasagem temporal entre a unidade de texto e a respectiva unidade de áudio reproduzida deve ser menor do que Skew_D (Relógio - HoraExibiçãoAudio[ET]) (ver Figura 4.6).

[1] Variável incrementada pela *thread* Dispositivo_Virtual_Rede.

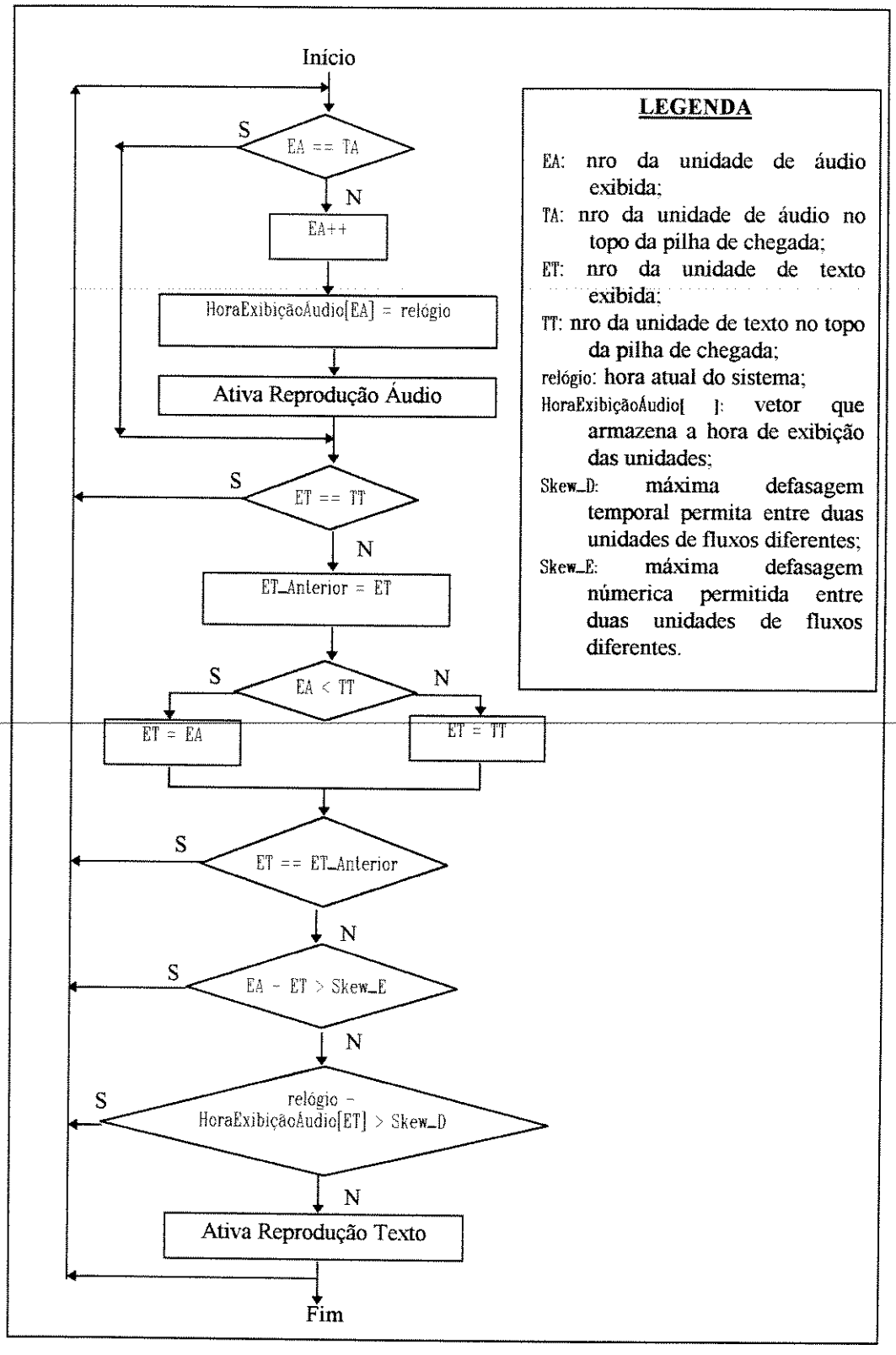

Figura 4.6 - Fluxograma da tomada de decisão do Sincronizador com fluxo mestre.

4.4.1.2 Sincronismo sem Fluxo Mestre

A sincronização sem a eleição de um fluxo mestre pode ser desejada por aplicações que não queiram deter o progresso de um fluxo em detrimento de um outro que esteja atrasado. Quando qualquer um dos fluxos se atrasar, o outro pode continuar seu avanço até os limites especificados pela aplicação. Se o fluxo conseguir se recuperar antes que este limite seja alcançado, o sincronismo será retomado sem ocorrer a interrupção do fluxo que estava com sua taxa de apresentação normal.

Quando em operação, Sincronizador poderá se encontrar em uma das três situações abaixo:

1) *Situação Normal*: é a situação que ocorre quando áudio e vídeo chegam intercalados de maneira tal que a defazagem temporal e numérica entre suas unidades não exceda os valores de Skew_D e Skew_E respectivamente. Para cada unidade de, por exemplo, áudio, o Sincronizador verifica se ela não está Skew_E unidades à frente das de texto (EA - ET < Skew_E) (se estiver, a situação é dada como atraso do texto). Após este, é feito o teste temporal, ou seja, caso a unidade de áudio esteja atrasada, é necessário verificar se já não passou tempo suficiente para a situação ser considerada como atraso do áudio. Para não ser considerada atrasada, o tempo decorrido depois da reprodução da respectiva unidade de texto não deve ser maior do que Skew_D (relógio - HoraExibiçãoTexto[EA] ≥ Skew_D).

2) *Atraso do Texto*: durante o tempo em que o texto permanecer atrasado, as unidades de áudio continuam sendo reproduzidas enquanto a defasagem numérica entre as unidades EA e ET for menor do que Skew_E (EA - ET < Skew_D). Quando esta situação não mais for verdadeira, o sistema pára até que novas unidades de texto cheguem. Como a exibição das unidades de áudio está à frente do topo das unidades de texto (EA > TT), quando da chegada de unidades de texto pode ocorrer uma das duas situações abaixo:

   - *Chegada de unidades de texto suficientes para alcançar o áudio*: neste caso, faz-se a variável de exibição de texto igual à variável de exibição do áudio (ET=EA), e a reprodução volta à situação normal;
   - *Chegada de unidades de texto insuficientes para alcançar o áudio*: faz-se ET++ e verifica se o atraso atual é menor do que o permitido, ou seja, se a defasagem entre o relógio e o momento em que foi exibida a respectiva unidade de áudio não ultrapassou o valor Skew_D (Relógio - HoraExibiçãoAudio[ET] ≥ Skew_D).

   Dependendo da aplicação, é desejável definir valores para Skew_E de forma a permitir que o fluxo que esteja à frente não seja interrompido devido ao atraso de outro fluxo. Esta situação é promovida definindo-se Skew_E com valores suficientemente grandes quando comparado aos atrasos médios ocorridos na rede;

3) *Atraso do Áudio*: analogamente ao que ocorre com o fluxo do texto, ocorre também com o fluxo de áudio, devido à não diferenciação entre fluxo mestre e escravo.

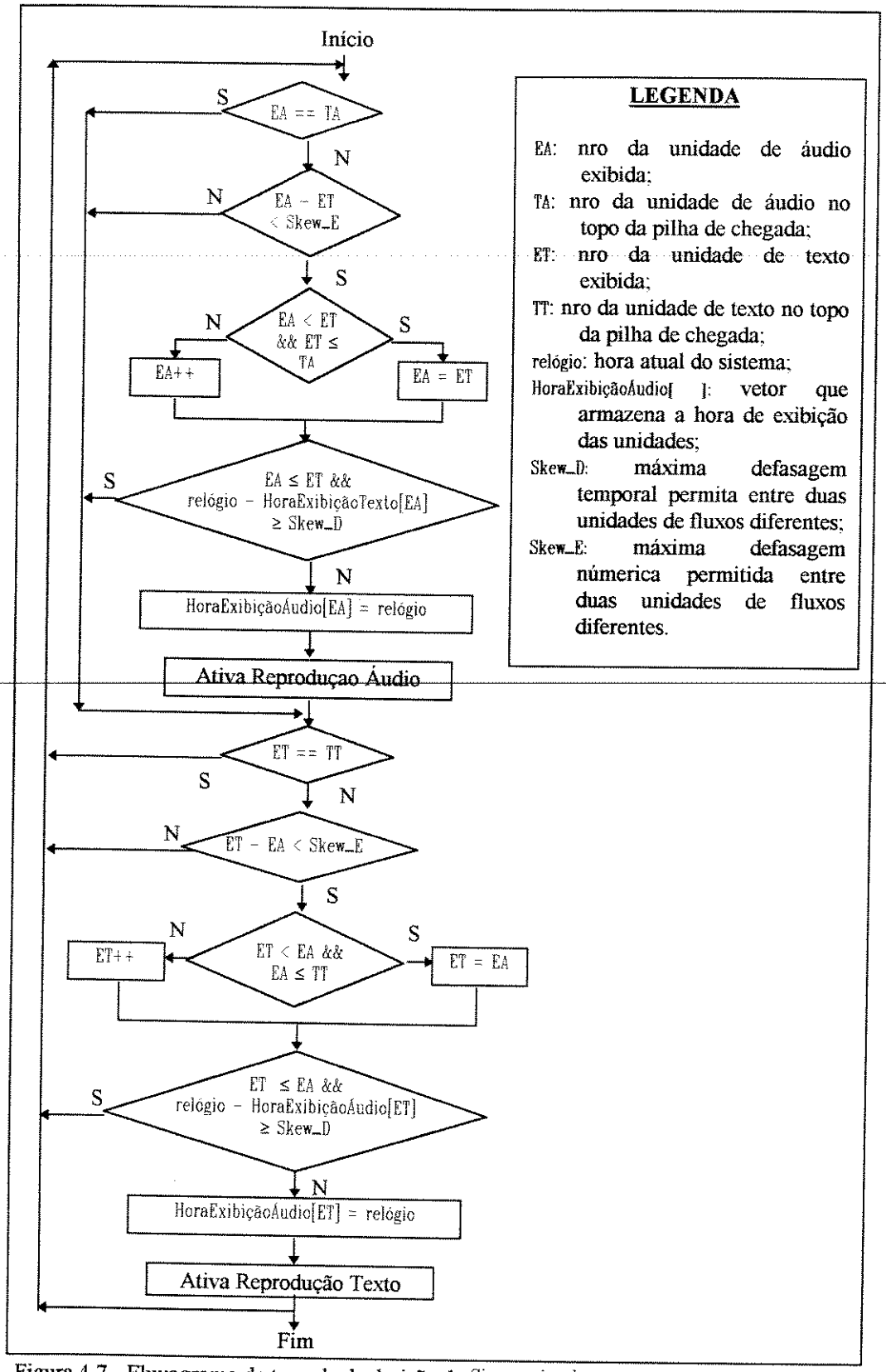

Figura 4.7 - Fluxograma da tomada de decisão do Sincronizador sem fluxo mestre.

A situação de atraso somente é considerada de um fluxo em relação ao outro. Quando áudio e texto se atrasam, mas de forma que a diferença entre eles se mantenha abaixo dos parâmetros Skew_E e Skew_D, considera-se a *situação normal*. Caso contrário, sempre haverá um fluxo mais atrasado do que o outro, e então, o comportamento recai em uma das situações anteriores (ver Figura 4.7).

4.5 Resultados Obtidos

A taxa de amostragem do áudio foi de 11025 vezes por segundo. Foi observado, na prática, que o melhor tamanho das unidades capturadas para a sincronização foi de 370ms. Entretanto, o envio de comandos à placa, e a transmissão das unidades gastavam cerca de 90ms, durante os quais não era capturado áudio. Tem-se então que a cada 460ms era capturada uma unidade de áudio de 370ms.

O tamanho da unidade, 370ms, foi o menor tamanho conseguido utilizando-se os recursos disponíveis. Aumentando-se este valor, o atraso entre a captura e reprodução também era aumentado na mesma quantidade, criando-se um eco entre os dois sistemas. E, o mais importante, aumentava muito a granulosidade temporal da sincronização: sincronizar um número maior de unidade menores é mais complexo do que sincronizar um número menor de unidades maiores.

Para simplificar o algoritmo de sincronização, a cada unidade de áudio, fez-se corresponder uma de texto. O tamanho destas unidades, que também foi encontrado na prática, deveria, então ser de 5 caracteres. O tempo gasto para lê-los é aproximadamente 370ms.

Para que os resultados da sincronização entre estas unidades pudessem ser verificados, atrasos, da ordem de grandeza do tamanho das unidades, foram inseridos na rede de forma aleatória mas única para todos os testes. Com isto, manteve-se os atrasos fixos e variou-se os parâmetros de sincronização para que as alterações sofridas fossem consequência somente destes últimos (Skew_E e Skew_D). Esta metodologia foi utilizada apenas para efeito de estudo dos resultados, uma vez que há uma equivalência entre fixar um e variar o outro. Inclusive, este comportamento irá permitir que as camadas superiores ao SESIM redefinam Skew_E e Skew_D, quando ocorrer variações consecutivas de atraso na rede, com o intuito de manter a QoS estabelecida pela aplicação.

A validação do SESIM foi sistematizada em uma tabela de testes (ver Tabela 4.1) que pudesse comprovar sua eficiência na ocorrência de pertubações na rede. Percorrendo-se a tabela na horizontal, encontra-se testes utilizando-se uma rede FDDI com pouco tráfego[1], uma rede FDDI com tráfego normal[2]e uma rede Ethernet com pouco tráfego. Em cada uma destas redes foi analisada a diferença existente entre a sincronização *com* e *sem* o fluxo mestre.

---

[1]Testes realizados durante os horários em que não havia outros usuários a rede, ou seja, somente tráfego gerado pelo sistema.

[2]Testes realizados durante os horários em que a rede era compartilhada por outros usuários.

Na vertical, encontram-se uma combinação de testes feitos *com* e *sem* sincronismo; *com* e *sem* atraso; *com* e *sem* pertubações na rede; e com a variação dos parâmetros Skew_D e Skew_E de forma a tornarem visíveis as diferenças ocorridas durante a sincronização devido às variações destes.

A Tabela 4.1 possui oito linhas e seis colunas com um total, consequentemente de 48 células. Em cada célula foram realizados três testes: o primeiro com perturbação somente do áudio, o segundo com perturbação somente do texto; e o terceiro com perturbação dos dois fluxos ao mesmo tempo. Portanto, cada tabela contém 144 testes e foram feitas, pelo menos, duas tabelas completas como esta, mais alguns testes isolados. Os resultados mostrados a seguir, foram escolhidos de modo a enfatizar o uso da sincronização e a mostrar as diferenças ocorridas com a variação dos parâmetros Skew_D e Skew_E.

Tabela 4.1 - Testes realizados.

TABELA DE TESTE	FDDI - com pouco tráfego		FDDI - c/ tráfego normal		ETHERNET com pouco tráfego	
	com mestre (A)	sem mestre (B)	com mestre (C)	sem mestre (D)	com mestre (E)	sem mestre (F)
(1) sem atraso sem sincronismo	A1	B1	C1	D1	E1	F1
(2) sem atraso Skew_E=5 - Skew_D=5000	A2	B2	C2	D2	E2	F2
(3) com atraso sem sincronismo	A3	B3	C3	D3	E3	F3
(4) com atraso Skew_E=1 - Skew_D =200	A4	B4	C4	D4	E4	F4
(5) com atraso Skew_E=1 - Skew_D =700	A5	B5	C5	D5	E5	F5
(6) com atraso Skew_E=3 - Skew_D=5.000	A6	B6	C6	D6	E6	F6
(7) com atraso Skew_E=5 - Skew_D=700	A7	B7	C7	D7	E7	F7
(8) com atraso Skew_E=5 - Skew_D =200	A8	B8	C8	D8	E8	F8

4.5.1 Sincronização sem Fluxo Mestre

O Gráfico 4.1 demonstra uma corrida realizada na Ethernet sem fluxo mestre, sem atraso e sem sincronismo. Já o Gráfico 4.2 é semelhante ao Gráfico 4.1, apesar de ter sido feito a sincronização. Esta semelhança , enfatiza uma situação ideal onde, não havendo atraso da rede, não há a necessidade de sincronização. Em ambos os casos, as unidades de áudio e texto são capturadas, transmitidas e reproduzidas sem que ocorra nenhuma pertubação nos seus fluxos.

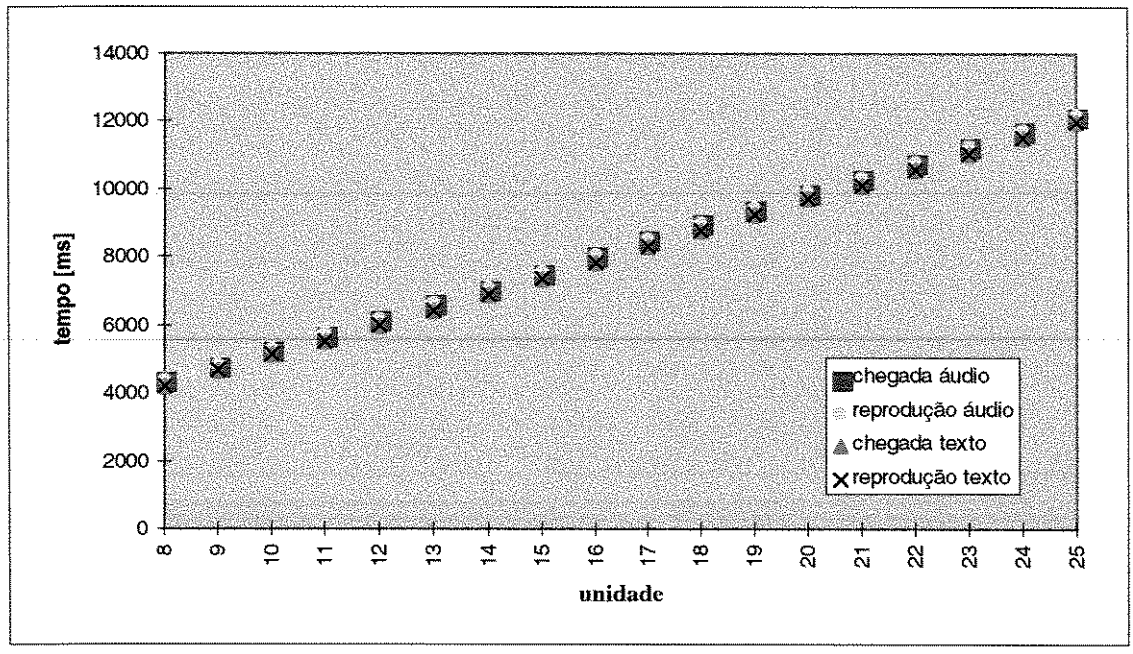

Gráfico 4.1 - FDDI com pouco tráfego, sem fluxo mestre, sem atraso e sem sincronismo (tabela B1).

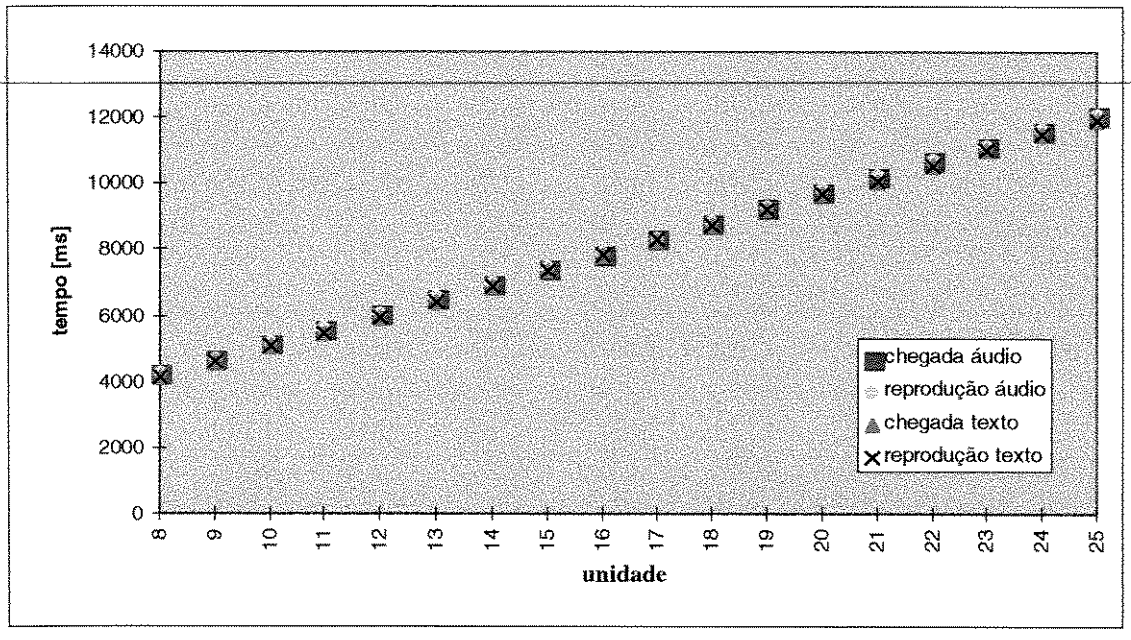

Gráfico 4.2 - FDDI com pouco tráfego, sem mestre, sem atraso e com sincronismo (tabela B2).

Entretanto, se provocada uma perturbação na rede tem-se o Gráfico 4.3, onde é possível notar que entre os intervalos 5.500ms e 9.200ms houve uma defasagem temporal entre as unidades de vídeo e áudio. Logo após, contudo, a rede conseguiu enviar as unidades atrasadas, o sistema as reproduziu com uma defasagem mínima, voltando à sua taxa normal de operação.

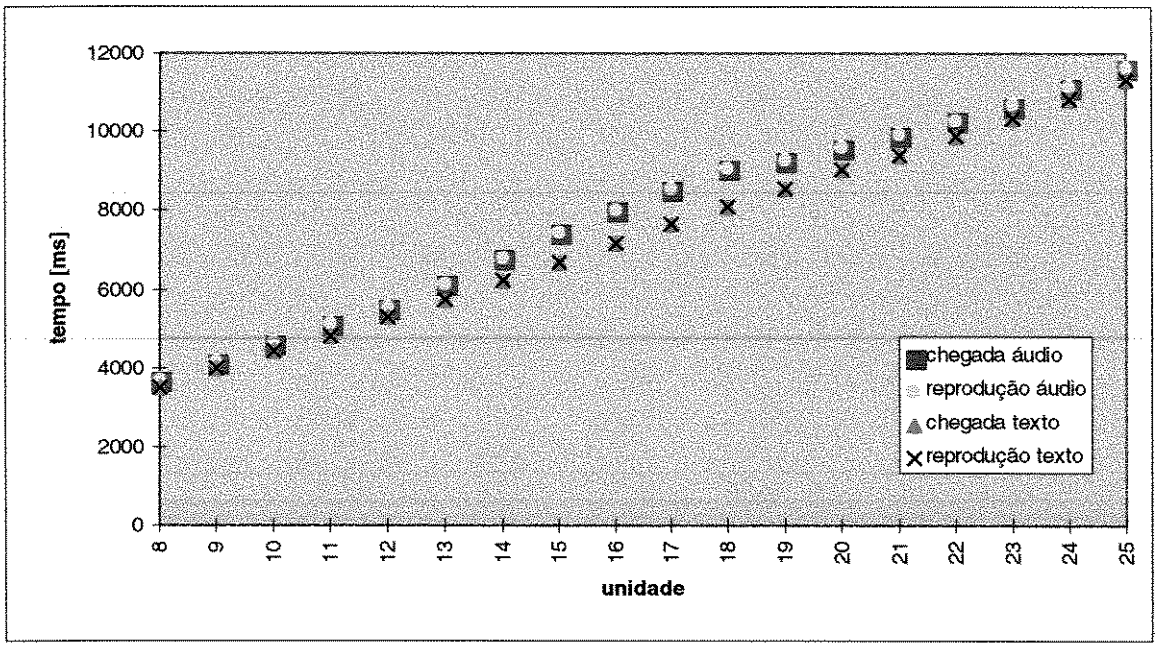

Gráfico 4.3 - FDDI com pouco tráfego, sem fluxo mestre, com atraso do áudio e sem sincronismo (tabela B3).

Aplicando-se a sincronização ao fluxo do gráfico anterior, é possível verificar a atuação do método. No Gráfico 4.4, no período em que ocorreu a desfazem (5.500ms a 9.200ms), a exibição das unidades de texto tentou acompanhar a chegada das unidades de áudio. Foi utilizado os parâmetros Skew_E=1 e Skew_D=700ms. Isto significa que a defazagem máxima permitida em relação ao número de unidades é igual a um e que a defazagem máxima em relação ao tempo é igual a 700ms. Quando a unidade 15 de texto chega, a 14 já foi exibida mas a unidade 14 de áudio ainda não chegou. Devido a isto a unidade 15 de texto deve esperar a chegada e, conseqüentemente, a reprodução da unidade 14 de áudio. Este comportamento continua até a unidade de número 19, quando ocorre a chegada das unidades atrasadas, que são reproduzidas o mais rápido possível na tentativa do sistema voltar à sua taxa normal de operação.

Alterando-se os valores das variáveis Skew_E e Skew_D pode-se obter os mais diversos resultados. O Gráfico 4.5 mostra que a má escolha pode piorar a reprodução das mídias.

Foram escolhidos os valores de Skew_E=1 e Skew_D=200ms. As unidades de áudio chegam atrasadas e as de texto se atrasam na tentativa de acompanhá-las. Entretanto, tal ação em nada se justificou, pois as unidades de áudio não puderam ser exibidas: estavam mais de 200ms (Skew_D) atrás das unidades de texto. Neste caso, seria melhor que as unidades de texto fossem reproduzidas à medida em que fossem chegando e que as unidades atrasadas de áudio simplesmente fossem eliminadas.

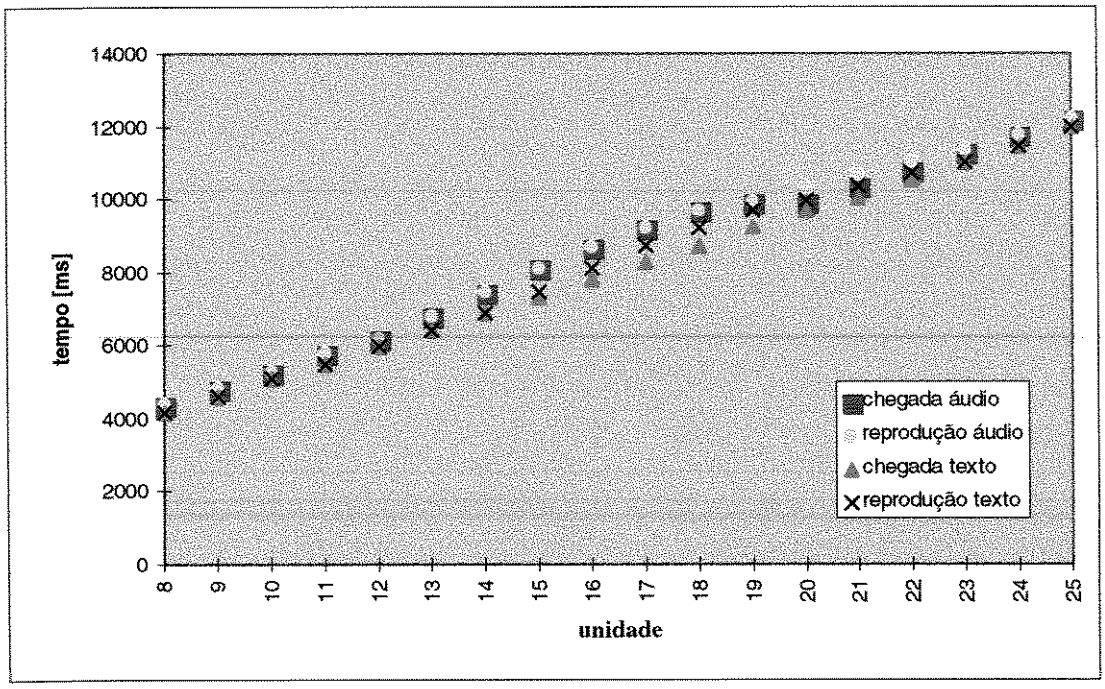

Gráfico 4.4 - FDDI com pouco tráfego, sem fluxo mestre, com atraso do áudio, Skew_E=1 e Skew_D=700ms (tabela B5).

Quando a unidade 13 de áudio chega (ver Gráfico 4.5), ela já está suficientemente atrasada (atraso > Skew_D) em relação à unidade 13 de texto para ser exibida e por isto não o é (o mesmo ocorre com as demais até a de número 20 quando o sistema consegue repor as atrasadas). O atraso de exibição das unidades de texto começa na unidade 16 e vai até a 19. Como Skew_E=1, a exibição de um dos fluxos não pode ficar mais do que uma unidade à frente do outro.

Inversamente ao gráfico anterior, ao aumentar demasiadamente os valores de Skew_D (5.000ms) e Skew_E (3) obtém-se um resultado parecido com o do Gráfico 4.3, onde não houve sincronização. A única diferença está na unidade 19 que não é reproduzida porque sua chegada ocorreu juntamente com a unidade 20 e, neste caso, o sistema atua de forma a eliminar o mais rápido possível os atrasos (ver Gráfico 4.6).

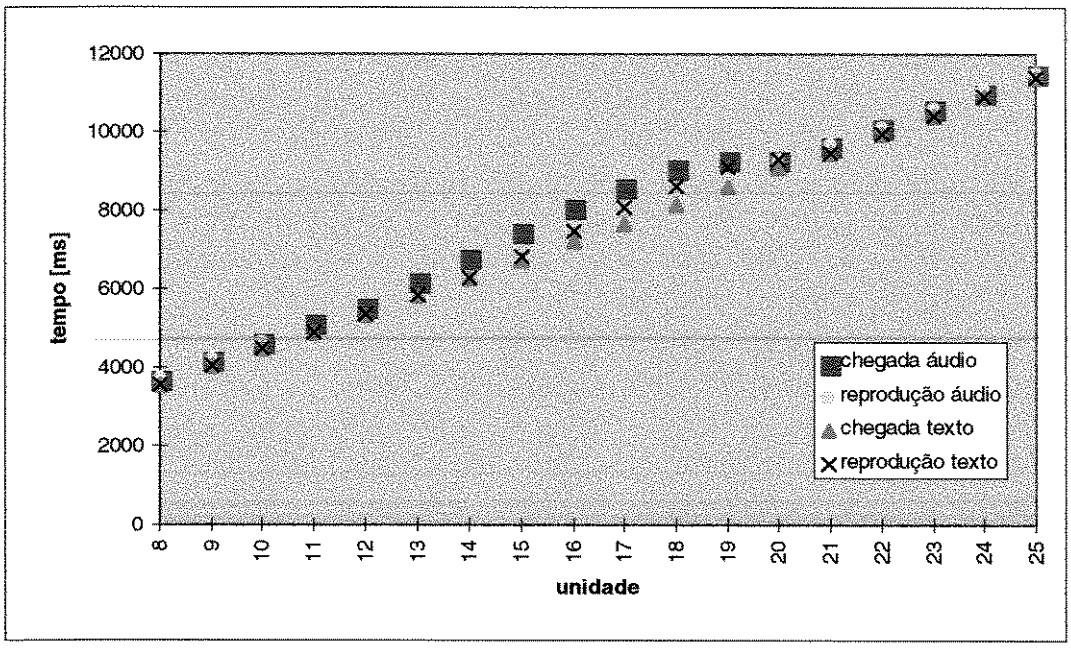

Gráfico 4.5 - FDDI com pouco tráfego, sem fluxo mestre, com atraso do áudio, Skew_E=1 e Skew_D=200ms (tabela B4).

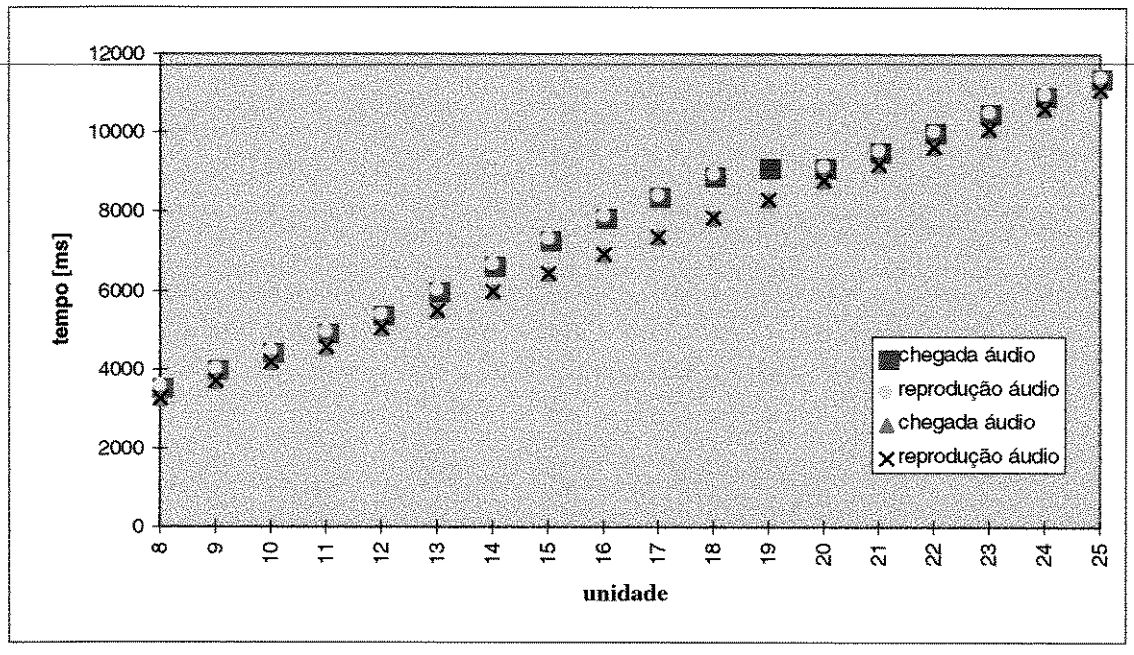

Gráfico 4.6 - FDDI com pouco tráfego, sem fluxo mestre, com atraso do áudio, Skew_E=3 e Skew_D=5.000ms (tabela B6).

Voltando-se ao Gráfico 4.5, suponha que se queira exibir o fluxo que não esteja atrasado. Para isto, basta aumentar o Skew_E para 5 e manter o Skew_D em 200ms, tem-se, então o Gráfico 4.7 em que o áudio atrasado não é reproduzido.

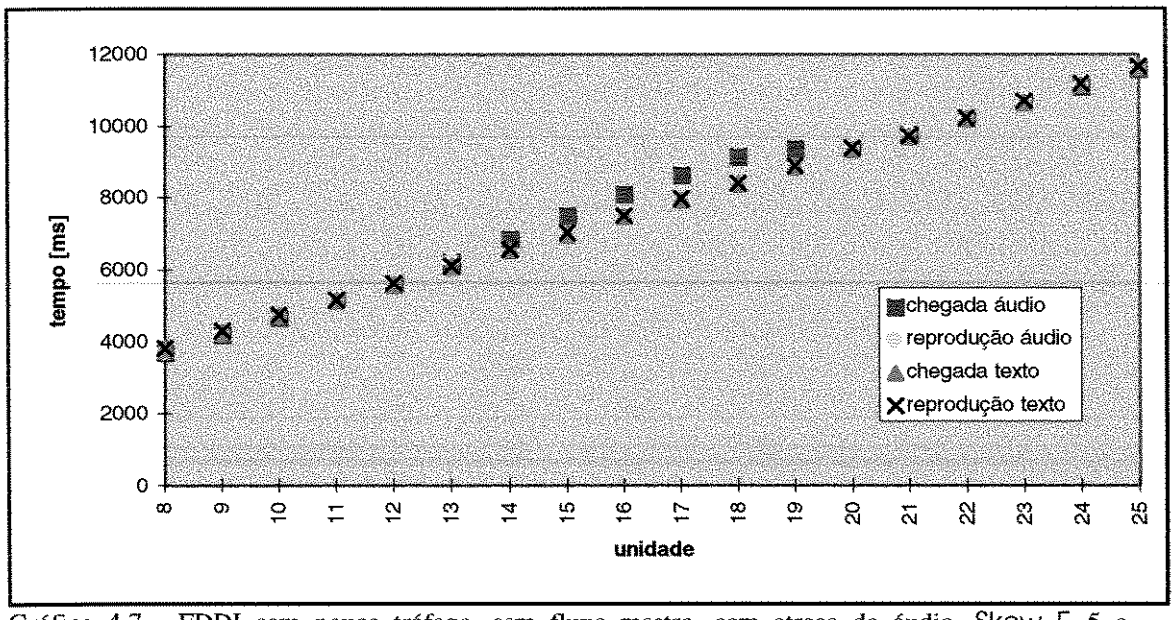

Gráfico 4.7 - FDDI com pouco tráfego, sem fluxo mestre, com atraso do áudio, Skew_E=5 e Skew_D=200ms (tabela B8).

Como não há fluxo mestre, o comportamento do algoritmo em caso de atraso do texto é idêntico ao atraso do áudio (ver Gráfico 4.8 e Gráfico 4.9) (compare estes com os Gráficos 4.4 e 4.7 respectivamente).

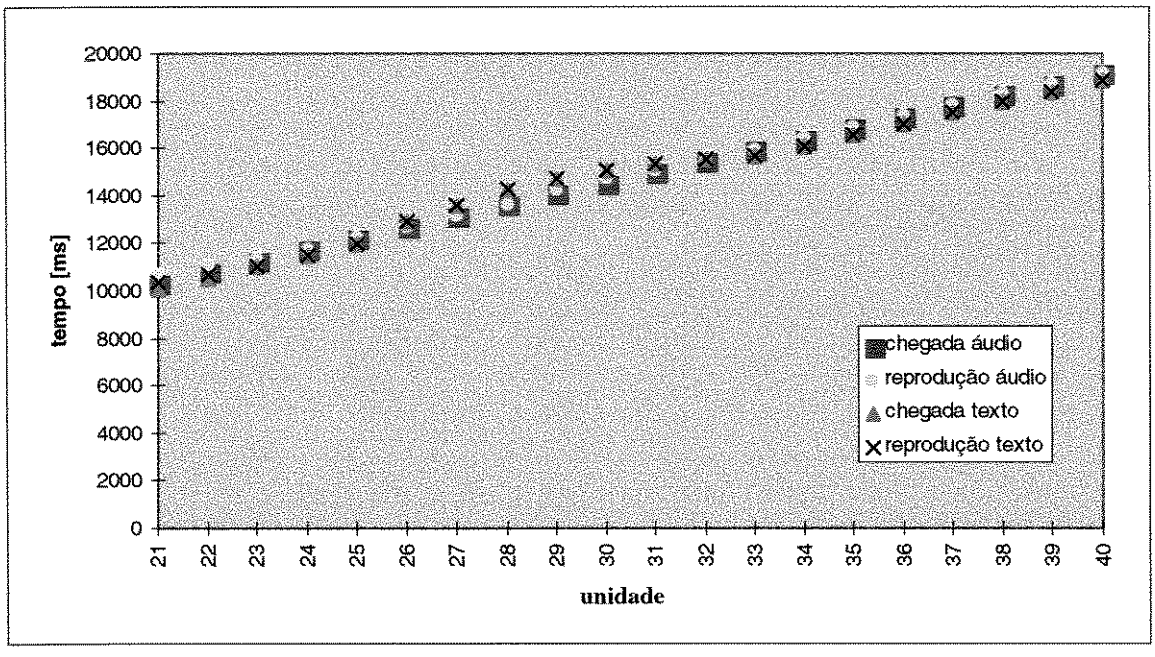

Gráfico 4.8 - FDDI com pouco tráfego, sem fluxo mestre, com atraso de texto, Skew_E=1 e Skew_D=700ms (tabela B5).

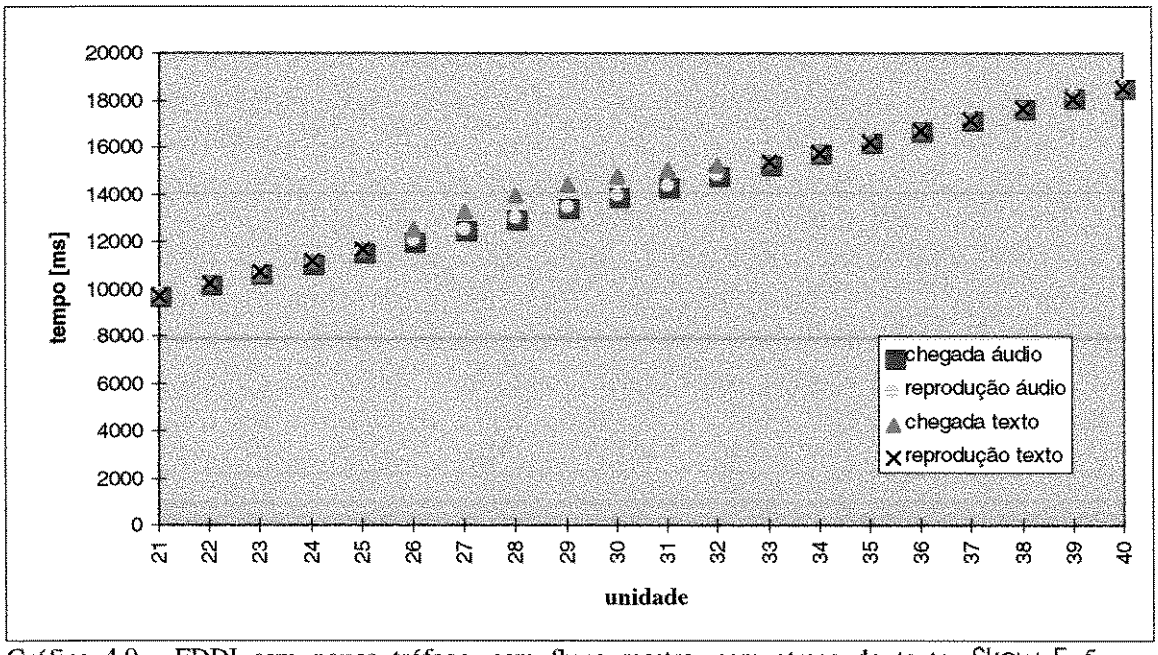

Gráfico 4.9 - FDDI com pouco tráfego, sem fluxo mestre, com atraso do texto, Skew_E=5 e Skew_D=200ms (tabela B8).

Quando há o atraso dos dois fluxos, o problema da sincronização recai em um dos dois casos estudados anteriormente: atraso do áudio ou atraso do vídeo. Isto porque sempre haverá um fluxo que estará mais atrasado do que o outro. Caso contrário, se a defasagem for a mesma para ambos, então, apesar de estarem atrasados, não estarão descasados (ver Gráfico 4.10).

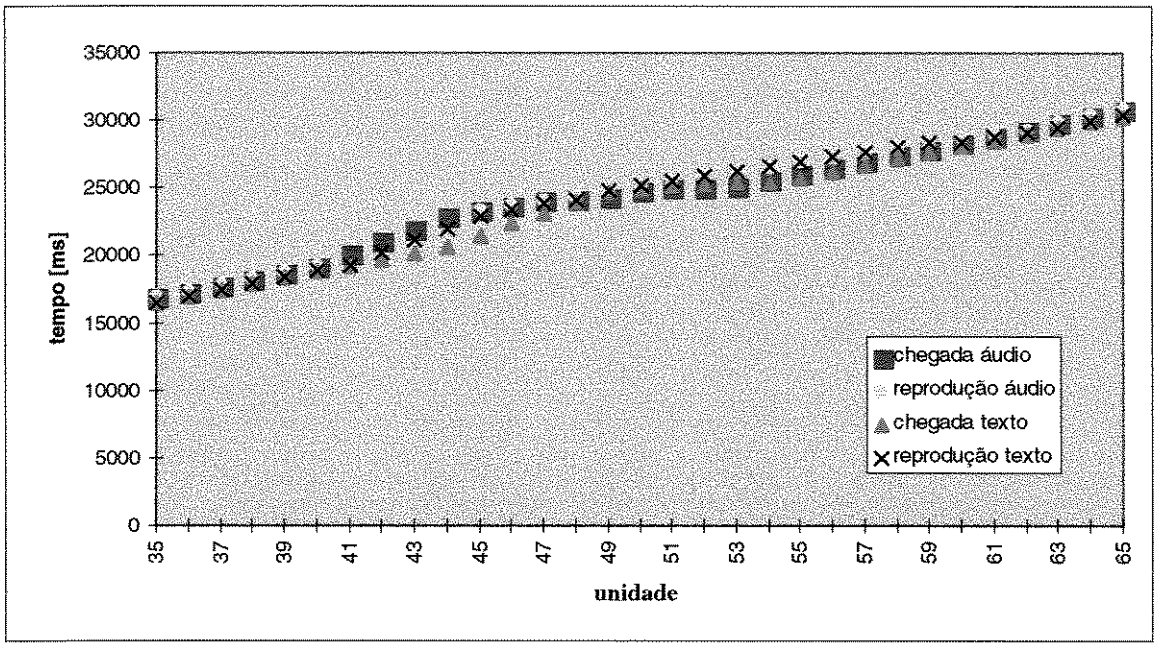

Gráfico 4.10 - FDDI com pouco tráfego, sem fluxo mestre, com atraso do áudio e do texto, Skew_E=1 e Skew_D=700ms (tabela B5).

A partir da unidade 41 o áudio começa a se atrasar e a unidade 42 de texto tenta acompanhá-la. Porém, a unidade 42 de áudio já chega suficientemente atrasada e não pode ser exibida (atraso maior do que Skew_D). Esta situação perdura até a chegada da unidade 45, quando ocorre uma aceleração na chegada das unidades de áudio que estavam atrasadas. Os fluxos conseguem casar-se novamente, uma vez que o texto também vinha sofrendo com a sobrecarga da rede e chega, inclusive, a ficar mais atrasado do que o fluxo de áudio. Com a chegada de todas as unidades atrasadas, o sistema as reproduz o mais rápido possível: não há mais o atraso da rede, pois todas as unidades já estão no *buffer* e assim, após alguns milisegundos, o sistema volta a operar em regime normal (a partir da unidade 60).

O Gráfico 4.11 é uma união dos comportamentos do Gráfico 4.7 e do Gráfico 4.10, ou seja, ocorre o atraso de ambos os fluxos, mas somente o fluxo que está à frente é reproduzido. Este procedimento é verificado entre as unidades 40 e 48, quando o áudio atrasado é eliminado. Com o atraso do texto, os fluxos voltam a se casarem após a unidade 48.

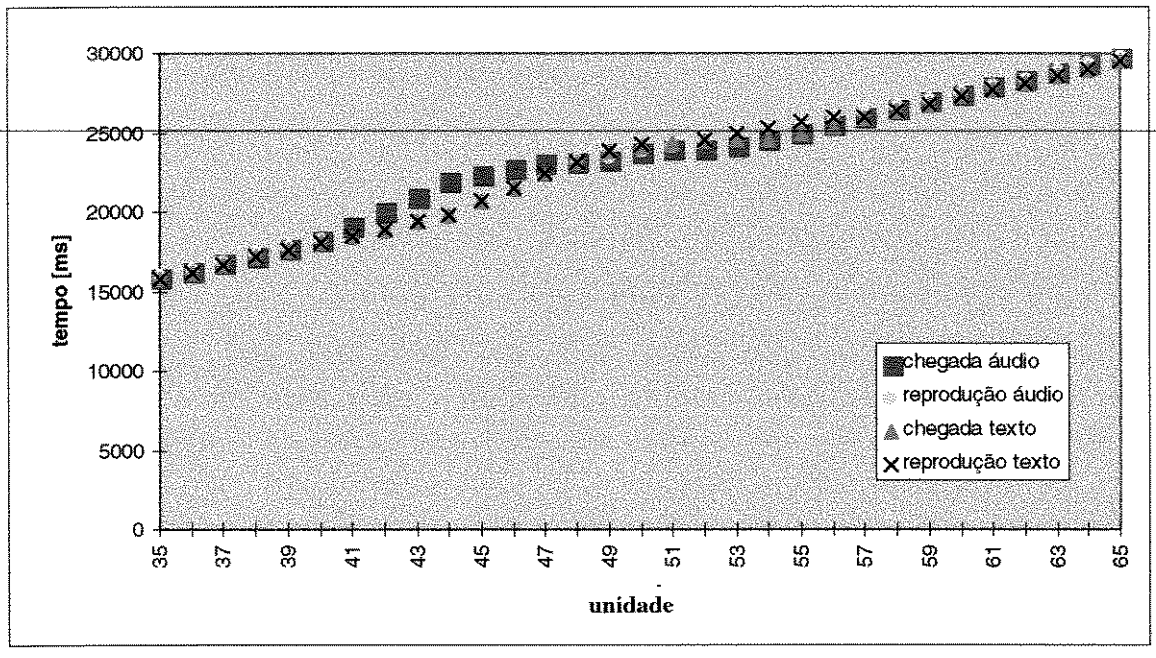

Gráfico 4.11 - FDDI com pouco tráfego, sem fluxo mestre, com atraso do áudio e do texto, Skew_E=5 e Skew_D=700ms (tabela B7).

4.5.2 Sincronização com Fluxo Mestre

A existência de um fluxo mestre obriga que os demais o sigam. O Gráfico 4.12 demonstra os resultados obtidos sob esta condição. Entre as unidades 14 e 19 o texto, apesar de não ter sofrido atraso, teve de esperar pelo áudio para ser reproduzido.

Quando ocorrer um atraso do fluxo escravo, ele só poderá ser reproduzido se a diferença entre eles não exceder os limites estabelecidos por Skew_E e Skew_D. No Gráfico 4.13 entre o intervalo que compreende as unidades 27 e 31 de texto ultrapassam estes limites e consequentemente elas não são reproduzidas.

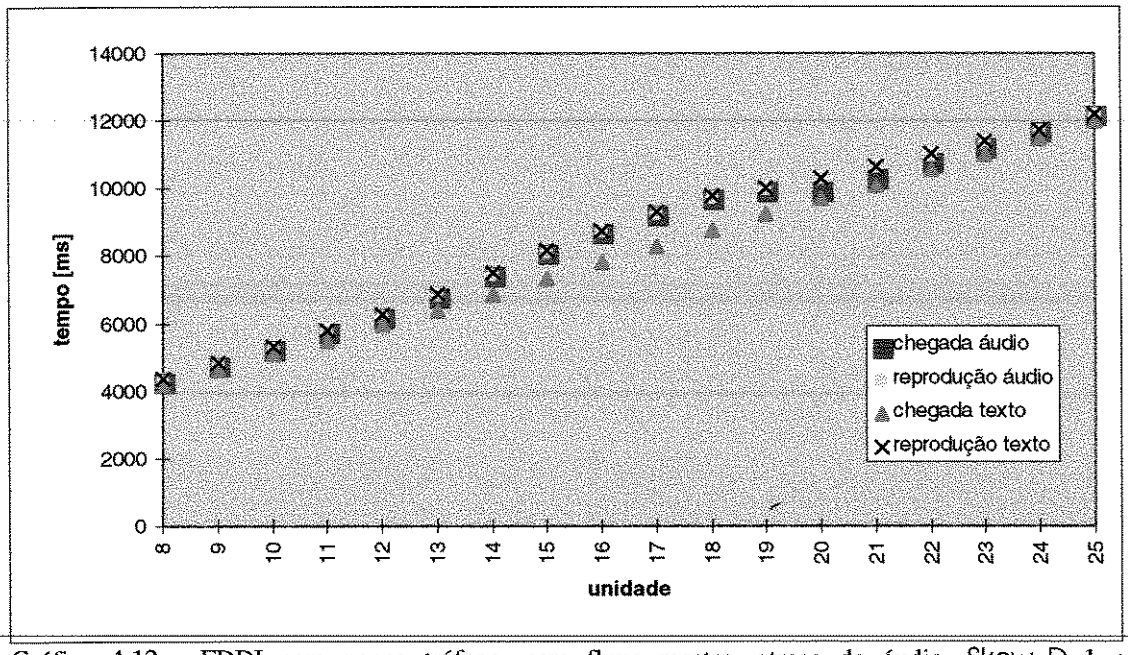

Gráfico 4.12 - FDDI com pouco tráfego, com fluxo mestre, atraso do áudio, Skew_D=1 e Skew_D=200ms (tabela A4).

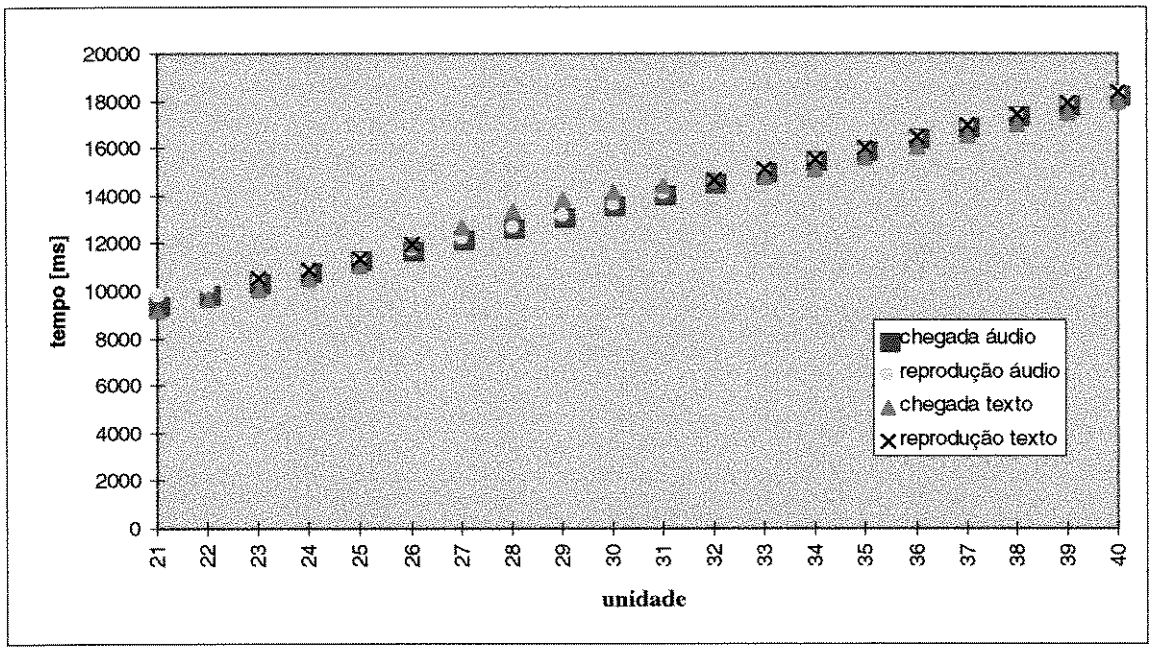

Gráfico 4.13 - FDDI com pouco tráfego, com fluxo mestre, com atraso do texto, Skew_D=1 e Skew_E=200ms (tabela A4).

No Gráfico 4.14, áudio e texto se atrasam. Entre as unidades 41 e 48 o áudio está mais atrasado: a reprodução do texto deve, então, acompanhá-lo. Após a unidade 48, a situação se inverte e o atraso do áudio é superado pelo do texto. Porém, entre as unidades 49 e 59, somente o áudio é exibido. Como o carimbo de tempo é marcado no início de cada unidade, após a exibição do áudio já se passou mais do que 200ms e é por isto que o texto não pode ser reproduzido (valor de Skew_D). (Isto também pode ser observado no Gráfico 4.13 com as unidades 20 e 21.)

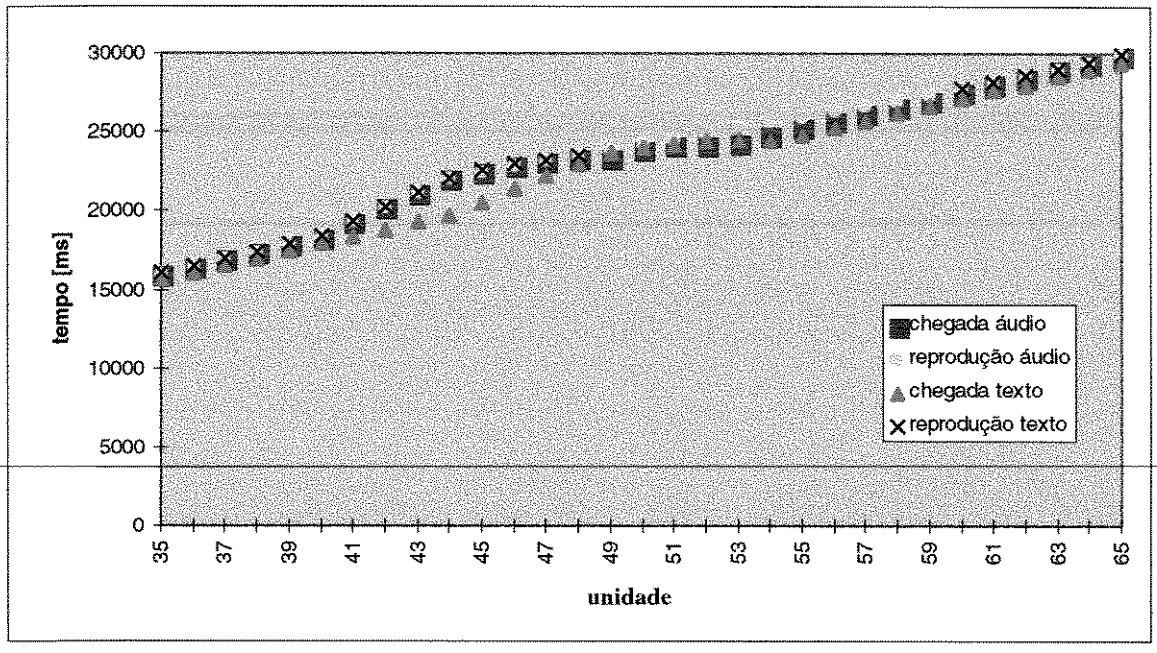

Gráfico 4.14 - FDDI com pouco tráfego, com fluxo mestre, com atraso do áudio e do texto, Skew_E=1 e Skew_D=200ms (tabela A4).

Para fazer com que as unidades de texto também sejam reproduzidas é necessário aumentar o valor de Skew_D. No Gráfico 4.15 as unidades de texto que estão acima de 51 passam a ser reproduzidas.

Quando, por algum motivo, o protocolo de comunicação atrasar a entrega das unidades de tal forma que não mais consiga repô-las, a defasagem entre as mídias ficará restrita aos limites definidos por Skew_E e Skew_D. No Gráfico 4.16, a partir da unidade 68, o áudio sofre um atraso definitivo. Mas como Skew_E=1, a reprodução da próxima unidade de texto já é retardada pelo Sincronizador.

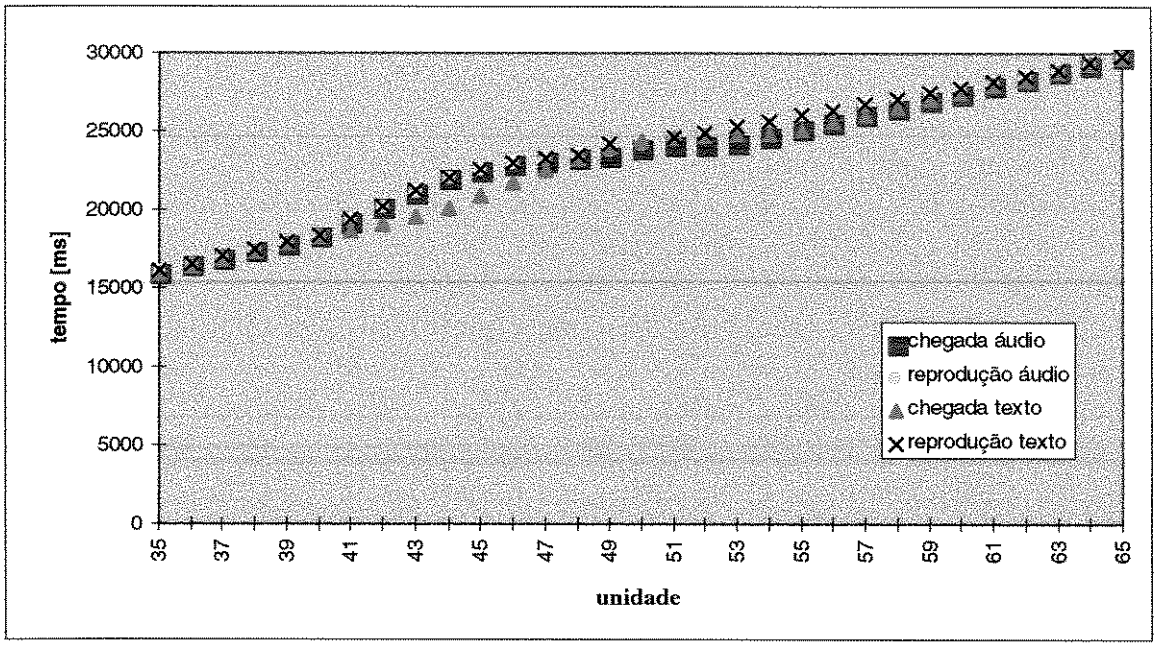

Gráfico 4.15 - FDDI com pouco tráfego, com fluxo mestre, com atraso do áudio e texto, Skew_E=5 e Skew_D=700ms (tabela A7).

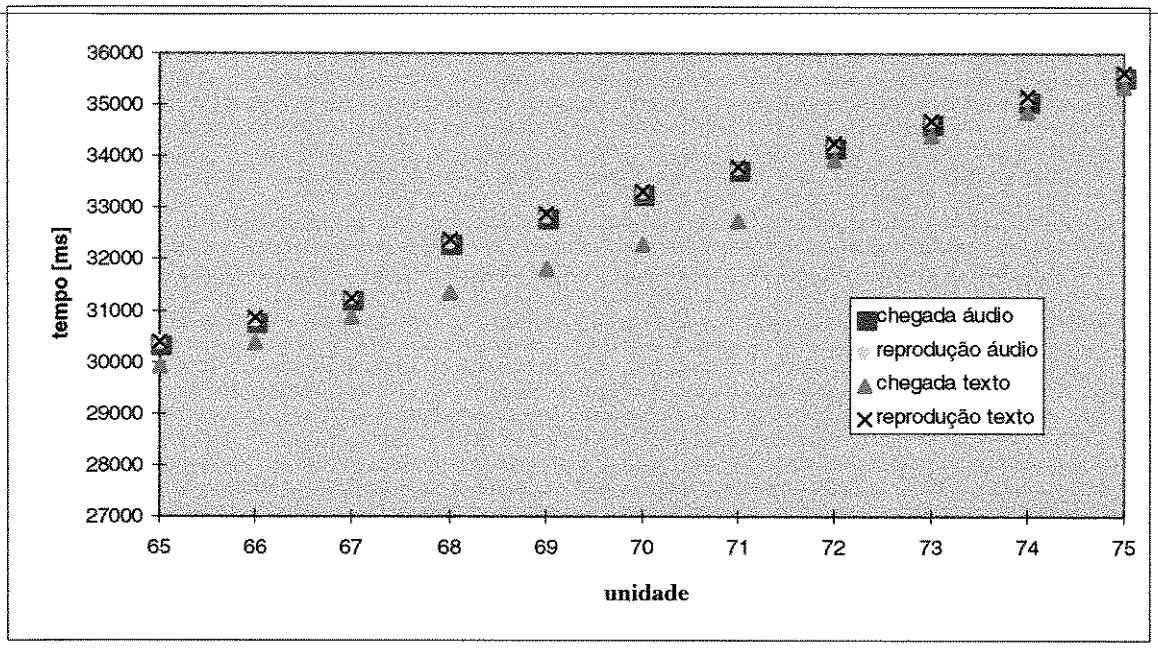

Gráfico 4.16 - FDDI com pouco tráfego, com fluxo mestre, com atraso do áudio, Skew_E=1 e Skew_D=700ms (tabela A7).

4.5.3 FDDI com Tráfego e Ethernet

Os testes na rede FDDI com tráfego foram realizados em situações de compartilhamento da rede, ou seja, quando a maioria das estações de trabalho do laboratório estavam em uso (a rede que interligava os computadores PS/2 era compartilhada por outras estações de trabalho do laboratório). Quanto à Ethernet, esta era uma rede privada, partilhada somente pelos computadores desta pesquisa.

Os resultados obtidos com estas duas redes e os da FDDI com pouco tráfego são bastante semelhantes. A explicação para tal fato deve-se à limitação da velocidade pela CPU utilizada, 33MHz, e principalmente à granulosidade das unidades, 460ms. (Compare os Gráficos 4.17, 4.18 e 4.15 e os Gráficos 4.19, 4.20 e 4.5.)

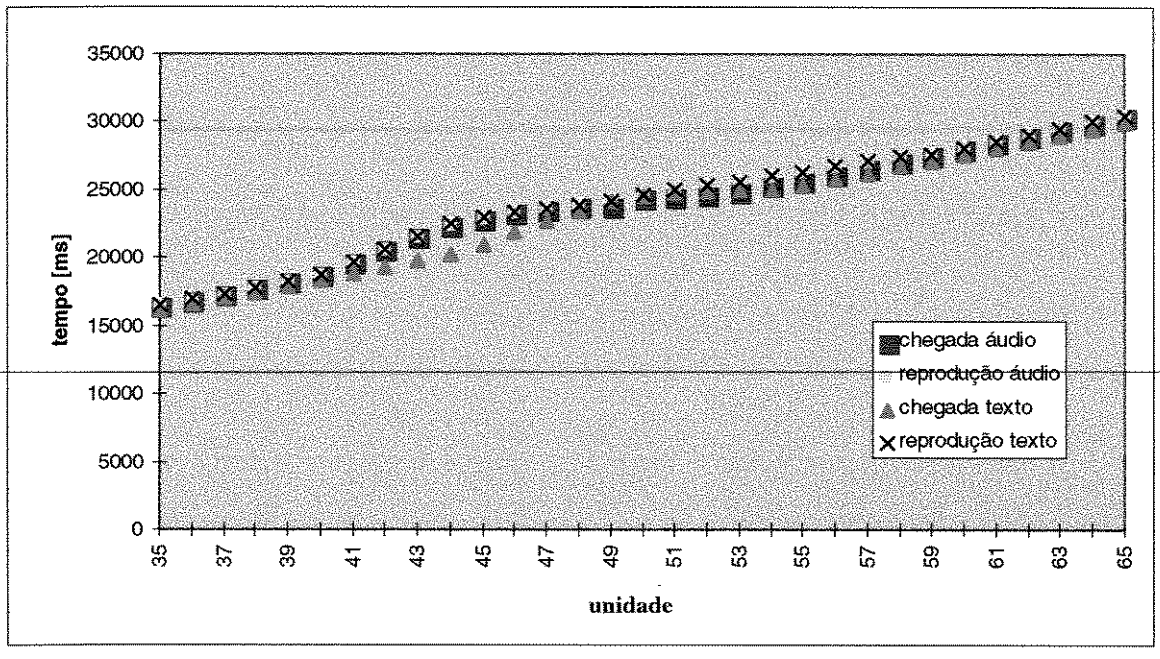

Gráfico 4.17 - FDDI com tráfego, com fluxo mestre, com atraso do áudio e do texto, Skew_E=5 e Skew_D=700ms (tabela C7).

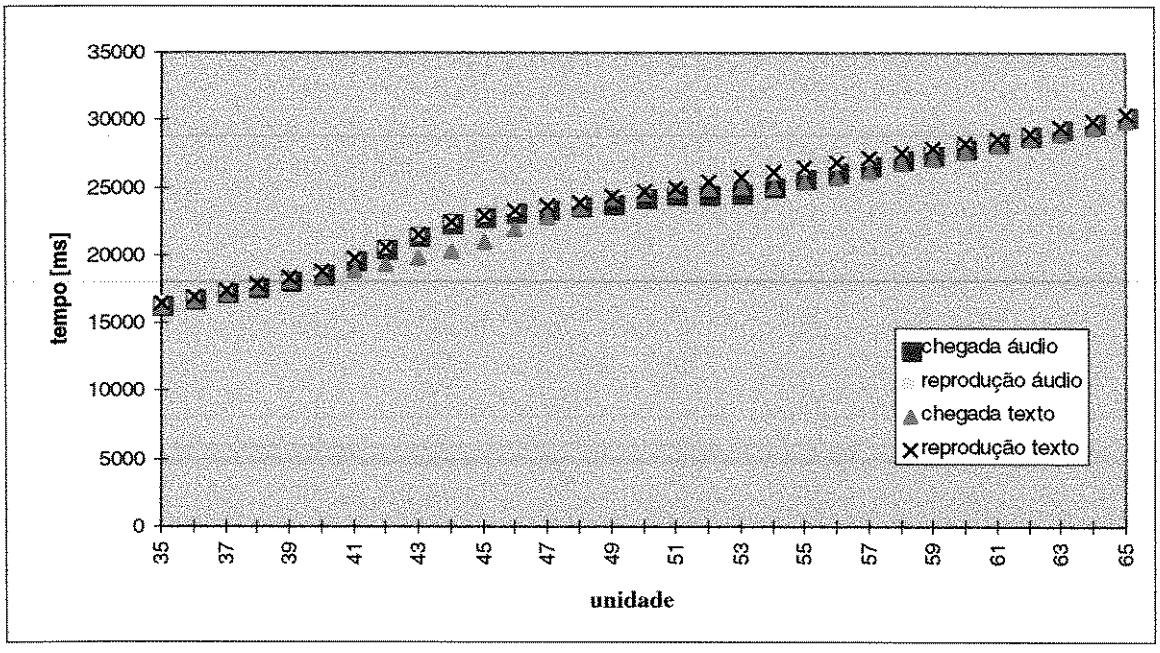

Gráfico 4.18 - Ethernet, com fluxo meste, com atraso do áudio e texto, Skew_E=5 e Skew_D=700ms (tabela E7).

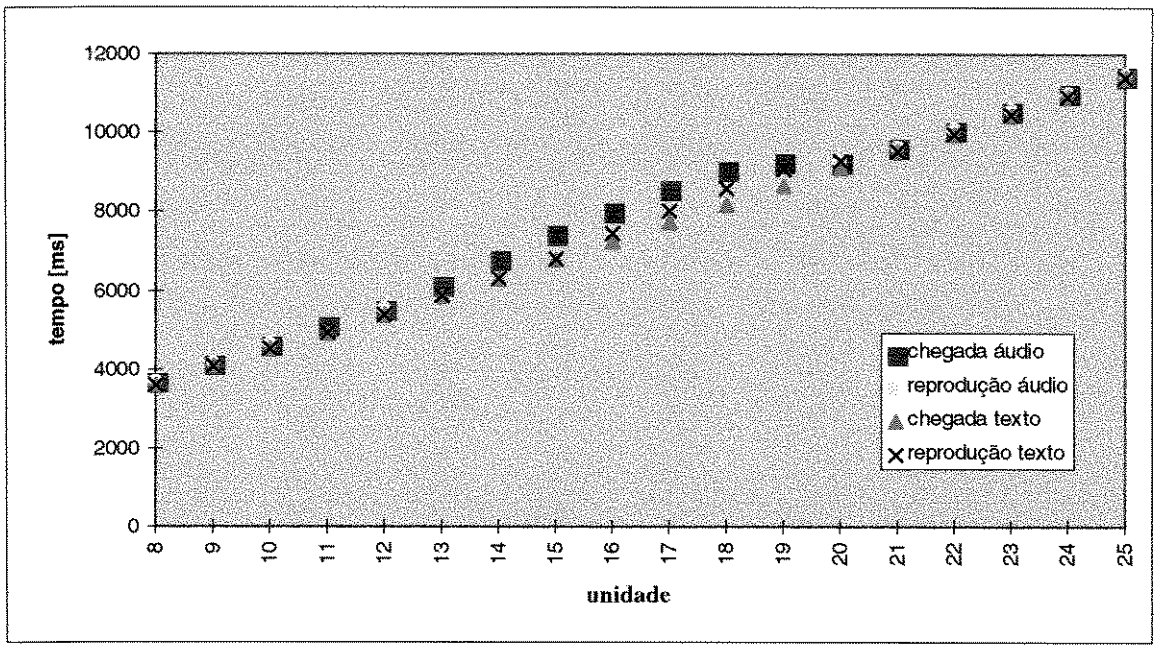

Gráfico 4.19 - FDDI com tráfego, sem fluxo mestre, atraso do áudio, Skew_E=1 e Skew_D=200ms (tabela D4).

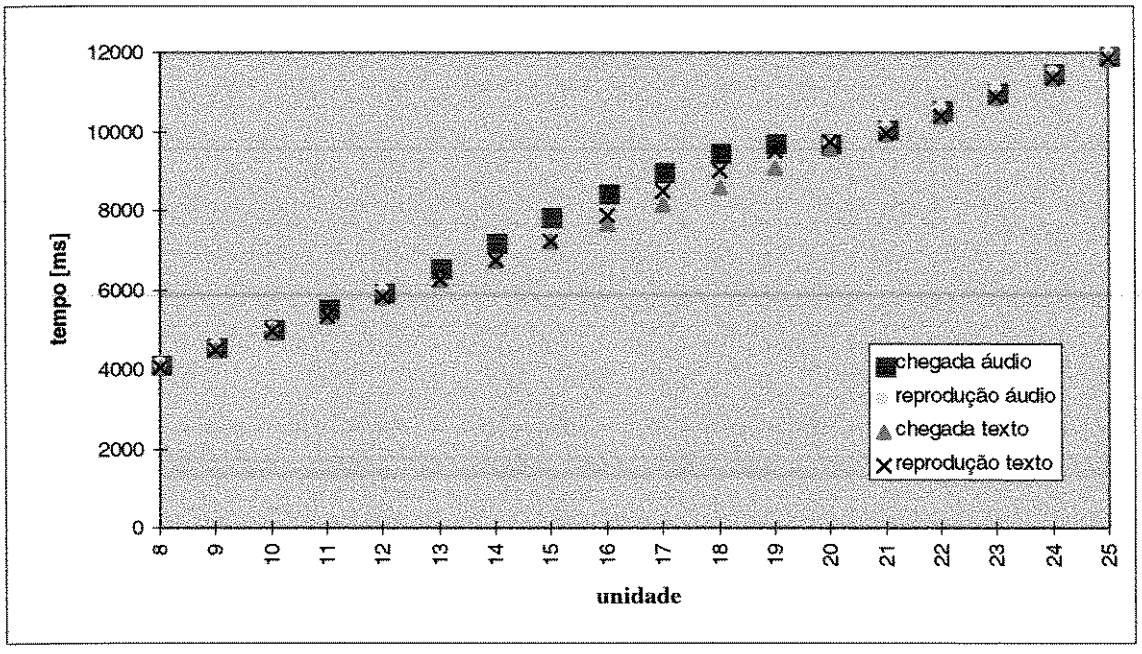

Gráfico 4.20 - Ethernet, sem fluxo mestre, com atraso do áudio, Skew_E=1 e Skew_D=200ms (Tabela F4).

Finalmente, após a análise de todos estes resultados é possível constatar que as Equações 2.3 e 2.4 foram respeitadas quando da reprodução de todas as unidades de ambas as mídias em qualquer uma das situações anteriormente apresentadas.

Capítulo 5

CONCLUSÕES E TRABALHOS FUTUROS

5.1 Conclusões e Trabalhos Futuros

Um dos principais objetivos deste trabalho era o domínio da manipulação de fluxos de dados (enviar, capturar, parar, retomar, etc.) para dispositivos físicos multimídia. A partir do estudo e uso de *softwares* comerciais (MMPM/2 e AVK) foi possível familiarizar-se com os conceitos e as técnicas empregadas para o controle destes dispositivos. Posteriormente, o domínio desta tecnologia seria utilizado, em outros trabalhos, para a implementação de uma plataforma multimídia que possibilitasse o desenvolvimento de aplicações independentes dos dispositivos físicos.

Entretanto, foi justamente na utilização e integração entre *software* e *hardware* multimídia que se depositaram as maiores dificuldades. Como já descrito anteriormente (Capítulos 1 e 4), vários foram os fatores que impossibilitaram o emprego de vídeo. Apesar desta perda, tal fato em nada comprometeu a validação do

algoritmo de sincronização, uma vez que este último deve funcionar independentemente das mídias com as quais esteja manipulando.

Outra dificuldade encontrada foi a especificação do tamanho das unidades capturadas. O menor tamanho conseguido, 370ms, é maior do que o realmente desejado. Portanto, uma das continuações deste trabalhos será o emprego de outros dispositivos físicos e/ou de outras ferramentas de programação que possam contribuir para a diminuição do tamanho das unidades e, consequentemente, validar o algoritmo neste novo contexto.

Com relação ao algoritmo de sincronização, pôde-se verificar a diferença entre a eleição ou não de um fluxo mestre. Sua presença é capaz de estreitar o casamento entre os fluxos ao custo da exclusão, quando do atraso do mestre, da reprodução do fluxo escravo que não se atrasou. Ao mesmo tempo, sua ausência é capaz de tornar a sincronização mais flexível, uma vez que todos os fluxos têm a mesma prioridade. Desta forma é possível privilegiar aquele fluxo que está mais à frente. Mas, neste caso, há um dessincronismo maior entre os fluxos.

O sincronismo é fortemente afetado pelo parâmetros Skew_E e Skew_D que especificam o desvio máximo permitido entre os fluxos. Skew_E determina o desvio em relação à ordem sequencial das unidades. Já o Skew_D estabelece limites à defazagem temporal. Manipulando-se adequadamente estes parâmetros, é possível obter, em função da QoS desejada/necessária, os mais diversos resultados.

Com relação ao estabelecimento de limites para o sincronismo, um ponto merece ainda ser estudado. Alguns trabalhos (por exemplo o ACME, ver Capítulo 2) desenvolvem metodologias de sincronização que levam em conta o atraso da CPU no atendimento às tarefas. Supondo-se que em alguns sistemas haja a reserva de recursos, um dos requisitos da QoS, pode-se assumir, sob estas condições, que a CPU irá atender a todas as tarefas com uma determinada probabilidade de falha já anteriormente negociada com a aplicação. Entretanto, nos casos onde isto não for possível, é certo que atrasos irão ocorrer. Mas qual será a influência deste atraso na percepção do usuário? Nas situações em que há transmissão do fluxo via rede, não seria o atraso da CPU insignificante diante do atraso ocorrido na rede? Em um experimento realizado no Centro de Rede de Computadores Europeu da IBM [Steinmetz90] foi medida a tolerância dos usuários à percepção de mídias fora de sincronismo. Desvios entre os intervalos -80ms (áudio atrás do vídeo) e +80ms (áudio na frente do vídeo) não foram detectados pelos usuários. A área determinada como fora de sincronismo estava acima dos valores -160ms e +160ms. Os intervalos de -80ms a -160ms e +80ms a +160ms foram considerados como uma área de transição, dependendo de fatores como a aproximação do usuário, era possível afirmar se havia ou não sincronismo. Ou seja, a partir de estudos como estes deve-se levantar quais são realmente os fatores que mais influenciam no resultado final de apresentação de uma determinada aplicação multimídia. Assim sendo, os futuros algoritmos de sincronização poder-se-ão focar nestes pontos.

Pode-se, enfim, concluir que os principais objetivos do trabalho foram alcançados. O estudo de técnicas de sincronização, a implementação de um sincronizador, o domínio da tecnologia de controle/monitoramento de dispositivos

físicos multimídia através da utilização de *softwares* comerciais para tais fins e, finalmente, a proposta para desenvolvimento de uma plataforma de sincronização de dados multimídia para ambiente distribuído aberto. Esta plataforma possibilitará a migração do sincronizador para um ambiente de estações de trabalho, tendo-se em vista sua integração com os demais trabalhos deste projeto.

REFERÊNCIAS BIBLIOGRÁFICAS

[ActionMediaII] "*ActionMedia/II - Technical Reference*", IBM Corporation, Version 1.0, 1992.

[Anderson91] D. P. Anderson, G. Homsy, "A Continuous Media I/O Server and Its Synchronization Mechanism", *IEEE Computer Magazine*, Vol. 24, No. 10, 1991, pp. 51-57.

[Blair92] Gordon Blair, Andrew Campbell, Geoff Coulson, Frankie Garcia, David Hutchison, Andrew Scott, Doug Shepherd, "Orchestration Services for Continous Media Communications", *IV Bangor Communication Symposium*, Bangor, Gwynedd, May, 1992.

[Blakowski96] Gerold Blakowski e Ralf Steinmetz, "A Media Synchronization Survey: Reference Model, Specification, and Case Studies", *IEEE Journal on Selected Areas in Communications*, Vol 14, No 1, Jan 96.

[Bulterman] Dick C. A. Bulterman, Robert van Lieve, "Multimedia Synchronization and Unix", Centrum voor Wiskunde en Informatica. Amsterdã, Holanda.

[Campbell] Andrew Campbell, Geoff Coulson, Francisco Garcia, David Hutchison, "Orchestration Services for Distribuited Multimedia Synchronisation", Computing Departament, Lancaster University, Lancaster LA1 4YR, UK. E-mail: mpg@comp.lancs.ac.uk.

[Chakrabarti94] S. Chakrabarti, R. Wang "An Adaptive Control for Packet Video", *Proc. of Int. Conference on Multimedia Computing*, May 1994, pp. 56-62.

[COPPEa] "Definição de um Serviço de Sincronização Contínua para Aplicações Multimídia", relatório interno, COPPE UFRJ.

[COPPEb] "Comunicação Sincronizada de Dados Pré-Orquestrados", relatório interno, COPPE UFRJ.

[COSS] "Common Object Services Specification (COSS)", Version 1.0, Technical Report, Object Management Group, March 1993.

[Eleftheriadis94] A. Eleftheriadis, S. Pejhan, D. Anastassiou, "Architecture and Algorithms of the Xphone Multimedia Communication System", *Multimedia Systems*, Vol. 2, August 1994, pp. 89-99.

[Escobar91] J. Escobar, D. Deutsch, C. Partridge, "A Multi-Service Flow Synchronization Protocol", *BBN Systems and Technologies Division*, 1991, pp. 1-16.

[Escobar92] J. Escobar, D. Deutsch, C. Partridge, "Flow Synchronization Protocol", *Proc. of the IEEE Globecom*, Vol. 3, 1992, pp.1381-1387.

[Furht94] Borko Furht, "Multimedia Systems: An Overview", *IEEE Multimedia*, Spring 1994, pp. 47-59.

[Guedes95] Luiz Affonso Guedes, "Qualidade de Serviço em Aplicações Multimídia", Relatório interno. Departamento de Computação e Automação, Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas, Outubro, 1995.

[Herman95] Ivan Herman e Graham J. Reynolds, "PREMO: An Emerging Standard for Mulmedia Presentation", Centrum voor Wiskunde en Informatica. E-mail: Ivan.Herman,Graham.Reynolds@cwi.nl. 1995.

[Köhler94] Daniel Köhler, Harald Müller, "Multimedia Playout Synchonization Using Buffer Level Control", In: Proc. of the 2nd Int Workshop on Multimedia: Advanced Teleservices and High-Speed Communication Architectures, 1994. *Lecture Notes in Computer Science*, Springer Verlag, Vol. 868, pp. 167-180.

[Little90] T. D. C. Little, A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects", *IEEE J. Selected Areas in Communications*, Vol. 8, No3, 1990, pp. 413-427.

[Little93a] T. D. C. Little, "A Framework for Synchronous Delivery of Time-dependent Multimedia Data", *Multimedia Systems*, Vol. 1, No2, 1993, pp. 87-94.

[Little93b] T. D. C. Little, F. Kao, "A Intermedia Skew Control System for Multimedia Data Presentation", In: Proc. of the 3rd Int. Workshop on Network and Operating System Support for Digital Audio and Video, 1992. *Lecture Notes in Computer Science*, Springer Verlag, Vol. 712, 1993, pp. 130-141.

[Lopes96] A. B. Lopes, L. Desiderá, J. C. Maia, "An ISO/PREMO-based Multimedia System Service Plataform and its Implementation on a CORBA Enviroment", relatório interno, DCA, FEE, Unicamp, 1996.

[Microsoft] New Multimedia Data Types and Data Techniques, Microsoft Multimedia, Standards Update, revision 1.0.97, march 1995.

[MMPM/2] *IBM Multimedia Presentation Manager Programming Reference and Programming Guide 1.0*, IBM Corporation, IBM Form: S41G-2919-00 and S41G-2920-00, Mar. 1992.

[PREMO] International Organisation for Standardization, Information Processing System, Computer Graphics, "Presentation Environment for Multimedia Objects". *Part 4: Multimedia System Services*, (PREMO); ISO/IEC 1447-4, September 1994.

[Pronios94] Nikos B. Pronios, Theodoros Bozios, "A Scheme for Multimedia and Hypermedia Synchronization", *Lecture Notes in Computer Science 882*, Nov. 1994, pp. 340-345.

[Rangan93] P. V. Rangan, S. Ramanathan, el al., "Techniques for Multimedia Synchronization in Network File Systems", *Computer Communications*, Vol. 16, No.3, March 1993.

[Shepherd92] Doug Shepherd, David Hutchinsom, Francisco Garcia, Geoff Coulson. "Protocol Support for Distributed Multimedia Applications", *Computer Communications*, Vol. 15, No. 6, jul/ago, 1992, pp. 359-366.

[Stefani95] Jean-Bernard Stefani, Elsevier Science B. V., "Open distributed processing: an architectural basis for information networks", *Computer Communications*, Vol. 18, No. 11, Nov 1995.

[Steinmetz90] Ralf Steinmetz, "Synchronization Properties in Multimedia Systems", *IEEE Journal on Selected Areas Communications*, Vol. 8, No. 3, pp. 401-412, April 1990.

[Steinmetz96] Ralf Steinmetz, "Human Perception of Jitter and Media Synchronization", *IEEE Journal on Selected Areas Communications*, Vol. 14, No. 1, pp. 61-72, January 1996.

[Williams91] Neil Williams, Gordon S. Blair, Nigel Davics, "Distribuited Multimedia Computing an Assessment of the State of the Art", *Information Services and Use*, (11), 1991, pp. 265-281.

Apêndice A

O SISTEMA OPERACIONAL OS/2

A.1 Introdução

Para a implementação do SESIM, foi escolhido o sistema operacional OS/2[1]. Esta escolha não se deu por mero acaso. O OS/2 possui qualidades imprescindíveis para o funcionamento do SESIM, podendo-se destacar: a multitarefa (tanto a nível de processos quando a nível de *threads*), a definição de prioridades, e a disponibilidade de *software* e *hardware* para o desenvolvimento de aplicações multimídia. Nas próximas sessões descrever-se-á o funcionamento de multitarefa e de prioridades no OS/2.

[1] Marca registrada da IBM

A.2 Multitarefa no OS/2

O OS/2 é um sistema operacional multitarefa[1]. Isto significa que á capaz de rodar dois ou mais programas ao mesmo tempo. Existem três tipos de multitarefa no OS/2:

- *Sessões*: a sessão é o nível mais alto de multitarefa. Consiste de um ou mais processos e é, em essência, um computador virtual: possui sua própria imagem de tela, seu próprio teclado e *mouse*;
- *Processos*: abaixo da sessão, o próximo nível de multitarefa é o processo. Cada sessão contém pelo menos um processo, mas pode conter vários. É conveniente pensar que uma aplicação é sinônimo de um processo. Cada processo possui um número de recursos alocados para o seu uso. Estes recursos incluem memórias, arquivos, semáforos e outros. Cada processo contém uma ou mais *threads*;
- *Threads*: é o nível mais baixo de multitarefa. Um programador em linguagem C pode pensar em uma *thread* como sendo uma função; uma que possa rodar ao mesmo tempo que outras do programa. Cada processo tem pelo menos uma *thread*, chamada de principal. As *threads* ficam competindo pela utilização da CPU, causando o chaveamento entre elas e dando ao usuário a sensação de que estão rodando concomitantemente. É importante ressaltar que este chaveamento também é possível entre os processos, porém ele é muito mais rápido entre as *threads*, isto porque os recursos de um processo não têm que ser alocados para uma nova *thread*, mas devem ser para um novo processo: as *threads* compartilham de todos os recursos alocados para o processo ao qual pertencem.

  Como em multimídia o tempo é um fator determinante, a possibilidade de sua utilização é uma qualidade essencial ao desenvolvimento de tais aplicações.

A.3 Prioridades no OS/2

A prioridade de uma *thread* determinará o instante em que o escalonador[2] de tarefas do sistema operacional a ativará. A prioridade consiste de duas partes: classes de prioridades e níveis de prioridades. Há quatro classes de prioridades, em ordem decrescente: tempo-crítico, alta-fixada, regular e tempo-ocioso. Cada classe de prioridade é subdividida em 32 níveis de prioridade, numerados de 0 a 31. Isto significa que cada classe de prioridade possui outros 32 níveis de precedência. Toda *thread* marcada como tempo-crítico tem a prioridade maior do que aquelas marcadas como alta-fixada, e esta última somente será executada quando não houver nenhuma *thread* tempo-crítico pronta, e assim por diante. Mas quando existir duas ou mais *threads* com a mesma classe de prioridade, o escalonador irá fazer o chaveamento entre elas para que cada uma receba ciclos da CPU. Por exemplo, se duas ou mais *threads* entre várias outras que estão sendo executadas no sistema são de tempo-

[1] Encontra-se também na literatura os termos multiprogramação e multiexecução

[2] É a parte do sistema operacional responsável pela distribuição de tempo de CPU às várias *threads* existente.

crítico, estas duas sempre receberão ciclos da CPU antes de qualquer outra do sistema. Entretanto, se uma dessas duas *threads* de tempo crítico tem prioridade 25 e a outra tem prioridade 10, a *thread* com prioridade 25 sempre será executada em primeiro, a não ser que esteja bloqueada ou suspensa[1].

Mais uma vez, como as restrições temporais têm uma forte relevância em aplicações multimídia, o uso de prioridades possibilita a definição do grau de importância de cada tarefa na apresentação das unidades de mídias envolvidos na aplicação.

[1] Uma *thread* pode estar em um dos três estados: rodando (está em execução), pronta para rodar (está esperando outras terminarem para entrar em execução) ou bloqueada (está esperando algum evento como, por exemplo, uma E/S, para passar para o estado pronta para rodar).

Apêndice B

MULTIMEDIA PRESENTATION MANAGER/2

B.1 Introdução

O sistema operacional OS/2 oferece duas ferramentas bastante úteis ao desenvolvimento de aplicações multimídia, são elas o PM (*Presentation Manager*) e o MMPM/2 (*Multimedia Presentation Manager/2*). O primeiro disponibiliza funções para a criação de interfaces homem-máquina e, o segundo, funções para o controle de dispositivos fisicos de mídia.

B.2 O Ambiente *Presentation Manager (PM)*

O ambiente de apresentação do OS/2 2.1 é o *Presentation Manager*. Sua organização é toda baseada no conceito de janelas que, em suas aplicações, não são apenas áreas retangulares na tela do monitor. Elas são um conjunto de estruturas, filas, dados, e características que definem como a aplicação, o usuário, e o sistema operacional vão se interagir. As janelas de uma aplicação são os meios pelos quais a aplicação e o usuário irão se comunicar.

B.3 O Software Multimedia Presentation Manager/2 (MMPM/2)

O *Multimedia Presentation Manager/2* versão 1.1 é uma ferramenta que permite aos desenvolvedores de aplicações tornarem-se mais produtivos em relação ao desenvolvimento de programas multimídia para o OS/2. Esta facilidade é provida através de uma interface chamada de *Media Control Interface (MCI)*. A MCI fornece às aplicações serviços para o envio de mensagens de controle aos dispositivos multimídia através das funções de uma *Application Programming Interface (API)*. Desta forma, é possível o desenvolvimento de aplicações sem se preocupar com o tipo de mídia ou *hardware* envolvidos, pois a programação fica independente destes.

O envio de mensagens aos dispositivos multimídia é feito através da interface de mensagem procedural chamada de mciSendCommand. Também é possível o envio de mensagens através de uma interface de *string* interpretativa chamada de mciSendString. Mas como o conjunto de mensagens disponíveis para esta interface é menor e mais restrito do que o conjunto de mensagens disponíveis para a interface mciSendCommand. Por este motivo, esta última foi a escolhida para se fazer a implementação. A sintaxe da função mciSendCommand é mostrada no Quadro B-1 (fazendo-se uso da linguagem C).

```
/* esta função envia um comando para o dispositivo de mídia especificado */
USHORT usDevice;     /* ID do dispositivo, retorno da mensagem MCI_OPEN */
USHORT usMessage;    /* a ação (mensagem) a ser executada */
ULONG  pParam1;      /* flags da mensagem. São definidos separadamente para cada */
                     /* mensagem */
PVOID  pParam2;      /* ponteiro para a estrutura de dados da mensagem */
USHORT usUserParm;   /* parâmetro especificado pelo usuário que é retornado quando */
                     /* em mensagens de notificação */
ULONG ulReturn;      /* código de retorno */

ulReturn = mciSendCommand (usDeviceID, usMessage, ulParam1, ulParam2, usUserParam);
```

Quadro B-1 - Sintaxe da função mciSendCommand.

Esta função envia todas as mensagens aos dispositivos multimídia envolvidos na aplicação. O Quadro B-2 exemplifica o envio da mensagem de pausa ao dispositivo identificado por usDeviceID:

O parâmetro usDeviceID é retornado pela mensagem MCI_OPEN, que, inclusive, deve ser a primeira mensagem a ser enviada. O ulParam2 definem os *flags*; em especial, existem dois que podem ser utilizados por quase todas as mensagens: MCI_WAIT e MCI_NOTIFY. O uso do flag MCI_WAIT provoca a espera da conclusão da mensagem para o retorno do controle, ou seja, para a execução das demais linhas de comando do programa. Já, de forma oposta, opera o flag MCI_NOTIFY: o retorno do controle é imediato após o envio da mensagem, mas é possível definir uma janela para

notificação quando a mensagem for completada. Estes dois *flag* são mutuamente exclusivos.

A sintaxe da mensagem MCI_PAUSE, assim como a de qualquer outra, é independente do dispositivo, desta forma, pode-se utilizar a mesma mensagem para controlar dispositivos de áudio ou vídeo indistintamente. A única diferença que deve ocorrer é no uso dos *flags*.

```
USHORT usDeviceID;                          /* identificação do dispositivo */
MCI_GENERIC_PARMS mciGenericParms;  /* estrutura dos parâmetros de mensagem */
                                            /* genérica requerida pela mensagem */
                                            /*  MCI_PAUSE */

mciSendCommand (usDeviceID,                  /* identificador do dispositivo */
                MCI_PAUSE,                   /* mensagem de pausa */
                MCI_WAIT,                    /* flag que inidica que a mensagem deve */
                                             /* ser concluída antes do retorno do controle */
                (PVOID) &mciGenericParms,    /* estrutura de dados para esta mensagem */
                0);                          /* não há mensagem de notificação */
```

Quadro B-2 - Código da função mciSendCommand para o envio da mensagem de pausa a um dispositivo.