CARLOS ALEXANDRE PICCIONI

MODELO E IMPLEMENTAÇÃO DE UM SERVIÇO DE DATACASTING PARA TELEVISÃO DIGITAL

FLORIANÓPOLIS
2005

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

MODELO E IMPLEMENTAÇÃO DE UM SERVIÇO DE DATACASTING PARA TELEVISÃO DIGITAL

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Mestre em Engenharia Elétrica.

CARLOS ALEXANDRE PICCIONI

Florianópolis, Fevereiro de 2005.

MODELO E IMPLEMENTAÇÃO DE UM SERVIÇO DE DATACASTING PARA TELEVISÃO DIGITAL

Carlos Alexandre Piccioni

‘Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica, Área de Concentração em *Automação e Sistemas*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.’

_____

Prof. Carlos Barros Montez, Dr.
Orientador

_____

Prof. Alexandre Trofino Neto, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

_____

Prof. Carlos Barros Montez, Dr.
Presidente

_____

Prof. Lau Cheuk Lung, Dr.

_____

Prof. Roberto Willrich, Dr.

_____

Prof. Rômulo da Silva Oliveira, Dr.

Aos meus pais, por tornar tudo isso possível…

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica.

MODELO E IMPLEMENTAÇÃO DE UM SERVIÇO DE DATACASTING PARA TELEVISÃO DIGITAL

Carlos Alexandre Piccioni

02/2005

Orientador: Prof. Carlos Barros Montez, Dr.
Área de Concentração: Automação e Sistemas
Palavras-chave: Datacasting, Televisão Digital (TVD), TVD Interativa, MPEG-2, DSM-CC
Número de Páginas: 102

A difusão de dados, conhecida como *datacasting*, se mostra como a base para o surgimento de novos serviços na Televisão Digital (TVD). Neste contexto, este trabalho busca propor e implementar um modelo fim a fim de *datacasting*, direcionado à difusão de aplicativos e demais dados correlacionados, que suporte esses novos serviços. O modelo foi baseado em especificações abertas de TVD, assim como em trabalhos encontrados na Literatura. Foram estudados os mecanismos de difusão de dados sobre os padrões de TVD, utilizados pelos sistemas abertos, assim como investigadas suas principais características, vantagens e desvantagens. A implementação foi concretizada através do desenvolvimento de parte dos componentes previstos no modelo, e da integração destes com determinadas soluções de *softwares* livres já existentes.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

MODEL AND IMPLEMENTATION OF A DATACASTING SERVICE FOR DIGITAL TELEVISION

Carlos Alexandre Piccioni

02/2005

Advisor: Prof. Carlos Barros Montez, Dr.
Area of Concentration: Automation and Systems
Key words: Datacasting, Digital Television (DTV), Interactive DTV, MPEG-2, DSM-CC
Number of Pages: 102

The Data Broadcasting, or datacasting for short, is the base of new services in Digital Television (DTV). In this context, this work considers and implements an end-to-end datacasting model, destined for the broadcast of applications and its correlated data, to support these new services. The model was based on DTV's open specifications, as well in works found in Literature. The datacasting mechanisms, which are also adopted by the DTV open systems, and its main characteristics, advantages and disadvantages, were investigated. This model was implemented, with the development of some DTV components, and the integration of others pre-existents.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com a evolução das tecnologias baseadas na digitalização da informação, diversos novos serviços foram criados nos últimos anos, como por exemplo, a Internet. Outros serviços migraram ou estão a caminho de se digitalizarem totalmente. Esse é o caso da televisão.

1.1 Motivação

A *Televisão Digital*, ou *TVD*, ainda é um termo desconhecido para muitas pessoas. Na última década, consórcios ao redor do mundo, formados por instituições de pesquisa, governos e empresas, surgiram buscando definir sistemas abertos de televisão digital. As primeiras especificações de TVD datam do início dos anos noventa. Padrões de televisão digital para radiodifusão, o mecanismo de difusão mais comum no Brasil na televisão analógica, foram aprovados apenas no final da década passada. Dessa forma, é seguro afirmar que a Televisão Digital realmente é um assunto recente.

São vários os setores acadêmicos e da indústria envolvidos no desenvolvimento de tecnologias de TVD. Podemos citar universidades, produtores de conteúdo, agregadores de conteúdo, difusores (*broadcasters*), fabricantes de *hardware* e desenvolvedores de *software*, e principalmente as organizações padronizadoras. Essas últimas exercem papel fundamental nesse contexto, visto que através da especificação de padrões abertos tornam possível a competição horizontal nesse mercado, ao contrário do que ocorre em sistemas proprietários.

Com o uso de padrões abertos na TVD várias oportunidades de pesquisas vêm surgido recentemente. Grande parte delas diz respeito a uma das principais características da televisão digital: a possibilidade de se difundir dados no mesmo meio de transporte utilizado pela programação audiovisual normal. Essa propriedade é conhecida como *Data Broadcasting*, ou *datacasting*.

O *datacasting* possibilita que aplicações sejam transmitidas até o receptor do telespectador, sendo a base para sistemas interativos de televisão digital. Também permite que dados independentes

da programação televisiva sejam difundidos, ou seja, as tecnologias de televisão digital podem ser utilizadas em aplicações além daquelas de simples entretenimento.

Atualmente, grande parte dos trabalhos na área é direcionado para o estudo do *datacasting* através de datagramas IP encapsulados no sinal da televisão digital [1]. O interesse por essa área se deve ao fato de que é desejada, por parte desses pesquisadores, a utilização de redes de difusão como suporte às redes IP. Outro fator motivador dessas pesquisas é o endereçamento de conteúdo, possível através do protocolo IP, aos usuários da TVD.

Além do *datacasting* através do encapsulamento de datagramas IP no sinal da TVD, outras formas de difusão de dados são possíveis. Um dos principais deles é conhecido como carrossel [2] [3]. São dois tipos básicos de carrosséis padronizados: *carrossel de dados* e *carrossel de objetos*. Ambos são adotados pela maioria dos sistemas abertos. Diferente do *datacasting* de datagramas IP, que surgiu voltado para redes bidirecionais, os carrosséis foram desenvolvidos tendo como alvo as redes unidirecionais de televisão digital.

Determinados padrões e especificações de tecnologias e *softwares* são adotados por praticamente todos sistemas de TVD abertos. Alguns desses são especificações para *datacasting*. Assim, vários estudos abordam questões relativas aos mesmos ([4], [5], [6], [7], [8], [9], [10], [11], [12] e [1]), como modelos de negócios para serviços de difusão de dados, arquiteturas de sistemas de *datacasting*, etc.

Através do *datacasting* a TVD passa a possuir uma grande capacidade de implementação de novos serviços. Portanto, também existe, atualmente, uma forte atividade da comunidade de pesquisa no sentido de propor aplicações e serviços inovadores usando tecnologias relacionadas a televisão digital. Com a possibilidade de difusão de dados, habilitando a implementação de serviços interativos na TVD, padrões em cada sistema foram sugeridos também com o intuito de especificar plataformas interativas. Na maioria dessas especificações, o carrossel de objetos foi definido como o padrão de *facto* para a transmissão de aplicações e seus recursos.

1.2 Objetivos

O principal objetivo deste trabalho é propor e implementar um modelo simplificado de *datacasting*, baseado em modelos para a TVD já existentes, utilizando carrosséis para o transporte de dados assim como outras formas de difusão de dados. O interesse desta pesquisa não se concentrou em apenas um dos pontos finais (*endpoints*) do sistema (multiplexador ou codificador, etc). O modelo proposto também prevê a existência no receptor de um ambiente de execução de aplicação, a qual ao ser difundida, é decodificada e executada neste, e a mesma pode também acessar os demais dados difundidos.

Diferentemente de grande parte das pesquisas que atualmente estão interessadas em suportar o protocolo IP na TVD, este trabalho se direcionou em explorar a potencialidade do uso dos carrosséis em serviços de *datacasting*. Por fim, outro objetivo desse trabalho é a implementação do modelo proposto.

Segundo [11], os principais desafios relacionados a implementação de soluções de *datacasting* fim a fim são: determinação do modelo de negócios; controle do fluxo de dados; gerenciamento da banda de difusão; segurança, prevenção de erros e compressão. Este trabalho, por outro lado, se concentra apenas nos aspectos relacionados aos fluxos de dados, como codificação, multiplexação, etc. Os outros desafios citados ficam como sugestão de continuação deste trabalho.

1.3 Metodologia

A metodologia adotada foi, inicialmente, levantar todas as propostas e especificações existentes relacionadas ao *datacasting*. Estudou-se também ferramentas disponíveis para o desenvolvimento do trabalho, como por exemplo, emuladores de *set-top boxes*, codificadores por *software*, etc.

Algumas das ferramentas analisadas não fazem parte deste texto. Outras por sua vez foram usadas de forma a verificar se as mesmas seguiam corretamente os padrões abertos de TVD. No decorrer do trabalho foram consultados também os desenvolvedores dessas aplicações, assim como especialistas na área de televisão digital.

A partir da literatura estudada, foi proposto e desenvolvido um modelo de *datacasting* fim a fim. Sua implementação foi realizada em um PC, com o desenvolvimento de soluções de *software* para suprir tarefas geralmente realizadas por *hardware* especializado em sistemas de TVD. Alguns componentes do modelo por sua vez foram implementados através de *softwares* livres já existentes para o sistema operacional utilizado, o GNU/Linux.

1.4 Organização do Texto

Este trabalho, no capítulo 2, introduz o conceito de televisão digital, descrevendo suas principais características assim como os sistemas abertos existentes atualmente. O capítulo 3, por sua vez, apresenta o padrão MPEG-2, base dos sistemas abertos apresentados no capítulo anterior, focando no suporte deste padrão ao *datacasting*. O capítulo 4 descreve os principais mecanismos de *datacasting* sobre o MPEG-2 citados na Literatura, assim como algumas taxonomias relativas ao assunto. O capítulo 5 apresenta o modelo proposto neste trabalho, baseado em algumas das formas de *datacasting* apresentadas no capítulo 4, utilizando os padrões abertos descritos no capítulo 3 de forma a tornar possível algumas aplicações sobre a TVD. O capítulo 6 apresenta a implementação deste modelo,

assim como os mecanismos adotados para tal. Por fim, o capítulo 7 finaliza esta dissertação com as conclusões pertinentes ao trabalho.

Capítulo 2

Televisão Digital

Este capítulo tem por objetivo apresentar uma visão geral sobre a televisão digital. Alguns dos principais componentes dessa nova tecnologia são apresentados, assim como os sistemas abertos de televisão digital atualmente em desenvolvimento, de forma a possibilitar o entendimento das questões introduzidas nos próximos capítulos com relação ao *datacasting*.

2.1 Introdução à Televisão Digital

O primeiro conceito a se entender sobre televisão, tanto digital quanto analógica, é o de difusão. A difusão é definida como um serviço de entrega de programas ou serviços, televisivos ou de rádio, para consumidores que possuem equipamento apropriado para a recepção dos mesmos. Essa entrega pode ser efetuada através de vários meios, como via radiodifusão, cabo, satélite, ou outros meios, ou a combinação destes [7].

As redes televisivas no princípio utilizavam-se de ferramentas analógicas tanto para captura de sinais, armazenamento, edição e difusão. Porém, nos últimos anos, a tecnologia digital passou a fazer parte das diversas fases de manipulação de conteúdo. Contudo, as técnicas de codificação digital atualmente são pouco empregadas no processo de difusão.

Nos últimos anos foi crescente o interesse pelo uso da tecnologia digital para a substituição da difusão puramente analógica. Verificou-se que a mesma apresenta várias vantagens em relação à tecnologia atual [13]. As primeiras vantagens dizem respeito a uma melhor qualidade de áudio e vídeo e um melhor uso do espectro eletromagnético. Devido a métodos de compressão empregados sobre fluxos digitais de vídeo e áudio, um maior número de informações pode ser transmitido se comparado com a difusão analógica. Em uma mesma faixa do espectro de freqüência por onde atualmente é difundido um canal analógico, em torno de 6 MHz em uma difusão terrestre, podem ser transmitidos

no mínimo quatro canais[1] codificados e comprimidos digitalmente com qualidade similar à atual. Ou também, pela mesma largura de banda, pode ser enviado ao menos um canal de alta definição, conhecido como HDTV (*High Definition Television*) [14].

A difusão digital necessita também de uma menor potência para a transmissão que a analógica, além de possuir uma maior tolerância a interferências eletromagnéticas. Qualquer tipo de interferência em um sinal de televisão analógica prejudica diretamente a recepção, resultando na presença de chuviscos ou fantasmas na imagem, além de ruídos no som. Em um sinal codificado digitalmente, pequenas interferências sobre o mesmo geralmente não alteram as unidades de informações, os *bits*. Além disso, mecanismos de correção de erros podem reconstruir o sinal original caso alguns *bits* sejam alterados. Caso a interferência seja relativamente forte, a ponto de alterar vários *bits* em um fluxo de dados, o receptor passa a não reconhecer mais o formato da informação e interrompe a exibição. Dessa forma, em uma recepção digital, a qualidade na recepção é constante, ou caso a interferência ultrapasse o limite aceitável, a recepção é interrompida [13].

Além dessas vantagens, deve-se destacar uma de grande importância: a possibilidade de se difundir, além de vídeo e áudio, qualquer outro tipo de informação digital. Como todo o conteúdo a ser difundido, na TVD, é codificado digitalmente, qualquer outro tipo de informação digital também pode ser difundida. Dessa forma vários outros serviços podem ser implementados através da TVD [14]. Essa característica da televisão digital, de possibilitar a difusão de dados, conhecida também como *datacasting*, é o foco deste trabalho e será discutida ao longo dos próximos capítulos.

2.2 Componentes da Televisão Digital

Existem vários modelos propostos para sistemas de televisão digital, formando cadeias de valores nas quais os componentes das mesmas exercem papéis variados. A cadeia de valores mais comum é citada em [15], válida também para a televisão analógica, e apresentada na figura 2.1.

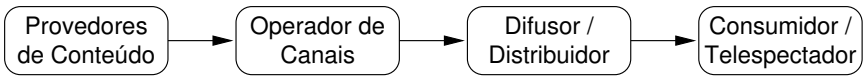

Figura 2.1: *Cadeia de valores típica em sistemas de televisão.*

O telespectador, ou consumidor, termo utilizado na televisão paga, é o usuário dos serviços da TVD. Esses serviços são entregues através de um distribuidor ou difusor. O distribuidor geralmente fornece seus serviços a um operador de canal, o qual é responsável por agregar conteúdo de vários provedores, e entregá-los ou vendê-los ao telespectador ou consumidor.

[1] A quantidade de canais possível de ser difundida em determinada largura de banda depende do mecanismo de modulação do sinal digital e do CODEC empregado na codificação de vídeo e áudio. Na difusão terrestre, dados os atuais padrões de modulação, é comum citar o número quatro como o número de canais a serem difundidos utilizando-se o padrão MPEG-2 para a codificação de mídia com qualidade similar à da televisão analógica

Uma única organização pode exercer mais de um papel na cadeia de valores apresentada na figura 2.1. Por exemplo, pode deter tanto os meios de difusão como a operação dos canais. É o que geralmente ocorre em sistemas de televisão a cabo. Em sistemas de televisão via satélite, geralmente, o operador de canais não detêm o meio de transmissão. Já na radiodifusão aberta, um operador de canal pode deter os três primeiros componentes da cadeia de valores: produção, operação e distribuição.

Além dos componentes da cadeia de valores apresentados na figura 2.1, outras entidades também exercem papel fundamental no desenvolvimento de sistemas de televisão digital. São os desenvolvedores de *hardware*, *software* e os órgãos padronizadores. Esses últimos são de grande importância para a televisão digital, pois permitem uma competição horizontal em sistemas de televisão digital ao especificarem padrões abertos como base para a tecnologia envolvida nesse contexto. Algumas dessas especificações são apresentadas no decorrer deste trabalho.

Independente do modelo de negócios adotado em um sistema de televisão digital, a informação a ser entregue ao consumidor passa por várias etapas desde em sua geração, assim como por algum processamento no receptor. Os próximos tópicos apresentam essas etapas básicas.

2.2.1 Geração do Sinal

O processo de construção do sinal a ser difundido é dividido em uma série de etapas. A Figura 2.2 ilustra um modelo simplificado desse processo. A primeira etapa é a codificação e compressão da informação seguindo um padrão digital bem definido [2]. A saída de cada codificador é um fluxo de dados denominado *fluxo elementar*. Quando vários fluxos elementares são relacionados entre si, como por exemplo, seqüências de vídeo com seqüências de áudio e possivelmente também com seqüências de dados, eles formam um *serviço* [16]. O conceito de serviço para a televisão digital é similar ao de um canal para a televisão analógica.

Na TVD, todos os serviços são multiplexados em uma seqüência de dados denominada de *fluxo de transporte*. Essa tarefa é geralmente implementada por um sistema encarregado de multiplexar além dos fluxos de vídeo, áudio e dados, tabelas que descrevem apropriadamente o conjunto de serviços transportados em um fluxo de transporte.

Na próxima etapa, o fluxo de transporte é então modulado em uma onda portadora e difundido via satélite, cabo ou radiodifusão. Ainda em fase de testes, existem soluções como redes digitais de alta velocidade para a transmissão do fluxo de transporte ao invés dos meios citados anteriormente. São baseadas em tecnologias já existentes, como xDSL, LANs, *Wireless* LANs, etc [15].

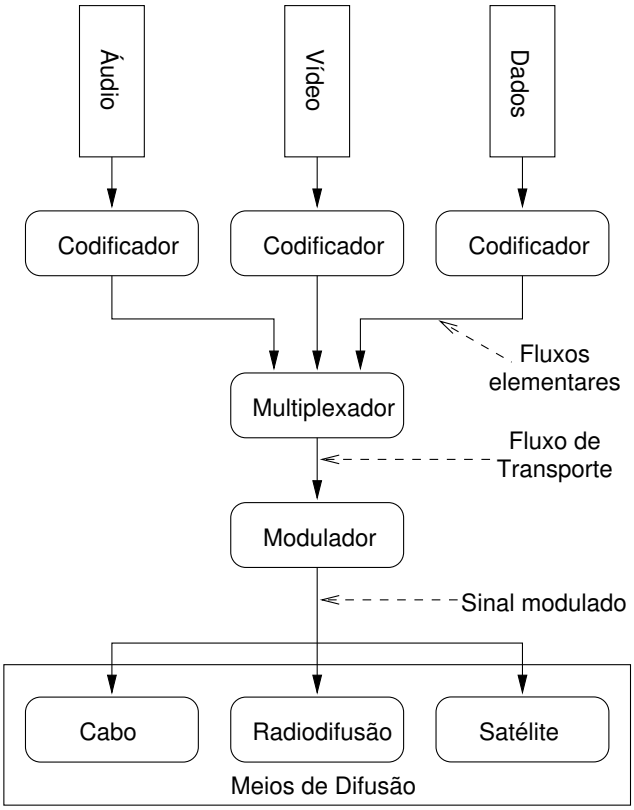

Figura 2.2: *Diagrama simplificado das etapas da difusão na televisão digital.*

2.2.2 Recepção do Sinal

O equipamento de recepção pode ser integrado a uma televisão, a televisão digital, ou ser um equipamento à parte. Nesse último caso, o mesmo é conhecido industrialmente como *set-top box*. A idéia básica desse dispositivo, quando não embutido em uma televisão digital, é o de uma pequena caixa com a função de processar e converter os sinais digitais para o formato analógico utilizado pelas televisões atuais. Uma terceira opção é também a existência de placas ou cartões acoplados a PCs com a função de recepção do sinal da TVD.

Neste trabalho é denominado *terminal de acesso* o equipamento responsável pelas tarefas comuns de um receptor de televisão digital. As principais etapas de processamento do sinal em um terminal de acesso são ilustradas na figura 2.3.

Antes de ser processado pelo terminal de acesso, o sinal difundido é captado por uma antena específica para a tecnologia, no caso da difusão via satélite ou radiodifusão, e isolado em determinado canal pelo sintonizador. O sinal passa então pelo demodulador, que restaura o fluxo de transporte difundido.

O demultiplexador é encarregado de extrair os fluxos elementares de determinado serviço,

alimentando os respectivos decodificadores. Geralmente existe um decodificador para áudio e um para vídeo, e o correto tratamento dos outros dados depende da implementação do sistema. O áudio decodificado é encaminhado para a saída de áudio do terminal de acesso, assim como o vídeo é direcionado para a tela de exibição. Outros tipos de dados são processados, executados e exibidos se necessários. Dependendo da implementação, o usuário pode interagir com alguns serviços.

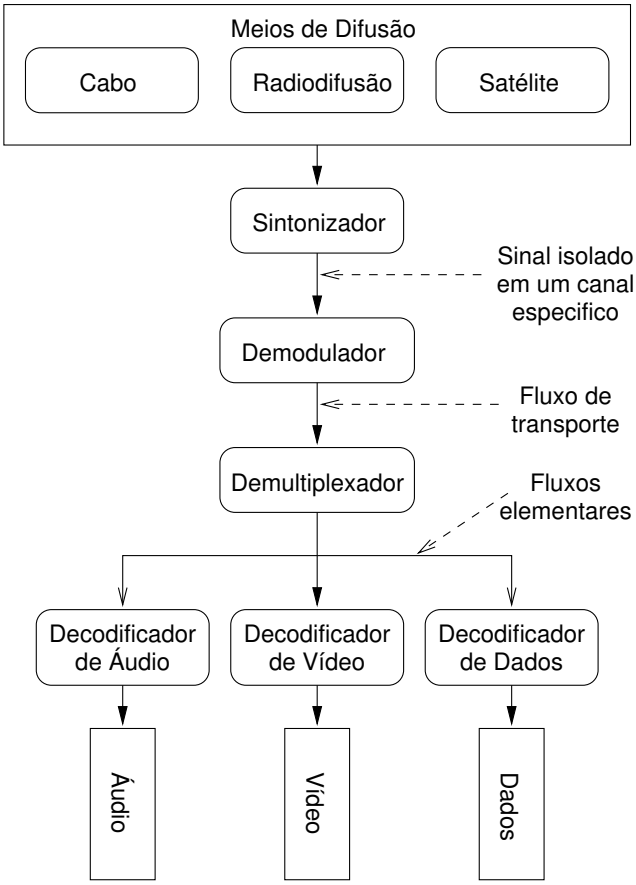

Figura 2.3: *Diagrama simplificado das etapas da recepção na televisão digital.*

2.2.3 Canal de Retorno

A televisão digital pode estabelecer mais de um meio de comunicação entre o produtor de conteúdo ou o difusor e o terminal de acesso. O canal de retorno, ou canal de interatividade para alguns autores [13], é esse meio por onde é possível a troca de informações no sentido inverso da difusão [17].

Várias tecnologias para o canal de retorno estão sendo desenvolvidas e testadas atualmente. A mais simples atualmente é a que utiliza a telefonia fixa. Apesar da sua largura de banda ser relativamente baixa, é considerada suficiente para a maioria das aplicações interativas atualmente previstas.

Outras tecnologias que também utilizam a linha telefônica também podem ser utilizadas, como o ADSL (*Assimetric Digital Subscriber Line*).

Existem também esforços com relação à utilização dos próprios meios de difusão como suporte ao canal de retorno. Na difusão via cabo, por exemplo, é possível a implementação do canal de retorno através do uso de *cable modems* nos terminais de acesso, fornecendo uma largura de banda de retorno consideravelmente alta. Na radiodifusão é possível o uso da própria largura de banda do meio de difusão, através da própria antena de recepção, para o envio de dados no sentido inverso. Na difusão via satélite, apesar das dificuldades técnicas e dos custos elevados, a implementação de um canal de retorno pode ser concretizada através de uma antena conectando o receptor ao satélite.

Existem também outras tecnologias alternativas em estudo. Uma delas é através das tecnologias de telefonia celular, como GSM (*Global System for Mobile Communication*) ou CDMA (*Code-Division Multiple Access*). Outra é o LMDS (*Local Multipoint Distribution System*), uma alternativa que utiliza sinais de microondas de curto alcance. Também, ainda em estudo, existe o PLC (*Power Line Communication*), que se propõe a utilizar a rede elétrica para a transmissão de dados. Porém, essa última tecnologia já está sendo pesquisada há mais de 30 anos e os resultados alcançados ainda não são satisfatórios [13].

2.3 Terminal de Acesso

Uma das principais características de um terminal de acesso é a capacidade de processamento. Sua arquitetura é semelhante a de um microcomputador. A princípio, possui uma menor capacidade de processamento e armazenamento, porém possui componentes de *hardware* específicos para um ambiente de televisão digital. Suas principais características de *hardware* e *software* são apresentadas a seguir.

2.3.1 *Hardware*

O terminal de acesso possui uma unidade central de processamento, CPU, responsável pela execução de determinado código, além de outras funções como administração de interrupções de *hardware*, gerenciamento de memória, e etc. Além da CPU, o terminal de acesso possuí também um processador gráfico responsável pelo processamento dos fluxos de vídeo e de outros elementos gráficos [18].

Com relação à memória, um terminal de acesso possui tanto a tipo RAM, necessária para o armazenamento temporário de aplicações em execução e de dados, quanto memória ROM, geralmente dos tipos *Flash* ou EEPROM, para o armazenamento permanente de aplicativos e configurações do aparelho.

Os terminais de acesso podem possuir também unidades de armazenamento de grande capacidade, utilizando interfaces comuns em um PC, como a IDE, além de unidades externas de armazenamento. Dessa forma, o terminal de acesso pode ter funções de armazenamento de fluxos audiovisuais, funcionando de uma forma similar a um vídeo-cassete digital. Um terminal com tal capacidade é conhecido como VDR (*Video Digital Recorder*).

Para a implementação do canal de retorno, o terminal de acesso pode possuir um *modem* que forneça suporte às tecnologias citadas na seção 2.2.3. Pode também apresentar várias outras interfaces para a comunicação com o mundo externo. Podemos citar dentre elas as seguintes: IEEE-1284, ou interface paralela; RS-232, a interface serial; USB; IEEE-1394, ou *Firewire*; 10/100 Base-T, utilizado pela *Ethernet*; PCMCIA; *SmartCards*; e controles remotos e teclados sem fio [18].

2.3.2 *Software*

Geralmente os componentes de *software* em um terminal de acesso são divididos em camadas como mostrado na figura 2.4. A camada inferior da pilha de software é formada por um conjunto de *devices drivers* necessários para que um sistema operacional de tempo real, a ser executado na camada adjacente, possa controlar o *hardware*. Esse sistema operacional possui características diferentes de um sistema convencional, já que deve estar apto para lidar com recursos mais escassos e com restrições temporais mais fortes que às presentes em um PC.

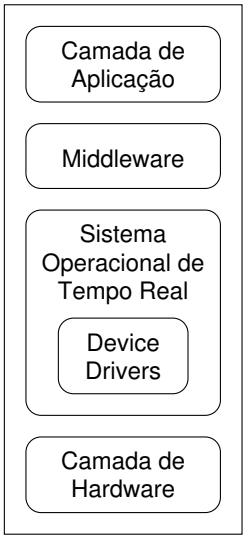

Figura 2.4: *Divisão típica das camadas de software de um terminal de acesso.*

Acima da camada do sistema operacional, encontra-se o *middleware*[2] [14], que é a camada responsável por fornecer uma interface de programação de aplicação, API (*Application Programming*

[2]O conceito de *middleware* em um terminal de acesso pode diferenciar do utilizado nas demais áreas da ciência da computação, como em sistemas distribuídos por exemplo. Neste trabalho é considerado como *middleware* a camada ilustrada na figura 2.4 que possui as funcionalidades apresentadas no decorrer do texto

Interface), às aplicações. Essa interface busca ser independente do *hardware* e das tecnologias de comunicação subjacentes, com a finalidade de afastar o desenvolvedor de aplicações para a televisão digital das peculiaridades de cada sistema operacional, ou do *hardware* de um terminal de acesso.

A última camada é a camada de aplicação, onde residem os aplicativos, que podem ser difundidos em fluxos de transportes, extraídos e executados no receptor. Também podem ser permanentes, ou seja, estarem presentes na ROM do terminal de acesso.

2.4 Sistemas Abertos de Televisão Digital

Atualmente três importantes sistemas abertos para a televisão digital encontram-se em desenvolvimento. O sistema predominantemente europeu, *Digital Vídeo Broadcasting*, o sistema norte-americano, *Advanced Television Systems Committee*, e o sistema japonês, conhecido como *Integrated Services Digital Broadcasting*.

Esses sistemas, DVB, ATSC e ISDB, especificam as etapas apresentadas neste capítulo com relação à produção, difusão e recepção de serviços de televisão digital, através da adoção de padrões para cada uma dessas etapas. Possuem várias similaridades entre si, porém divergem consideravelmente com relação aos padrões adotados na modulação do sinal, além das especificações referentes ao *middleware*. As próximas seções apresentam os principais aspectos técnicos relacionados a esses sistemas [13], com relação às suas especificações para a difusão terrestre no caso dos mecanismos de modulação.

2.4.1 DVB - *Digital Vídeo Broadcasting*

O grupo DVB foi fundado em 1993 através da união de diversas empresas públicas e privadas de vários países europeus, além da Austrália, que o adotou em 2001. O DVB-T, especificação para a radiodifusão, utiliza uma modulação do tipo COFDM (*Coded Orthogonal Frequency Division Multiplexing*), e a taxa de transmissão pode chegar a 31 Mbps, dependendo dos parâmetros utilizados na modulação do sinal. A largura de banda original é de 8 MHz, com a possibilidade de ser escalonado para 6 MHz.

A compressão e codificação de vídeo é feita sobre o padrão MPEG-2 video[3]. A codificação de áudio é realizada através do padrão MPEG-2 *Layer II Audio*. A multiplexação segue o padrão *MPEG-2 Sistemas*. O *middleware* utilizado é conhecido como *Multimedia Home Plataform*, ou MHP.

[3]As últimas especificações do sistema DVB incluem o uso do padrão MPEG-4 Parte 10, também conhecido como H.264, para a codificação de vídeo, assim como o padrão MPEG-4 AAC para o áudio.

Uma de suas vantagens, principalmente sobre o padrão ATSC, é uma maior imunidade aos problemas de multi-percurso[4], além da possibilidade de recepção móvel. Um desvantagem, porém, é a baixa imunidade à interferência causadas por equipamentos eletrônicos.

O país destaque no uso do DVB é a Inglaterra, o qual já possuí mais de um milhão de usuários. Neste e nos demais países, parte da televisão digital terrestre é um serviço pago, e os terminais de acesso são subsidiados pelas operadoras televisivas.

2.4.2 ATSC - *Advanced Television Systems Committee*

Esse sistema se consolidou em novembro de 1998, e além dos Estados Unidos, foi adotado pelo Canadá, México, Coréia do Sul e Taiwan. Utiliza a modulação 8-VSB (*Vestigial Side Band*) em uma largura de banda de 6 MHz (com a possibilidade de ser escalonado para 8 MHz), alcançando uma taxa de transmissão de 19,4 Mbps.

A codificação de vídeo é feita sobre o padrão MPEG-2 Video. Já a codificação de áudio é realizada através do padrão *Dolby AC-3*. A multiplexação segue o padrão MPEG-2 Sistemas. O *middleware* utilizado é o *DTV Application Software Enviroment*, ou DASE.

O ATSC foi projetado para a recepção por antenas externas, voltado para a televisão de alta definição (HDTV), e não permite em sua versão original recepção móvel. Outra desvantagem é a necessidade de uso de equalizadores para a recepção de sinais em situações de multi-percurso.

Até este momento, este sistema não obteve sucesso basicamente devido aos altos preços dos aparelhos de alta definição e da maioria dos terminais de acesso. Outro fator negativo é o fato de que mais de 80% da população norte-americana possui TV por assinatura via cabo ou via satélite, onde os difusores adotam seus próprios padrões para a migração da tecnologia analógica para a digital.

2.4.3 ISDB - *Integrated Services Digital Broadcasting*

Estabelecido em 1999 por várias empresas e operadoras de televisão japonesas, o ISDB-T é o sistema de transmissão terrestre adotado somente por esse país no momento. Utiliza o padrão de modulação COFDM, com algumas variações, e possui uma taxa de transferência de 24 Mbits/s, e uma largura de banda de 6 MHz (escalonável a 8 MHz).

A codificação de vídeo, como nos dois sistemas anteriores, segue o padrão *MPEG-2 Vídeo*. A codificação de áudio utiliza a especificação *MPEG-2 AAC* (*Advanced Audio Coding*). A multiplexação segue o padrão *MPEG2 Sistemas*. O *middleware* é conhecido como ARIB-Std-B24, de

[4] A situação conhecida como *multi-percurso* ocorre quando um mesmo sinal percorre caminhos diferentes, chegando em tempos também diferentes ao terminal de acesso. Dessa forma, o sinal que percorreu um dos caminhos gera interferência sobre o outro. Essa situação é comum em grandes metrópoles, onde o sinal é refletido em grandes construções.

Association of Radio Industries and Businesses. As suas maiores vantagens são a grande flexibilidade de operação e seu maior potencial para transmissões móveis e portáteis que o sistema europeu, além da maior imunidade ao problema de multi-percurso que o sistema americano.

2.5 *Middlewares* para Televisão Digital Interativa

Como citado anteriormente, neste texto considera-se como um *middleware* a camada de *software* encarregada de fornecer uma interface de programação de aplicação, API, comum às aplicações a serem executadas em um ambiente de televisão digital, independente do sistema operacional e do *hardware* do terminal de acesso. A existência desse tipo de *middleware* se deve ao modelo de negócios visado pelos sistemas de televisão digital, onde é desejado que uma aplicação criada para determinada região funcione também em outras, ou que funcione para diversos terminais de acesso de diferentes fabricantes. Dessa forma, com sistemas operacionais também diferentes, é função do *middleware* esconder as heterogeneidades de *hardware* e *software* da aplicação. Assim, caso uma aplicação siga estritamente as especificações de determinado *middleware*, poderá ser executada em qualquer terminal de acesso que o possua implementado [14].

Ao menos quatro tecnologias de *middlewares* proprietárias estão disponíveis atualmente: *Canal+ Media Hightway*; *Liberate Technologies*; *Microsoft TV* e *OpenTV*. Além desses, os três sistemas abertos de televisão digital trabalham em suas especificações abertas de *middleware*. Concorrem a especificação européia MHP, a americana DASE e a japonesa ARIB [13]. O MHP é apresentado a seguir, pois atualmente é uma especificação aberta com um grande número de funcionalidades e também grande aceitação na pesquisa e desenvolvimento de sistemas de televisão digital interativa [16]. Apesar de pertencerem a outros sistemas de televisão digital, as outras duas especificações de *middleware* possuem várias similaridades com a especificação européia.

2.5.1 MHP - *Multimedia Home Platform*

Um *middleware* MHP é composto basicamente por uma máquina virtual Java, capaz de executar *bytecodes* dessa linguagem, e por um conjunto de bibliotecas encarregadas de fornecer diversas funções às aplicações específicas de um ambiente de televisão [19]. Lançada em junho de 2000 [13], a especificação de versão 1.0.3 define as funcionalidades necessárias em um receptor e o conjunto de APIs que deve fornecer para as aplicações. A versão 1.1 define o uso de APIs adicionais, para suporte a *smart cards*, suporte a recepção de aplicações pelo canal de retorno, além da possibilidade de armazená-las em disco.

Uma das mais importantes APIs do MHP é a JavaTV [20]. Lançada independentemente pela *Sun Microsystems* em 1998, foi adotada também pelo DASE e pela especificação japonesa. Ela é

responsável pelo conceito de *Xlet*, que foi posteriormente adotado no *Java Microedition*, que em um ambiente de televisão interativa equivale a uma *applet* Java em um PC. Assim como a *applet*, que é carregada pelo navegador em determinada URL da Internet, a *Xlet* é difundida em um fluxo de transporte, extraída pelo terminal de acesso, e executada quando necessária.

Assim como uma *applet*, uma *Xlet* possui uma interface que permite que uma fonte externa, no caso o gerenciador de aplicação do terminal de acesso, inicie-a, pare-a, ou a controle de várias outras formas. A figura 2.5 ilustra os estados que uma *Xlet* pode possuir [21]. Primeiramente, o gerenciador de aplicação carrega a classe principal da *Xlet*, difundida em um fluxo de transporte MPEG-2, e cria uma instância da mesma. Assim, a *Xlet* inicia no estado *carregada*.

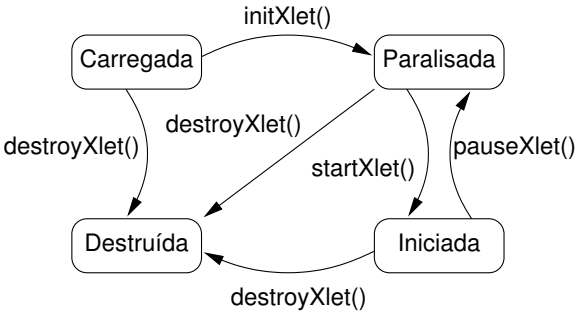

Figura 2.5: *Possíveis estados de uma Xlet.*

No momento em que determinado serviço contendo uma *Xlet* é selecionado no terminal de acesso ou quando outra aplicação determina que a mesma deve ser iniciada, o *gerenciador de aplicação* (*application manager*) do terminal invoca o método de inicialização da *Xlet*, carregando os demais recursos necessários para sua execução. Após isso, a *Xlet* passa para o estado *paralisada*, no qual está pronta para ser iniciada imediatamente.

Quando a *Xlet* retornar do método de inicialização, o gerenciador de aplicação chama outro método para iniciar a mesma. A *Xlet* passa então ao estado *iniciada*, e estará apta para interagir com o telespectador ou com o ambiente onde se encontra. Nesse ponto, o gerenciador de aplicação pode determinar que a *Xlet* deve ser paralisada, retornando ao estado anterior.

Em qualquer um dos estados em que a *Xlet* se encontrar, o gerenciador pode também destruí-la, liberando assim todos os recursos ocupados por ela. O telespectador, ou outra entidade que interage com a *Xlet*, também pode ter o controle sobre seus estados, através de diferentes interfaces, como por exemplo, interfaces visuais da telivisão, na forma de menus, ícones e etc.

A *Xlet* é o item fundamental em termos de interação para o telespectador. Ela pode receber dados de entrada, através do controle remoto ou de algum teclado acoplado ao terminal de acesso, processá-los, usar fluxos elementares presentes no fluxo de transporte, enviar e receber informações pelo canal de retorno, e gerar dados audiovisuais ou de controles de saída.

2.6 Novos Serviços na Televisão Digital

Dadas as características da televisão digital apresentadas neste capítulo, novos serviços podem ser oferecidos ao telespectador típico ou até mesmo a outros clientes de um sistema de TVD. Um primeiro exemplo são os serviços interativos. Um primeiro tipo de interatividade pode ocorrer sem a presença do canal de retorno. Devido à capacidade de difusão dos mais diversos tipos de dados, e da capacidade de processamento de terminal de acesso, aplicativos podem ser transmitidos e executados no receptor. Dessa forma, por exemplo, jogos podem ser executados, legendas em várias línguas disponibilizadas, ou informações detalhadas sobre certos produtos podem ser difundidos com seu comercial e acessados pelo telespectador.

O conceito de interatividade na televisão se torna mais amplo com a presença do canal de retorno, por onde o telespectador pode retornar algum tipo de informação. As aplicações imaginadas nesse contexto dependem fortemente da capacidade do canal de retorno e da integração da entidade responsável por gerenciar a rede de suporte a esse meio com o difusor do sinal televisivo. Algumas aplicações possíveis de implementação são: navegação *web*; correio eletrônico (*e-mail*); serviços de mensagens instantâneas (ICQ, msn, etc.); vídeo sob-demanda (*VOD - Video On-Demand*); comércio eletrônico (*t-comercing*); banco eletrônico (*t-banking*); governo eletrônico (*t-government*); bate-papo (*chat*) e jogos.

Outras aplicações onde o destinatário do serviço é um cliente diferente de um telespectador normal são previstas. Como exemplos podemos citar o uso do canal de difusão como meio de suporte ao acesso à Internet; meio de distribuição de conteúdo digital diversificado; suporte ao estabelecimento de redes entre corporações; envio de informações de controle ou de atualização a equipamentos remotos, inclusive móveis, dentre outros. Alguns desses exemplos são abordados também no capítulo 5.

A base de todos esses serviços, por sua vez, é a difusão de diferentes tipos de dados, conhecida como *datacasting*. Dessa forma, o *datacasting* é considerado como o pilar para a implementação desses serviços, justificativa pela qual o mesmo é tema central desse trabalho e será descrito em mais detalhes nos demais capítulos.

2.7 Considerações Finais

Esse capítulo apresentou as principais características da televisão digital, assim como os três sistemas abertos em desenvolvimento. Conforme foi visto, além de uma série de novas características e de vantagens, a televisão digital possibilita o surgimento de novos serviços, dentre eles os dependentes do *datacasting*. O *datacasting* é tema fundamental nesta dissertação e será visto em detalhes no capítulo 4.

Como apresentado na seção 2.4, o padrão adotado pelos sistemas DVB, ATSC e ISDB, para a multiplexação de fluxos elementares em um fluxo de transporte é a especificação *MPEG-2 Sistemas*. Dessa forma, essa especificação é objetivo de estudo do próximo capítulo, assim como outras extensões do MPEG-2 relevantes para a difusão de dados em sistemas de televisão digital.

Capítulo 3

MPEG-2

Os sistemas de televisão digital abertos citados no capítulo 2 são compostos por conjuntos de especificações, dentre os quais os produzidos pela MPEG desempenham papel central. A MPEG, ou *Moving Pictures Experts Groups*, é um grupo de trabalho da ISO/IEC responsável pelo desenvolvimento de padrões principalmente para a codificação de áudio e vídeo [22]. Desde 1988 são produzidas diversas especificações, como o MPEG-1, direcionado ao suporte de vídeo em qualidade de CD, o MPEG-2, para a televisão digital e DVD, o MPEG-4, padrão para multimídia em geral, o MPEG-7, para a descrição e busca de conteúdo audiovisual, e o MPEG-21, também para multimídia.

Além de especificações para codificação de áudio e vídeo, o padrão MPEG-2 define a forma como os vários serviços de televisão digital são multiplexados, sendo adotado pelos sistemas DVB, ATSC e ISDB. Dessa forma, a difusão de dados é fortemente dependente dessa especificação, a qual é objetivo de estudo deste capítulo. A seção 3.1 apresenta uma visão geral do padrão MPEG-2, enquanto as seções 3.2 e 3.3 descrevem respectivamente as duas partes do padrão MPEG-2 mais importantes para este trabalho: o *MPEG-2 Sistemas* e o *Digital Storage Media Command and Control*, ou *DSM-CC*.

3.1 O Padrão MPEG-2

O MPEG-2 é um conjunto de nove especificações. A primeira parte, conhecida como ISO/IEC 13818-1 ou *MPEG-2 Sistemas* (*MPEG-2 Systems*), define a estrutura e a sintaxe do fluxo de transporte da chamada *camada de sistema*. A camada de sistema é responsável pela multiplexação em um único fluxo de *bits*, apropriado para a transmissão ou armazenamento dos fluxos de vídeo, áudio e dados da camada adjacente, a chamada *camada de compressão*. As duas especificações seguintes, ISO/IEC 13818-1 e 13818-2, ou respectivamente, *MPEG-2 Vídeo* e *MPEG-2 Áudio*, especificam a codificação de mídia audiovisual da camada de compressão.

Das partes restantes, a de maior interesse deste trabalho é a ISO/IEC 13818-6, ou *Extensões para o DSM-CC*. O DSM-CC, *Digital Storage Media Command and Control*, padroniza um conjunto de protocolos que fornece funções de controle para o gerenciamento de fluxos de *bits* MPEG-2 [22]. Partes dessa especificação são adotadas pelos sistemas de televisão digital. Na seção 3.3 é apresentada uma visão geral desse padrão, assim como são descritos dois de seus protocolos para a difusão de dados em fluxos de transporte: os *carrosséis de dados* e os *carrosséis de objetos*.

3.2 MPEG-2 Sistemas

O *MPEG-2 Sistemas* especifica o formato do fluxo de *bits* resultante da multiplexação dos fluxos da camada de compressão. Essa especificação não define, porém, como devem ser projetados os equipamentos multiplexadores e demultiplexadores. Existem duas formas distintas de multiplexação: através de *Fluxos de Transporte* (*Transport Streams*) ou de *Fluxos de Programa* (*Program Streams*) [23]. A figura 3.1 ilustra o escopo de trabalho da especificação *MPEG-2 Sistemas*.

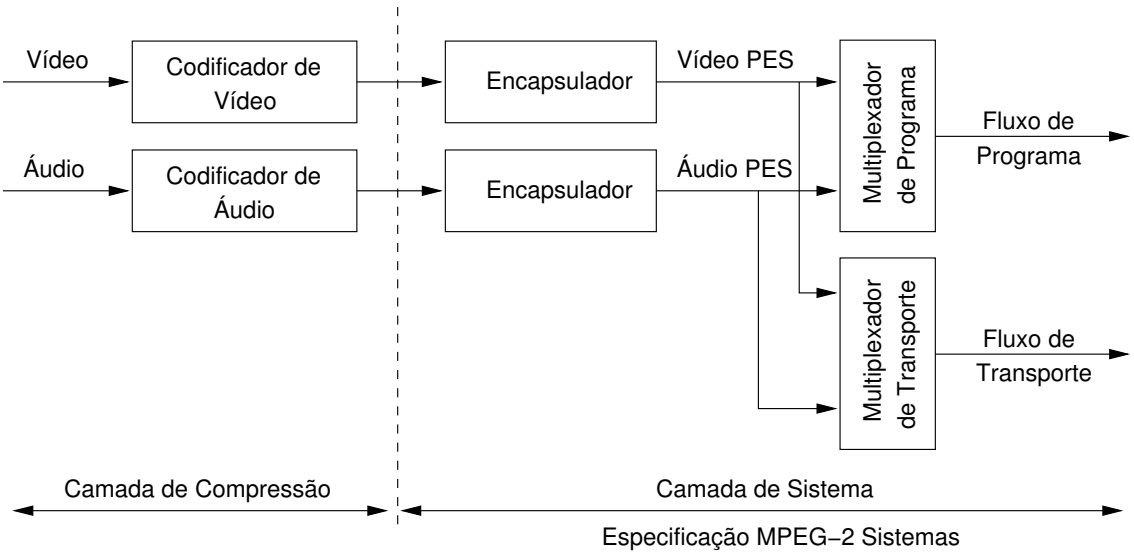

Figura 3.1: *Escopo simplificado da especificação MPEG-2 Sistemas.*

Um *Fluxo de Programa* é otimizado para o uso em ambientes com menor ocorrência de erros, como por exemplo, em sistemas de armazenamento de conteúdo multimídia. É o sistema de multiplexação utilizado atualmente pelos DVDs. O *Fluxo de Transporte* é otimizado para ambientes onde a presença de erros é mais freqüente, como em difusões ou em outras transmissões suscetíveis a ruídos, um dos motivos pelo qual foi adotado DVB, ATSC e ISDB, seja na difusão via cabo, terrestre ou satélite.

3.2.1 Fluxo de Transporte

Em um fluxo de transporte os fluxos elementares de áudio, vídeo e demais dados são organizados logicamente formando unidades conhecidas como programas ou *serviços*[1]. Cada serviço pode conter um ou vários fluxos de vídeo, áudio e dados. Os serviços são equivalentes aos canais na televisão analógica. A figura 3.2 ilustra a organização lógica dos diversos fluxos elementares em serviços em um Fluxo de Transporte.

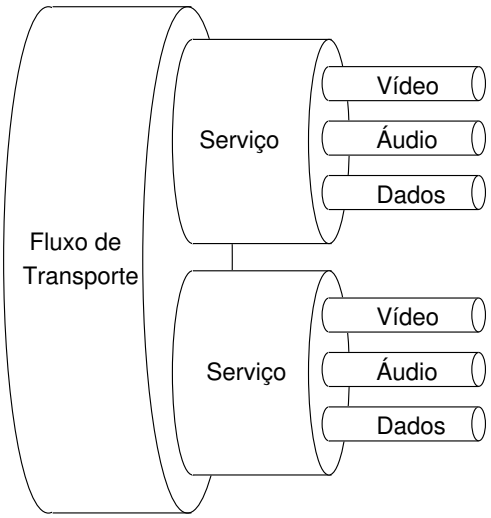

Figura 3.2: *Estrutura lógica de um fluxo de transporte.*

Um Fluxo de Transporte é dividido em duas camadas de encapsulamento de dados. A camada inferior é formada por *pacotes de transporte*, onde cada pacote é encarregado de carregar um fragmento de determinado fluxo elementar ou estrutura da camada imediatamente superior. As estruturas da camada superior são divididas em dois tipos, *seções* e *PES* (*Packetized Elementary Streams*). Essa camada fornece funcionalidades adicionais à camada inferior, como sincronização e descrição de serviços. A figura 3.3 ilustra a pilha de estruturas presentes em um fluxo de transporte, que são descritas nos próximos tópicos.

3.2.2 Pacotes de Transporte

Cada pacote de transporte possui um tamanho fixo de 188 *bytes*. Esse valor se deve ao fato de que em redes ATM, cada célula possui 47 *bytes* de carga, ou seja, um pacote de 188 *bytes* pode ser fragmentado em exatamente quatro células. Cada pacote de transporte possui um cabeçalho obrigatório de no mínimo 4 *bytes*, que pode ser estendido até o final do pacote. Os *bytes* posteriores ao cabeçalho, chamados de carga, transportam as estruturas da camada superior, seções ou PES, que

[1]Um *Programa* para o MPEG-2 corresponde a um *Serviço* para o sistema DVB. *Serviço* será o termo utilizado para referenciar um conjunto de fluxos elementares relacionados entre si no decorrer deste trabalho

geralmente são fragmentadas em mais de um pacote de transporte. A figura 3.4 ilustra a sintaxe de um pacote de transporte. Os números abaixo de cada campo indicam o número de *bits* ocupados por cada um.

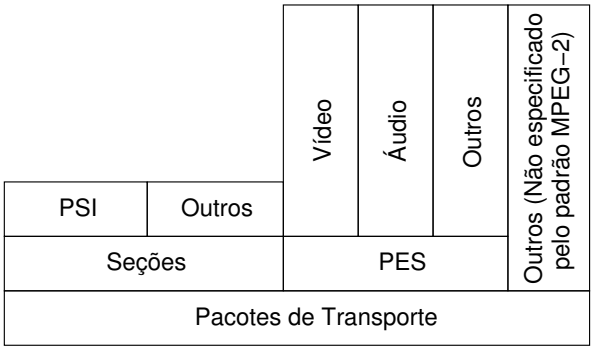

Figura 3.3: *Pilha de camadas de encapsulamento de um fluxo de transporte.*

Como o fluxo de transporte possui apenas pacotes de tamanho fixo dispostos seqüencialmente, o demultiplexador do terminal de acesso, para sincronizar corretamente esse fluxo, deve localizar apenas o primeiro *byte* de determinado pacote de transporte. Essa é a razão do primeiro *byte* de um pacote de transporte ser denominado *byte de sincronismo*, o qual possui um valor fixo. Porém, o mesmo valor pode estar presente esporadicamente em qualquer outro *byte* da carga. Dessa forma, o demultiplexador deve, na primeira tentativa de sincronização, verificar se esse valor realmente é do primeiro *byte* do cabeçalho. Isso é realizado verificando-se no fluxo de transporte se existe realmente outro *byte* de sincronismo a cada 188 *bytes*.

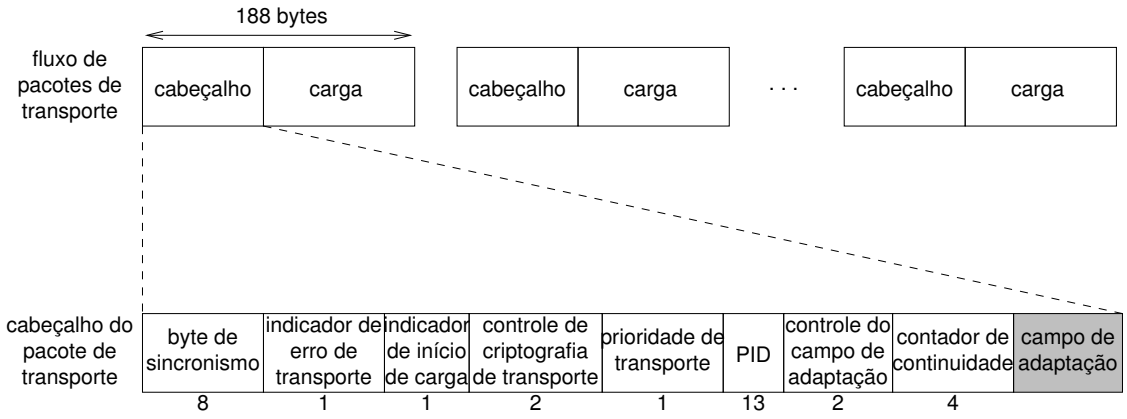

Figura 3.4: *Sintaxe de um pacote de transporte.*

Após ser entregue pelo multiplexador, um pacote de transporte pode sofrer interferências ou estar sujeito a erros nas demais etapas da transmissão. Quando esses erros ou interferências são detectados, e não é possível afirmar se os dados foram afetados ou não, os equipamentos que detectam esses erros alteram um *bit* do pacote de transporte de forma a informar ao demultiplexador que o pacote

apresenta erros indeterminados. Tal *bit* é o *indicador de erro de transporte*, presente no cabeçalho do pacote de transporte.

Devido ao seu pequeno tamanho, 188 *bytes*, um pacote de transporte geralmente carrega apenas fragmentos das estruturas da camada superior, as PES e as seções. Assim, para informar ao demultiplexador quando o pacote de transporte contém o início de uma seção ou de uma PES, ou apenas a continuação dessas estruturas, é usado um *bit* denominado *indicador de início de carga*.

O *MPEG-2 Sistemas* permite que apenas uma PES ou fragmento de uma PES seja encapsulada em um pacote de transporte. Por conseguinte, quando o *indicador de início de carga* for igual a um, o primeiro *byte* após o cabeçalho do pacote de transporte corresponde ao primeiro *byte* da PES. Porém, com relação às seções, é permitido que o início de uma seção seja transportada após o final de outra no mesmo pacote. Dessa forma, diferentemente do que ocorre com as PES, o cabeçalho do pacote de transporte é estendido de forma a incluir um *byte* adicional denominado *ponteiro* quando o *indicador de início de carga* é igual a um. Esse campo contém o número de *bytes* após o final do cabeçalho onde se encontra o início da seção. Nesses pacotes, os *bytes* após o *ponteiro* correspondem ao último fragmento da seção anterior. A figura 3.5 ilustra essa situação.

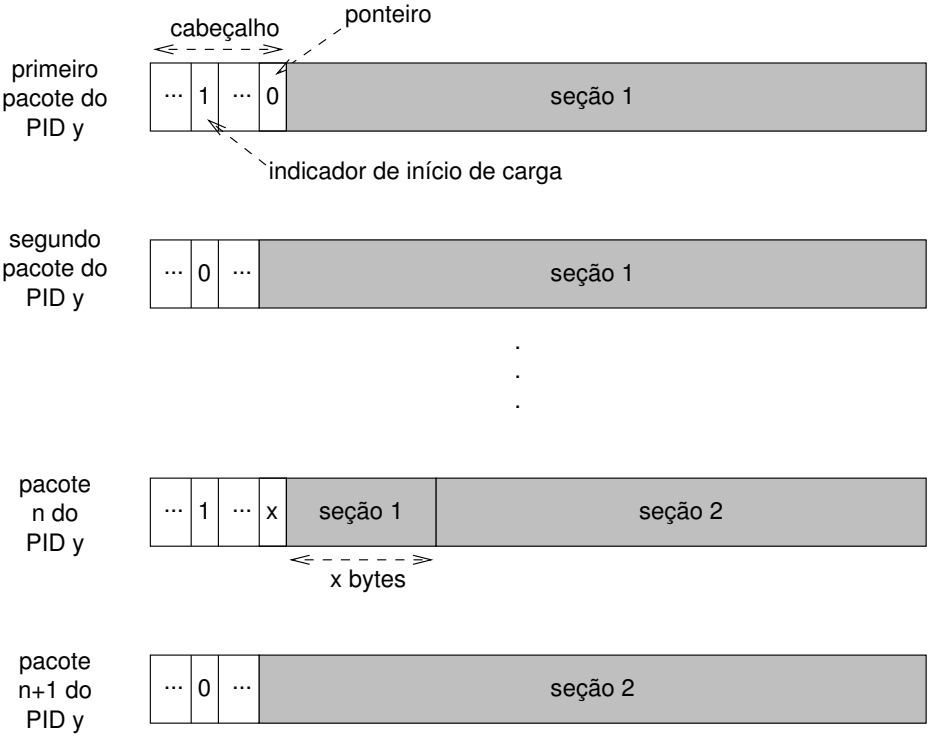

Figura 3.5: *Função do campo ponteiro no pacote de transporte.*

O conteúdo de determinados pacotes pode ser criptografado de forma a restringir o seu acesso a determinados grupos de consumidores ou telespectadores. Tais mecanismos de criptografia são implementados por sistemas conhecidos como *Sistemas de Acesso Condicional*. O padrão MPEG-2

não especifica como esses sistemas devem ser implementados, mas apenas oferece suporte a eles. No pacote de transporte existe um campo responsável por informar o demultiplexador se a carga do pacote está ou não criptografada, o campo *controle de criptografia de transporte*.

Em determinados casos, alguns pacotes de transporte podem ser mais importantes, ou necessitarem de tratamento preferencial sobre outros pacotes. Cada pacote de transporte possui um campo responsável pela delegação de prioridade simples, o *bit* de *prioridade de transporte*. Através dele é possível informar ao demultiplexador que determinado pacote ou conjunto de pacotes possui prioridade superior aos outros.

Como vários fluxos elementares são multiplexados em milhares de pacotes de transporte, deve existir um meio de identificar quais pacotes carregam qual fluxo. Dessa forma, cada pacote de transporte possui um identificador, o *PID*, ao qual é atribuido um valor único para o conjunto de pacotes que transporta PES de determinada seqüência de vídeo, por exemplo, ou fragmentos da mesma seção.

Apesar de geralmente possuir apenas 4 *bytes*, o cabeçalho de um pacote de transporte pode ser estendido de modo a apresentar recursos adicionais. A extensão do cabeçalho é chamada de *campo de adaptação*, e dentre os seus campos se encontra o *PCR, Program Clock Reference*. A função do PCR é carregar em cada pacote a estampilha de tempo do momento de sua geração no multiplexador, de forma a permitir que o terminal de acesso reconstrua a base de tempo utilizada por determinado serviço. Assim, os fluxos de vídeo e áudio são sincronizados baseando-se no valor do PCR. Dessa forma, um serviço de mídia deve possuir ao menos um fluxo de pacotes de transporte contendo o PCR. A existência ou não do *campo de adaptação* em pacote de transporte é indicado pelo campo *controle do campo de adaptação* presente no cabeçalho padrão do pacote de transporte.

O último campo em um cabeçalho padrão é o *contador de continuidade*. Esse campo tem a função de implementar um mecanismo simples de verificação de continuidade, visto que o mesmo é incrementado a cada novo pacote de mesmo *PID*. Dessa forma, o demultiplexador pode reconhecer, em alguns casos, a perda de pacotes em um fluxo de transporte.

Os equipamentos demultiplexadores MPEG mais simples trabalham apenas no nível dessa primeira camada, a dos pacotes de transporte, apresentada nessa seção. Dessa forma, os mesmos ficam limitados apenas às funcionalidades presentes em seus cabeçalhos. Geralmente, para se demultiplexar conteúdo transportado diretamente em pacotes de transporte, e decodificar seus dados, é necessário que o demultiplexador saiba qual o valor do *PID* em que os dados alvos estão sendo difundidos. Além disso, a forma como os dados são codificados e decodificados é de responsabilidade da implementação. De forma a implementar estruturas mais avançadas para o encapsulamento de dados e fluxos elementares, assim como maneiras do demultiplexador resolver os PIDs que transportam informações de seu interesse, o *MPEG-2 Sistemas* define estruturas que são apresentadas na próxima seção.

3.2.3 PES e Seções

Equipamentos mais avançados, como receptores de televisão digital, são capazes de filtrar os dados encapsulados nas estruturas da camada de codificação, imediatamente superior a camada de transporte. Funcionalidades não apresentadas nos pacotes de transporte estão disponíveis nessas estruturas, como por exemplo, correção de erros e sincronização. De forma a atender diferentes requisitos no transporte de dados, são dois os tipos de estruturas definidos nessa camada: as *PES*, ou *Packetized Elementary Streams*, e as *Seções*.

PES - *Packetized Elementary Streams*

As PES são utilizadas principalmente como forma de encapsulamento de fluxos elementares de áudio e vídeo. Geralmente, para um fluxo de vídeo MPEG-2, cada quadro é conduzido em um pacote PES. Com relação ao áudio, um pacote PES carrega em torno de 24 milisegundos de amostragem. Além de áudio e vídeo MPEG-2, as PES também podem carregar outros tipos de dados. O tamanho de cada pacote PES é variável, chegando a aproximandamente 65 mil *bytes*.

As PES carregam em seu cabeçalho informações como a identificação do tipo de conteúdo, além de estampilhas de tempo para a sincronização do mesmo. São duas estampilhas de tempo com essa finalidade, o *PTS, Presentation Time Stamp*, e o *DTS, Decoding Time Stamp*. Ambas são baseadas no relógio de referência de programa, o *PCR*, difundido em campos de adaptação de alguns pacotes de transporte.

O *PTS* corresponde ao instante em que o conteúdo transportado por uma PES deve ser apresentado. Por exemplo, pode corresponder ao instante em que um quadro de um fluxo de vídeo deve ser exibido.

O *DTS* indica o instante em que a carga da PES deve ser decodificada. Essa estampilha de tempo se mostra útil em situações onde determinada informação a ser apresentada deve ser decodificada depois de outra informação que ainda não foi entregue ao receptor. Isso ocorre, por exemplo, na decodificação e apresentação de vídeo MPEG-2 onde alguns quadros, os quadros do tipo B[2], necessitam de informações de um quadro posterior para sua decodificação.

Seções

As seções são utilizadas principalmente no transporte de determinadas tabelas e contêm informações referentes aos serviços presentes em um fluxo de transporte. Essas tabelas são conhecidas como *PSI*, ou *Program Specific Information*, e podem ser transmitidas em uma seção ou fragmentadas em mais de uma. São quatro as PSI definidas pelo padrão MPEG-2:

[2]Quadros Bidirecionais MPEG-2. Sua codificação é baseada em informações de quadros anteriores e posteriores

- Tabela de Associação de Programas ou Serviços (*PAT - Program Association Table*)
- Tabela de Mapeamento de Programa ou Serviço (*PMT - Program Maple Table*)
- Tabela de Informação de Rede (*NIT - Network Information Table*)
- Tabela de Acesso Condicional (*CAT - Conditional Access Table*)

A *Tabela de Informação de Rede* é uma tabela opcional. Ela transporta informações sobre parâmetros físicos da rede por onde o fluxo de transporte está sendo difundido. A *Tabela de Acesso Condicional* é utilizada para gerenciar os sistemas de criptografia de conteúdo, citados na seção 3.2.2. Por não ser objetivo deste trabalho, tal tópico não é abordado.

Como apresentado neste capítulo, um fluxo de transporte contêm vários serviços, uma organização lógica de vários fluxos elementares que são multiplexados no fluxo de transporte. O multiplexador, ao receber determinado fluxo de transporte, deve ser capaz de descobrir quais serviços estão presentes no mesmo. Tal informação é fornecida pela *Tabela de Associação de Programas*. Ela contém a lista de todos os serviços presentes em um fluxo de transporte, associando cada serviço a um determinado PID.

A informação sobre quais fluxos elementares fazem parte de determinado serviço estão presentes na *Tabela de Mapeamento de Programa*. É o PID dos pacotes que transportam essa tabela que é referenciado pela PAT para cada serviço. A PMT contém a lista dos fluxos elementares do serviço, o tipo de cada fluxo, assim como o PID de seus pacotes de transporte.

Para um terminal de acesso, a PAT é ponto de partida para se localizar determinado serviço ou fluxo elementar em um fluxo de transporte. Esse é motivo pelo qual todos os pacotes com PID igual a zero são reservados para o transporte da PAT como definido pelo *MPEG-2 Sistemas*. A figura 3.6 ilustra a sintaxe de uma PAT.

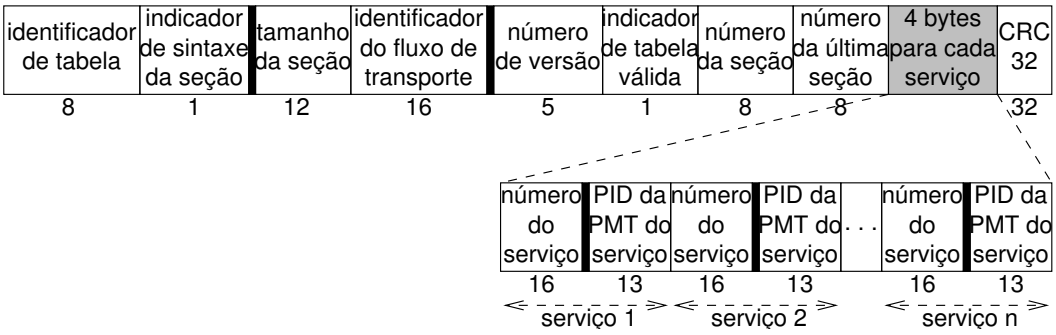

Figure 3.6: *Sintaxe de uma Tabela de Associação de Programas.*

A estrutura de uma seção carregando uma PAT é semelhante às demais PSI. Através do *identificador de tabela* é possível determinar se a tabela é uma PAT, PMT, etc. Especificamente para a PAT

existe o campo denominado *identificador do fluxo de transporte*, que como o próprio nome indica, serve como um identificador para o fluxo de transporte no qual os serviços referenciados pela PAT são transportados.

Alguns campos são comuns a todas as PSI. Como essas tabelas podem ser atualizadas periodicamente, suas modificações são informadas ao decodificador através de campos como o *número de versão* e o *indicador de tabela válida*. O *campo indicador de tabela válida* informa se a tabela que está sendo transmitida atualmente, de determinado número de versão, já deve substituir a tabela de número de versão anterior, ou se apenas está sendo difundida previamente para que o terminal de acesso se prepare para as futuras modificações.

As seções possuem tamanho variável, indicado pelo campo *tamanho da seção*, porém não pode ser maior que 1024 *bytes*. Caso a PAT ou a PMT seja maior que esse valor, a mesma deve ser segmentada em mais de uma seção. De forma a numerar essas seções, é utilizado o campo *número da seção*, que é incrementado até a última seção de determinada tabela. Para prever o tamanho da tabela nesse caso, o número total de seções pode ser obtido pelo decodificador desde a primeira seção através do campo *número da última seção*. As seções implementam também mecanismos de correção de erros através do CRC de 32 *bits*, calculado no momento de codificação e inserido nos últimos 4 *bytes* da seção.

Além do *identificador do fluxo de transporte*, uma estrutura única das PAT é o conjunto de associações para cada serviço presente no fluxo de transporte. Cada serviço é referenciado por um número de programa, ao qual é atribuído um PID. Esse PID indica em qual fluxo está sendo transportada a *Tabela de Mapeamento de Programa*. Dessa forma, ao selecionar determinado serviço, o demultiplexador passa a filtrar os pacotes com o PID em questão, informado pela PAT, obtendo assim a PMT referente ao mesmo. A sintaxe das seções das PMTs é apresentada na figura 3.7.

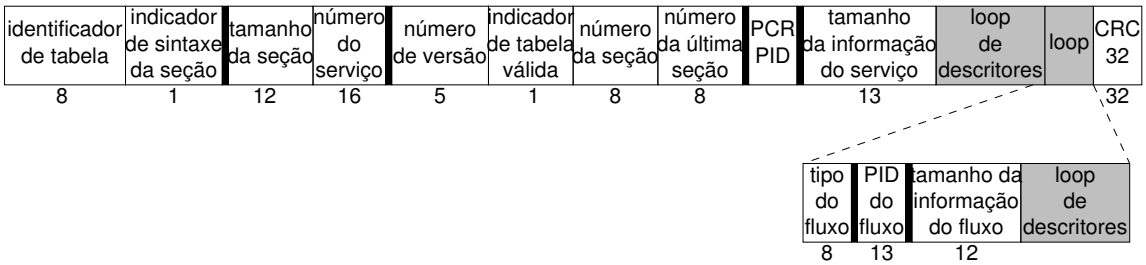

Figura 3.7: *Sintaxe de uma Tabela de Mapeamento de Programa.*

Alguns campos da seção da PMT são semelhantes aos da PAT, possuindo as mesmas funções. Porém, um campo único na PMT é o *número do programa*, que é um identificador para o serviço, com o mesmo valor referenciado pela PAT. Um campo adicional é o *PCR PID*, que indica em quais pacotes está sendo transportado o relógio base do programa, PCR, como mostrado na seção 3.2.2. A seção da PMT contém ainda um campo de tamanho variável, definido pelo campo *tamanho da*

informação de programa, que possui informações adicionais sobre o mesmo.

Após a *informação de programa*, a PMT possui uma lista relacionando todos os fluxos elementares que fazem parte do serviço. Cada entrada dessa lista é dividida em três partes: *tipo do fluxo*, *PID* e *descritores*. O primeiro informa qual o tipo de fluxo carregado pelos pacotes com o PID do segundo campo: vídeo, áudio, etc. Os descritores são opcionais, dependentes geralmente da implementação, e fornecem informações adicionais sobre o fluxo referenciado. Ao ler a lista fornecida pela PMT, o demultiplexador decide quais fluxos irá repassar aos decodificadores. A figura 3.8 ilustra a relação entre as PSI e os demais fluxos elementares.

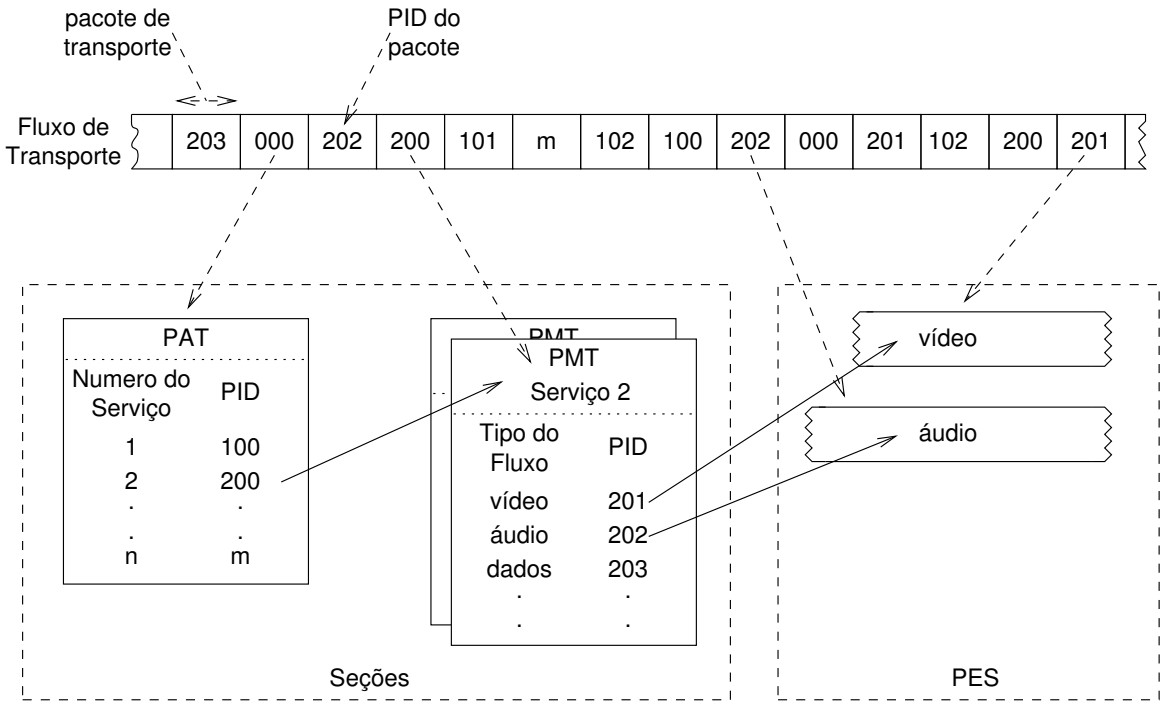

Figura 3.8: *Relação entre PAT, PMT e demais fluxos elementares.*

Além das seções utilizadas na codificação das PSI descritas até então, existe um tipo genérico de seção denominado *seção privada*. As seções privadas podem ser utilizadas para transportar outras tabelas, definidas em implementações de sistemas de TVD, ou qualquer outro tipo de informação. A figura 3.9 ilustra a sintaxe das seções privadas.

É possivel encapsular dados em uma seção privada utilizando apenas três *bytes* de cabeçalho. A existência dos demais cinco *bytes* comuns em uma seção são sinalizadas pelo campo *indicador de sintaxe da seção*. A maioria dos outros campos possui a mesma função dos presentes na PAT e PMT. Além do *indicador de sintaxe da seção*, a única diferença do cabeçalho completo da seção privada é o campo *identificador estendido de tabela*, cuja funcionalidade é determinada pela implementação. Outra diferença é que, segundo o padrão MPEG-2, uma seção privada pode chegar até 4096 *bytes*, contra os 1024 das seções utilizadas na codificação das PSI.

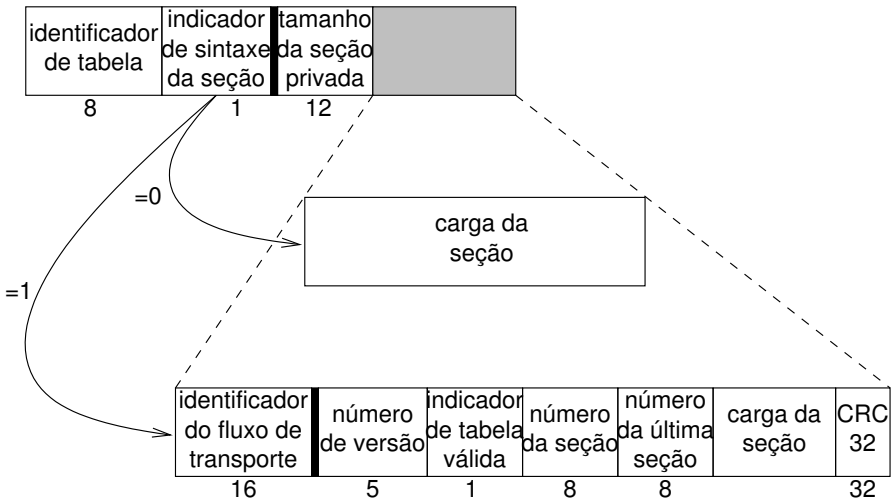

Figura 3.9: *Sintaxe das seções privadas.*

3.3 *Digital Storage Media, Command and Control* - DSM-CC

O *Digital Storage Media, Command and Control*, foi originalmente desenvolvido com o objetivo de fornecer funções semelhantes as presentes em aparelhos de vídeo cassete para o controle de fluxos de áudio e vídeo presentes em um fluxo de transporte [24]. Posteriormente o DSM-CC foi estendido e dividido em várias partes, com o intuito de fornecer funções como seleção, acesso e controle de fontes distribuídas de vídeo e suporte para a transmissão de dados através de fluxos de transporte, além de outras [16] [20].

Dividido em várias partes, o DSM-CC pode ser visto como um conjunto de ferramentas, apresentando várias configurações e funcionalidades. Dessa forma, vários sistemas, como os de televisão digital, adotam apenas algumas de suas partes. Os mecanismos de *carrossel de dados* e *carrossel de objetos* do DSM-CC são dois dos mecanismos mais utilizados para a difusão de dados nos sistemas DVB e ATSC. Dessa forma, serão apresentados em mais detalhes nas próximas seções.

3.3.1 Carrossel de Dados

O protocolo denominado *carrossel de dados* foi desenvolvido com o intuito de possibilitar a difusão de dados, de forma periódica, à terminais de acesso. A idéia básica desse protocolo é de módulos de dados difundidos ciclicamente, de modo que, quando o receptor necessitar determinado módulo, deve apenas aguardar o instante de sua próxima repetição no fluxo de dados. O carrossel de dados não prevê o uso de um canal de retorno, e assim é assumido que todos os parâmetros de transferência de dados são aceitos a priori pelo receptor.

Existem dois tipos de mensagem nos carrosséis de dados: as *mensagens de controle*, ou *Down-*

load Control Messages e as *mensagens de dados*, ou *Download Data Messages*. As *Download Data Messages* usam uma estrutura sintática conhecida como *DDB - Download Data Block* para o encapsulamento de dados, e sua sintaxe é apresentada no apêndice A. Já as *download control messages* podem ser de dois tipos diferentes: as *DSI - Download Server Initiate* ou as *DIIs*, *Download Info Indication*. A sintaxe de ambas mensagens também é apresentada no apêndice A.

Um módulo, em um carrossel de dados, é definido como um item simples e completo de dados. Por exemplo, um único arquivo de texto ou vários outros arquivos de texto podem ser definidos como um módulo. Cada módulo é então dividido em um ou mais blocos, e cada bloco é inserido na carga de uma *Download Data Blocks*, ou DDBs. As *Download Control Messages* nas formas de DSI ou DII carregam as informações sobre o conjunto de módulos de um determinado carrossel. Tanto as DDBs, DIIs e DSIs são construídas baseadas nas estruturas das seções privadas apresentadas na seção 3.2.3. A pilha de protocolos de um carrossel de dados é ilustrado na figura 3.10.

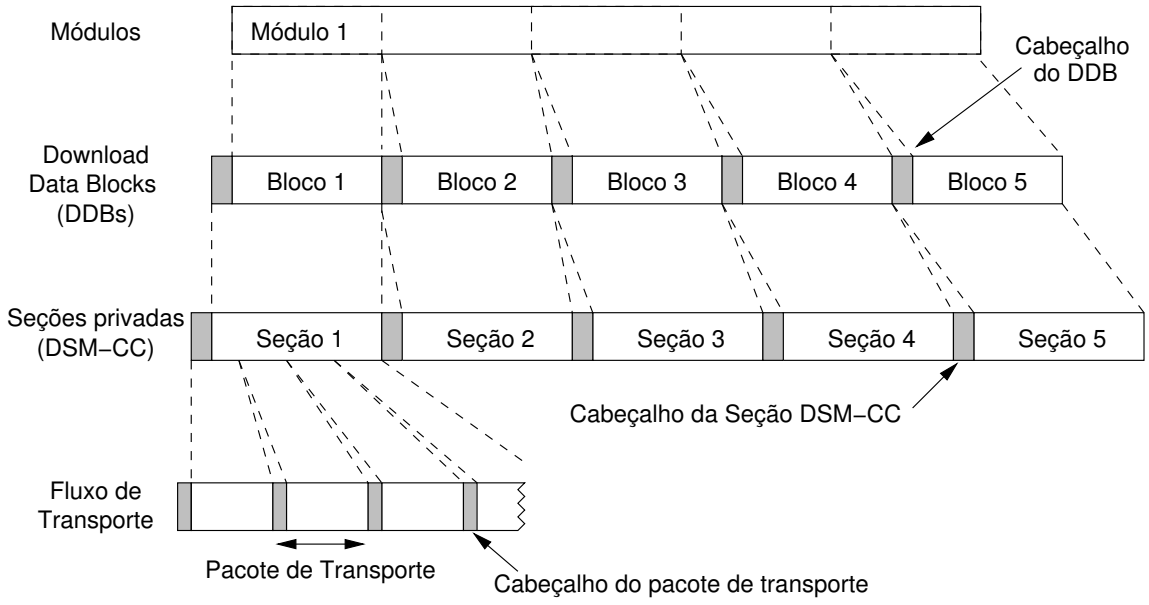

Figura 3.10: *Encapsulamento de um módulo nas estruturas DSM-CC e MPEG-2 Sistemas.*

A figura 3.11 ilustra um exemplo simples de carrossel de dados, onde três arquivos são encapsulados em três módulos diferentes, que são fragmentados em DDBs e inseridos no carrossel. Precedendo estes, uma *Download Control Message* transporta informações sobre os módulos e suas respectivas DDBs.

No exemplo da figura 3.11, os módulos são inseridos apenas uma vez na chamada rotação do carrossel. Os DDBs de cada módulo na figura 3.11 estão dispostos de forma ordenada. Contudo, não há restrições com relação à ordem dos DDBs e com a quantidade de vezes em que são inseridos no carrossel. Dessa forma, determinados módulos podem estar presentes mais de uma vez por rotação, fazendo com que os mesmo sejam difundidos a uma freqüência maior que os outros.

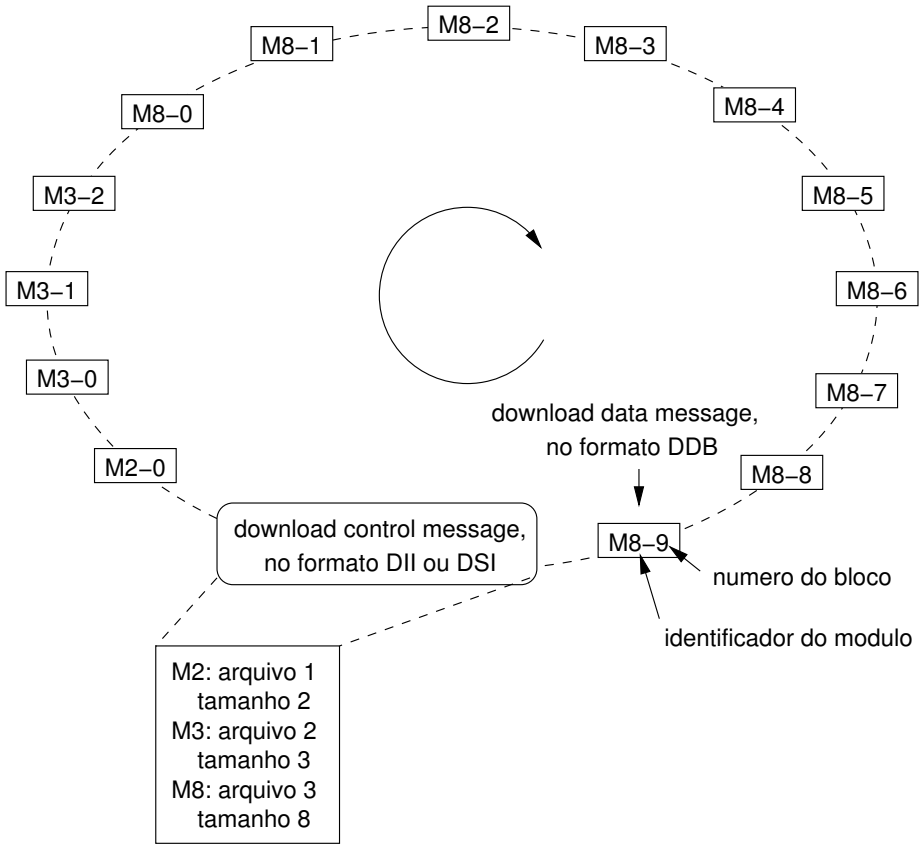

Figura 3.11: *Exemplo de um carrossel de dados.*

O carrossel de dados permite que um conjunto de diferentes módulos possa ser organizado logicamente formando grupos. Não há restrições sobre quantos módulos podem pertencer a um grupo, e um módulo pode fazer parte de um ou mais grupos. Para cada grupo, sempre deve haver uma DII que é responsável pela descrição de cada módulo pertencente ao mesmo.

Um conjunto de grupos pode ser organizado logicamente formando supergrupos. Assim como em um grupo, cada supergrupo deve possuir uma mensagem de controle. Porém, no caso dos supergrupos são as DSI as mensagens responsáveis por descrever cada um de seus membros. A figura 3.12 ilustra a estrutura de um carrossel de dados baseado em grupos e supergrupos.

Quando o número de módulos é pequeno, e apenas um grupo pode ser formado, não há necessidade de supergrupos. Nesse caso, conceitua-se o carrossel como um carrossel de dados de uma camada, onde apenas uma DII é necessária para descrever todo o conjunto de módulos. Por outro lado, um carrossel de dados de duas camadas é aquele onde existem supergrupos, ou seja, há a presença de mensagens de controle do tipo DII assim como do tipo DSIs. Geralmente um carrossel desse tipo se faz necessário quando o número de módulos é demasiadamente grande para ser descrito por apenas uma DII [2].

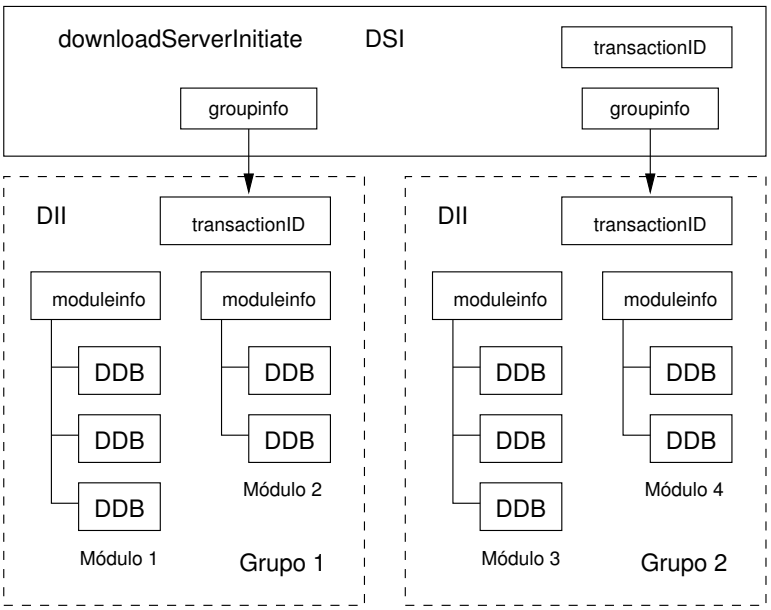

Figura 3.12: *Grupos em um carrossel de dados.*

Os carrosséis de dados podem ser referenciados nas PSIs, mais especificamente pelas PMTs. Dessa forma, um carrossel pode fazer parte de um serviço, e a localização de suas mensagens de controle facilmente encontradas pelo terminal de acesso através de consultas a PMT de determinado serviço.

3.3.2 Carrossel de Objetos

Carrosséis de Objetos são construídos sobre as estruturas dos carrosséis de dados, apresentadas na seção 3.3.1, adicionando uma nova camada a aqueles, onde a informação é encapsulada na forma de objetos. Nessa nova camada, os objetos são transportados em mensagens conhecidas como BIOPs, ou *Broadcast Inter ORB Protocol* [25]. O BIOP é uma extensão do ORB, *Object Request Broker*, definido pelo CORBA [26].

BIOPs

Uma BIOP é dividida primeiramente em 3 partes: cabeçalho (*MessageHeader*), sub-cabeçalho (*MessageSubHeader*) e corpo da mensagem (*MessageBody*). O cabeçalho carrega informações sobre a versão do protocolo BIOP utilizado e sobre o tamanho da mensagem. O sub-cabeçalho traz informações a respeito do objeto carregado, seu tipo, e um identificador, o *objectKey*. O corpo depende do tipo do objeto, e contém os dados que serão difundidos. A estrutura completa de uma BIOP é apresentada no apêndice B [3].

As BIOPs são carregadas dentro de módulos, os mesmos definidos pelo carrossel de dados. Cada módulo pode encapsular uma ou mais BIOPs, onde cada objeto é identificado dentro do módulo pelo seu *objectKey*. Dessa forma, o decodificador pode encontrar facilmente um objeto pela análise seqüencial de cada *objectKey*, encontrado em intervalos dentro do módulo de acordo com o tamanho de cada objeto, atributo informado pelo cabeçalho da BIOP. A figura 3.13 ilustra a organização dos objetos dentro de cada módulo.

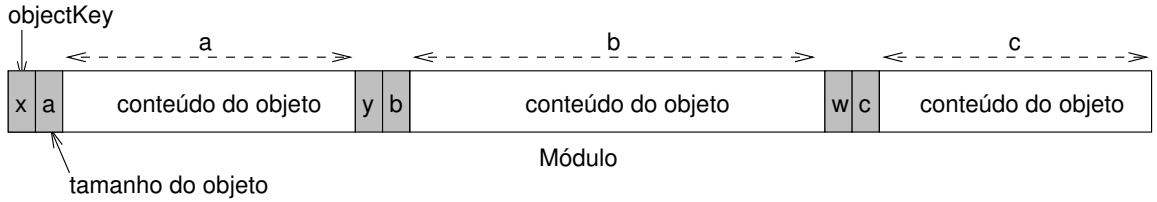

Figura 3.13: *Seqüência de objetos encapsulados em um módulo.*

Assim como acontece com o carrossel de dados, cada módulo é fragmentado em um ou mais blocos, os quais são inseridos nas cargas de *Download Data Blocks* - DDBs. Os DDBs possuem tamanho fixo, com exceção do último DDB que carrega um módulo, que pode ser menor. Cada DDB é encapsulado em uma seção DSM-CC, uma extensão das seções privadas. As seções por fim são fragmentadas em vários pacotes de transporte. A figura 3.14 ilustra o encapsulamento das BIOPs.

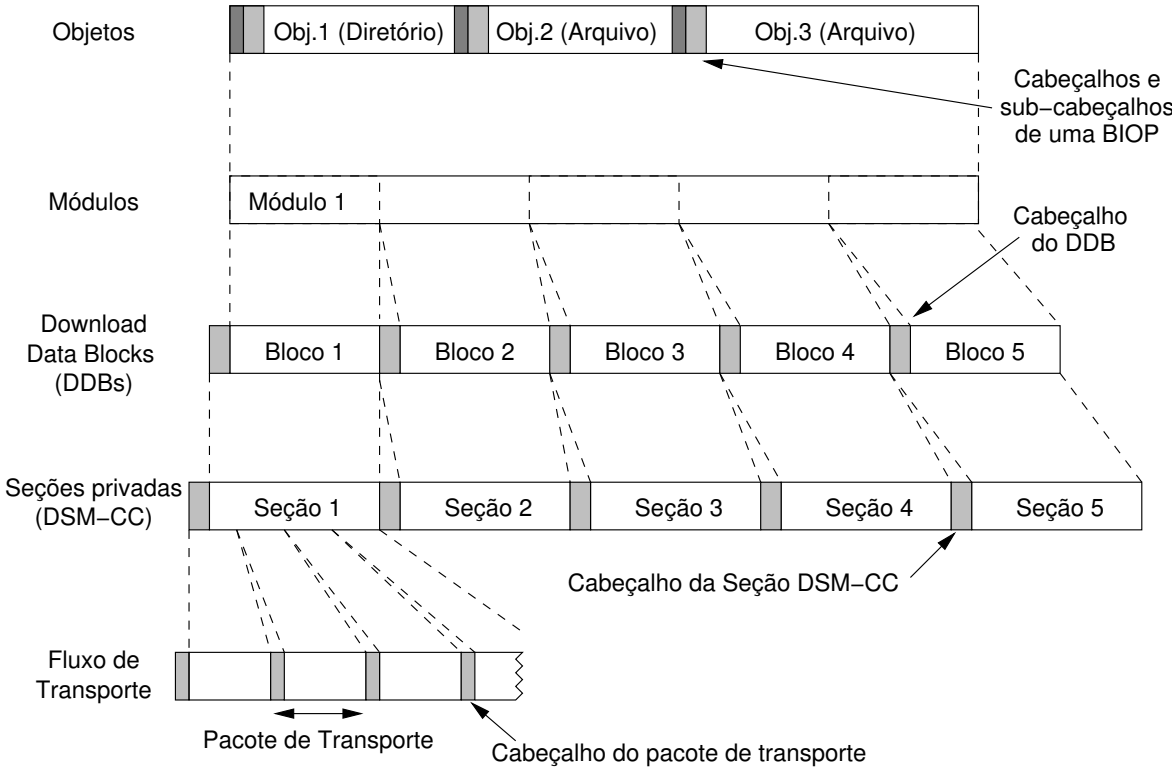

Figura 3.14: *Encapsulamento dos objetos nas estruturas DSM-CC e MPEG-2 Sistemas.*

Tipos de objetos

Existem basicamente três tipos de objetos em um carrossel: *arquivo*, *diretório* e *stream*[3] [2]. Um conjunto de objetos que forma um carrossel de objetos é conhecido como *Service Domain*.

Os objetos do tipo *arquivo* são destinados ao transporte de dados da mesma forma que os mesmos são armazenados em sistemas de arquivos simples. Um objeto desse tipo pode conter o *bytecode* de uma *Xlet*, ou demais recursos utilizados por ela, como arquivos de imagens ou conteúdo textual, por exemplo.

Um conjunto de objetos do tipo *arquivo* sempre deve pertencer a um diretório, assim como na maioria dos sistemas de arquivos atuais. Essa é a função do objeto do tipo *diretório*, que contém uma lista com o nome dos demais objetos pertencentes a esse, ligados a uma *IOR - Interoperable Object Reference*. A lista contendo os nomes dos objetos e suas IORs é conhecida como *lista de ligações* (*List of Bindings*). Em todo *Service Domain* deve haver um diretório raiz, a partir do qual todos os demais objetos são encontrados. O diretório raiz é implementado por um objeto especial do tipo diretório denominado *Service Gateway*.

IOR e *Profile Body*

A função da IOR é fornecer ao decodificador do carrossel de objetos informações suficientes para a localização do objeto ao qual é ligado na lista de ligações. Dessa forma, cada IOR pode conter uma ou mais referências a determinado objeto. Cada referência presente em uma IOR é conhecida como *profile body*.

Existem dois tipos de *profile body*, o *BIOP Profile Body* e o *Lite Options Profile Body*. O *BIOP Profile Body* é utilizado para referenciar objetos localizados em um mesmo *Service Domain*, ou seja, em um mesmo carrossel de objetos. Já o *Lite Options Profile Body* de uma *IOR* se refere a objetos localizados em outros carrosséis [2]. Esse último não será apresentado em mais detalhes, visto que é similar em vários aspectos ao primeiro, e na maioria das vezes os objetos são referenciados em um mesmo carrossel de objetos.

Um *BIOP Profile Body* possui basicamente duas estruturas: o *ObjectLocation* e o *ConnBinder*. O *ObjectLocation* é utilizado para referenciar o objeto em um módulo. Ele fornece o número do *indentificador do módulo* (*moduleId*) assim como o *objectKey* do objeto. Como os módulos no carrossel de objetos são referenciados através das mensagens de controle, as DII, o decodificador precisa primeiro consultar essa mensagem para encontrar o módulo desejado. Essa é a função do *ConnBinder* do *BIOP Profile Body*: fornecer informações ao decodificador sobre onde se encontra a DII que indica onde está o módulo que contem o objeto desejado.

[3] Um objeto do tipo *stream* transporta uma lista de identificadores, onde cada identificador se refere a um fluxo elementar ou a um serviço inteiro. Esse tipo de objeto não será abordado em mais detalhes nesse trabalho

Para se encontrar a DII que contém a localização do módulo, o decodificador precisa informar ao demultiplexador em qual fluxo elementar a mesma está sendo difundida. Como um fluxo elementar é referenciado através do valor do PID dos pacotes que o transporta, conhecendo-se de antemão qual o PID do fluxo que transporta a DII é suficiente. Porém, a geração do carrossel de objetos geralmente é um processo anterior e independente do processo de multiplexação. Em alguns casos, o multiplexador altera os valores dos PIDs dos pacotes de entrada, ou seja, o fluxo de transporte resultante possui fluxos elementares com PIDs diferentes dos originais. Dessa forma, caso o *ConnBinder* referencie determinada DII através do PID na qual ela se encontra, o multiplexador ao alterar esse valor deve alterar também o valor referenciado pelo *ConnBinder*. Essa abordagem não é adotada devido à sua complexidade, pois o multiplexador necessitaria estar apto a manipular estruturas de mais alto nível que as definidas pelo *MPEG-2 Sistemas*. Esse problema é resolvido através de estruturas denominadas *Taps* no *ConnBinder*.

Um *Tap* contém três campos de maior importância para a localização de uma DII: o *carouselId*, o *association tag* e o *transactionId*. O *carouselId* serve para referenciar a PMT que contém em sua lista de fluxos elementares o fluxo que transporta a DII. Na PMT em questão deve haver um descritor no campo *informações de programa*, conhecido como *carouselId descriptor*, que deve conter o mesmo valor referenciado pelo *Tap* de forma que o terminal de acesso identifique corretamente a PMT.

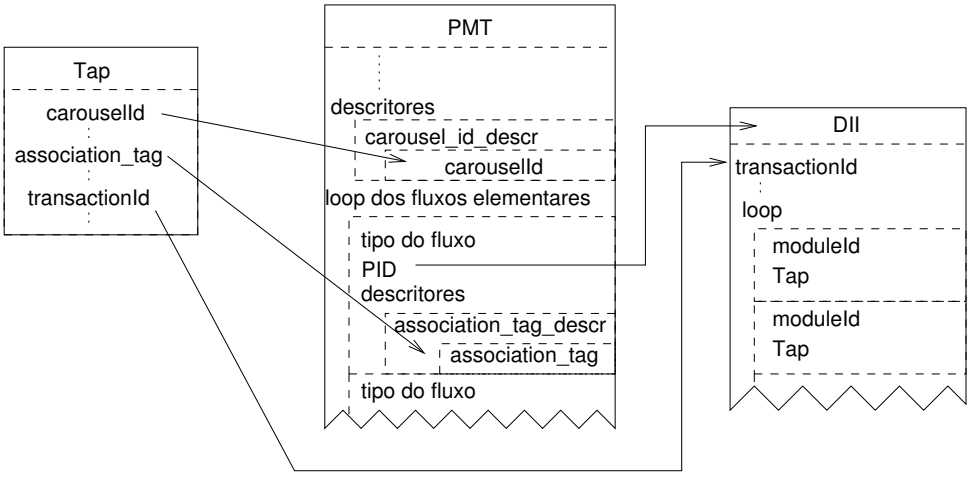

Figura 3.15: *Uso do Tap para referenciar fluxos elementares.*

O *association tag* referencia indiretamente o fluxo elementar que transporta a DII em questão. Para que o fluxo possa ser encontrado, é necessário a inserção de um descritor no *loop* de descritores do fluxo da PMT. Esse descritor é conhecido como *association tag descriptor*, e contém o mesmo valor referenciado pelo *association tag* do *Tap*. Dessa forma, o terminal de acesso pode localizar o fluxo elementar que transporta a DII consultando a lista de fluxos da PMT procurando pelo mesmo valor do *association tag* informado pelo *Tap* em descritores dos fluxos. Para se identificar qual a DII entre as várias que podem ser transmitidas em um mesmo fluxo elementar é utilizado por fim

o *transactionId* fornecido pela *Tap*, que deve ser o mesmo apresentado pelo cabeçalho da DII. A figura 3.15 ilustra como através da *Tap* é possível localizar uma DII [3].

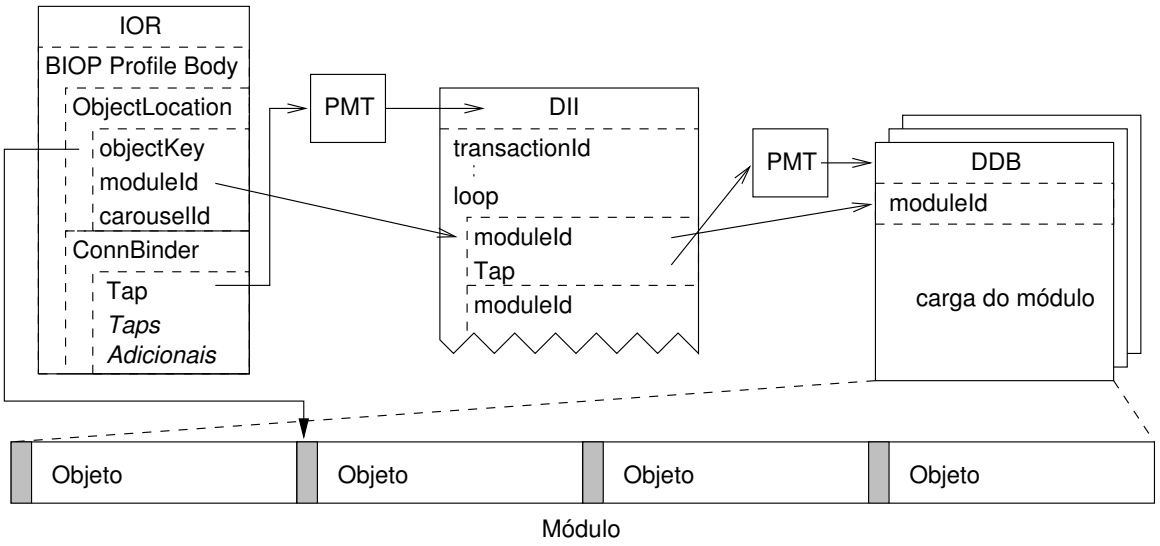

Figura 3.16: *Resolução de um objeto em um BIOP Profile Body.*

Localizada a DII, o módulo contendo o objeto desejado é também referenciado através de uma *Tap* presente nessa. Assim, o fluxo contendo as DDBs que carregam o módulo são encontradas de forma semelhante a utilizada para encontrar a DII. O objeto alvo é então localizado dentro do módulo através de seu *objectKey*. A figura 3.16 ilustra esse processo [3].

Service Gateway

As DIIs contêm informações necessárias para que a partir de um IOR seja possível localizar determinado módulo e seus objetos. O *Service Gateway* é o objeto do tipo diretório que representa o diretório raiz de um carrossel de objetos, ou seja, a partir dele é possível encontrar todos os demais objetos dos tipos *arquivo* e *diretório*. Dessa forma, o *Service Gateway* geralmente é o primeiro objeto alvo do decodificador.

De forma a facilitar a localização do *Service Gateway*, a *IOR* que referencia o mesmo é transportada em uma DSI. Isso se deve ao fato de que, para o carrossel de dados, as PSIs apontam para o fluxo elementar que contém a DSI, já que essa é a mensagem de controle de maior nível desse protocolo e deve ser a primeira a ser localizada. A figura 3.17 ilustra esse processo.

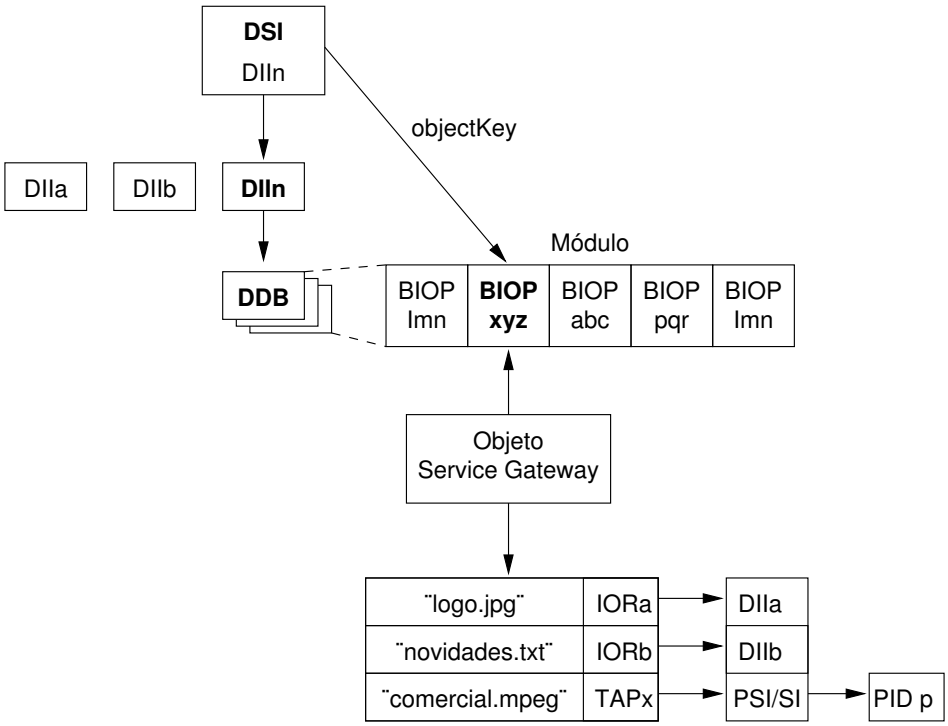

Figura 3.17: *Localizando objetos através da DSI em um carrossel de objetos.*

3.4 Considerações Finais

A especificação *MPEG-2 Sistemas*, primeira parte do MPEG-2, é adotada como padrão para a camada de multiplexação de serviços nos sistemas abertos de televisão digital, DVB, ATSC e ISDB. A mesma possibilita a difusão de dados através de algumas das estruturas apresentadas neste capítulo, como em PES ou seções privadas. Já o DSM-CC adiciona funcionalidades extras para a transmissão de dados sobre o MPEG-2. Tanto as formas de difusão de dados sobre as estruturas MPEG-2 como sobre as extensões DSM-CC utilizadas pelos sistemas de televisão digital em serviços de *datacasting* são objetivos de estudo do próximo capítulo.

Capítulo 4

Datacasting

O *Data Broadcasting*, ou simplesmente *datacasting*, pode ser definido como um serviço, diferente do serviço de difusão normal, de entrega de informação (seja ela no formato de dados, texto, som, imagem ou em qualquer outro) para clientes com terminais de acesso adequados, através do mesmo meio utilizado na difusão [7]. Este capítulo apresenta algumas taxonomias referentes ao *datacasting* em ambientes de televisão digital, assim como o uso das tecnologias apresentadas no capítulo 3 nesse contexto.

4.1 Taxonomias

Antes de classificar o *datacasting* em si, é importante classificar os dados a serem difundidos. Duas classificações são usualmente empregadas: uma com relação ao formato dos dados e outra com relação aos requisitos temporais ou de sincronização dos mesmos. A primeira divide os dados em três categorias: *delimitados* (*bounded*), *não delimitados* (*unbounded* ou *streams*) e *datagramas*. Dados *delimitados* são aqueles que podem ser divididos em unidades de tamanho determinado, como em objetos, arquivos, etc. Dados *não delimitados* não atendem a esse requisito, sendo considerados como fluxos contínuos de *bits*. O terceiro tipo corresponde a fragmentação dos dados, independente de sua natureza, em pacotes denominados datagramas seguindo algum protocolo de comunicação [27].

Uma segunda classificação divide também os dados em três tipos: *síncronos*, *sincronizados* e *assíncronos*. Dados *síncronos* possuem requisitos de sincronização com outros dados do mesmo fluxo, ou seja, possuem sincronização *intra-mídia*. Dados *sincronizados* são aqueles que devem ser decodificados ou apresentados em instantes pré-determinados, sincronizados com elementos de outras mídias, por exemplo, com determinado fluxo de vídeo. Essa forma de sincronismo é denominada *inter-mídia*. Já os dados *assíncronos* não possuem informações temporais relativas a sincronização [28].

As classificações dos dados aqui apresentadas são úteis para o entendimento das diversas técnicas de *datacasting* discutidas no decorrer do texto. Já o *datacasting* em si também pode ser classificado de várias maneiras. As duas classificações mais comuns são definidas de acordo com quem é o destinatário dos dados difundidos e também com relação ao grau de dependência dos dados difundidos com a programação televisiva.

4.1.1 Classificação de acordo com o Usuário dos Dados

A primeira classificação divide o *datacasting* em dois segmentos, de acordo com quem é o destinatário dos dados difundidos. São duas possibilidades: *Consumidor* ou *Empresa*[1] [6] [11]. O consumidor tem acesso aos serviços de *datacasting* de forma gratuita ou paga, geralmente com a finalidade de entretenimento. As empresas por sua vez pagam pelos serviços ou pela utilização da banda da televisão digital de forma a suprir necessidades operacionais através do uso do *datacasting*.

Para o primeiro caso, o *datacasting* funciona geralmente como um fator de enriquecimento à programação televisiva. A justificativa de uso do *datacasting* com essa finalidade para a televisão digital é de que provavelmente apenas os benefícios de melhoria na qualidade de imagem e som não atraia recursos suficientes (consumidores, telespectadores, anunciantes, todas as possíveis fontes financiadoras) que compensem a transição para a tecnologia digital [6]. Dessa forma, a interatividade é um recurso adicional que pode acelerar o interesse por parte do uso da tecnologia digital, e o *datacasting* é a base para que esse serviço seja possível.

Alguns autores [6] afirmam, porém, que atualmente o modelo de negócios para o segundo tipo de *datacasting*, em aplicações empresariais, é economicamente mais viável. Essa afirmação é baseada no fato de que a infra-estrutura tecnológica necessária para suportar esse tipo de *datacasting* é consideravelmente mais simples e barata que no primeiro caso, assim como os benefícios são mais atraentes.

Terminais de acesso neste último caso podem ser mais baratos, visto que não há a necessidade de se implementar decodificadores de mídia e *middlewares* em sua totalidade como especificados para receptores domésticos. Dessa forma, padrões de *middleware* como o MHP e o DASE podem ser apenas implementados parcialmente, atendendo somente requisitos específicos de determinada aplicação, ou substituídos por soluções proprietárias mais simples.

[1] Telespectador, consumidor ou cliente são termos utilizados nesta dissertação e em outros trabalhos para referenciar a pessoa que é beneficiária dos serviços de *datacasting*

4.1.2 Classificação de acordo com o Acoplamento dos Dados à Programação Televisiva

O *datacasting* pode ser classificado de acordo com o grau de acoplamento, ou de relação, dos dados transmitidos com a programação televisiva. São três classificações possíveis: *fortemente acoplado*, *fracamente acoplado* e *desacoplado* [6] [11].

Um serviço de *datacasting* é definido como *fortemente acoplado* quando o mesmo é utilizado com a finalidade de enriquecer a programação televisiva em tempo real. Na maioria dos casos existem restrições temporais fortes e necessidade de sincronização dos dados com alguma mídia audiovisual. O *datacasting* é *fracamente acoplado* quando possui alguma relação com determinado programa televisivo, porém, não é fortemente sincronizado com o mesmo. Quando os dados difundidos não possuem nenhuma relação com algum serviço televisivo, o *datacasting* é definido como *desacoplado*. Os mesmos podem ser difundidos como um serviço de televisão digital, e reconhecidos e tratados por um terminal de acesso como tal, ou serem completamente direcionados para aplicações específicas.

4.2 Mecanismos de Datacasting

Existem basicamente quatro mecanismos, ou áreas de aplicação [2], para a difusão de dados utilizando os fluxos de transporte MPEG-2 na televisão digital. São eles: *Data Piping*, *Data Streaming*, *MPE* ou *Multiprotocol Encapsulation* e Carrosséis [3, 28, 2], sendo esses últimos divididos em *Carrosséis de Dados* e *Carrosséis de Objetos*. Os quatro mecanismos são explicados em mais detalhes a seguir.

4.2.1 *Data Piping*

O *data piping*, ou canalização de dados, é o método mais simples de inserção de dados em um fluxo de transporte MPEG-2. Baseia-se em encapsular dados brutos diretamente nas cargas de pacotes de transporte. É geralmente utilizado por sistemas proprietários. Pode ser utilizado para o transporte de dados delimitados, não-delimitados e datagramas, porém a forma como os dados são fragmentados e organizados sobre os pacotes de transporte, e a sua posterior recuperação, é de responsabilidade da implementação [3].

A principal vantagem do *data piping* reside na simplicidade. A tarefa de um codificador para essa abordagem é gerar pacotes de transporte como apresentados na seção 3.2. É atribuido um valor PID conhecido pela aplicação destinatária, e os dados inseridos nos 184 *bytes* de carga de cada pacote, antes da seqüência de pacotes ser enviada ao multiplexador. O receptor precisa apenas filtrar os pacotes com esse PID e extrair de sua carga os dados transmitidos. Essa abordagem envolve um

baixo custo de *hardware* e *software* para implementação, sendo recomendada para o envio de dados proprietários.

Qualquer tentativa de acoplamento dos dados transmitidos via *data piping* com demais fluxos elementares é também responsabilidade da implementação, que deve criar mecanismos extras aos fornecidos pelo *MPEG-2 Sistemas*. O *data piping* é uma forma de *datacasting* assíncrono, considerando o uso de apenas estruturas MPEG-2 a nível dos pacotes de transporte. Por conseguinte, a grande desvantagem do *data piping* é a falta de estruturas sintáticas mais poderosas para a codificação de dados. Esses recursos são supridos pela segunda camada do *MPEG-2 Sistemas*, como introduzido na seção 3.2, e utilizados pelas demais formas de *datacasting*.

4.2.2 *Data Streaming*

O *data streaming*, ou fluxo de dados, pode ser definido como uma área de *datacasting* onde dados, na maioria das vezes *não delimitados*, são continuamente difundidos e geralmente alimentam alguma aplicação do terminal de acesso. Dessa forma, o *data streaming* pode ser implementado de duas formas básicas: através de *seções privadas* ou *PES*.

Também seria possível definir o *data piping* como uma forma de *data streaming*. Porém como apresentado na seção 4.2.1, os pacotes de transporte carecem de uma melhor forma de encapsulamento dos dados, o que dificulta o acesso a esses dados para aplicações genéricas (não proprietárias) no terminal de acesso. Tal fato não ocorre com relação ao uso das *seções privadas* e *PES* para essa finalidade, visto que os padrões de *middleware* dos sistemas de televisão digital geralmente oferecem APIs de acesso a dados difundidos nessas estruturas.

Seções Privadas

As *seções privadas*, como apresentado na seção 3.2.3, podem ser utilizadas para o encapsulamento de dados ordinários. A sintaxe das *seções privadas* fornecem algumas funcionalidades úteis para a implementação do *data streaming*, como a possibilidade de numeração das seções de várias formas. Também é uma forma de envio de dados com um baixo *overhead*. Através da configuração de cabeçalho simples, são necessários apenas 3 *bytes* para cada seqüência de 4093 *bytes* de dados.

Utilizando-se o cabeçalho estendido das seções privadas, podem ser transmitidos 4084 *bytes* de dados por seção. Neste caso é previsto que os últimos 4 *bytes* da seção contenham um valor de CRC, o mecanismo de detecção de erros implementado nessa estrutura. A figura 3.9 ilustra o uso do cabeçalho compacto e do cabeçalho estendido.

A grande vantagem do uso de *seções privadas* para o *data streaming* também reside no fato da simplicidade. As seções são facilmente referenciadas nas PSIs, e existem mecanismos de *hardware*

nos terminais de acesso conhecidos como *filtros de seções*, encarregados de extrair conteúdo das mesmas. Os *middlewares* fornecem funções de acesso a esses dados através de suas APIs de forma que qualquer aplicação pode ter acesso aos mesmos.

A desvantagem das seções privadas nesse contexto é a falta de estruturas para o transporte de informações temporais, ou seja, a princípio é utilizado como um meio de *datacasting* assíncrono. Novamente, qualquer tentativa de sincronismo é de responsabilidade da implementação, ou através do uso de estruturas de mais alto nível.

PES

Ao contrário das *seções privadas*, o uso das *PES* para o *data streaming* permite o *datacasting síncrono* e *sincronizado*, devido ao uso das estampilhas de tempo *DTS* e *PTS*, como apresentado na seção 3.2.3. Essa é a sua grande vantagem em relação as outras formas de difusão de dados. O *datacasting* via *PES* é geralmente empregado em aplicações que requisitam forte acoplamento dos dados com outros fluxos elementares.

4.2.3 *MPE - Multiprotocol Encapsulation*

O *Multiprotocol Encapsulation*, MPE, é utilizado para transportar datagramas de diversos protocolos através de *seções privadas* do padrão MPEG-2. O padrão DSM-CC estende as seções privadas, definindo a chamada *seção de datagrama*. Uma *seção de datagrama* pode ser codificada de acordo com qualquer tipo de protocolo de rede da terceira camada do modelo ISO/OSI [2].

Sistemas de televisão digital como o DVB otimizaram a sintaxe das *seções de datagrama* de forma a facilitar o transporte de datagramas IP. A preferência pelo protocolo IP se deve a dois fatores: sua popularidade e, caso utilize-se essa forma de *datacasting* para oferecer serviços de internet via canal de difusão (*downstream*), a mesma é considerada uma escolha natural. Vários trabalhos, como [1], apontam o *IP datacasting* através do *MPE* como solução para diversas aplicações utilizando o canal de difusão da televisão digital, principalmente no que se refere à recepção móvel.

A grande diferença dos dois meios de *datacasting* apontados até então, o *data piping* e o *data streaming* reside no fato de que no MPE do tipo *IP Datacasting* os dados podem ser facilmente endereçados a um terminal de acesso (*unicast*) ou a grupos de terminais de acesso (*multicast*). Não que isso não seja possível com os dois tipos anteriores, mas o endereçamento, nesse último caso, é uma característica intrínseca ao protocolo IP (e conseqüentemente, ao *IP datacasting*).

O endereçamento dos receptores ocorre de forma similar a uma rede IP sobre Ethernet: cada receptor possui um endereço de *hardware* fixo, o endereço MAC (*Media Access Control*). Também podem haver outros tipos de endereçamento, como o número de série do aparelho ou o número de

algum *smart card* acoplado ao receptor. Esses endereços são mapeados para endereços IPs para que os datagramas IP alcancem corretamente seus destinatários. Tal tarefa é realizada por descritores de roteamento nas PSIs do fluxo de transporte ou, como definido no sistema DVB, através de uma tabela adicional denominada *IP/MAC Notification Table*, ou INT.

Outra diferença com relação ao *data streaming* através das *PES* é que as *seções de datagrama* não contêm informações temporais. Dessa forma, qualquer tentativa de sincronização deve ser de responsabilidade da implementação ou efetivada através de outros recursos do DSM-CC.

4.2.4 Carrosséis

Os carrosséis são protocolos de difusão de dados definidos pelo padrão DSM-CC como mostrado na seção 3.3. O nome deriva de seu propósito que é o de repetir ciclicamente determinado conjunto de dados em um fluxo de transporte. Essa característica auxilia o receptor no acesso aos dados, visto que o mesmo, em busca de determinada informação, necessita apenas aguardar sua próxima repetição. Também é possível, à nível do terminal de acesso, adotar mecanismos de armazenamento prévio de conteúdo de um carrossel, de forma a agilizar a disponibilidade dos dados à alguma aplicação.

Carrossel de Dados

O carrossel de dados, tipo de carrossel mais simples definido pelo DSM-CC, organiza os dados logicamente em módulos. Como os dados são encapsulados nessas estruturas, essa forma de *datacasting* é fortemente recomendada para a difusão de dados delimitados. Porém, não há nenhuma descrição sobre qual o tipo do dado que é transportado por um módulo, e seu correto tratamento é de responsabilidade da aplicação. O carrossel de dados permite também que determinados módulos sejam difundidos em intervalos regulares de tempo menores que outros. Dessa forma, é possível otimizar o acesso a determinados módulos reduzindo seu tempo de espera por parte do terminal de acesso. Trabalhos como [10], [12] e [4] apresentam algoritmos de otimização de alocação de módulos (e objetos) em carrosséis.

A princípio, o DSM-CC não define nenhum mecanismos na codificação, das próprias mensagens do carrossel, de forma a incluir estampilhas de tempo para a implementação de *datacasting* sincronizado. Porém, alguns sistemas de televisão digital, como [28], recomendam o uso de determinados campos das mensagens do carrossel para a inserção dessas estampilhas de tempo. Outra solução para o problema do sincronismo é a utilização de outras estruturas, como tabelas adicionais as PSIs, as quais fornecem informações de sincronismo dos fluxos elementares do carrossel. O DSM-CC define alguma dessas soluções, as quais podem ser implementadas também de forma proprietária pelos sistemas de televisão digital.

Um carrossel de dados é localizado em um fluxo de transporte através de referências de seu fluxo elementar nas PSI. O carrossel pode também ser fragmentado em vários fluxos elementares. Nesse caso, a PSI contêm uma referência ao fluxo que carrega a mensagem de controle de maior nível: uma *DSI* em um carrossel de duas camadas ou uma *DII* em um de uma camada. A partir da localização dessa mensagem, todos os módulos que pertencem ao carrossel podem facilmente ser encontrados no fluxo de transporte.

Carrossel de Objetos

Como apresentado na seção 3.3.2, os carrosséis de objetos baseiam-se nas definições dos carrosséis de dados, porém passam a tratar a informação na forma de objetos. Para a difusão de dados, dois objetos são de maior importância em um carrossel deste tipo: objetos do tipo *arquivo* e do tipo *diretório*. Com esses dois tipos de objetos é possível formar um sistema de arquivo simples. Dessa forma, o terminal de acesso pode acessar arquivos de um sistema de arquivos que está sendo difundido em um carrossel de objetos como se os mesmos estivessem disponíveis localmente. Devido a essa característica, o carrossel de objetos pode ser definido como um sistema de arquivos de difusão [3].

Os carrosséis de objetos são a escolha natural para a difusão de dados delimitados armazenados na forma de arquivos, como ocorre na maioria dos sistemas de arquivos atuais. Dessa forma, também é a solução mais empregada na difusão de aplicações, como as *Xlets*, assim como os demais recursos utilizados por essa.

Da mesma forma como ocorre com o carrossel de dados, o carrossel de objetos pode ser utilizado para *datacasting* assíncrono ou sincronizado. Nesse último caso, como citado no tópico anterior, os mecanismos de sincronização podem ser implementados por outras estruturas, e não diretamente nas mensagens do carrossel de objetos.

4.2.5 Comparação entre os Mecanismos de *datacasting*

Dadas as classificações dos dados apresentadas na seção 4.1, a tabela 4.1 mostra a matriz de seleção proposta para o emprego do mecanismo de *datacasting* de acordo com o tipo dos dados e seus requisitos temporais [28].

Como mostrado na tabela 4.1 e citado na seção 4.2.4, os carrosséis são preferidos para a difusão de dados delimitados, visto que os mesmos partem do princípio que os dados são estruturados em módulos ou objetos. O envio de dados não delimitados e assíncronos pode por sua vez ser implementado via *data streaming* através de *seções privadas*. Já para o caso síncrono e sincronizado, a melhor opção como apresentado na seção 4.2.2 é através de encapsulamento em *PES*.

O MPE é a escolha natural quando se deseja difundir dados em datagramas, visto que esse protocolo foi desenvolvido justamente com a finalidade de atender esse propósito. Quando a difusão

Tabela 4.1: *Matriz de recomendação do mecanismo de* datacasting *de acordo com o tipo de dado a ser difundido e seus requisitos temporais.*

	Dados Delimitados	**Dados Não Delimitados**	**Datagramas**
Assíncrono	Carrosséis	*Data Streaming* via Seções Privadas	MPE
Síncronos	Sem recomendação [28]	*Data Streaming* via PES	*Data Streaming* via PES
Sincronizado	Carrosséis	*Data Streaming* via PES	*Data Streaming* via PES

de datagramas passa a ser síncrona ou sincronizada, o mecanismo preferido passa a ser através das PES. Porém, é possível a implementação de aplicações de *datacastig* de datagramas síncronos e sincronizados via MPE, mas nesse caso, as estruturas de controle de sincronização são dependentes da implementação.

O *data piping* não foi incluído na seleção apresentada pela tabela 4.1. Isso se deve ao fato de que os requisitos que os dados difundidos possuem, no caso dessa solução, devem ser atendidos geralmente de forma proprietária, já que o *data piping* não possuí recursos nativos de identificação de dados nem de temporização, como apresentado na seção 4.2.1.

4.3 Considerações Finais

O envio de dados tanto em sistemas de televisão como nos de rádio analógicos é possível. Um exemplo é o serviço de teletexto, comum na televisão analógica da Europa. Porém, devido a restrições técnicas, enriquecer a programação, ou difundir dados independentes, fez com que essa abordagem se restringisse a poucas aplicações. Com as tecnologias da televisão digital, essas restrições são suportadas. Utilizando a mesma infra-estrutura necessária para a transmissão de vídeo e áudio codificados digitalmente, o *datacasting* pode ser viabilizado sem grandes custos adicionais na difusão dos mais diversos tipos de informação.

Especificações já estabelecidas e adotadas pelos três principais sistemas de televisão digital fornecem diversos mecanismos de suporte ao *datacasting*. O *MPEG-2 Sistemas*, adotado pelos três, possibilita a difusão de informações tanto fortemente sincronizadas com a programação normal (fortemente acopladas) como o envio de informações totalmente independentes dessa (não-acopladas). Assim, o *datacasting* pode ter por finalidade tanto agregar valor aos programas televisivos das emissoras bem como disponibilizar serviços tanto para instituições públicas como privadas. Independente de qual finalidade, o *datacasting* passa a ter importância fundamental em arquiteturas e modelos de sistemas de televisão digital, motivo pelo qual, foi descrito nesse capítulo.

Capítulo 5

Modelo de um Serviço de Datacasting

Sistemas de Televisão Digital vêm sendo desenvolvidos e padronizados nos últimos dez anos. Inúmeras propostas de cadeias de valores, arquiteturas e novos serviços têm surgido durante esse tempo, envolvendo os três principais sistemas abertos, não existindo, contudo, um consenso ou uma proposta vencedora.

Serviços de *datacasting* pela TVD tendem a se tornar estratégicos nos próximos anos [1]. Existem esforços de padronização nesse sentido, contudo muitos deles são direcionados ao encapsulamento de dados em datagramas IP, como proposto por [1], [29] e [30].

Este trabalho se insere nesse contexto propondo um modelo simplificado de *datacasting*, fim a fim, baseado na composição de modelos parciais já existentes nos sistemas DVB e ATSC, através do uso de padrões abertos empregados também por esses dois sistemas de televisão digital. O modelo é direcionado para a difusão de aplicações e demais dados delimitados via *carrossel de objetos*, assim como para *data streaming* assíncrono via *seções privadas*.

5.1 Modelo de Datacasting

O modelo de *datacasting* proposto neste trabalho pode ser dividido primeiramente em quatro componentes básicos: *Provedor de Conteúdo*, *Difusor*, *Meio de Difusão* e *Terminal de Acesso*. O primeiro, como o próprio nome indica, fornece conteúdo ao *difusor*. O conteúdo pode ser na forma de mídia audiovisual, dados delimitados e *data streaming* assíncrono.

O *difusor* é encarregado de multiplexar o conteúdo fornecido pelo *provedor de conteúdo* em um único fluxo de transporte de acordo com os padrão *MPEG-2 Sistemas*, de forma que o mesmo possa ser difundidos através do *meio de difusão*. Esse pode entregar os dados ao terminal de acesso de diversas formas, via satélite, cabo, radiodifusão ou via redes digitais de alta velocidade, como

apresentado no capítulo 2. A função do meio de difusão é receber um fluxo de transporte MPEG-2 do difusor e transportá-lo até o terminal de acesso, independentemente da tecnologia utilizada para isso.

O *terminal de acesso* por sua vez é o beneficiário do serviço. A figura 5.1 ilustra o fluxo de informação entre esses quatro componentes. Tanto o *difusor* como o *terminal de acesso* foram previstos com mais detalhes, os quais são apresentados a seguir.

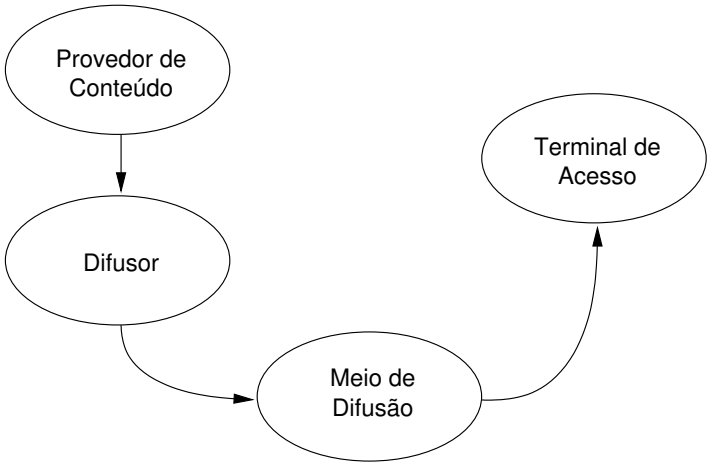

Figura 5.1: *Visão geral de um modelo de serviço de* datacasting.

5.1.1 Difusor

A objetivo do *difusor* é gerar um fluxo de transporte MPEG-2, contendo um ou mais serviços de mídia audiovisual, dados, ou ambos. A figura 5.2 ilustra o modelo do *difusor*, indicando o fluxo de dados entre seus componentes assim como as informações de controle. Este modelo do *difusor* é uma simplificação[1] do modelo de serviço de *datacasting* introduzido por [27].

O *servidor de conteúdo* pode tanto oferecer mídia audiovisual como outros tipos de dados ao *difusor*. A mídia audiovisual é direcionada para os codificadores de vídeo e áudio. As saídas desses, para cada conjunto de fluxos de vídeo e áudio relacionados, são direcionadas a um *multiplexador de serviço*. A saída do *multiplexador de serviço* é um fluxo de transporte MPEG-2 contendo um serviço com as PSIs básicas configuradas, como mostrado na seção 3.2.3. Vários multiplexadores de mídia podem trabalhar em paralelo, gerando vários serviços, os quais são direcionados ao *multiplexador de emissão*.

Os dados a serem difundidos, fornecidos pelo *provedor de conteúdo*, são encaminhados ao *codificador de dados*. A interface entre o *provedor de conteúdo* e o *codificador de dados* não é foco deste modelo, mas é abordada em trabalhos como [31].

[1] O modelo apresentado nesse trabalho não possui o módulo de *acesso condicional*, presente no modelo sugerido por [27]

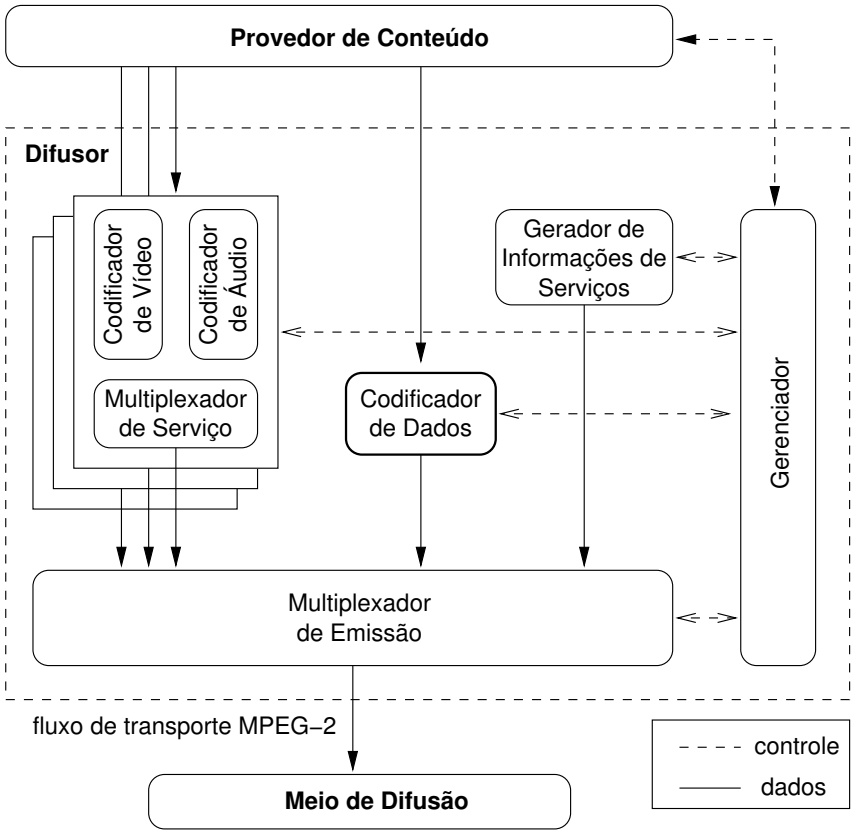

Figura 5.2: *Fluxo de dados entre os componentes do difusor.*

A figura 5.3 ilustra o fluxo de dados no codificador de dados. Dados delimitados, como arquivos, aplicações, etc, são codificados em *carrosséis de objetos*. Essa tarefa é realizada pelo *codificador de carrosel*. A saída desse codificador é um fluxo elementar, já encapsulado em pacotes de transporte, contendo as estruturas apresentadas na figura 3.14 e codificadas de acordo a especificação [25]. O *gerador* por sua vez é responsável por repetir ciclicamente esse fluxo de dados em uma das entradas do multiplexador de emissão, de forma que a difusão dos dados adquira o comportamento de carrossel (repetição cíclica dos dados, conforme a figura 3.11).

Já os dados não delimitados, ou *data streaming*, são encaminhados ao codificador de seções privadas, responsável por encapsular os dados de acordo com as estruturas sintáticas introduzidas na seção 3.2.3. O fluxo elementar de dados de saída do codificador, já encapsulado em pacotes de transporte, é direcionado também a uma das entradas do multiplexador de emissão. A figura 5.4 ilustra de forma simplificado o modelo de um codificador desse tipo.

Os serviços de *datacasting* gerados pelo *codificador de dados* podem ser incluídos nos serviços de mídia, definindo um *datacasting fracamente acoplado*, ou serem definidos como serviços não acoplados. Tais informações se encontram nas PSI, PAT e PMT, e tabelas adicionais, produzidas pelo *gerador de informações de serviço*, conforme apresentado na seção 3.2.3. Nestas tabelas também são

inseridas informações descrevendo os serviços assim como as sinalizações de aplicações difundidas em carrosséis de objetos.

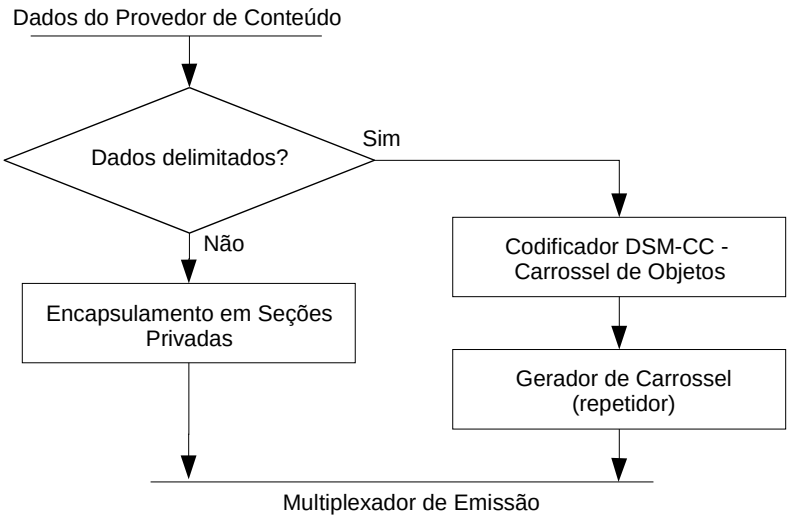

Figura 5.3: *Fluxo de dados no codificador de dados.*

O *gerenciador* por sua vez é o componente responsável pelo controle desses componentes. O mesmo passa parâmetros ao *gerador de informações de serviço* para a correta geração das tabelas ligando os fluxos elementares dos serviços, obtidos a partir do *provedor de conteúdo* e dos componentes do difusor.

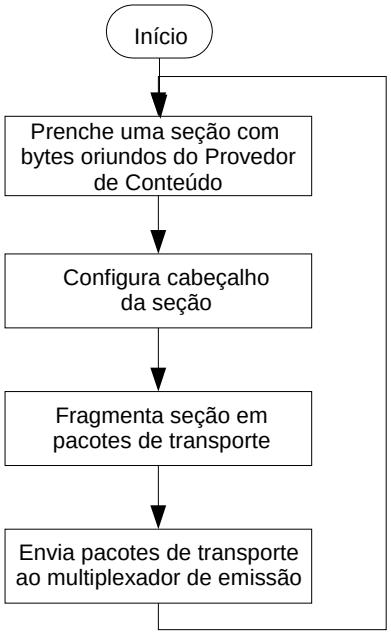

Figura 5.4: *Fluxograma do mecanismo de encapsulamento de dados em seções privadas.*

5.1.2 Terminal de Acesso

A figura 5.5 ilustra em mais detalhes o fluxo de dados previsto em um *terminal de acesso*. O modelo pressupõe que determinada aplicação está ou vai ser executada no terminal de acesso, e a mesma fará uso de dados difundidos. A aplicação pode, inclusive, ser difundida com os dados, carregada e executada pelo terminal de acesso. Esse modelo de terminal de acesso é uma versão simplificada[2] do modelo proposto por [19].

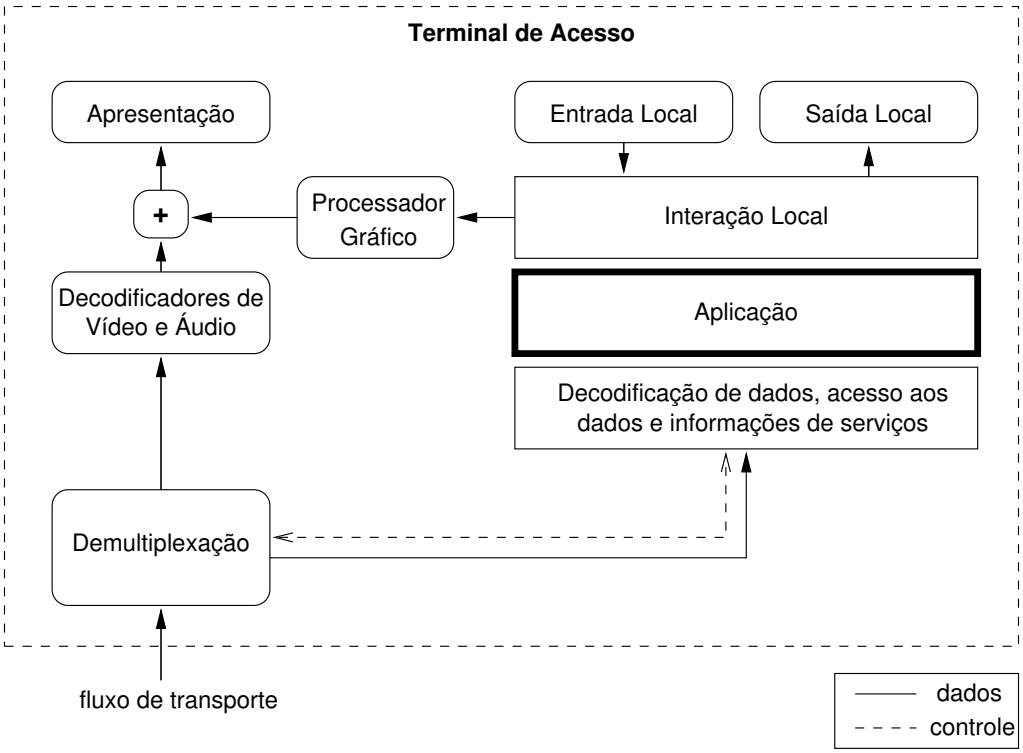

Figura 5.5: *Fluxo de dados em um terminal de acesso.*

O sinal de entrada do terminal, como mostra a figura 5.5, é um fluxo de transporte MPEG-2 com vários serviços multiplexados, incluindo os serviços de *datacasting*. As etapas comuns de sintonização e demodulação que ocorrem em um terminal de acesso não foram citadas no modelo, visto que dependem da forma como o sinal digital é modulado e transmitido.

A primeira tarefa do demultiplexador é encontrar as tabelas que descrevem logicamente os serviços transportados no fluxo de transporte entregue pelo meio de difusão. A primeira dessas tabelas é PAT, que declara os serviços presentes, como apresentado na seção 3.2.3. Após determinado serviço ser selecionado, o demultiplexador passa a procurar a PMT correspondente. Após encontrada, os fluxos de mídia e de dados do serviço são localizados. A figura 5.6 ilustra os estados que o

[2] O modelo apresentado por [19] apresenta alguns componentes adicionais em relação ao apresentado nesse trabalho, como por exemplo, módulo de acesso condicional, controle sobre um sistema de apresentação de mídia assim como a presença de um canal de retorno

demultiplexador assume nesse processo.

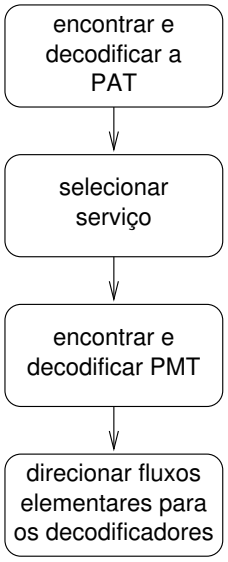

Figura 5.6: *Etapas de processamento típicas de um demultiplexador.*

Após identificados os fluxos de mídia audiovisual no serviço selecionado pelo terminal de acesso, os mesmos são direcionados a seus respectivos decodificadores. Já o processo de decodificação dos dados é diferente: após a seleção do serviço, o terminal de acesso deve verificar se alguma aplicação em execução (residente ou pertencente a outro serviço previamente selecionado) está requisitando dados do serviço. Em caso positivo, o terminal de acesso indentifica em qual fluxo elementar do serviço esses dados são transportados, inicia sua decodificação, e encaminha os dados em sua forma original à aplicação. O fluxograma da figura 5.7 ilustra esse processo.

Caso nenhuma aplicação esteja requisitando dados, ou o terminal de acesso já tenha encaminhado os dados requisitados, ele passa a procurar por tabelas de sinalização de aplicações no serviço. Caso encontre alguma, e a mesma é nova (não foi tratada anteriormente), o terminal de acesso localiza o fluxo elementar em que está sendo difundido o carrossel de objetos que carrega a aplicação. Após localizado, o carrossel é totalmente decodificado, e a aplicação carregada e executada.

As aplicações podem interagir com dois componentes vizinhos, como mostra a figura 5.6: *Acesso aos dados e informações do serviço* e *Interação local*. Através do primeiro, a aplicação pode identificar em quais fluxos do serviço estão sendo difundidos dados, assim como selecionar determinado fluxo e acessar os dados difundidos neste. Através da interação local, pode receber ou enviar dados ao ambiente em que se encontra. Dessa forma, esse modelo é válido em aplicações com interatividade local.

A borda que envolve a aplicação na figura 5.5 representa a API fornecida pelo *middleware* do terminal de acesso, possibilitando que a aplicação interaja com os componentes vizinhos também ilustrados na figura.

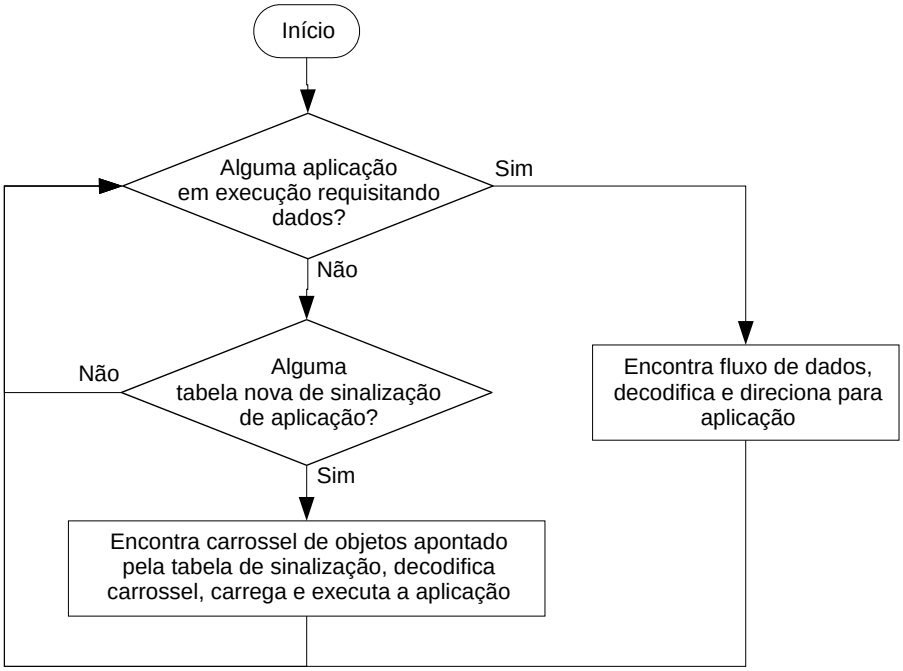

Figura 5.7: *Fluxograma descrevendo as etapas de decisão do terminal de acesso com relação a decodificação de dados.*

5.2 Exemplos de Uso do Modelo

O modelo apresentado não cobre todos os tipos de *datacasting*, contudo pode ser utilizado em uma grande gama de aplicações, sejam elas direcionadas a consumidores ou empresas, como apresentado na seção 4.1. Os próximos tópicos cobrem alguns dos possíveis exemplos de uso desse modelo nesses dois contextos.

5.2.1 Suporte à Interatividade na Televisão Digital

Quando o destinatário do serviço é o consumidor ou telespectador, alguns exemplos do uso do modelo podem ser dados, como o suporte à interatividade na televisão digital. Aplicações podem ser codificadas em carrosséis de objetos pelo difusor e difundidas como parte de algum serviço comum de televisão digital, contendo mídia audiovisual, gerada pelo multiplexador de serviço. A correta sinalização da aplicação é determinada por tabelas adicionais às PSIs produzidas pelo *gerador de informações de serviços*.

O terminal de acesso, ao receber determinada aplicação, sinalizada por tabelas de informação de serviços, a decodifica e executa. Essa aplicação pode também acessar recursos difundidos,

no próprio carrossel de objetos ou em seções privadas, e fornecer meios de interação local com o telespectador.

5.2.2 Difusão de Informações de Interesse Público e Alertas em Tempo Real

Outro uso do modelo é no envio de informações relacionadas a condições climáticas, situações emergenciais, trânsito, acidentes em rodovias, boletins policiais, etc. Aplicações no terminal de acesso podem fazer uso de dados relacionados a essas situações, continuamente difundidos em seções privadas, apresentando-as propriamente ao usuário. O fluxo de transporte contendo as informações importantes pode ser difundido por meios que permitam recepção móvel, possibilitando o acesso as mesmas em equipamentos com o modelo do terminal de acesso implementado, como celulares, dispositivos em automóveis, e etc. A sugestão apresentada em [32], de transmitir via FM dados sobre a situação dos ônibus de um sistema de transporte, por exemplo, pode ser viabilizado via sinal de TVD a partir do modelo aqui proposto.

5.2.3 Distribuição de Material Digital

O modelo pode ser utilizado para a implementação de sistemas de distribuição de qualquer tipo de informação, como por exemplo, material escolar no formato digital. Dados desse tipo podem ser codificados em carrosséis de objetos, e o fluxo de transporte enviado a distribuidores locais. Esses podem remultiplexar o fluxo de transporte com os demais serviços da programação local.

Os terminais de acesso podem ser estendidos de forma a possuir unidades de armazenamento, como apresentadas na seção 2.3.1, nos quais as operações de armazenamento são fornecidas pelo *middleware* para a aplicação. Também é possível estender o terminal de acesso através de uso de interfaces de comunicação com PCs, apresentadas na seção 2.3.1 de forma a possibilitar que o conteúdo difundido seja transferido a microcomputadores.

5.2.4 Atualização de Dados em Terminais Fixos ou Móveis

O modelo pode ser empregado na atualização de dados em terminais móveis ou fixos, como quiosques, por exemplo [6]. Os dados, codificados em carrosséis ou seções privadas, são filtrados por esses terminais. Apenas uma pequena parte da API comum em *middlewares* nesse caso precisa ser implementada, de forma a fornecer à aplicação acesso a esses dados.

5.2.5 Controle de Dispositivos Remotos

Informações de controle também podem ser difundidas utilizando-se o modelo. Um cenário possível é a difusão de informações de controle de semáforos em grandes centros urbanos. Esses, com

uma implementação parcial do modelo do terminal de acesso, acessam os dados multiplexados nos sinais de TVD, fornecidos pelo distribuidor local. Esses dados podem ser informações de controle, alterando as configurações desses equipamentos sem a necessidade de intervenção local ou através do estabelecimento de redes de outra natureza. Esse cenário pode ser estendido para qualquer outro equipamento que necessite de atualização periódica de dados, estando distante de sua central de controle.

5.3 Trabalhos Relacionados

Esta seção apresenta alguns trabalhos na área de televisão digital que abordam o *datacasting* como um novo serviço nesse contexto. Esses trabalhos em geral buscam definir cadeias de valores, arquiteturas e modelos, porém, como citado por [1], o assunto ainda não é totalmente coberto por padrões abertos em todos os níveis de um sistema completo de *datacasting*. Dessa forma, não existe consenso de uma arquitetura ou modelo definitivo.

5.3.1 DIWG - *Data Implementation Work Group*

A *DIWG* é um grupo de trabalho da ATSC responsável por produzir recomendações para a implementação de serviços de *datacasting* para o sistema de televisão digital norte-americano. Um dos primeiros trabalhos na área, [27], datado de 1999, propõem um modelo para a geração de serviços de *datacasting* baseado em arquitetura de difusores de televisão digital. O objetivo do grupo é de tornar possível a implementação de serviços de *datacasting* utilizando equipamentos já previstos em estações de emissão ATSC com o menor número possível de modificações.

Ao contrário da maioria dos trabalhos na área, as quais buscam propor modelos de negócios para o *datacasting*, as recomendações da DWIG focam na arquitetura do sistema de difusão, ou seja, o núcleo do sistema que viabiliza a implementação dos serviços. Em [27] são abordados tópicos como a interconexão entre componentes de um difusor de TVD, o gerenciamento e a passagem de parâmetros para os equipamentos pertencentes ao sistema, controle de dados para *datacasting oportunístico*, assim como a sincronização dos dados.

O modelo proposto nesta dissertação é baseado no apresentado por [27], com algumas simplificações. Uma delas é com relação a não inclusão do módulo de acesso condicional, descrito por [27]. A outra é com relação ao *datacasting oportunístico* assim como a difusão de dados com algum mecanismo de sincronismo a nível do *MPEG-2 Sistemas*, que são tratadas por [27].

5.3.2 TVD Interativa Baseado na Proposta da DWIG

O trabalho [5], baseado na recomendação da DIWG [27], propõe um sistema de difusão de dados interativo, seguindo o sistema de TVD da ATSC. Foi formulado por membros da ETRI, *Electronics and Telecommunications Research Institute*, uma organização de pesquisa do governo sul coreano na área de tecnologia da informação.

Nesse trabalho, o modelo da DIWG é estendido de forma a fornecer serviços interativos. Para tal, o trabalho [5] adiciona um injetor de dados ao modelo apresentado por [27], que calcula e configura as estampilhas de tempo das estruturas transportando os dados de forma a sincronizá-las com determinado fluxo de mídia. Tal componente foi julgado necessário pela equipe da ETRI, visto que multiplexadores de serviços normais, como sugeridos pela DWIG, não apresentam a propriedade de multiplexar dados com os fluxos audiovisuais. Dessa forma, são previstos serviços interativos, fortemente acoplados com a programação televisiva.

O modelo de [27] também foi estendido por [5] de forma a incorporar um canal de retorno. Uma experiência de testes do sistema foi realizada, e as discussões sobre os resultados obtidos também são apresentados em [5].

5.3.3 Mecanismos Determinísticos e Estocásticos de Transmissão de Dados

Azimi, em sua dissertação de mestrado [33], propõe um sistema interativo de televisão onde, concorrente aos fluxos principais de vídeo e áudio, são difundidos fluxos de mídia oportunísticos, além de dados diversos. Esse trabalho prevê dois métodos nesse contexto: um determinístico, onde o fluxo concorrente preenche a banda sobressalente do fluxo principal, sem perdas de pacotes; e um método estocástico, onde possíveis perdas de pacotes são previstas. Para ambos, apresenta uma estratégia de codificação escalar de vídeo, de forma a possibilitar a melhor qualidade possível de imagem na banda disponível. Também propõe e implementa um sistema de multiplexação desses fluxos, suportando as teorias propostas. A partir desse sistema, são realizadas experiências que apresentam resultados quantitativos validando a teoria apresentada.

5.3.4 IPDC Forum

O *IPDC Forum* [34], ou *IP Datacasting Fórum*, é uma associação, sem fins lucrativos, de diversas indústrias com o objetivo de tornar possível a implementação de serviços multimídia baseados no protocolo IP sobre redes de difusão como o DVB e o DAB (*Digital Audio Broadcasting*). Estabelecido no início de 2002, inclui provedores de serviços, difusores e fabricantes de terminais de acesso. Apesar de prever a recepção fixa, tem como grande meta a entrega de serviços baseados em IP a terminais de acesso móveis, apostando no sucesso de especificações como o DVB-H, uma especificação originada da especificação de radiodifusão, destinada à recepção móvel.

Trabalhos como [29] descrevem as tecnologias e desafios envolvidos no estabelecimento de serviços propostos pelo IPDC. Alguns padrões já estão bem definidos, como a utilização do padrão *MPEG-2 Sistemas* para o transporte de datagramas IP encapsulados em *MPE* sobre meios de difusão DVB. Também foca em áreas relacionadas à segurança e garantia de entrega de serviços a consumidores selecionados, através da adoção de padrões como o DRM (*Digital Rights Management*) e sistemas de acesso condicional.

O documento [29] levanta algumas vantagens e desvantagens do IPDC com relação a outras formas de transmissão de dados. Como vantagens, cita a largura de banda disponível em um canal de difusão comum[3], difusão de serviços para receptores móveis, além da transmissão em *multicast*. Como desvantagens, são citadas a natureza da difusão, que é unidirecional, além do fato de transmissões em *unicast* ou *point-to-point* serem empregadas de melhor forma por outras tecnologias.

Staffans, em sua dissertação [1], apresenta uma visão geral das tecnologias envolvidas no IPDC, introduzidas por [29], decorrendo sobre os protocolos e padrões envolvidos no *datacasting*. Também propõem um modelo de serviço IP *datacasting*, porém reconhece que apenas algumas especificações abertas são padrões de *facto* para determinados componentes previstos em um sistema IPDC. Sistemas de servidores de conteúdo, assim como suas interfaces com os sistemas encarregados do encapsulamento e multiplexação carecem de especificações bem definidas. Também apresenta o ULE, *Ultra Lightweight Encapsulation*, como uma possível solução de encapsulamento de datagramas IP, concorrente ao MPE, como é apresentado na seção 5.3.5.

Ambos, [29] e [1], reconhecem que a divisão estabelecida em arquiteturas propostas de IPDC e o papel de cada elemento na cadeia de valores ainda não estão claramente definidos. Tal fato provavelmente se deve ao pouco tempo que o assunto está em pauta, sendo que os primeiros artigos datam de 1998. Dessa forma, a divisão nos modelos propostos e o papel de cada componente são temas de estudos de trabalhos como [9].

5.3.5 IETF IP sobre DVB

A IETF possui um grupo de trabalho, *IETF IP over DVB Working Group* [30], com o objetivo de desenvolver novos protocolos e arquiteturas, de forma a melhorar a implementação de serviços baseados em IP sobre o padrão *MPEG-2 Sistemas*. Dessa forma, pretende tornar possível o estabelecimento de redes IP sobre sistemas de televisão digital como o DVB e o ATSC.

Além do uso de MPE para o transporte de datagramas IP sobre MPEG-2, o grupo de trabalho estuda a adoção de um sistema de encapsulamento alternativo, o *ULE, Ultra Lightweight Encapsulation*. No ULE, ao contrário do MPE, os datagramas são encapsulados diretamente nas cargas dos

[3] Segundo [29], em torno de 22 Mbits para recepção fixa e 11 Mbits para recepçao móvel através do uso dos padrões da DVB

pacotes de transporte, sem o uso de seções DSM-CC. Também é esperado um mecanismo de suporte à sincronização com demais fluxos elementares. O último *draft* [35] da especificação do ULE, no momento de publicação deste trabalho, data de janeiro de 2005, e o RFC da mesma é previsto para o final desse ano.

5.3.6 Dotcast

O *Dotcast* [36] faz parte de algumas implementações de serviços de *datacasting* proprietários. A empresa que implementa o serviço sugere que o mesmo pode ser utilizado em aplicações de video sob demanda, *download* de aplicações via difusão, *e-learning*, etc. Tanto o processo de encapsulamento do conteúdo como os terminais de acesso compatíveis com o sistema são fornecidos pela *Dotcast*. A mesma utiliza banda sobressalente de emissoras de TVD para a difusão de seus serviços.

5.3.7 Moviebeam

O *Moviebeam* [37] é outra solução comercial de *datacasting*, através de tecnologia proprietária da *Disney*, para a difusão de filmes via tecnologias de TVD. Assinando o serviço da *Disney*, o consumidor recebe um terminal de acesso, que possui uma unidade de armazenamento de grande porte, com vários filmes em qualidade de DVD previamente armazenados. Outros filmes, por sua vez, são difundidos terrestrialmente, periodicamente, na área de cobertura do serviço através de seu sistema de *datacasting* proprietário. Esses são recebidos e armazenados automaticamente pelo terminal de acesso do assinante.

5.4 Considerações Finais

O modelo apresentado neste capítulo define um serviço de *datacasting* onde dados delimitados e não delimitados assíncronos podem ser difundidos. Aplicações no terminal de acesso fazem uso desses dados implementando serviços cujos consumidores são tanto telespectadores quanto empresas, como ilustrado nos exemplos da seção 5.2.

O modelo foi baseado em trabalhos recentes da área, como os apresentados na seção 5.3. Porém, enquanto a maioria desses trabalhos foca na difusão de dados via datagramas IP sobre MPEG-2, o modelo prevê o *datacasting* através do uso de mecanismos de codificação de dados via *carrossel de objetos* e *seções privadas*. O mesmo pode atender aplicações sem necessidades de endereçamento a nível de datagramas, assim como aplicações sem requisitos temporais fortes. Dessa forma, o modelo pode ser utilizado para *datacasting fracamente acoplado* ou *desacoplado*. O próximo capítulo por sua vez apresenta a implementação deste modelo.

Capítulo 6

Implementação do Modelo de Datacasting

Este capítulo tem por objetivo descrever a implementação do modelo fim a fim de *datacasting* proposto no capítulo 5. O protótipo foi implementado em uma arquitetura *intel x86*, sobre o sistema operacional *GNU/Linux*. Para tanto, foram integrados vários aplicativos, *softwares* livres já existentes, em sua maioria distribuídos sob a licença *GPL*, cada qual exercendo alguma das funções previstas no modelo, como será apresentado no decorrer deste capítulo.

Em etapas onde não foram encontradas soluções de *software* livre disponíveis, que cobrissem satisfatoriamente determinada funcionalidade, as mesmas foram desenvolvidas em POSIX C. São os casos do codificador de dados em seções privadas, demultiplexador e decodificadores de dados de carrosséis de objetos e de seções privadas. As próximas seções, de 6.1 a 6.4, apresentam a implementação, que é dividida neste capítulo em quatro partes, seguindo o modelo apresentado na figura 5.1. Por fim, a seção 6.6 apresenta algumas considerações sobre as experiências realizadas com a implementação.

6.1 Provedor de Conteúdo

Por ainda não existir padronização com relação à interface entre o *provedor de conteúdo* e o *difusor*, o primeiro foi abstraído através de recursos locais da máquina onde o *difusor* foi implementado. A mídia audiovisual é obtida através de uma *webcam*, que captura vídeo e áudio. O acesso ao conteúdo capturado se dá através de uma interface *video4linux*.

Os dados são armazenados no sistema de arquivos local. Dados delimitados são organizados em diretórios e arquivos, e como será visto no decorrer deste capítulo, o caminho dos mesmos é o parâmetro passado ao *codificador de dados*. Dados não delimitados, na forma de *data streaming*,

podem ser obtidos de diversas fontes e direcionados a um *named pipe*, ou *fifo*, do Linux. Esse *fifo* também serve como entrada para o *codificador de dados*.

6.2 Difusor

O objetivo do *difusor* é gerar um fluxo de *bits* válido seguindo a especificação *MPEG-2 Sistemas*, contendo um ou mais serviços de *datacasting* atrelados ou não a serviços audiovisuais de televisão digital. Para os serviços de *datacasting*, a implementação permite que os mesmos sejam codificados em carrosséis de objetos ou em seções privadas. Permite também que aplicações sejam codificadas em um carrossel de objetos e seu ciclo de vida sinalizado por tabelas adicionais às PSIs. Com relação à mídia audiovisual, a implementação oferece suporte à codificação de vídeo e áudio provenientes do *provedor de conteúdo*.

Sistemas de Acesso Condicional não foram abordados nessa implementação. Dessa forma, as tabelas CAT do *MPEG-2 Sistemas* não foram utilizadas. Também não foi prevista a geração da tabela NIT, visto que, como o difusor é abstraído na forma de um *fifo*, como é apresentado na seção 6.3, a mesma não possui utilidade, já que sua função é transportar informações sobre esse meio.

Assim como mostra a figura 5.2 do modelo proposto, o *difusor* é dividido em diversos componentes. Para cada um desses buscou-se soluções de *software* livre para GNU/Linux que suprissem as funções previstas no modelo. Com exceção do codificador de dados via seções privadas, parte do *codificador de dados*, todos os outros componentes do difusor foram implementados através de soluções já existentes.

Uma simplificação na implementação com relação ao modelo foi a não implementação do gerenciador de serviços, que seria responsável pela parametrização e controle de todos os demais componentes do *difusor*. Dessa forma, essa parametrização e configuração dos mesmos é realizada individualmente e no momento da inicialização do protótipo. Essa abordagem confere ao sistema uma menor dinamicidade, ou seja, a impossibilidade de se realizar determinadas alterações em seu comportamento em tempo real. Porém, resultou em uma implementação de menor complexidade visto que, por natureza, cada componente utilizado na implementação provem de uma fonte diferente, não existindo interfaces de comunicação ou de controle entre os mesmos.

As próximas seções apresentam os aplicativos utilizados para implementação de cada componente previsto na figura 5.2. O codificador em seções privadas, integrante do *codificador de dados*, será apresentado em mais detalhes visto que o mesmo foi desenvolvido como parte deste trabalho.

6.2.1 Codificador de Dados

O *codificador de dados* é composto por dois aplicativos diferentes: um capaz de codificar dados em carrosséis de objetos e o outro em seções privadas. A saída dos dois aplicativos corresponde à saída de dados do *codificador de dados* como mostrado na figura 5.2. Cada uma dessas saídas é direcionada a um *fifo* diferente, os quais são entradas do *multiplexador de emissão*.

Codificador de dados em Carrosséis de Objetos

Para a codificação de dados em carrosséis de objetos foi utilizado o aplicativo *dsmcc-oc* pertencente ao conjunto de ferramentas *dsmcc-mhp-tools* [38]. O mesmo é capaz de codificar um conjunto de arquivos e diretórios, formando um fluxo elementar encapsulado em pacotes de transporte como ilustrado na figura 3.14, seguindo os padrões MPEG-2 e DSM-CC[1].

A entrada para o *dsmcc-oc* é o caminho de um diretório, da abstração do provedor de conteúdo, que deve conter os arquivos e demais diretórios a serem codificados. A saída é um arquivo codificado na forma de uma seqüência de transporte MPEG-2. O *dsmcc-oc* permite que alguns parâmetros de codificação sejam especificados na chamada do aplicativo. Os principais são o PID do fluxo do carrossel de objetos, a identificação do carrossel (*CarouselId*) e o *component tag* ou *association tag*. Os dois últimos, como explicado na seção 3.3.2, auxiliam na localização do carrossel no momento da demultiplexação. Também é possível, através do *dsmcc-oc*, dividir um carrossel em vários fluxos com valores de PID diferentes.

Os carrosséis de objetos gerados pelo *dsmcc-oc* não possuem informações temporais ou de sincronismo em nível de codificação de dados. Apesar do guia de implementação de *datacasting* da ATSC [28] indicar a possibilidade do uso de um campo das estruturas dos carrosséis de dados para a inserção de informações desse tipo, a abordagem mais comum para atender aplicações desse gênero é através do uso de tabelas NPT e *stream events*. Apesar de ser possível a criação dessas tabelas e fluxos contendo *streams events* através do *dsmcc-mhp-tools* (*dsmcc-npt* e *dsmcc-ste*), tal tentativa de sincronização não é implementada neste trabalho.

Como apresentado até então, a saída do *dsmcc-oc* é um arquivo contendo um fluxo de transporte, que contém um carrossel de objetos. Como, por definição, o conteúdo de um carrossel de objetos deve ser difundido ciclicamente, executar repetitivamente o *dsmcc-oc* e direcionar sua saída para um *fifo* seria uma das possíveis soluções para a geração de um carrossel de objetos. Porém, a cada chamada desse aplicativo, todo o carrossel é reconstruído. De forma a evitar consecutivas chamadas do *dsmcc-oc*, foi utilizado outro aplicativo, o *repeatts*, pertencente ao conjunto *multiplexer* [39]. O *repeatts* lê um arquivo de entrada, no caso o arquivo contendo a estrutura do carrossel de objetos,

[1] Como o *dsmcc-oc* foi desenvolvido para a geração de carrosséis de objetos seguindo a especificação MHP, o mesmo apresenta algumas restrições e simplificações impostas por essa especificação ao DSM-CC.

e ciclicamente o repete em sua saída, no caso da implementação, em um *fifo*. Parâmetros como a freqüência dos ciclos de repetição são fornecidos na chamada desse aplicativo.

Codificador de dados em Seções Privadas

A codificação em seções privadas foi implementada através do desenvolvimento de um aplicativo, que obtém os dados provenientes do *provedor de conteúdo* através de um *fifo*, e os codifica de acordo com a sintaxe das seções privadas especificadas pelo *MPEG-2 Sistemas*. Para fins de testes, foi implementada também a opção de encapsular, em seções privadas, a informação sobre a hora atual no sistema onde o aplicativo está sendo executado, na forma de horas, minutos e segundos. Tal opção pode ser ativada via linha de comando, no momento da ativação da aplicação.

O aplicativo escreve a estrutura codificada em um arquivo, ou seja, sua saída pode ser direcionada para um *fifo*, forma que é utilizado na implementação. O valor do PID do fluxo gerado é especificado via passagem de parâmetros em sua inicialização.

Descrição do mecanismo de codificação

O codificador utiliza dois contadores para numerar os pacotes de transporte e seções: o *contador de continuidade* e o *identificador estendido de tabelas*. O primeiro é incrementado e inserido em cada pacote de transporte, gerado para carregar os fragmentos de uma seção. Sua utilidade está em permitir ao decodificador identificar a ocorrência de perdas de pacotes. Já o *identificador estendido de tabelas* é incrementado a cada nova seção, abordagem adotada na numeração de seções.

Após a inicialização do aplicativo, o mesmo entra em um *loop* onde a primeira tarefa é ler a entrada de dados. São lidos no máximo 4084 *bytes* a cada ciclo, tamanho máximo de dados que uma seção privada com cabeçalho completo pode transportar. Com os dados a serem codificados, seu tamanho, e o contador de seções representado pelo campo *identificador estendido de tabelas*, os demais campos presentes no cabeçalho de uma seção, como mostrado na seção 3.2, são configurados. Ao *identificador de tabelas* é atribuído um valor qualquer, dentro dos valores não reservados e permitidos pela especificação *MPEG-2 Sistemas*; o *bit* do *indicador de sintaxe* da seção é ligado, indicando que a mesma possui um cabeçalho estendido contendo os demais campos; os campos *número de seção* e *número da última seção* possuem valores igual a zero, já que nessa implementação o contador utilizado é o *identificador estendido de tabelas*.

A partir dos valores de cada campo do cabeçalho da seção, esta é codificada em uma seqüência de *bytes* de acordo com a especificação MPEG-2. Os primeiros 8 *bytes* formam o cabeçalho, os *bytes* posteriores contêm os dados a serem transportados, e a partir desses é calculado o valor do CRC de 32 *bits*. Esse último é inserido ao final da seção, ocupando os últimos 4 *bytes*.

O próximo passo é encapsular a seção em pacotes de transporte. O processo entra em um novo *loop*, onde a seção é fragmentada em vários pacotes. Para a codificação dos pacotes de transporte, os campos do cabeçalho devem ser corretamente configurados: ao valor do *PID* é atribuído o valor obtido como parâmetro fornecido na inicialização da aplicação; para o primeiro pacote, que carrega o primeiro fragmento de seção, o *bit indicador de início de carga* é ligado; o *controle de criptografia* indica que o conteúdo do pacote não está criptografado; e por fim o *controle do campo de adaptação* indica que o pacote não possuí essa extensão do cabeçalho. No quinto *byte* do pacote se encontra o ponteiro, indicando que o início da seção se dá após o mesmo.

A cada ciclo do *loop*, para cada novo pacote criado, contendo os demais fragmentos da seção, os valores dos campos do cabeçalho do pacote de transporte permanecem os mesmos, com exceção do *contador de continuidade*, que é incrementado a cada novo pacote. Além desse, o *bit indicador de início de carga* é desligado para todos os pacotes que não transportam o primeiro fragmento da seção. Também não existe o campo ponteiro nesses pacotes, como explicado na seção 3.2.3, e o quinto *byte* do pacote corresponde ao primeiro *byte* do fragmento.

O último passo do *loop* principal é escrever a seqüência de *bytes*, representado os pacotes de transporte que carregam uma seção, no *fifo* de saída da aplicação. Ao final desse *loop*, o *identificador estendido de tabelas* é incrementado, valor que será utilizado no próximo ciclo para numerar a próxima seção. A figura 6.1 ilustra as etapas de cada *loop* descritas até então.

6.2.2 Codificadores de Mídia e Multiplexador de Serviço

Para a implementação do codificador de mídia e do multiplexador de serviço, foi utilizada uma única solução capaz tanto de codificar áudio e vídeo segundo as especificações MPEG-1 ou MPEG-2, em PES, assim como encapsular as PES em um fluxo de transporte MPEG-2: o *ffmpeg* [40]. A entrada desse aplicativo é proveniente, nessa implementação, do dispositivo de captura de vídeo do *provedor de conteúdo* através da interface *video4linux*.

Além de gerar os pacotes de transporte com as PES encapsuladas, o *ffmpeg* gera também as tabelas PAT e PMT, contendo as estruturas mínimas exigidas pela especificação MPEG-2. A saída deste aplicativo é direcionada a um *fifo*, que é uma das entradas do *multiplexador de emissão*.

6.2.3 Gerador de Informações de Serviços

A geração das PSIs, e demais tabelas adicionais presentes em um fluxo de transporte, depende forte e claramente das informações referentes aos serviços transportados no fluxo. O *gerenciador*, de acordo com o modelo proposto na figura 5.2, é o componente responsável pela obtenção dessas informações, e de fornecê-las ao *gerador de informações de serviços* para que o mesmo possa criar as

tabelas corretamente referenciando e sinalizando os serviços disponíveis. Como o *gerenciador* não foi implementado, o *difusor* assume um comportamento estático com relação à produção dessas tabelas, visto que não foi implementada uma forma de se trocar informações entre o *gerador de informações de serviços* e os demais componentes de forma a criar e atualizar essas tabelas dinamicamente. Dessa forma, essas tabelas são criadas, na maioria das vezes, na inicialização de todo o sistema.

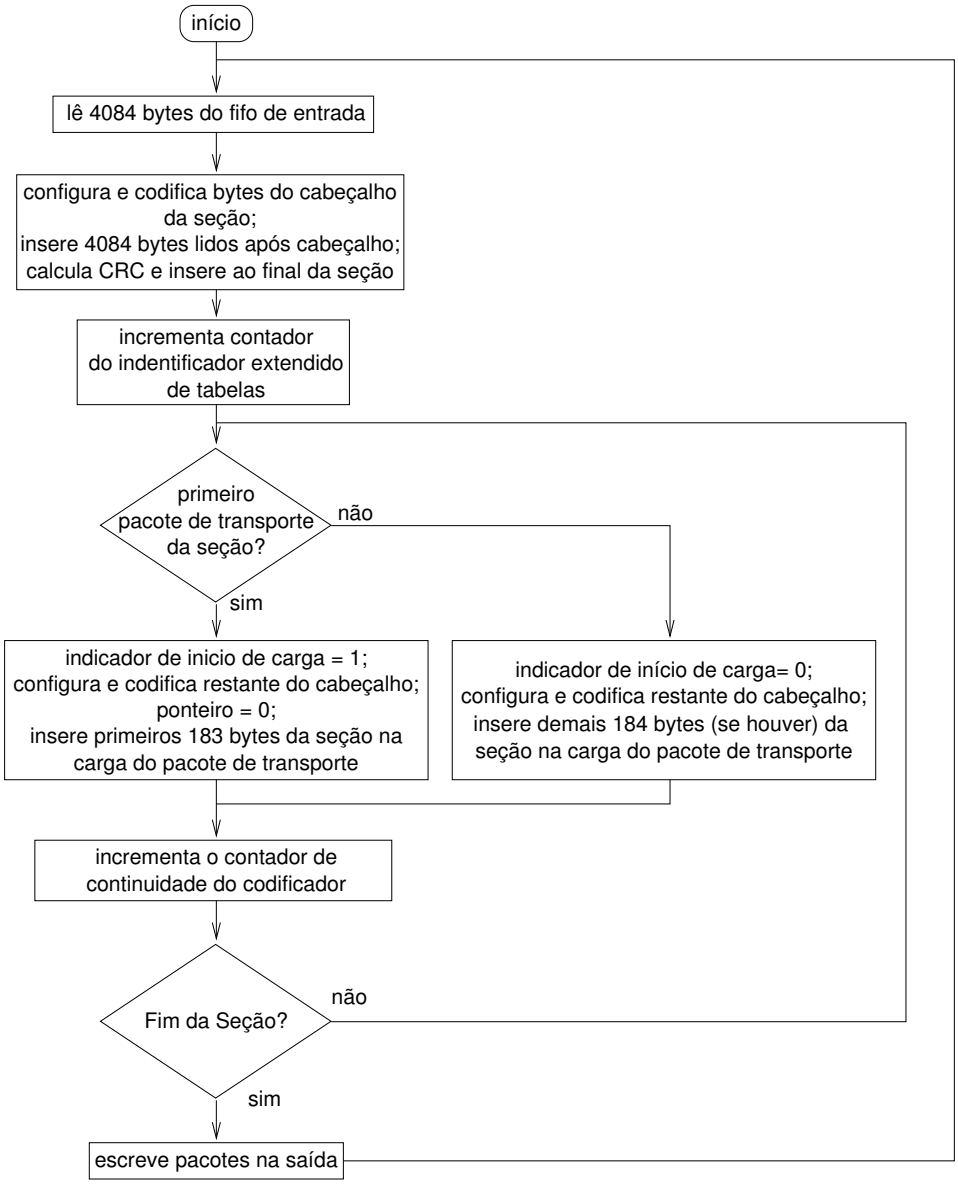

Figura 6.1: *Etapas de processamento dos dados do codificador de seções privadas.*

Devido à capacidade apresentada pelo *multiplexador de emissão*, presente nessa implementação, de possibilitar a edição das tabelas PAT e PMT através de parâmetros passados ao mesmo no momento da execução, não foi necessário gerar essas duas tabelas através de um aplicativo a parte. Como o *ffmpeg*, em seu fluxo de saída, já fornece essas duas tabelas com uma estruturação mínima,

as mesmas podem ser editadas de forma a referenciar e especificar corretamente os demais serviços e fluxos elementares via *multiplexador de emissão*.

O modelo proposto na seção 5.1.2 prevê que determinada aplicação será encapsulada, difundida e executada no receptor. As PSIs não fornecem em sua sintaxe informações suficientes relacionadas à sinalização de aplicações difundidas em fluxos de transporte. Dessa forma, a solução escolhida nesta implementação foi o uso das tabelas AIT, *Application Information Table*, definidas pelo padrão MHP [19] do sistema DVB.

Para a geração de uma AIT, um aplicativo pertencente ao conjunto *dsmcc-mhp-tools* foi utilizado, o *dsmcc-ait*. O mesmo produz a tabela e a insere em uma seqüência de transporte de determinado PID, especificado na chamada do aplicativo. Outros parâmetros devem ser passados ao *dsmcc-ait* para a sinalização da aplicação. Os mais importantes são o código de controle da aplicação, a localização do fluxo onde a mesma esta sendo difundida, o protocolo utilizado na codificação (carrossel), e o nome e local da aplicação no carrossel de objetos. Após criado, o arquivo contendo a AIT é repetido ciclicamente em um *fifo*, através do *repeatts*, que é direcionado para uma das entradas do *multiplexador de emissão*.

6.2.4 Multiplexador de Emissão

O *multiplexador de emissão* é o componente responsável pela multiplexação de várias seqüências elementares, ou serviços MPEG-2, em um único fluxo de *bits*, que é encaminhado ao *meio de difusão*. Para a implementação desse componente, foi utilizado o *software iso13818ts*, pertencente ao pacote *multiplexer* [39]. As entradas para o multiplexador são, nesta implementação, os fluxos de vídeo e áudio encapsulados em pacotes de transporte do *codificador de mídia e multiplexador de serviço*, os fluxos dos dois *codificadores de dados* e o fluxo do *gerador de informações de serviços*.

Os PIDs dos fluxos de entrada do multiplexador são mantidos no fluxo de saída, opção informada ao aplicativo via linha de comando. A preferência por se manter os valores dos PIDs se deve ao fato de que os mesmos podem ser livremente configurados em todas as aplicações geradoras dos fluxos, com exceção do *ffmpeg*. Neste último caso, nenhum parâmetro é passado ao multiplexador referente à opção de não interferir no valor do PID do fluxo de mídia, fazendo com que o mesmo rearranje os valores de forma própria. Independente dos valores originais, o multiplexador mantém a PAT no PID 0x000, e atribui à primeira PMT o valor 0x100. Os fluxos elementares que fazem parte do serviço referenciado por essa PMT são numerados a partir do PID 0x101.

Como justificado na seção 6.2.3, em vez de novas tabelas serem geradas, as PMTs provenientes dos fluxos de saída do *ffmpeg* são alteradas via linha de comando no multiplexador de forma a apontar para os fluxos elementares adicionais, gerados pelo *codificador de dados* e pelo *gerador de informações de serviços*. Os parâmetros que devem ser passados ao multiplexador são o número

do serviço que deve ser alterado, o PID a ser incluído como referência na PMT, e o tipo de fluxo do mesmo.

Cada fluxo elementar referenciado em uma PMT pode conter descritores com informações adicionais sobre o mesmo, como mostrado na seção 3.2.3. O multiplexador permite a inserção desses descritores via linha de comando. Os mais importantes são os que devem estar presentes no anúncio da AIT e do fluxo do carrossel de objetos. De acordo com a especificação MHP, para a AIT, deve estar presente o *descritor de sinalização de aplicação*, indicando que tal fluxo contém informações sobre o ciclo de vida de determinada aplicação. No caso do carrossel de objetos, deve haver um *descritor de identificação de fluxo*, informando qual o *association tag* relacionado ao carrossel com o PID apontando pela PMT.

O fluxo de saída do multiplexador principal é um fluxo de transporte contendo uma PAT e uma PMT para cada serviço, que contêm referências para todos os demais fluxos multiplexados, sejam eles de dados, mídia ou tabelas como a AIT. A taxa de repetição das PAT e PMT é definida também via linha de comando ao multiplexador. A saída é direcionada a um *fifo*, que simula o *meio de difusão*, como será apresentado a seguir.

6.3 Meio de Difusão

Como o fluxo de saída do difusor é o mesmo esperado na entrada do *terminal de acesso*[2], o *meio de difusão* foi abstraído nesta implementação através do uso de um *fifo*. Dessa forma, o *difusor* escreve o fluxo de transporte MPEG-2 no *fifo*, que é lido pelo *terminal de acesso*. O *fifo* funciona como a conexão entre o *terminal de acesso* e o *difusor*. A figura 6.2 ilustra essa abstração.

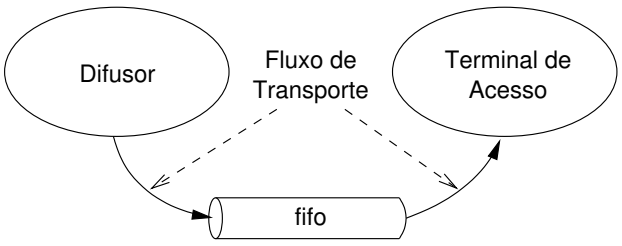

Figura 6.2: *Abstração do difusor através do uso de um fifo na implementação.*

[2]Desconsiderando o fato de que a sintonização e demodulação em sistemas reais são de responsabilidade do terminal de acesso

6.4 Terminal de Acesso

A função do *terminal de acesso* é selecionar determinado serviço presente no fluxo de entrada, proveniente do difusor, e processar todos os componentes de mídia e dados que integram o mesmo. De acordo com o modelo apresentado na seção 5.1.2, o *terminal de acesso* inclui componentes como um demultiplexador e decodificadores de mídia e dados. Em *terminais de acesso* reais, tais componentes são geralmente implementados via *hardware* especializado.

Neste trabalho, o demultiplexador e o decodificador de dados, apresentados no decorrer deste capítulo, foram desenvolvidos totalmente através da linguagem de programação C, seguindo o padrão POSIX, com exceção do uso de algumas bibliotecas específicas do GNU/Linux para a interface com o usuário. Apenas o decodificador de mídia foi implementado através de um aplicativo a parte, como será descrito ainda nesse capítulo.

O demultiplexador e o decodificador de dados fazem parte do mesmo código fonte desenvolvido. Isso se deve ao fato da integração necessária entre esses dois componentes, visto que os mesmos utilizam várias informações em comum, como será descrito ao longo desta seção. Dessa forma, desenvolver os dois como parte do mesmo aplicativo facilitou o intercâmbio de dados entre eles, resultando em uma implementação mais simples. O aplicativo integrando o demultiplexador e o decodificador de dados apresenta uma interface com o usuário, implementada através de uma *thread*, através da qual é possível verificar seu estado de processamento, assim como selecionar determinado serviço.

O modelo prevê a existência de um ambiente de execução de aplicação no terminal de acesso, onde a mesma pode utilizar funcionalidades através de uma API, assim como acessar os demais dados difundidos, tabelas de informação de serviço, e interagir localmente. Em terminais de acesso reais, como mostrado na seção 2.5, o *middleware* é responsável por fornecer essas funcionalidades à aplicação. A princípio, desejou-se utilizar o padrão MHP na implementação de uma espécie de *middleware* para cumprir esse papel, mas a carência de implementações de referências completas, livres, que não despendessem demasiado esforço de adaptação aos demais componentes fez com que fosse implementada uma abordagem simples e proprietária com relação a API, como será apresentado ao final desta seção.

6.4.1 Descrição do Mecanismo de Demultiplexação

Como apresentado no modelo da figura 5.5, o fluxo de entrada de dados do demultiplexador corresponde a um fluxo de transporte MPEG-2, produzido pelo difusor. Este é obtido através da leitura do *fifo* que simula o meio de difusão. O fluxo MPEG-2 é composto por um seqüência de pacotes de transporte, de 188 *bytes* cada. Dessa forma, o demultiplexador foi implementado como um *loop*, onde em cada ciclo são lidos 188 *bytes* correspondentes a um pacote de transporte, que é

armazenado em um *buffer* para posterior manipulação. Porém, o demultiplexador precisa sincronizar corretamente os *bytes* lidos com os *bytes* de um pacote de transporte, como mostrado no próximo tópico.

Sincronização

No início de cada *loop*, é verificado se o primeiro *byte*, dos 188 lidos, corresponde ao *byte de sincronismo*, ou seja, realmente indica o início de um pacote de transporte. Caso o *byte de sincronismo* seja confirmado, o restante do processamento do *loop* principal é realizado. Caso contrário, o demultiplexador entra no modo de sincronia, onde passa a ler a entrada até encontrar um *byte* com o mesmo valor do *byte de sincronismo*. Ao encontrá-lo, continua lendo a entrada, procurando 188 *bytes* após esse primeiro *byte* de sincronismo a existência de outro. Essa tarefa é repetida quatro vezes. Dessa forma, após verificar a presença do *byte* de sincronismo em 5 pacotes sucessivos, é descartada a possibilidade de leitura de valores aleatórios com o mesmo valor do *byte de sincronismo*, e os próximos 188 *bytes* realmente correspondem a um pacote de transporte válido. A figura 6.3 ilustra esse processo.

Tarefas do Demultiplexador

A partir desse ponto, a cada ciclo, o demultiplexador extrai um pacote do fluxo de transporte, que está disponível para processamento pelo restante da aplicação. As próximas tarefas do terminal de acesso são encontrar e decodificar a PAT, que apresenta os serviços presentes no fluxo de transporte; selecionar determinado serviço apresentado pela PAT; encontrar e decodificar a PMT correspondente ao serviço selecionado e processar o serviço, decodificando os fluxos elementares presentes no mesmo. A figura 5.6 ilustra os estados que o demultiplexador assume nesse processo.

Para encontrar a PAT, a PMT e os demais fluxos, o multiplexador precisa filtrar os pacotes de cada um com base no valor do PID. Para facilitar o acesso a essas informações, no início de cada *loop*, um descritor de pacote, baseado em uma estrutura em C, é preenchido com as informações referentes ao pacote, obtidas através da decodificação de seu cabeçalho. São extraídos o PID do pacote, a *indicação de início de carga*, o *contador de continuidade* e o *código de controle do campo de adaptação*.

O descritor também apresenta um ponteiro, que indica em qual *byte* do *buffer* onde está armazenado o pacote se dá o início da carga do mesmo. Quando um pacote não possui o campo de adaptação, esse *byte* é sempre o quinto do pacote. Quando há a presença do *campo de adaptação*, indicado pelo código do *controle do campo de adaptação* no cabeçalho do pacote, o início da carga é encontrado calculando-se o tamanho do cabeçalho normal mais o tamanho dessa extensão. Tal informação é obtida sempre no quinto byte do pacote, quando o *campo de adaptação* está presente. Nesse

caso, o demultiplexador aponta o ponteiro presente no descritor de pacote para esse *byte*, facilitando assim o uso da carga do pacote pelas demais funções presentes no demultiplexador.

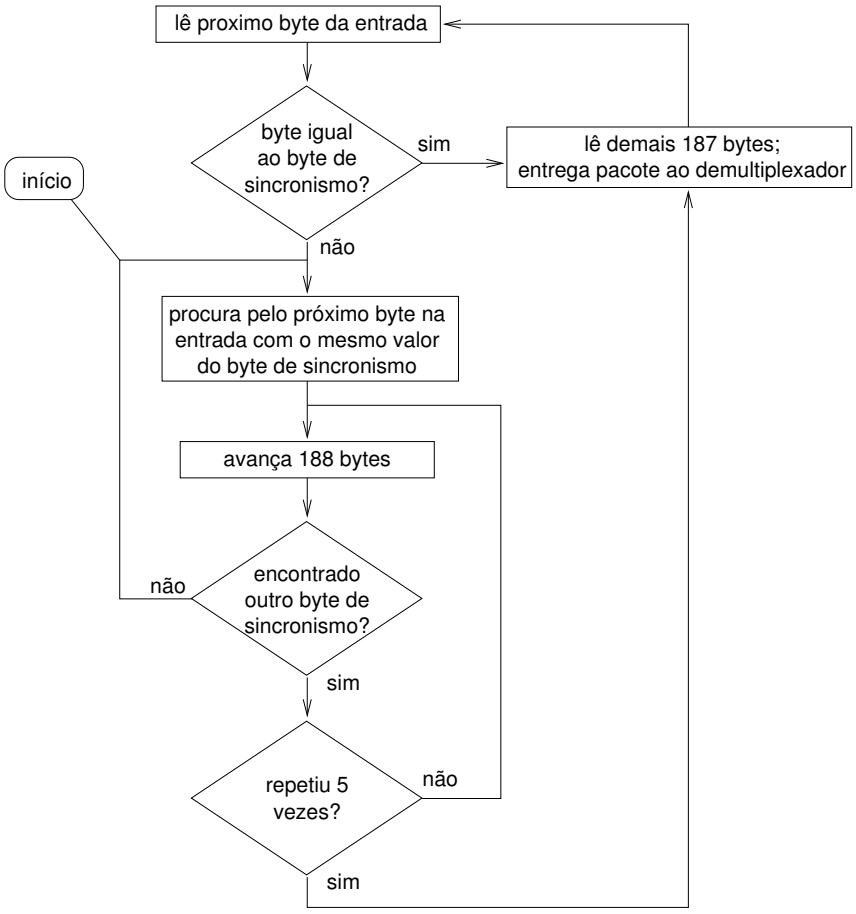

Figura 6.3: *Sincronização dos pacotes de transporte.*

Como as PSI são encapsuladas em seções, foi necessária a implementação de um filtro de seções responsável por extrair corretamente essas estruturas de um seqüência de pacotes de transporte. Também da mesma forma como ocorre com os pacotes de transporte, após cada seção completa ser armazenada em um *buffer*, é criado um descritor contendo as informações necessárias ou úteis aos outros mecanismos do demultiplexador.

Filtro de Seções

O primeiro passo na reconstrução de uma seção é identificar o pacote de transporte, com o PID desejado, contendo seu primeiro fragmento. Tal tarefa é realizada procurando-se por um pacote que apresente o bit *indicador de início de carga* ligado. Após identificado esse pacote, é lido o campo de ponteiro ao final do cabeçalho, que indica a quantos *bytes* após esse se inicia a carga. Determinado o início de carga do pacote, o primeiro fragmento da seção é extraído.

Através do primeiro fragmento, o demultiplexador determina o tamanho da seção, verificando quantos pacotes de transporte foram necessários para transportá-la. Nos oito primeiros *bytes* do primeiro fragmento se encontra o cabeçalho da seção, que contém o campo indicando seu tamanho. Este é lido, e verificado se o tamanho indicado é maior que a quantidade de *bytes* extraída do pacote de transporte. Se for menor ou igual, conclui-se que foi necessário apenas um pacote de transporte para encapsular a seção, ou seja, o fragmento retirado corresponde a seção inteira.

Caso a seção seja fragmentada em mais de um pacote de transporte, o demultiplexador procura pelos pacotes seguintes do mesmo fluxo, ou seja, de mesmo PID. A verificação de perda de pacotes é realizada através da análise do *contador de continuidade* de cada pacote. Caso o valor desse campo, em determinado pacote, seja diferente do esperado, o processo de decodificação de seção é reinicializado. Caso os demais pacotes de transporte sejam recebidos na ordem correta, os fragmentos da seção são extraídos de suas cargas, até que o número total de *bytes* obtidos corresponda ao tamanho da seção.

A seqüência de *bytes* da seção é armazenada em um *buffer*, onde os últimos quatro *bytes* correspondem ao valor do CRC de 32 bits. É realizado então o mesmo cálculo de CRC sobre os demais *bytes* da seção, e verificado se o resultado confere com o CRC presente. Caso os dois valores não sejam indênticos, existe a indicação de erro na transmissão ou decodificação, e o filtro de seções é reinicializado. Caso o CRC esteja correto, a seção é armazenada em outro *buffer* para posterior manipulação, e o demultiplexador é informado que a mesma está disponível para processamento. A figura 6.4 sintetiza o processo de reconstrução da seção.

Com a seção armazenada em um *buffer*, é criado um descritor a partir deste com o intuito de armazenar suas principais informações. Através da análise dos oito primeiros *bytes*, correspondentes ao cabeçalho, são obtidos, além do tamanho da seção, o valor do *identificador de tabelas*, o *identificador estendido de tabelas* e o *número de versão*.

Implementação das Tarefas do Demultiplexador

Como mostrado na figura 5.6, o multiplexador possuí quatro estados básicos: procura e decodificação da PAT; seleção de serviço; procura e decodificação da PMT referente ao serviço selecionado; e decodificação e processamento dos fluxos elementares do serviço.

Na primeira etapa, a de encontrar a PAT, o demultiplexador procura por pacotes com o PID igual 0, valor de identificação do fluxo que contém essa tabela segundo o padrão MPEG-2. Sobre esses pacotes é aplicado o filtro de seções, descrito anteriormente, que reconstrói a seção preenchendo seu descritor com as informações pertinentes.

O próximo passo é decodificar e analisar a PAT, e extrair dessa as informações sobre os serviços presentes no fluxo de transporte. Como mostrado na seção 3.2.3, existem quatro *bytes* para cada

serviço referenciado na PAT. Esses *bytes* são analisados, e deles extraídos o número de programa e o PID da PMT do serviço. Essas informações são registradas em uma lista encadeada, apontada pelo descritor da PAT, que é criado nesse processo.

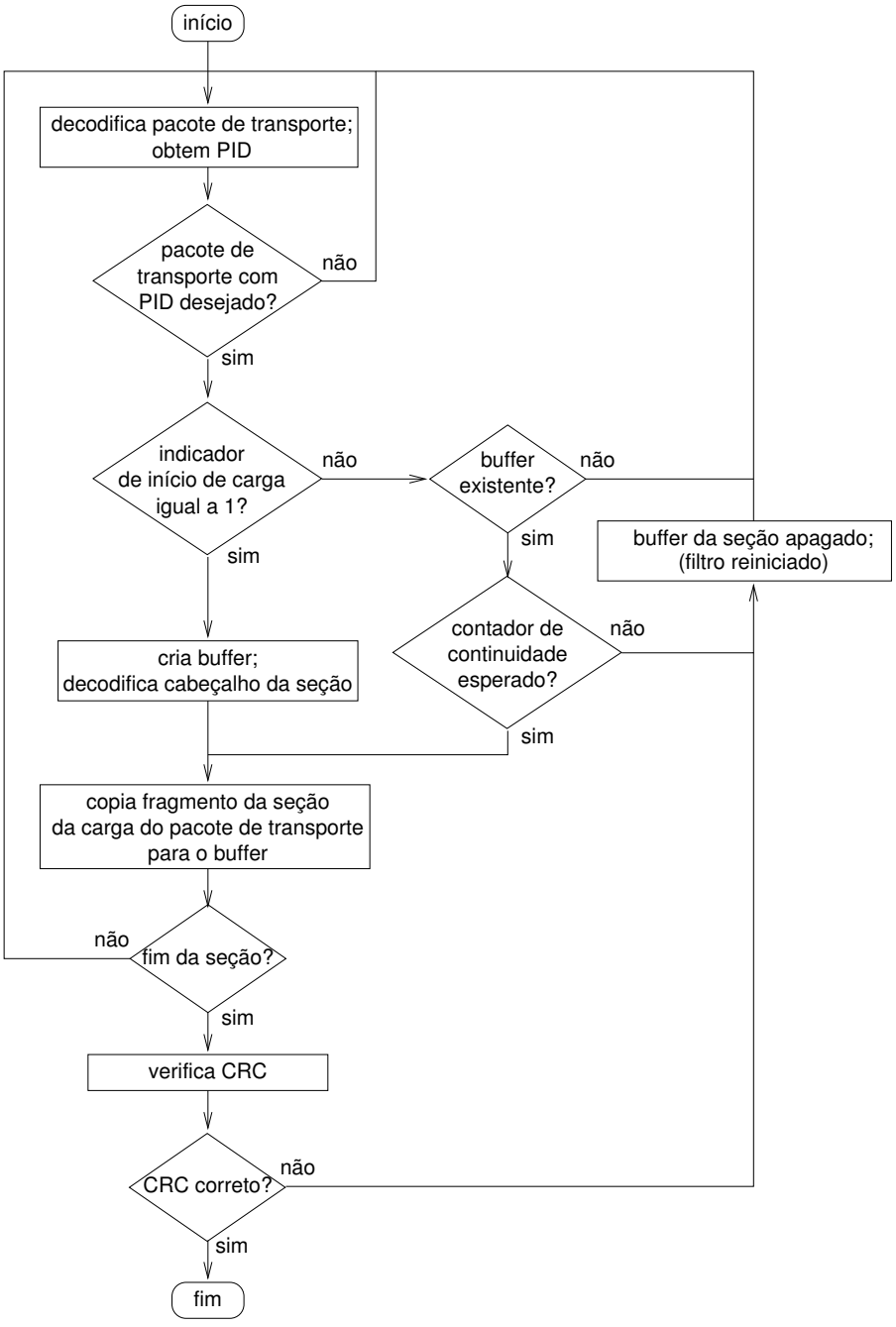

Figura 6.4: *Processo de reconstrução de uma seção.*

A partir desse ponto, o demultiplexador possui um descritor da PAT com um ponteiro para uma lista encadeada que contém o número e o PID de cada serviço. Neste momento, o terminal de acesso entra no estado de seleção de serviço. A *thread* responsável pela interface gráfica apresenta

os serviços disponíveis a partir da consulta à lista, apontada pelo descritor da PAT, e possibilita ao usuário do terminal de acesso selecionar o serviço.

Após selecionado determinado serviço, é repassado ao demultiplexador um ponteiro para a entrada na lista encadeada que contém o número do programa e o PID do serviço a ser decodificado. Em posse do PID da PMT do serviço, o demultiplexador entra no estado de procura e decodificação da PMT. O processo é semelhante ao descrito anteriormente. O demultiplexador aplica o filtro de seções sobre os pacotes com o PID da PMT desejada, e a partir da seção obtida é criado um descritor com as informações a seu respeito.

A partir da decodificação da PMT, é criada outra lista encadeada, desta vez listando os fluxos elementares que fazem parte do serviço. É registrado em cada entrada dessa lista o *tipo de fluxo*, seu *PID* e seus descritores. Dessa forma, o demultiplexador obtém a relação de todos os fluxos elementares pertencentes ao serviço selecionado.

Algumas restrições fazem parte da implementação até esse ponto. A primeira diz respeito à atualização das PSIs. O demultiplexador implementado não é capaz de reconhecer alterações na PAT e PMT após ter decodificado as mesmas, nem de possibilitar ao usuário a seleção de outros serviços após a primeira escolha ter sido realizada. Outra restrição é que, nesta implementação, tanto a PAT como a PMT podem ter no máximo 1024 *bytes*, tamanho máximo de uma seção transportando uma PSI. Essas, tabelas quando maiores, são fragmentadas em mais seções, e tal capacidade de decodificação pode ser implementada através da análise dos campos *número da seção* e *número da última seção*, como mostrado na seção 3.2.3.

Depois de decodificar a PAT, e obter a lista dos fluxos elementares, o demultiplexador entra no estado de decodificação e processamento dos fluxos elementares do serviço. A partir desse ponto, o mesmo deve acionar os decodificadores específicos de cada fluxo. Neste trabalho foram implementados decodificadores de mídia, áudio e vídeo, através de *softwares* já existentes, e totalmente desenvolvidos decodificadores de carrosséis de objetos e de dados transportados em seções privadas.

6.4.2 Decodificação de Mídia

Para a decodificação e processamento dos fluxos elementares, o demultiplexador consulta a lista encadeada contendo a relação dos fluxos pertencentes ao serviço. Também verifica quais deles, através da informação do campo *tipo de fluxo*, correspondem a fluxos de áudio ou vídeo MPEG-1 ou MPEG-2, como mostrado na seção 3.2.3. Após isso, o demultiplexador registra o PID do primeiro fluxo de vídeo e do primeiro de áudio encontrado na lista encadeada.

Com o demultiplexador conhecendo os PIDs dos fluxos de mídia, o mesmo passa a rotear os pacotes para dois *fifos*, um para o vídeo e outro para o áudio. Cada *fifo* é lido por uma instância do software *ffplay*, que é responsável pela decodificação e apresentação da mídia. O *ffplay* é distribuído

com o *software ffmpeg* e foi utilizado na implementação por ser o decodificador mais simples analisado. Outros decodificadores ou *players* já bem estabelecidos na comunidade GNU/Linux, como o VLC [41], utilizam as mesmas bibliotecas do *ffplay* para decodificação, e possuem funcionalidades adicionais que não eram de interesse dessa implementação.

A abordagem mais elegante no ponto de vista de demultiplexação a ser implementada no caso da decodificação de mídia seria a de retirar as PES dos pacotes de transporte e enviá-las aos decodificadores. Porém, verificou-se que o *ffplay* não só aceita PES, como também é capaz de processar a camada mais inferior do MPEG-2, a dos pacotes de transporte. Dessa forma, de modo a simplificar a implementação do roteador de mídia, os pacotes de transporte são encaminhados sem processamento aos decodificadores.

6.4.3 Descrição dos Mecanismos de Decodificação de Dados

A implementação prevê que uma aplicação, em execução no terminal, faça requisições de uso de algum conjunto de dados difundidos no fluxo de transporte. Tal comportamento do terminal de acesso foi materializado através de uma biblioteca simulando a implementação de uma API, como será apresentado ao final do capítulo. A implementação permite também, como prevê o modelo, que essas aplicações sejam difundidas, decodificadas e executadas. Para tanto, é necessário que a mesma seja sinalizada. Como mostrado na seção 6.2.3, isso é possível através da difusão da AIT que, quando o terminal de acesso se encontra no estado de processamento dos fluxos elementares, passa a ser alvo de decodificação.

Decodificação da AIT

Quando determinado serviço é selecionado, o demultiplexador verifica se em algum fluxo é transportada uma AIT. Ao contrário dos fluxos de mídia, isso não é verificado apenas através do campo *tipo de fluxo*, e sim através dos descritores do mesmo. O campo *tipo de fluxo* da AIT indica que o mesmo é do tipo seção privada. Em fluxos desse tipo o demultiplexador deve procurar pelo descritor de sinalização de aplicação. Caso encontrado, o demultiplexador registra o PID do fluxo e passa a procurar seus pacotes.

Sobre os pacotes carregando a AIT, é aplicado o filtro de seção, que extrai a seção correspondente e a armazena em um *buffer*, da mesma forma como ocorre com as PSIs. A partir da decodificação dessa seção é criado um descritor com as informações presentes na AIT. As informações mais importantes para o decodificador são o código de controle da aplicação, o fluxo onde ela está sendo difundida, indicado indiretamente por um *component tag*, e o caminho da aplicação dentro da estrutura de diretórios. A informação sobre o protocolo utilizado na codificação, também extraído da decodifi-

cação da AIT, é usado pelo decodificador para verificar se a aplicação foi codificada em um carrossel de objetos, único protocolo de decodificação de aplicações suportado por esta implementação[3].

No protótipo foi implementado apenas o tratamento do código de controle *auto-iniciar*. Esse código indica que o terminal de acesso deve imediatamente localizar e executar a aplicação. Dessa forma, é acionado o decodificador de carrossel de objetos, e no retorno do mesmo, o terminal de acesso é informado que a aplicação já está disponível para execução.

Decodificação do Carrossel de Objetos

O primeiro passo na tentativa de se decodificar um carrossel de objetos é o de encontrar a mensagem de mais alto nível de controle, a DSI, como mostrado na seção 3.3.2. Ela é referenciada indiretamente pelo campo *component tag* da AIT. Esse valor é repassado ao demultiplexador, que se encarrega de procurar nos fluxos do serviço qual deles contém carrosséis de objetos. Para isso, o demultiplexador varre sua lista de descrição dos fluxos a procura de entradas que contenham o descritor de identificação de carrossel. Quando encontrada determinada entrada desse tipo, é verificado se existe nessa o descritor de identificação do fluxo, que carrega o *component tag* associado ao mesmo. Se o valor do *component tag* é o mesmo indicado pela AIT, o fluxo que carrega a DSI do carrossel de objetos foi localizado, e seu PID é registrado pelo demultiplexador, para iniciar a decodificação.

A partir do PID do fluxo do carrossel de objetos, o demultiplexador direciona os pacotes de transporte correspondentes para o filtro de seções. Todo o conteúdo extraído da seção a partir dos pacotes de transporte é copiado para um *buffer*, seu cabeçalho é decodificado, e as principais informações armazenadas em um descritor. Ambos são repassados ao decodificador de carrossel de objetos.

O decodificador utiliza informações do descritor da seção para verificar se ela contém a mensagem de controle desejada. Através do identificador de tabelas, é verificada se ela é uma mensagem de controle, e caso seja, é verificado o campo estendido de tabela a fim de determinar se a mensagem é uma DSI ou uma DII. Para o primeiro caso, o decodificador reconhece a seção como válida e tenta decodificar a DSI. Caso contrário, continua a aplicar o filtro de seções sobre os demais pacotes de transporte do mesmo PID.

A DSI é extraída da seção e armazenada em um *buffer*. Ela contém, após seus 36 *bytes* de cabeçalho, o *Service Gateway Info*, que possui a *IOR* que referencia o objeto do tipo diretório *Service Gateway*, o diretório raiz do carrossel de objetos. No protótipo foi implementado a localização e recuperação de objetos em um único fluxo de transporte, ou seja, apenas o *Profile Body* foi implementado. São recuperados do *Service Gateway Info* as seguintes informações:

- identificador do carrossel que transporta o *Service Gateway*;

[3] O padrão MHP, através da AIT, permite também a sinalização de aplicações encapsuladas em datagramas IP

- identificador do módulo;
- *objectKey*;
- *association tag*;
- *transactionId*.

Essas informações são suficientes para encontrar qualquer objeto transportado em um carrossel. O identificador do módulo e o *objectKey* são utilizados para localizar o módulo e o objeto dentro deste. Porém, como todo módulo é referenciado e gerenciado por uma DII, como mostrado na seção 3.3.2, através dessa é obtida sua localização. Para se encontrar a DII, são utilizados o *association tag* e o *transactionId*. O processo de recuperação de um objeto é igual, a partir desse ponto, tanto para o *Service Gateway* como para qualquer outro objeto do tipo diretório ou arquivo. A forma como o procedimento, apresentado na seção 3.3.2, foi implementado, é apresentado a seguir.

Localização de um objeto em um carrossel

O primeiro passo para se encontrar um objeto é obter a DII que o descreve. O PID é resolvido através do *association tag* obtido da IOR que referencia o objeto. Esse campo possui a mesma funcionalidade do *component tag*. O demultiplexador varre a lista encadeada dos fluxos procurando por algum descritor contendo um *component tag* igual. Quando encontrado, o demultiplexador tem em mãos o valor do PID do fluxo onde se encontra a DII.

O procedimento de procura e decodificação da DII é semelhante ao da DSI. A seção que transporta ela é filtrada, e é verificado se a mesma carrega uma mensagem de controle através da consulta ao campo *identificador de tabela*. A confirmação se a mensagem é uma DII se dá através da análise do campo *indetificador estendido de tabela*. Passados por esses testes, a DII é extraída da tabela e armazenada em um *buffer*.

A partir do *buffer* contendo a DII é criado um descritor que contém o *transactionId* dessa mensagem. Neste ponto, o decodificador sabe que a mensagem que está sendo processada é uma DII, mas não se é a que referencia o objeto *Service Gateway*. Tal confirmação se dá pela comparação do *transactionId* indicado pelo *Tap* do *Service Gateway Info* com o da DII em processamento. Caso sejam iguais, o decodificador continua a processar a DII. Caso contrário, descarta a mesma e repete o processo de busca.

Como a DII contém informações sobre vários módulos, a partir dela é criada um lista encadeada, com uma entrada para cada módulo. Em cada entrada são armazenados o *identificador do módulo*, o tamanho do mesmo, e sua *association tag*.

Com a DII indicando a localização do objeto, o decodificador precisa descobrir em qual PID está o módulo que o transporta. Para tanto, é percorrida a lista encadeada construída a partir da DII, a procura de alguma entrada que contenha o identificador do módulo igual ao referenciado pelo IOR do objeto. Quando encontrado, o decodificador descobre qual a *association tag* referente ao módulo, e requisita ao demultiplexador o valor do PID do fluxo referente ao *association tag*. O demultiplexador varre sua lista de fluxos elementares procurando por um que possua em seus descritores o mesmo valor no *component tag*, determinando o PID que carrega as DDBs. A partir desse ponto, o decodificador começa a filtrar os pacotes de transporte do PID obtido.

Como apresentado na seção 3.3.1, os módulos são fragmentados em DDBs. As seções carregando as DDBs são filtradas, através da verificação de seu *identificador de tabelas*. Cada DDB é extraída de uma seção, armazenada em um *buffer*, do qual é extraído o *identificador do módulo* de seu cabeçalho. O mesmo é comparado ao valor declarado pelo IOR do objeto. Caso sejam iguais, o DDB é valido. Com base no tamanho do módulo, obtido pela entrada na lista encadeada da descrição da DII referente ao módulo em questão, o decodificador o reconstrói, concatenando as cargas das DDBs filtradas, verificando a quantidade de *bytes* obtidos com o tamanho do módulo previsto. Ao final da extração das cargas das DDBs, o módulo é armazenado em um *buffer*.

O partir do *buffer* do módulo, o decodificador procura o objeto dentro do mesmo. Isso é realizado através da pesquisa do *objectKey*, obtido a partir da IOR do objeto, como descrito anteriormente. Como mostrado na seção 3.3.2, dentro de um módulo, os objetos são armazenados seqüencialmente. O decodificador inicia a leitura pelo primeiro *byte* do módulo, que deve corresponder ao primeiro *byte* do primeiro objeto armazenado no mesmo. É verificado se os quatro primeiros *bytes* correspondem ao valor do campo *magic* de um objeto, como mostrado no anexo B, e em caso afirmativo, a partir do nono *byte* são obtidos os quatro *bytes* que informam seu tamanho. Outros quatro bytes a partir do décimo quarto contem o *objectKey* do objeto. Caso esse valor seja igual ao do *objectKey* procurado, o objeto foi encontrado, e o decodificador pode extrair o mesmo a partir desse ponto, que é referenciado através de um ponteiro, no *buffer*. Caso contrário, o decodificador pula para o primeiro *byte* do próximo objeto dentro do módulo, calculado a partir do tamanho do objeto anterior, e repete o processo. Tal distribuição de objetos em um módulo é ilustrada na figura 3.13

Com um ponteiro apontando para o início do objeto desejado no *buffer* do módulo, o decodificador obtém o tipo do objeto a partir de quatro *bytes* localizados no cabeçalho do mesmo. A codificação de dois dos três tipos básicos de objetos foi implementada: *diretório* e *arquivo*. Para cada um desses tipos, um flag é ligado indicando ao decodificador se o mesmo deve proceder com espécie de decodificação adequada. O decodificador cria também uma lista encadeada de objetos a serem processados, a princípio vazia, que vai ser manipulada como explicado a seguir.

Decodificação de um objeto do tipo diretório

Um objeto do tipo diretório é codificado de acordo com a sintaxe apresentada no anexo B. Este transporta um lista de ligações, que associa um nome de objeto a uma referência, uma IOR. O decodificador inicia o processamento do objeto varrendo o *buffer* do módulo onde este se encontra, através de um ponteiro. Como alguns campos das mensagens BIOP não possuem tamanho fixo, e seu tamanho é declarado geralmente por outro campo, o decodificador deve calcular várias vezes o número de *bytes* que o ponteiro deve avançar sobre os campos de tamanho variável.

O campo de maior interesse para o decodificador é o *contador de ligações*, que informa o número de objetos referenciados. O decodificador cria uma lista encadeada de nomes de objetos, e entra em um *loop*, com o número de ciclos igual ao número de ligações. Em cada ciclo o nome de um objeto é decodificado, e as informações pertinentes de sua IOR capturadas, preenchendo uma entrada da lista encadeada. Ao final do *loop*, o decodificador possui a listagem de todos os objetos e suas referências, obtidos do objeto do tipo diretório.

Decodificação de um objeto do tipo arquivo

A decodificação de um objeto do tipo arquivo é mais simples do que o do tipo diretório. Como no caso anterior, um ponteiro percorre o *buffer* do módulo até encontrar o início da carga da mensagem BIOP. A partir desse ponto, copia todo o conteúdo da carga, informado pelo campo de tamanho, para outro *buffer*, que será utilizado posteriormente para a reconstrução do arquivo.

Gerenciamento da lista de objetos

Após a decodificação do primeiro objeto do tipo diretório, o *Service Gateway*, que é automática após o processamento da AIT, a lista de ligações obtida do mesmo é copiada para lista encadeada de objetos do decodificador. A partir dessa lista de objetos, que corresponde aos objetos no diretório raiz do sistema de arquivos do carrossel de objetos, o decodificador tomará futuras decisões de localização e decodificação dos demais objetos.

Enquanto a lista de objetos não for vazia, o decodificador tenta processar seu primeiro elemento. A localização do objeto se dá da mesma forma apresentada nos tópicos anteriores, através de parâmetros como o *objectKey*, o identificador de módulo, a *association tag* e o *transaction id* da DII. Essas informações são copiadas da lista de ligação de cada objeto diretório, após sua decodificação, para a lista de objetos.

Caso o primeiro objeto da lista for do tipo diretório, após localizado e decodificado, a entrada correspondente ao mesmo é apagada da lista do decodificador. Porém a lista de ligações obtida do

objeto diretório é adicionada. Dessa forma, sempre que um objeto do tipo diretório contendo mais uma ligação, é decodificado, a lista de objetos do decodificador é incrementada. Como o objeto processado foi retirado da lista, o próximo objeto passa a ser o primeiro, ou seja, será o próximo a ser decodificado.

Caso seja um objeto do tipo arquivo o primeiro da lista, o mesmo é localizado e decodificado como descrito anteriormente. Um arquivo no diretório que o objeto pertence é criado com o nome presente na lista de objetos referente ao mesmo, e o conteúdo do *buffer* gerado após sua decodificação é copiado para o arquivo. Após esse processo, a entrada correspondente ao objeto do tipo arquivo é retirada da lista de objetos.

O processo apresentando nos parágrafos anteriores repete-se até que a lista de objetos fique novamente vazia, ou seja, todos os objetos de um carrossel de objetos foram reconstruídos. O diagrama da figura 6.5 ilustra esse processo. Após essa etapa, o decodificador de carrossel de objetos informa ao demultiplexador que o mesmo já está disponível, ou seja, alguma aplicação sinalizada já pode ser executada.

Decodificação de Seções Privadas

Atualmente não são previstas sinalizações de aplicações transportadas em seções privadas pelo padrão MHP, ou seja, a AIT não possui esse tipo de funcionalidade. Isso de deve ao fato de que seções privadas são mais utilizadas para fornecer dados a uma aplicação do que propriamente transportá-la. Tal fato se comprova na existência de APIs, como definida no padrão MHP, para o acesso de dados transportados nesses tipos de seções.

Neste protótipo, os dados carregados em uma seção privada são processados e redirecionados apenas quando alguma aplicação faz a requisição dos mesmos. Como não foi possível implementar ou integrar alguma implementação do MHP, foi criada uma API simples para simular tal comportamento. Através dela, como será apresentado em mais detalhes ao final do capítulo, alguma aplicação no terminal de acesso faz a requisição de acesso a dados transportados em seções privadas. O método de localização de fluxos contendo esses dados foi o mesmo utilizado pelos carrosséis de objetos: via *component tag* ou *association tag*.

Após um aplicativo fazer uma requisição de acesso a dados, um *flag* do demultiplexador é ligado, indicando que nos próximos ciclos o mesmo deve localizar o fluxo através do *component tag* informado pela aplicação. O demultiplexador varre a lista de fluxos do serviço em processamento, buscando um descritor de fluxo contendo uma *component tag* igual. Ao encontrar o PID do fluxo correspondente, inicia o roteamento de pacotes para um filtro de seção, que redireciona o conteúdo das seções para um *fifo*, que será utilizado pelo aplicativo para ler os dados recebidos. O *fifo* no qual o aplicativo receberá os dados é repassado via requisição da aplicação através da API do terminal de acesso.

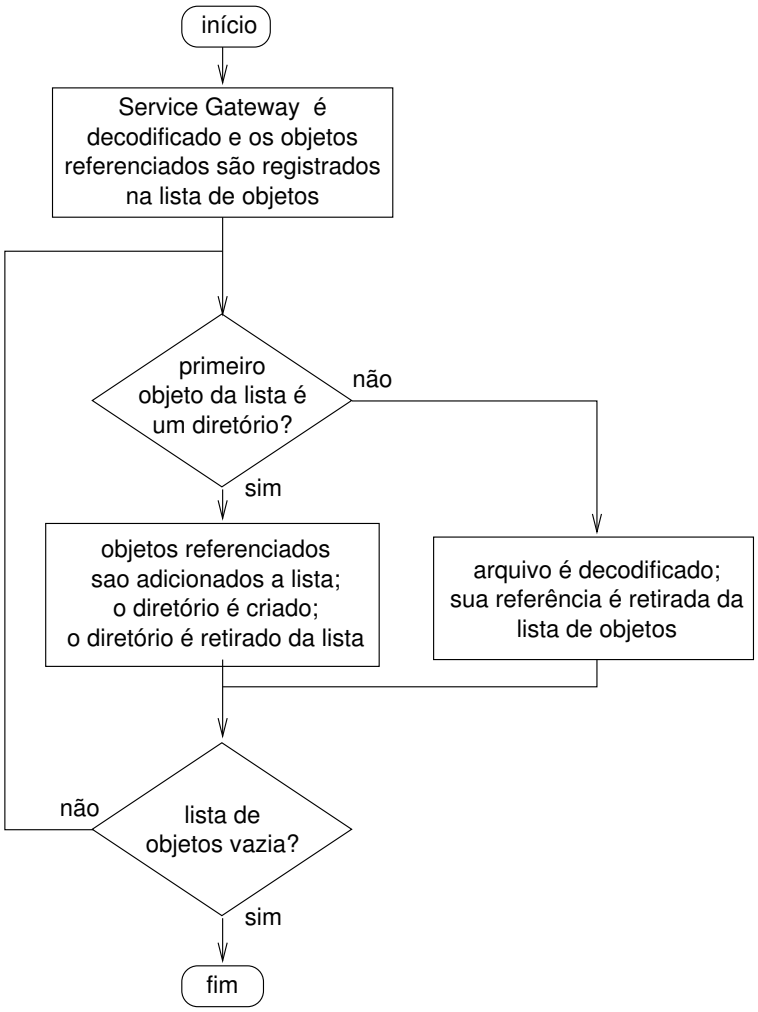

Figura 6.5: *Processamento de diretórios e arquivos em um carrossel de objetos: inserção e remoção de entradas na lista de objetos.*

6.4.4 API, Gerenciamento do Ciclo de Vida da Aplicação e Interação Local

Após todo o carrossel ser decodificado, e a estrutura de arquivos reconstruída, o demultiplexador é informado que a aplicação sinalizada pela AIT está disponível. Este se encarrega de executar uma chamada sistema que carrega a aplicação alvo. A essa é fornecida uma pequena API, implementada através de uma biblioteca em C, fornecendo uma função de acesso aos demais dados difundidos em seções privadas.

O componente de interação local, apresentado na seção 5.1.2, não foi implementado via API. A aplicação utiliza, para o mesmo, a entrada padrão de um terminal do Linux como entrada local e a saída padrão como saída local. Também não foi implementado nenhum processador gráfico capaz de sobrepor alguma interface visual sobre a vídeo apresentado pelo decodificador de mídia. A saída gráfica é a mesma da saída local, ou seja, a saída padrão do terminal em modo texto.

A pequena parte da API implementada é referente à interface de comunicação entre a aplicação e o demultiplexador. A aplicação tem a sua disposição uma função pela qual solicita o acesso aos dados difundidos em seções privadas, passando como argumento o valor do *component tag* do fluxo desejado. Em caso de sucesso, a função retorna o *fifo* no qual a aplicação deve se conectar para começar a receber os dados. Tal API visa simular um comportamento semelhante do processo que ocorre em chamadas de acesso a dados difundidos em APIs como a do padrão MHP.

Não foi necessário implementar uma API de acesso a objetos em um carrossel. A solução simples adotada é a de decodificar todo o carrossel, reconstruindo a estrutura de arquivos originalmente codificada antes de executar a aplicação. Dessa forma, a mesma pode acessar os demais arquivos presentes no mesmo carrossel como arquivos locais.

6.5 Experimentos com o Protótipo

A implementação foi desenvolvida de modo a ser executada em um PC sobre o sistema operacional GNU/Linux. Como mostrado na figura 6.2, o difusor alimenta um *fifo* com um fluxo de transporte contendo mídia e dados multiplexados. O terminal de acesso consome esse fluxo do *fifo*, processando-o de maneira adequada.

Uma pequena aplicação foi desenvolvida e compilada em C, utilizando a biblioteca da API apresentada na seção 6.4.4. Através dela, a aplicação requisita o acesso a outros dados difundidos em seções privadas. Essa aplicação foi codificada em um carrossel de objetos através do *dsmcc-oc*, como descrito na seção 6.2.1. Uma AIT estática foi criada através do *dsmcc-ait*, sinalizando a aplicação como auto-executável.

Duas instâncias do *reppeats* alimentam dois *fifos*, um com o carrossel de objetos e o outro com a AIT. O codificador de seções privadas é inicializado em modo teste, onde codifica a informação sobra a hora atual no ambiente em que está sendo executado, e direciona o fluxo codificado para um *fifo*. A mídia é proveniente da *webcam* da abstração do *provedor de conteúdo*. Esta é configurada de modo a fornecer vídeo via uma interface *video4linux* para o *ffmpeg*, como apresentado na seção 6.2.2, no formato NTSC a 24 quadros por segundo. O *ffmpeg* desfaz o entrelaçamento da seqüência de vídeo, e altera sua resolução para 352x240, comprimindo a uma taxa de 1200 *kbps* no formato MPEG-2 Vídeo. A saída do *ffmpeg* é direcionada a outro *fifo*, entrada do *multiplexador de emissão*.

O *multiplexador de emissão* é inicializado, alimentado pelos quatro *fifos* de entrada, que são as saídas dos componentes citados no parágrafo anterior. Em sua inicialização, são passados parâmetros para a modificação da PMT gerada pelo *ffmpeg*, de forma a inserir na mesma as referências aos fluxos elementares do carrossel de objetos, do codificador de seções privadas e do fluxo contendo a AIT.

Também são fornecidos, através de parâmetros ao multiplexador, os descritores a serem inseridos na PMT para cada um desses fluxos. Após sua inicialização, o multiplexador produz um fluxo de

transporte contento um serviço, que é formado por um fluxo de vídeo, um com o carrossel de objetos, um com dados codificados em seções privadas e outro com a AIT.

O terminal de acesso inicia a demultiplexação, apresentando o serviço difundido como único a ser decodificado. Quando confirmada a decodificações deste, através da interface disponível ao usuário, o *ffplay* é executado, passando a exibir o vídeo do serviço. O carrossel de objetos é automaticamente decodificado, e a aplicação difundida iniciada. A mesma passa a exibir na tela as estampilhas de tempo difundidas e acessadas pela mesma via seções privadas.

6.6 Considerações sobre as Experiências Realizadas

A execução do protótipo buscou simular o comportamento de um sistema de televisão digital onde alguma aplicação faz uso de dados difundidos no mesmo meio de difusão utilizado para os demais programas televisivos, seguindo o padrão *MPEG-2 Sistemas*. O comportamento obtido foi o mesmo esperado em um sistema de TVD, como por exemplo o DVB, onde através do padrão MHP, aplicações difundidas em carrosséis de objetos são decodificadas e executadas no receptor, interagindo com o telespectador e acessando demais dados difundidos.

Como apresentado na seção 6.4.4, a API que implementa o acesso a dados possui apenas uma função de suporte a esse fim. Especificações como o MHP fazem uso de várias bibliotecas, de modo a fornecer suporte ao acesso de dados difundidos à aplicação através de seções privadas, carrosséis de objetos, MPE e outros métodos, e também através de um possível canal de retorno. O MHP encontra-se em constante desenvolvimento, onde novas versões da especificação buscam adicionar recursos demandados por novas aplicações ou cenários desejados. Uma natural extensão da implementação aqui apresentada seria acrescentar um *middleware* MHP ao terminal de acesso. Isso pode ser feito através da adaptação de emuladores como o XleTView [42] ou OpenMHP [43], e do desenvolvimento de um *gerenciador de aplicações*, integrando-os aos componentes desenvolvidos: o demultiplexador e decodificador de dados.

Na execução do protótipo, foi transmitida uma pequena aplicação desenvolvida e compilada em C, diferentemente do que ocorre em sistemas de televisão digital, onde o padrão de *facto* para aplicações é a linguagem Java. Porém, os mecanismos de difusão de dados utilizados na implementação são os mesmos padrões de *facto* nesses sistemas: codificação de aplicações em carrossel de objetos e difusão de dados ordinários em seções privadas. O mecanismo de transporte, o MPEG-2, também segue o padrão adotado mundialmente. Dessa forma, os mecanismos de codificação, multiplexação, demultiplexação e decodificação implementados seguem os mesmos princípios de sistemas já implementados ou em desenvolvimento.

Algumas simplificações foram realizadas no protótipo, principalmente no terminal de acesso, de forma a tornar possível sua implementação. Outras funcionalidades, como a codificação de dados

através dos outros métodos apresentados no capítulo 4, podem ser acrescentadas ao mesmo. Exemplos são a codificação de dados em MPE e PES.

No caso da codificação de dados em PES, um esforço adicional de forma a aproveitar as capacidades de sincronismo de dados fornecidas por esse método deve ser realizado. Para a implementação de sistemas com esse tipo de codificação, o difusor deve ser capaz de inserir estampilhas de sincronismo nos pacotes PES, que deve ser baseado no relógio base do serviço que os dados fazem parte, o PCR. Dessa forma, o sistema de codificação deve ser capaz de acessar e manipular esse relógio, como introduzido na seção 5.1.1.

Para a decodificação de PES com informações de sincronismo, o terminal de acesso deve ser capaz de reconstruir o relógio base do serviço e decodificar e apresentar o conteúdo das PES quando a estampilha de tempo carregadas por essas atingir o valor do PCR. A leitura, controle e manipulação das estampilhas de tempo, PCR, DTS e PTS não foram implementadas nesse trabalho, e permanecem como sugestão para trabalhos futuros, como o desenvolvido por [5].

Também não foi adotada nenhuma prática de *datacasting* do tipo oportunística. Tal espécie de *datacasting* não foi implementado devido a alta complexidade exigida para a adaptação dos componentes, utilizados na implementação do protótipo, de modo a fornecer essa funcionalidade.

6.7 Considerações Finais

Neste trabalho, um sistema fim a fim com os mesmos princípios de um sistema real de difusão de dados sobre padrões abertos foi concretizado. Todos os componentes do sistema buscam seguir as especificações MPEG-2 e DSM-CC, padrões de *facto* em ambientes de televisão digital. A exceção se dá com relação à API fornecida à aplicação, que busca implementar uma função similar as previstas pela maioria dos padrões de *middleware* abertos.

Por ser apenas um protótipo a ser executado em um PC, carece de algumas funcionalidades. Porém, os conceitos e mecanismos aqui implementados, justamente por seguirem os padrões utilizados pelos sistemas de TVD abertos, podem servir de referência para futuros trabalhos na área.

Capítulo 7

Conclusão

7.1 Revisão das Motivações e Objetivos

A *Televisão Digital* é um assunto recente em termos da definição das tecnologias e especificações de *hardware* e *software* a serem adotadas. Alguns dos padrões abertos de sistemas de TVD foram ou estão sendo definidos nesta década. Nesse contexto, as especificações relacionadas à difusão de dados também são recentes, ou ainda se encontram em fase de desenvolvimento [35, 19, 3, 28].

Nessa área específica, várias pesquisas acadêmicas vêm surgindo ultimamente. São propostos novos serviços para a Televisão Digital, inclusive serviços interativos, a fim de enriquecer a programação televisiva convencional. Também são sugeridos outros serviços, os quais fazem uso do canal de difusão da TVD para a entrega de dados à aplicações independentes da programação televisiva. Ou seja, a televisão passa a servir de meio para outras aplicações além do simples entretenimento do telespectador. Segundo [11], o uso da difusão de TVD para serviços Empresa-Empresa deverá ser estratégico nos próximos anos.

Considerando todo esse contexto, a questão que este trabalho se propôs a abordar é:

- desenvolver e implementar um modelo fim a fim de *datacasting* – baseado em especificações abertas e trabalhos encontrados na Literatura – que suporte esses novos serviços e aplicações.

O modelo proposto é mais especificamente direcionado à difusão de aplicativos e demais dados correlacionados. Questões como modelos de negócios, gerenciamento da banda de difusão, segurança, prevenção de erros e compressão de dados em serviços de *datacasting* não foram o foco deste trabalho. O mesmo se concentrou nas questões relativas ao fluxo de dados do sistema fim a fim de *datacasting*, como codificação, multiplexação, demultiplexação e decodificação.

7.2 Visão Geral do Trabalho

Neste trabalho foram levantadas e estudadas as especificações de *facto* em termos de difusão de dados previstos pelos sistemas abertos de TVD. A partir dessas especificações e de outros trabalhos relacionados e encontrados na Literatura, foi sugerido um modelo simplificado de *datacasting* fim a fim. O mesmo incluiu o mecanismo de codificação de dados em carrosséis de objetos e seções privadas para a difusão de dados sobre os protocolos da televisão digital.

Além dos mecanismos de codificação e decodificação, foi proposta a existência de um ambiente de execução de aplicação no modelo. Através deste, aplicações difundidas, decodificadas, e executadas no receptor possuem acesso aos recursos do terminal de acesso comuns em um sistema de TVD, via uma API, assim como aos demais dados difundidos.

Uma implementação a partir do modelo proposto foi concretizada, seguindo os padrões abertos já citados. Essa implementação foi realizada em um PC sobre o sistema operacional GNU/Linux. Alguns aplicativos já existentes para esse sistema operacional foram integrados a implementação, como os usados nas etapas de codificação de dados em carrosséis de objetos, codificação e decodificação de vídeo, e multiplexação.

Alguns componentes do sistema foram desenvolvidos totalmente. É o caso de parte do terminal de acesso, incluindo o demultiplexador e os decodificadores de dados (carrosséis e seções privadas). Programados em C, os mesmos fornecem as funcionalidades básicas esperadas, com algumas restrições, de seus similares implementados na maioria via *hardware* em terminais de acesso reais.

O ambiente de execução por sua vez não foi implementado da forma ideal. Foi disponibilizada a uma aplicação em C apenas uma função especifica de um sistemas de TVD: o acesso aos dados difundidos em seções privadas. As demais funcionalidades da aplicação ficam restritas ao ambiente de execução da mesma: o sistema operacional GNU/Linux.

7.3 Contribuições e Escopo do Trabalho

A partir do modelo é possível criar novos serviços com o uso das tecnologias de TVD. Além dos exemplos citados no trabalho, outros que façam uso de aplicações difundidas e executadas nos terminais de acesso, assim como aplicações que utilizem demais dados difundidos, podem ser imaginadas.

O modelo apresentado nesse trabalho foi baseado em dois modelos já existentes: uma proposta de ATSC [27] e um modelo da DVB - MHP [19]. Entretanto, o primeiro cobre somente a etapa de emissão, enquanto o segundo foca na recepção. Esse trabalho propôs unir essas duas propostas em um modelo simplificado, porém fim a fim, descrevendo a arquitetura de um sistema de *datacasting*.

Algumas modificações, como a inclusão de determinadas funcionalidades, foram realizadas sobre os modelos da ATSC e DVB. A principal delas se deu no modelo de emissão da ATSC, onde a codificação dos dados através de carrosséis do objeto não é prevista, e tal mecanismo foi contemplado no modelo e implementação deste trabalho.

O mecanismo de carrosséis de objetos mostrou ser um método recomendado de codificação e transmissão de aplicações e demais dados delimitados via rede de difusão. Adotado pelo MHP, foi criado justamente para atender esse tipo de aplicação em redes unidirecionais, como é o caso da TVD sem canal de retorno. Essa tecnologia disponibiliza ao terminal de acesso um sistema de arquivos simples, também definida como um sistema de arquivos de difusão [2]. Com arquivos e seus respectivos diretórios transmitidos ciclicamente, o terminal de acesso, ao necessitar determinado objeto (uma aplicação por exemplo), apenas aguarda até sua próxima repetição no carrossel difundido. Porém, devido a restrições de testes e falta de um ambiente real para tal, questões como desempenho não foram avaliadas, o que restringe a validação desse mecanismo nesse trabalho.

O carrossel de objetos por sua vez apresenta algumas desvantagens. A maior delas é a falta de um mecanismo de endereçamento de terminais de acesso inerente ao protocolo utilizado na transmissão. Essa não é uma desvantagem, por exemplo, de mecanismos de *datacasting* baseados na difusão de datagramas IP via MPE.

Os carrosséis, assim como as seções privadas, não possuem mecanismos de sincronização a nível do próprio protocolo de transmissão de dados. Assim como as MPE, nesse caso, possuem desvantagem em relação ao datacasting via PES, que carrega nas próprias mensagens de dados estampilhas de tempo com relação aos instantes de decodificação e apresentação desejados. Por sua vez, os carrosséis de objetos podem ser sincronizados com outros elementos do serviço do qual pertencem através do uso de outras estruturas, como por exemplo através de tabelas de sinalização.

A implementação, apesar de abstrair alguns componentes dos sistemas de TVD, possibilitou que um conjunto de arquivos qualquer fosse codificado e multiplexado em um fluxo de transporte. Conforme determina o padrão aberto de *facto* para tal em sistemas de TVD, também possibilitou que o conjunto de arquivos fosse decodificado no terminal de acesso simulado.

Quando o sistema brasileiro de TVD for definido, e as primeiras difusões digitais terrestres realizadas, será crescente no país o interesse pela aquisição de placas sintonizadoras e demoduladores do sinal de televisão digital para PC. A implementação desse trabalho pode servir como referência principalmente para a disponibilização de decodificadores de carrosséis de objetos livres, fornecendo aos usuários de PCs acesso aos arquivos difundidos em determinados serviços. Com a integração dos emuladores MHP livres (Caso esse padrão de *middleware* sejá adotado pelo Sistema Brasileiro de Televisão), terminais de acesso reais em PC podem ser implementados.

7.4 Perspectivas Futuras

A implementação abstraiu o meio de difusão através de um *fifo* do Linux, resultando em um sistema com o difusor e terminal de acesso executados na mesma máquina. Tal implementação pode ser estendida, de forma que o fluxo de transporte produzido seja transmitido através de uma rede qualquer, em *multicast*, via RTP por exemplo, a várias outras máquinas executando a implementação do terminal de acesso.

Um outra sugestão de continuação do trabalho é integração dos componentes implementados (demultiplexador e decodificadores) com ferramentas de emulação de terminais de acesso como o XleTView [42] e o OpenMHP [43]. Através do mapeamento da API MHP desses, integrando-a com as funcionalidades do demultiplexador e decodificador, é possível emular de forma mais realista um aplicativo MHP em um PC. Com o desenvolvimento então de um *gerenciador de aplicação* (*application manager*), o PC passa a funcionar de forma similar a um terminal de acesso real.

O modelo também pode ser estendido de forma a suportar as outras formas de *datacasting*, também descritas no trabalho, como MPE, PES e até mesmo o mecanismo em fase final de padronização, o ULE, pesquisado atualmente pela IETF. Dessa forma, o modelo pode ser adaptado para fornecer suporte ao envio de dados endereçados, via IP, a terminais de acesso. Como descrito brevemente nos capítulos anteriores, o mapeamento dos endereços IP para os endereços MAC depende de tabelas adicionais de cada sistema de TVD, não existindo ainda um consenso ou padrão universal para tal (motivo pelo qual o ULE é tema de pesquisa da IETF).

O suporte ao *datacasting* via PES impõe alguns desafios ao modelo e implementação, de forma a suportar sua principal vantagem, o suporte nativo ao *datacasting* síncrono e sincronizado. Tais desafios dizem respeito a recuperação de relógios de base de serviços e outros fluxos elementares, de forma a inserir estampilhas de tempo nas PES sincronizando-as com esses. Apesar de já abordado em trabalhos como [5], fica também como sugestão de trabalhos futuros.

Uma evolução final do modelo é a adição do canal de retorno. Dessa forma, as aplicações difundidas passam a possuir um meio de enviar informações de volta ao difusor ou ao provedor de conteúdo. Essa é a base para aplicações de interatividade real esperadas nos terminas de acesso.

Apêndice A

Sintaxe das Mensagens DSM-CC do Carrossel de Dados

A.1 dsmccMessageHeader

Tabela A.1: *Sintaxe do cabeçalho da DSI e DII.*

Sintaxe	**Número de Bytes**
dsmccMessageHeader () {	
protocolDiscriminator	1
dsmccType	1
messageId	2
transactionId	4
reserved	1
adaptationLength	1
messageLength	2

if (adaptationLength > 0) {
dsmccAdaptationHeader
}	

}	

A.2 dsmccDownloadDataHeader

Tabela A.2: *Sintaxe do cabeçalho da DDB*

Sintaxe	**Número de Bytes**
dsmccDownloadDataHeader () {	
protocolDiscriminator	1
dsmccType	1
messageId	2
downloadId	4
reserved	1
adaptationLength	1
messageLength	2

for (adaptationLength > 0) {
dsmccAdaptationHeader
}	

}	

A.3 DownloadServerInitiate - DSI

Tabela A.3: *Sintaxe da DSI.*

Sintaxe	Número de Bytes
DownloadServerInitiate () {	
dsmccMessageHeader ()	
serverId	20
compatibilityDescriptor ()	
privateDataLength	2

for (i=0; i<privateDataLength; i++) {
privateDataByte
}
	1

}	

A.4 DownloadInfoIndication - DII

Tabela A.4: *Sintaxe da DII.*

Sintaxe	Número de Bytes
DownloadInfoIndication () {	
dsmccMessageHeader ()	
downloadId	4
blockSize	2
windowSize	1
ackPeriod	1
tCDownloadWindow	4
tCDownloadScenario	4
compatibilityDescriptor ()	
numberOfModules	2
for (i=0; i<numberOfModules; i++) { moduleId	2
moduleSize	4
moduleVersion	1
moduleInfoLength	1

for (i=0; i<moduleInfoLength; i++) {
moduleInfoByte
}
	1

}	
privateDataLength	2

for (i=0; i<privateDataLength; i++) {
privateDataByte
}
	1

}	

A.5 DownloadDataBlock - DDB

Tabela A.5: *Sintaxe da DDB.*

Sintaxe	Número de Bytes
DownloadDataBlock () {	
dsmccDownloadDataHeader ()	
moduleId	2
moduleVersion	1
reserved	1
blockNumber	2

for (i=0; i<N; i++) {
blockDataByte
}
	1

}	

Apêndice B

Sintaxe das Mensagens DSM-CC do Carrossel de Objetos

B.1 IOR

Tabela B.1: *Sintaxe da IOR*

Sintaxe	**Número de Bytes**
IOR () {	
type_id_length	4 (N1)
for (i=0; i<N1; i++) {	
type_id_byte }	1
if (N1 % 4 != 0) {	
for (i=0; i<(4-(N1 % 4)); i++) {	
alignment_gap } }	1
taggedProfiles_count	4 (N2)
for (i=0; i<N2; i++) {	
taggedProfile () {	
profileId_tag	4
profile_data_length	4 (N3)
for (j=0; j<N3; j++) {	
profile_data_byte } }	
}	
}	

B.2 BIOP Profile Body

Tabela B.2: *Sintaxe da BIOP Profile Body*

Sintaxe	Número de Bytes
BIOPProfileBody {	
profileId_tag	4
profile_data_length	4
profile_data_byte_order	1
liteComponents_count	1 (N1)
ObjectLocation {	
componentId_tag	4
component_data_length	1
carouselId	4
moduleId	2
version.major	1
version.minor	1
objectKey_length	1 (N2)
for (i=0; i<N2; i++) {	
objectKey_data_byte } }	1
ConnBinder {	
componentId_tag	4
component_data_length	1
taps_count	1 (N3)
Tap { id	2
use	2
association_tag	2
selector_length	1
selector_type	2
transactionId	4
timeout }	4
for (i=0; i<(N3-1); i++) {	
Tap {	
id	2
use	2
association_tag	2
selector_length	1 (N4)
for (j=0; j<N4; j++) {	
selector_data_byte } } } }	1
for (i=0; i<(N1-2); i++) {	
LiteComponent {	
componentId_tag	4
component_data_length	1 (N6)
for (j=0; j<N6; i++) {	
component_data_byte } } }	1
}	

B.3 BIOP Directory Message

Tabela B.3: *Sintaxe da BIOP Directory Message*

Sintaxe	Número de Bytes
BIOPDirectoryMessage () {	
magic	4
biop_version.major	1
biop_version.minor	1
byte_order	1
message_type	1
message_size	4
objectKey_length	1 (N1)
for (i=0; i<N1; i++) {	
objectKey_data_byte }	1
objectKind_length	4
objectKind_data	4
objectInfo_length	2 (N2)
for (i=0; i<N2; i++) {	
objectInfo_data_byte }	1
serviceContextList_count	1 (N3)
for (i=0; i<N3; i++) {	
context_id	4
context_data_length	2 (N9)
for (j=0; j<N9; j++) {	
context_data_byte } }	1
messageBody_length	4
bindings_count	2 (N4)
for (i=0; i<N4; i++) {	
Name {	
nameComponents_count	1 (N5)
for (j=0; j<N5; j++) {	
id_length	1 (N6)
for (k=0; k<N6; k++) {	
id_data_byte }	1
kind_length	1 (N7)
for (k=0; k<N7; k++) {	
kind_data_byte } } }	1
bindingType	1
IOR ()	
objectInfo_length	2 (N8)
for (k=0; k<N7; k++) {	
objectInfo_data_byte} }	4
}	

B.4 BIOP File Message

Tabela B.4: *Sintaxe da BIOP File Message*

Sintaxe	Número de Bytes
BIOPFileMessage () {	
magic	4
biop_version.major	1
biop_version.minor	1
byte_order	1
message_type	1
message_size	4
objectKey_length	1 (N1)
for (i=0; i<N1; i++) { objectKey_data_byte }	1
objectKind_length	4
objectKind_data	4
objectInfo_length	2 (N2)
ContentSize	8
for (i=0; i<(N2-8); i++) { objectInfo_data_byte }	1
serviceContextList_count	1 (N3)
for (i=0; i<N3; i++) {	
context_id	4
context_data_length	2 (N9)
for (j=0; j<N9; j++) { context_data_byte } }	1
messageBody_length	4
content_length	4 (N4)
for (i=0; i<N4; i++) { content_data_byte }	1
}	

B.5 BIOP Module Info Message

Tabela B.5: *Sintaxe da BIOP Module Info Message*

Sintaxe	Número de Bytes
BIOPModuleInfo () {	
ModuleTimeOut	4
BlockTimeOut	4
MinBlockTime	4
taps_count	1 (N1)
for (i=0; i<N1; i++) {	
id	2
use	2
association_tag	2
selector_length	1
}	
UserInfoLength	1 (N2)
for (i=0; i<N2; i++) {	
userInfo_data_byte	1
}	
}	

B.6 ServiceGatewayInfo

Tabela B.6: *Sintaxe do Service Gateway Info*

Sintaxe	Número de Bytes
ServiceGatewayInfo () {	
IOR ()	
downloadTaps_count	1 (N1)
for (i=0; i<N1; i++) {	
Tap	
}	
serviceContextList_count	1 (N2)
for (i=0; i<N2; i++) {	
context_id	4
context_data_length	2 (N3)
for (j=0; j<N3; j++) {	
context_data_byte	1
}	
}	
userInfoLength	2 (N4)
for (i=0; i<N4; i++) {	
userInfo_data_byte	1
}	
}	

Nomenclatura

AAC Advanced Audio Coding

ADSL Assimetric Digital Subscriber Line

AIT Application Information Table

API Application Programming Interface

ARIB Association of Radio Industries and Businesses

ATM Asynchronous Transfer Mode

ATSC Advanced Television Systems Committee

BIOP Broadcast Inter ORB Protocol

CAT Conditional Access Table

CDMA Code-Division Multiple Access

CODED Codificator-DECodificator

COFDM Coded Orthogonal Frequency Division Multiplexing

CORBA Common Object Request Broker Architecture

CPU Central Processing Unit

CRC Cyclic Redundancy Check

DAB Digital Audio Broadcasting

DASE DTV Application Software Enviroment

DDB Download Data Block

DII Download Info Indication

DRM Digital Rights Management

DSI Download Server Initiate

DTS Decoding Time Stamp

DVB Digital Vídeo Broadcasting

DVD Digital Versatile Disk

EEPROM Electrically Erasable Programmable Read-Only Memory

ETRI Electronics and Telecommunications Research Institute

GNU GNU is Not Unix

GPL GNU General Public License

GSM Global System for Mobile Communication

HDTV High Definition Television

IDE Integrated Drive Electronics

IEC International Electrotechnical Commission

INT IP/MAC Notification Table

IOR Interoperable Object Reference

IP Internet Protocol

IPDC IP Datacasting

ISDB Integrated Services Digital Broadcasting

ISO International Organization for Standardization

LMDS Local Multipoint Distribution System

MAC Media Access Control

Mbps Megabits per second

MHP Multimedia Home Plataform

MHz Mega Hz: Milhões de ciclos por segundo

MPE Multiprotocol Encapsulation

MPEG Moving Pictures Experts Groups

NIT Network Information Table

NTSC National Television System Committee

ORB Object Request Broker

OSI Open Systems Interconnection

PAT Program Association Table

PC Personal Computer

PCMCIA Personal Computer Memory Card International Association

PCR Program Clock Reference

PES Packetized Elementary Streams

PLC Power Line Communication

PMT Program Maple Table

PSI Program Specific Information

PTS Presentation Time Stamp

RAM Random Access Memory

RFC Requests for Comments

ROM Read Only Memory

RTP Real Time Protocol

TVD Televisão Digital

ULE Ultra Lightweight Encapsulation

URL Uniform Resource Locator

USB Universal Serial Bus

VDR Video Digital Recorder

VSB Vestigial Side Band

Referências Bibliográficas

[1] L. Staffans. Internet protocol datacasting, a technology overview. Master's thesis, Helsinki University of Technology, 2004.

[2] Tektronix. *A Guide to MPEG Fundamentals and Protocol Analysis*, 2002. URL `http://www.tek.com/Measurement/App_Notes/25_11418/eng/25W_11418_4.pdf`.

[3] European Telecommunications Standards Institute. *Digital Video Broadcasting: Implementation guidelines for Data Broadcasting*, 2003. ETSI TR 101 202.

[4] G. Zhiqi, Y. Songyu, and Z. Wenjun. Using object multiplex technique in data broadcast on digital CATV channel. *IEEE Transactions on Broadcasting*, 50(2):113–119, Jun. 2004.

[5] J.M. Jeong et al. Development of interactive data broadcasting system compliant with ATSC standards. *ETRI Journal*, 26(2):149–160, Apr. 2004.

[6] D. Catapano et. al. DTV data broadcasting: Opportunities and experiences. Technical report, Triveni Digital Inc., Harris Corporation, 2003.

[7] J. Forster and C. Lovell. Sorting out the bits - Digital Television and Datacasting in Australia. *International Journal of Communications Law and Policy*, 1(6), 2000.

[8] B. Christos, K. Vaggelis, and M. Ioannis. A platform for gaming in Digital Interactive TV. In *10th International Conference on Software, Telecommunications and Computer Networks*, pages 105–109, Croatia, Italy, Oct. 2002. Technology Institute and University of Patras.

[9] Y. Choi. Impacts of digital technologies on the broadcast industries - production, distribution, and organizational operations. In *12th KISDI International Conference - The Future of Digital Television*, Nov. 2001.

[10] S. Bushholz, A. Schill, and T. Ziegert. A simulation study of update techniques for cyclic data broadcast. In *4th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 115–122, Rome, Italy, Jul. 2001.

[11] G. Thomas. ATSC Datacasting: Opportunities and challenges. In *NAB2000 Broadcasting Eng. Conf.*, Apr. 2002.

[12] E.A. Heredia. Optimal object allocation for multimedia broadcast. In *Int. Conf. Acoustics, Speech, and Signal Processing*, pages 3717–3720, May 1998.

[13] V. Becker e C. Montez. TV Digital Interativa: Conceitos, desafios e perspectivas para o brasil, 2004.

[14] M. Pagani. *Multimedia and Interactive Digital TV: Managing the Opportunities Created by Digital Convergence*. IRM Press, 2003.

[15] W. Leister et al. Digital TV - a survey. Technical report, Norsk Regnesentral, Dec. 2002.

[16] E.M. Schwalb. *iTV Handbook: Technologies and Standards*. Prentice Hall PTR, 2003.

[17] M.T. Andrade. Set-top boxes, 2005. URL `http://telecom.inescn.pt/people/mandrade/tvd/2000/galeria/trabalho1-1/set-top.htm`. Último acesso em 26 de janeiro.

[18] TV Cabo Portugal. Televisão interactiva, 2005. URL `http://pwp.netcabo.pt/workpage/iTV/files/indice.html`. Último acesso em 26 de janeiro.

[19] European Telecommunications Standards Institute. *Digital Video Broadcasting: Multimedia Home Platform Specification 1.0.3*, 2003. ETSI ES 201 812 V1.1.1.

[20] S. Morris. Mhp interactive, 2005. URL `http://www.interactivetvweb.org/tutorial/mhp/index.shtml`. Último acesso em 26 de janeiro.

[21] Sun Microsystems. Java TV Technology, 2005. URL `http://java.sun.com/products/javatv`. Último acesso em 26 de janeiro.

[22] Moving Picture Experts Group. The MPEG home page, 2005. URL `http://www.chiariglione.org/mpeg`. Último acesso em 27 de janeiro.

[23] International Organization for Standardization. *Coding of Moving Pictures and Associated Audio - MPEG-2 Systems*, 2000. ISO/IEC 13818-1.

[24] G. Fairhurst. Data transmission using MPEG-2 and DVB, 2005. URL `http://www.erg.abdn.ac.uk/research/future-net/digital-video/dsm-cc.html`. Último acesso em 27 de janeiro.

[25] International Organization for Standardization. *Coding of Moving Pictures and Associated Audio - Extension for Digital Storage Media Command and Controls*, 1996. ISO/IEC 13818-6.

[26] Object Management Group, 2005. URL http://www.omg.org. Último acesso em 27 de janeiro.

[27] ATSC Implementation Subcommittee Informational Document. Implementation of data broadcasting in a DTV station. Technical report, Advanced Television Systems Committee, 1999.

[28] Advanced Television Systems Committee. *ATSC Recommended Guidelines for the ATSC Data Broadcasting Standard*, 2001. ATSC A/91.

[29] Sonera MediaLab. IP Datacasting content services white paper. Technical report, Sonera MediaLab, May 2003.

[30] IETF IP over DVB Working Group, 2005. URL http://www.ietf.org/html.charters/ipdvb-charter.html. Último acesso em 16 de fevereiro.

[31] D. Bhat et. al. An open interface to a DTV data server. In *NAB2001 Broadcast Engineering Conference*, Apr. 2001.

[32] C. Wei and S. Hong. An FM subcarrier data broadcast on bus transportation status indication system. *IEEE Transactions on Broadcasting*, 47(1):76–79, Mar. 2001.

[33] M. Azimi. Data transmission schemes for a new generation of interactive digital television. Master's thesis, The University of British Columbia, Canada, Mar. 2004.

[34] IPDC Fórum, 2005. URL http://www.ipdc-forum.org. Último acesso em 15 de fevereiro.

[35] Internet Engineering Task Force. *Ultra Lightweight Encapsulation (ULE) for transmission of IP datagrams over MPEG-2/DVB networks*, Jan. 2005.

[36] Dotcast, 2005. URL http://www.dotcast.com. Último acesso em 03 de fevereiro.

[37] Moviebeam, 2005. URL http://www.moviebeam.com. Último acesso em 03 de fevereiro.

[38] M. Pikarski. DSM-CC MHP object carousel MPEG2 encoding tools, 2005. URL http://www.linuxtv.org/dsmcc-mhp-tools. Último acesso em 27 de janeiro.

[39] O. Schirmer. ISO 13818 stream multiplexer, 2005. URL http://www.linuxtv.org/multiplexer/. Último acesso em 27 de janeiro.

[40] F. Bellard. FFmpeg Multimedia System, 2005. URL http://ffmpeg.sourceforge.net. Último acesso em 27 de janeiro.

[41] A. Cellerier. VLC media player, 2005. URL http://www.videolan.org. Último acesso em 27 de janeiro.

[42] M. Sveden. XleTView, 2005. URL `http://xletview.sourceforge.net`. Último acesso em 27 de janeiro.

[43] Axel Technologies and Turku Centre for Computer Science. OpenMHP, 2005. URL `http://www.openmhp.org`. Último acesso em 27 de janeiro.