UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTAT[I]STICA

BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

Uma Implementação de Broadcast File System para Transmissão de Dados de TV Digital

George Elias Ferreira da Silva

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

Florianópolis – SC

2008/1

George Elias Ferreira da Silva

Uma Implementação de Broadcast File System para Transmissão de Dados de TV Digital

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Professor Doutor Antônio Augusto Medeiros Fröhlich

Banca Examinadora

Msc. Lucas Francisco Wanner

Hugo Marcondes

"Quem se vinga depois da vitória é indigno de vencer."

- Voltaire

Agradecimentos

Primeiramente gostaria de agradecer a meus pais, que sempre me apoiaram em todas as minhas decisões. Sempre se esforaçaram para que oportunidades surgissem para mim. Por estarem sempre ao meu lado, em momentos de tristeza e alegria, pela família que criaram e pelo ambiente sempre agradável em casa. A minha namorada que sempre me apoiou, sempre me alegrou e sempre me incentivou a perseguir meus sonhos e ideais. Agredeço a minha irmã que também manteve meus dias alegres e a casa limpa. Agradeço também ao meu orientador que sempre acreditou no meu potencial para realização deste trabalho e sempre me deu plena liberdade para execução do mesmo. Agradeço também a todos os meus amigos, principalmente ao Pit que desde a infância foi amigo leal e me ajudou quando precisei, ao Lep que estava sempre disposto a fazer festa mas também a estudar, ao Bruno que com suas piadinhas sempre alegra o pessoal, ao Igor que não tem noção nenhuma das coisas, ao Alexandre que é nosso vereador. Também quero agradecer ao pessoal do Lisha que sempre que precisei me ajudou, possibilitando a execução deste trabalho e também as risadas nos churrascos do laboratório. E finalmente agradecer a Deus, por colocar pessoas maravilhosas ao meu redor e sempre iluminar meu caminho ...

Sumário

Lista de Abreviaturas

ATM: Asyncronous Transfer Mode

BIOP: Broadcast Interoperability Protocol

DDB: DownloadDataBlock

DII: DownloadInfoIndication

DSI: DownloadServerInitiate

DSM-CC: Digital Storage Media - Command and Control

EPOS: Embedded Parallel Operating System

IP: Internet Protocol

ISO/IEC: International Organization for Standards/International Electrotechnical Comission

MPEG-2: Motion Picture Experts Group version 2

MPEG-2 TS: MPEG-2 Transport Stream

RPC: Remote Procedure Call

SBTVD: Sistema Brasileiro de Televisão Digital

SRM: Source and Session Manager

STB: Set-Top Box

U-N: Usuário para rede (do ingles user to network)

U-U: Usuário para usuário (do ingles user to user)

UDP: User Data Protocol

VFS: Virtual Filesystem Switch

Lista de Figuras

1 Introdução

Este trabalho consiste na adaptação do sistema de arquivos do EPOS (MARCONDES, 2004) para ser utilizado juntamente com uma implementação de Digital Storage Media - Command and Control (DSM-CC) seguindo as normas definidas para o SBTVD.

1.1 Motivação

Embora não sejam muito evidentes aos olhos dos consumidores, os sistemas embarcados e embutidos praticamente dominam o mercado da computação (FRÖHLICH, 2003), estando cada vez mais presentes nos automóveis, PDA's, microondas, celulares, aparelhos de imagem e áudio, etc. Os sistemas operacionais mais populares no mercado atendem com eficiência apenas os requisitos da computação de propósito geral, principalmente por serem sistemas de grande extensibilidade (FRÖHLICH, 2003), uma vez que não se sabe antecipadamente quais aplicações irão rodar nele.

Essa extensibilidade é inadequada ao mundo dos sistemas embarcados, que geralmente possuem restrições severas em relação a processamento e armazenamento, e nesse mundo, a aplicação é conhecida de antemão.

Assim, o sistema operacional EPOS, define um sistema altamente adaptável a aplicação para suportar sistemas de computação dedicados, mais especificamente, sistemas paralelos e embutidos (FRÖHLICH, 2002).

"Toda aplicação necessita armazenar e recuperar informações"(TANENBAUM; WOODHULL, 1997). Para sanar esta necessiade, o sistema operacional fornece um sistema de arquivos à aplicação, permitindo que dados sejam armazenados de forma persistente em um dispostivo físico, permitindo que diversos arquivos coexistam e se organizem de forma lógica do ponto de vista do usuário do sistema operacional.

Sistemas de arquivos também fazem parte de aplicações multimedia, como na televisão digital, que não transmite apenas imagem e som, mas também dados. Desta forma, há a neces-

sidade de se enviar estes dados garantindo que chegarão ao seu destino. Para isto foi definido o DSM-CC, padrão ISO/IEC que oferece protocolos para entrega de uma aplicação completa, como vídeo sob demanda. Através do carrosel de dados, o DSM-CC envia os pacotes de dados repetidamente, garantindo que todos chegarão ao destino. O carrosel de dados é mecanismo que permite a um servidor de aplicações faça *broadcast* de forma cíclica de um conjunto de dados, repetindo o conteúdo do carrosel uma ou mais vezes.

1.2 Justificativa

Assim como informações de áudio e vídeo são submetidos a processos de codificação e decodificação para serem transmitidas, a codificação de dados é necessária para o funcionamento da especificação Sistema Brasileiro de Televisão Digital (SBTVD). Esta pode ser feita através de dois mecanismos: Carrosel de Dados ou Encapsulamento Multiprotocolo que permite que um datagrama de qualquer protocolo de comunicação seja transmitido através do DSM-CC.

Neste trabalho será implementado o carrosel de dados como mecanismo de codificação para o SBTVD. A implementação se dará utilizando o sistema operacional EPOS, que fornecerá os mecanismos de interação com o hardware e o sistema de arquivos que suportará a estrutura do DSM-CC.

A escolha do sistema operacional EPOS justifica-se por ser um sistema orientado a aplicação. Permitindo assim que o sistema gerado atenda apenas as necessidades da aplicação, tanto em software quanto em hardware. Desta forma, o sistema torna-se um sistema com menor custo, mais eficiente e seu projeto torna-se mais rápido.

1.3 Objetivos

Este trabalho possui dois objetivos. Primeiro é refatorar o projeto de sistema de arquivos desenvolvido por (MARCONDES, 2004), possibilitando que o EPOS atenda as seguintes necessidades do SBTVD:

- Carrosel de dados para transmissão de arquivos
- Broadcast File System (BFS) para gerenciamento do sistema de arquivos
- Implementação do BFS no servidor (transmissão) e no cliente (recepção)

Segundo implementar a fração necessária da definição do DSM-CC para que seja possível utilizar o EPOS como sistema operacional para aplicações de TV Digital que utilizem o SBTVD.

Para isto, é necessário:

- Estudar o padrão ISO/IEC que define o DSM-CC, identificando as partes a serem implementadas para atender os requisitos do SBTVD. Estudar sistemas de arquivos, com anteção especial ao trabalho de (MARCONDES, 2004). Estudar o sistema operacional escolhido para desenvolvimento do trabalho. Além de consolidar conhecimentos na área de engenharia de software e software básico.
- Projetar as mudanças no sistema operacinal EPOS, utilizando modelagem UML, facilitando a compreensão e a implementação do trabalho.
- Implementar as mudanças necessárias no sistema de arquivos do EPOS e as definições DSM-CC relativas ao SBTVD.
- Testar o sistema gerado através de aplicação teste.

2 *Sistemas de Arquivos*

Sistema de arquivos é a parte do sistema operacional responsável por gerenciar arquivos, o modo como são estruturados, nomeados, acessados, usados, protegidos e implementados.

O sistema de arquivos visa solucionar três fatores limitantes para execução de uma aplicação:

- Quantidade limitada de memória para armazenamento de informações no espaço de endereçamento.
- Perda das informações do espaço de endereçamento após o término da execução de um processo.
- Compartilhamento de informações entre distintos processos em um sistema.

Segundo [Tanenbaum e Woodhull 1997], a solução mais comum para esses fatores é o armazenamento de informações em discos ou outras mídias externas em unidades chamadas de arquivos. Assim os processos podem ler e escrever em arquivos as informações que necessitam ser compartilhadas, mantidas após a sua execução e as informações que são grandes o suficiente para não caberem na memória principal do sistema, dentro do espaço de endereçamento do processo.

Assim os arquivos são gerenciados pelo sistema operacional e a maneira como eles são estruturados, nomeados, acessados, utilizados, protegidos e implementados são tópicos importantes no desenvolvimento de um sistema operacional [Tanenbaum e Woodhull 1997]. Dá-se o nome de sistemas de arquivos a parte do sistema operacional que manipula os arquivos.

2.1 Armazenamento

Apesar de já existirem tecnologias de armazenamento em memórias não voláteis, como memórias FLASH, que se assemelham a memórias SDRAM mas que garantem a persistência

dos dados, grande parte dos dados são armazenados em discos rígidos magnéticos ou ópticos devido ao menor custo, de forma que não inviabilizem dispositivos que necessitam armazenar grandes quantidades de dados.

Ambas as tecnologias trabalham orientados a blocos, sendo seu espaço de armazenamento acessado e divido como blocos físicos. Tamanho do bloco físico varia de acordo com o projeto de hardware e forma geométrica no caso de discos. Em geral, discos magnéticos possuem blocos de 512bytes isso significa que cada endereço físico acessa blocos de 512bytes.

É comum encontrarmos discos magnéticos variando entre alguns megabytes e a terabytes. Hoje usuários domésticos em geral, possuem discos em torno de 80Gb, estes discos se divididos em blocos 512bytes possuiriam espaço de endereçamento com 167.772.160 endereços. O sistema de arquivos precisa saber quais desses blocos possuem dados que precisam ser mantidos ou que podem ser realocados para criação e redimensionamento de arquivos. Para atingir este objetivo uma estrutura de controle é necessária.

O gerenciamento dessa estrutura pode ser uma terefa de alto custo para o sistema se o número de blocos físicos for elevado. Por este motivo, existe um outro nível de blocos, chamado de blocos lógicos que é utilizado pelo sistema.

A escolha do tamanho dos blocos pelo sistema de arquivos é vital para o desempenho do sistema. Blocos pequenos significam arquivos com muitos blocos, reduzindo a performance devido ao aumento do tempo de busca dos blocos no disco. Já blocos muito grandes aumentam a fragmentação interna, desperdiçando capacidade de armazenamento. Um estudo de [Mullender e Tannembaum 1984] mostra que o tamanho médio de um arquivo no sistema UNIX é por volta de 1Kb.

Uma possível solução para este problema é a utilização de blocos de tamanho variável, infelizmente esta implementação não é nada trivial, fazendo com que a escolha de um tamanho fixo ideal para os blocos de acordo com o tamanho do disco seja uma solução mais utilizada.

Um disco pode ser particionado em diversos volumes, de capacidade inferior a capacidade do disco como um todo. Os sistemas de arquivos estão contidos nos volumes e só conhecem o volume a qual pertencem, sendo assim, cada volume é uma unidade independente para o sistema. Somente uma parte do sistema operacional diferencia um disco físico e um volume, sendo que o resto do sistema conhece apenas os volumes. [Fröhlich 1994]

Segundo [Fröhlich 1994] a principal vantagem no uso de volumes é a sua maior tolerância a falhas, uma vez que se algum dado for corrompido, apenas o volume ao qual ele pertence é corrompido.

2.2 Arquivos

"Arquivo é um mecanismo de abstração. Ele oferece meios de armazenar informações no disco e de tê-las depois. Isso deve ser feito de um modo que isole o usuário dos detalhes sobre como e onde a informação está armazenada e como os discos na verdade funcionam." (TANENBAUM; WOODHULL, 1997)

Partindo da definição de (TANENBAUM; WOODHULL, 1997), descreveremos algun aspectos importantes sobre arquivos facilitando sua compreensão.

2.2.1 Nomeação

Segundo Tanenbaum, talvez esta seja a característica mais importante de um mecanismo de abstração qualquer: o modo como os objetos são gerenciados e nomeados. Um processo cria um arquivo que continua existindo após o processo ser encerrado e pode ser acessado por outros processos através de seu nome.

Apesar de cada sistema operacional possuir suas próprias regras para realizar a nomeação de arquivos, todos os sistemas atuais permitem cadeias de caracteres de um até oito letras como nomes válidos para arquivos. Frequentemente, dígitos e caracteres especiais também são permitidos. Alguns sistemas fazem distinção entre caracteres maiúsculos e minúsculos.

Muitos sistemas suportam nomes de arquivos divididos em duas partes separados por um ponto. A parte que segue o ponto é chamada de extensão do arquivo, esta em alguns sistemas é utilizada para indicar algo sobre o arquivo, como qual programa será utilizado para ler determinado arquivo.

2.2.2 Estrutura

Existem três formas de se estruturar um arquivo, como uma sequência de bytes, onde o sistema operacional não sabe o que contém o arquivo ou simplesmente não se interessa por isso, deixando isso a cargo do usuário. Essa estratégia permite máxima flexibilidade, deixando que os programas do usuário coloquem qualquer informação no arquivo.

Outra abordagem é ter arquivos como uma sequência de registros de tamanho fixo, cada um com alguma estrutura interna. A ideia básica desta abordagem é que as operações de leitura e escrita de um arquivo são facilitadas, a primeira retorna um registro, a segunda apenas sobrepõe ou anexa registros.

A terceira abordagem é a mais elaborada, tendo um arquivo como uma árvore de registros, não necessariamente de mesmo tamanho, cada um possui um campo chave e uma posição fixa no registro. Esta árvore é ordenada pelo campo chave, facilitando a busca por registros.

2.2.3 Descritores

Estruturas contidas no sistema de arquivos que visam descrever um arquivo no sistema. Transparentes ao usuário são utilizadas para controle do sistema de arquivos e segundo (TANENBAUM; WOODHULL, 1997) possuem 4 implementações diferentes.

Alocação Contínua

É a forma mais simples de se descrever a alocação dos dados de um arquivo. Os dados ficam contidos em um conjunto de blocos sequenciais, seu descritor precisa apenas apontar a posição do primeiro bloco e o tamanho do arquivo, assim os blocos seguintes são acessados somando-se o deslocamento.

Apesar de sua fácil implementação e rapidez na busca por blocos no disco, uma vez que são sequenciais, há a desvantagem de se definir um tamanho de arquivo na hora de sua criação, dificultando sua expansão.

Alocação em Lista Ligada

Semelhante a alocação contínua, porém cada bloco aponta para o bloco seguinte, permitindo que os blocos sejam armazenados em qualquer lugar do disco.

Sua principal desvantagem ocorre em um acesso randômico, sendo necessário percorrer toda a sequência, partindo do primeiro bloco para se atingir o bloco desejado.

Alocação em Lista Ligada Indexada

Para tentar sanar as desvantagens da lista ligada, este método mantém uma tabela indexada na memória principal para armazenar a sequência da lista ligada. Cada posição da tabela representa um bloco lógico e seu conteúdo o índice do próximo bloco na cadeia. O último bloco possui conteúdo nulo.

Esta tabela é armazenada no disco e precisa ser lida apenas quando o sistema de arquivos é iniciado. Assim seguir o encadeamento não é uma tarefa cara, devido a alta velocidade de

leitura da memória principal.

A desvantagem é que esta tabela precisa estar inteiramente na memória principal e nem sempre este é um recurso abundante e ela ainda cresce de acordo com o tamanho do disco.

Alocação em Inodos

Método que utiliza uma pequena tabela de tamanho fixo para descrever os arquivos. Ela possui atributos de um arquivo e ainda os endereços dos blocos que o compõem.

Os primeiros endereços do bloco são armazenados no próprio inodo, carregado no momento em que o arquivo é aberto, favorecendo a performance de arquivos pequenos. Para tentar evitar a limitação no tamanho do arquivo, cada inodo possui uma referência indireta simples, que indica um bloco que possui endereços de blocos de dados adicionais. Se esta referência simples não for suficiente, um segundo nível, chamado de referência indireta dupla, é criado, endereçando um bloco que contém uma lista de blocos com referências indiretas simples. Pode ser utilizado ainda um terceiro nível de referências indiretas caso necessário.

2.3 Diretórios

Um sistema de arquivos pode possuir milhares de arquivos, daí surge o conceito de diretórios, criando uma forma de organização para os arquivos dentro do sistema de arquivos.

Os diretórios são sistemas de tradução de nomes de arquivos, são como tabelas de símbolos que traduzem nomes de arquivos em entradas de diretório. Cada arquivo possui sua entrada, que pode ser seu próprio descritor ou seu nome e uma referência para o descritor. A maneira como uma entrada de diretório é implementada, depende da especificação do sistema de arquivos em questão.

Vários modelos estruturais podem ser utilizados para implementar diretórios, alguns deles são descritos a seguir.

2.3.1 Nível Único

É a forma mais simples de se organizar arquivos em diretórios. Existe apenas um diretório, onde estão relacionados todos os arquivos.

Em um sistema grande, ou multi usuários, esta abordagem é inadequada, uma vez que cada entrada deve possuir um identificador único, geralmente o nome do arquivo. Em sistemas multi

usuários significa que nenhum usuário pode possuir arquivo com nome idêntico ao de outro usuário.

2.3.2 Nível Duplo

Este modelo sana uma das desvantagens dos sistemas de arquivos com diretórios de nível único. Existe um diretório central que aponta para diretórios de usuários, onde cada usuário pode armazenar seus arquivos, acabando com o possível conflito de nomes. Novos diretórios de arquivos de usuários podem ser criados e removidos conforme necessário.

2.3.3 Árvores de Diretórios

É a abordagem mais comum. É mais generalizada, organizando os diretórios em árvores de altura indefinida, assim, usuários podem criar diretórios dentro de seus diretórios, formando uma estrutura em árvore.

Neste modelo, cada arquivo possui um único nome, chamado caminho, que pode ser absoluto, formado pelo caminho da raiz da árvore de diretórios até a folha (próprio arquivo). Ou relativo, partindo de um nodo (subdiretório) da árvore de diretórios.

2.3.4 Grafos Acíclicos

Como na estrutura em árvores não é possível compartilhar arquivos entre usuários em localizações diferentes, foi criado o modelo de grafos acíclios de diretórios. Neste é possível compartilhar arquivos e diretórios utilizando entradas especiais de diretório. Vale lembrar que esse compartilhamento não é o mesmo que possuir duas cópias do mesmo arquivo, onde a alteração em uma nada reflete na outra.

Em sistemas UNIX são chamadas links e em sistemas Windows de atalhos.

Alguns problemas emergem com o uso destes grafos, como a exclusão de arquivos dificultada ou ainda o fato de um mesmo arquivo ser lido diversas vezes quando a árvore é atravessada (na busca de um arqvuio por exemplo).

2.3.5 Grafos de Diretórios

Se um sistema de arquivos permitir a criação de links diretos a diretórios, pode-se criar ciclos no grafo de diretórios. Um problema decorrente disto é a dificuldade em saber se um

arquivo pode ser marcado como excluído, uma solução para isto é o uso de um coletor de lixo para identificar quando não há mais nenhuma referência a um arquivo.

2.4 Sistemas de Arquivos no EPOS

O sistema operacional *EPOS (Embedded Parallel Operation System)* foi desenvolvido por (FRÖHLICH, 2002). Em sua primeira implementação, buscava-se utilizar o EPOS como sistema operacional dedicado para nós de cluster de computadores pessoais baseados na arquitetura ix86 e interconectados por um sistema de computação de alto desempenho (MYRINET). Atualmente, os alunos do LISHA (Laboratório de Integração Software e Hardware) da UFSC (Universidade Federal de Santa Catarina), coordenados pelo Prof. Dr. Fröhlich, continuam seu desenvolvimento e realizam suas pesquisas no âmbito deste sistema.

O EPOS é composto basicamente por três tipos de famílias de componentes: Abstração, Mediadores e Aspectos. Características, funcionalidades e estruturas independentes de arquitetura, pertencem a família das Abstrações. Possuem grande reusabilidade, compondo grande parte das necessidades de uma aplicação. As características de hardware, com as implementação das funcionalidades necessárias às aplicações e às abstrações necessitam do hardware do sistema, caracterizam os Mediadores. A configurabilidade do sistema é implementada pela família de Aspectos.

O EPOS ainda oferece um conjunto de ferramentes que permitem que o usuário configure o sistema conforme as necessidades de sua aplicação.

Toda esta infraestrutura contribui para o desenvolvimento do trabalho, permitindo que este seja implementado em um sistema com mínimo *overhead*, trazendo eficiência e portabilidade, permitindo que o resultado do trablaho seja utilizado em dispositivos com poucos ou limitados recursos, tão bem quanto em dispositivos sem restrições.

A modelagem do sistema de arquivos no EPOS, desenvolvida por (MARCONDES, 2004), utiliza conceitos desenvolvidos por (FRÖHLICH, 2001) em sua tese de doutorado. Para isto primeiramente (MARCONDES, 2004) analisou o domínio de sistemas de arquivos e identificou as **famílias de componentes de software** (FRÖHLICH, 2001) que o compõe.

Assim, o domínio em questão foi divido em duas partes: a visão do sistema operacional e a visão do usuário em relação ao sistema de arquivos.

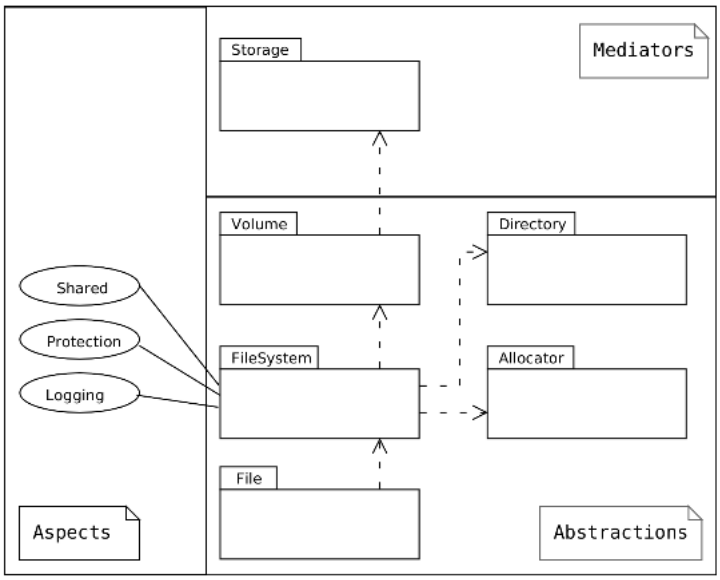

Figura 2.1: Famílias do Sistema de Arquivos. Fonte:Marcondes, 2004

2.4.1 Visão do Sistema Operacional

"Pensando-se na visão do sistema operacional, um sistema de arquivo é uma coleção de estruturas que contém meta-dados e um dispositivo de armazenamento utilizado para gravar os dados." (MARCONDES, 2004)

Dentro desta visão, o sistema de arquivos desenvolvido foi dividido em famílias, as quais serão brevemente descritas a seguir.

- **Família Storage**

  Esta família tem por objetivo implementar mediadores para permitir acesso a um dispositivo de armazenamento de dados persistentes baseados em blocos. Foi implementado "emulando"um dispositivo orientado a blocos, sobre a memória RAM do sistema.

- **Família Volume**

  Visando garantir, se necesário, a tradução de endereços de blocos lógicos para blocos físicos, o acesso de um determinado sistema de arquivos exclusivamente a região que lhe foi destinada e encapsular o conceito de tabela de volumes. Esta família é um implementação de uma unidade lógica chamado VOLUME (partição em alguns sistemas), onde cada membro da família encapsula o conhecimento de uma determinada especificação da tabela de volumes.

- **Família Allocator**

Implementada como uma classe utilitária do EPOS, ou seja, seus membros são genéricos para que todos o sistema e até mesmo a aplicação possam utilizá-la para alocação de recursos. Através do paradigma de programação genérica, utilizando-se o recurso de templates da linguagem C++, buscou-se separar o algoritmo de alocação e a estrutura de dados utilizada por ele para gerenciamento dos blocos livres.

- **Família FileSystem**

  Família central de um sistema de arquivos, engloba todos os algoritmos de acesso a meta-dados, permitindo assim que a implementação da família File seja genérica. São implementadas aqui, algumas operações definidas para arquivos e diretórios, visto que o contexto destas operações modificam diversas estruturas internas ao sistema de arquivos. É ainda implementada um classe denominada FileDescriptor que fornece acesso aos meta-dados necessários aos algoritmos da família File.

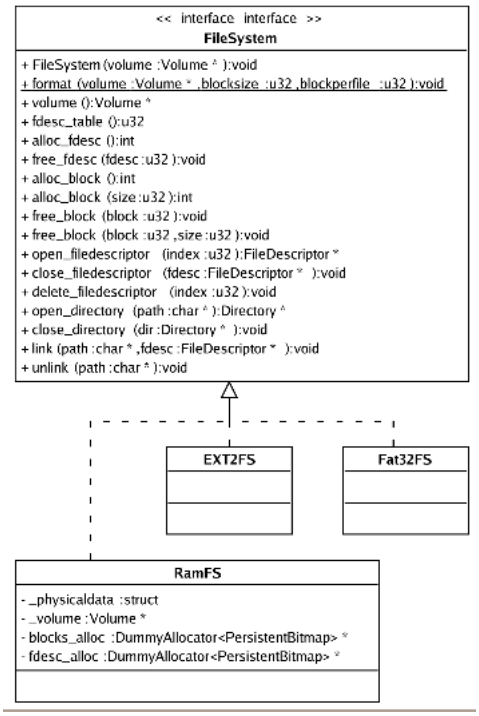

Figura 2.2: Família File System. Fonte:Marcondes, 2004

2.4.2 Visão do Usuário

Aqui, encontramos duas abstrações, diretórios e arquivos, descritas a seguir.

- **Família File**

Como este é o principal componente da modelagem, foi implementado de forma a ser o mais genérico possível, permitindo que componentes realmente genéricos que podem ser aproveitados independentemente do sistema de arquivos fossem criados. Foram criados três membros distintos para esta família, *ContinuousFile*, *RandomFile* e *CircularFile*, onde a diferença se dá na forma como o arquivo é acessado.

- **Família Directory**

  Responsável pelos recursos de nomes a arquivos, esta família da esse suporte ao EPOS, de forma que não precise necessariamente ser utilizada apenas para implementações de sistemas de arquivos, mas também por outros componentes que possuam a característica de tradução de um nome para um algum outro tipo de informação.

3 DSM-CC e BFS

3.1 Definição

Em MPEG-2 ISO/IEC 13816-1 é definida uma versão preliminar do DSM-CC, no qual é definido um simples protocolo de controle de streamming dos pacotes MPEG-2 TS. Esta primeira versão evoluiu para um padrão de mais de 400 páginas denominado ISO/IEC 13816-6 International Standard. Este padrão oferece protocolos para entrega de uma aplicação completa, como video sob demanda.

O DSM-CC é independente de camada de transporte, assim uma aplicação escrita para utilizá-lo não precisa se preocupar com a comunicação entre cliente e servidor. Assim uma aplicação pode ser distribuída por várias redes diferentes. O DSM-CC ainda abrange um número distinto de áreas de protocolo:

- Sessões de rede e controle de recursos
- Configuração de um cliente
- Download para um cliente
- Controle sobre streaming de video
- Serviços de aplicações interativas
- Serviços de aplicações de transmissão genéricas: carrosel de objetos ou de dados e mudança comutada de canal

Cada área de protocolo, pode ser utilizada individualmente ou em conjunto, dependendo da aplicação endereçada.

3.2 Modelo de referência funcional

O DSM-CC é definido com um modelo de referência funcional. O modelo é um cliente ou um servidor, juntos chamados de usuários, que utilizam uma rede para se comunicarem.

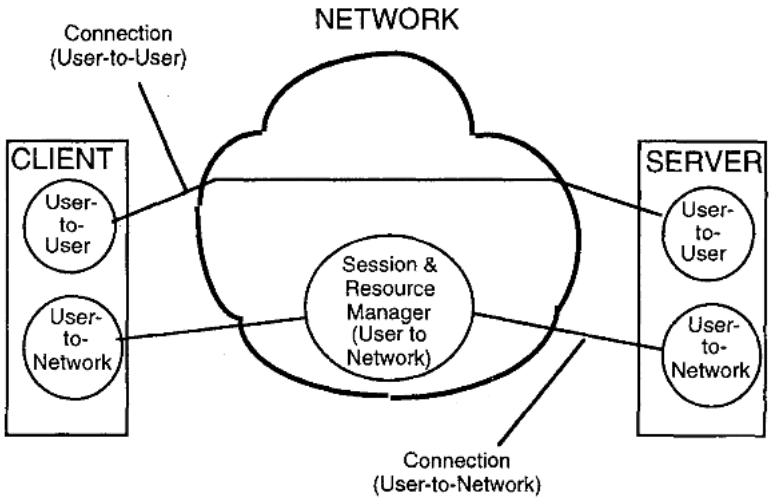

Figura 3.1: DSM-CC modelo de referência funcional

Geralmente clientes são set-top boxes que consomem conteúdo multimedia, enquanto servidores são entidades que transmitem este conteúdo. Um servidor ainda pode ser um sistema distribuído, contendo diferentes plataformas computacionais, algumas especializadas, e também conter múltiplos pontos de conexão de rede.

Os conceitos de rede e conexão são muito amplos. Rede é qualquer conjunto de elementos de comunicação que provê conexão a um usuário. Enquanto conexão é a capacidade de transportar informação entre dois ou mais pontos. A intenção é que o DSM-CC seja aplicável a uma vasta gama de redes físicas, incluindo one-way point-to-multipoint (broadcast).

Um fluxo de informação U-U é usado entre cliente e servidor. O fluxo de informação U-N trafega entre a rede e o client ou servidor. Mensagens U-N são trocadas através da conexão U-N. Seu propósito é o de controlar sessões e recursos da rede. Há uma entidade chamada SRM que é responsável por encerrar a conexão U-N do usuário. Um SRM poderia ser distribuído em uma região geográfica para atender uma rede global vários provedores locais. A especificação do DSM-CC define um protocolo U-N, porém ainda não é definida qualquer comunicação intranet.

O SRM pode gerenciar conexões cliente/servidor baseado em políticas setadas na "assinatura"do serviço, sendo o ponto da rede que fornece informações de configuração para usuários e autenticando os clientes.

3.3 Mensagens U-N

O DSM-CC define um cabeçalho padrão para todas as mensagens U-N. Assume-se que as mensagens serão enviadas através de uma camada de transporte utilizando um protocolo qualquer, que deve atender requerimentos mínimos. Não é necessária a entrega confiável de mensagens, mas deve detectar e discartar mensagens corrompidas. O serviço de transporte deve ser capaz de entregar mensagens U-N completas (camadas inferiores alguma segmentação e remontagem) mas não precisam entregá-las em ordem. Protocolos como UDP IP e também ATM atendem a estes requisitos.

3.4 Conexões U-U

Espera-se que existam várias conexões U-U entre um cliente e um servidor, entretanto, em geral o protocolo usado nestas conexões não é definido pelo DSM-CC, é definido apenas um "acordo"entre cliente e servidor. DSM-CC define um conjunto de serviços genéricos que um servidor pode prover aos clientes. As chamadas destes serviços de procedimento remoto sobre uma conexão U-U.

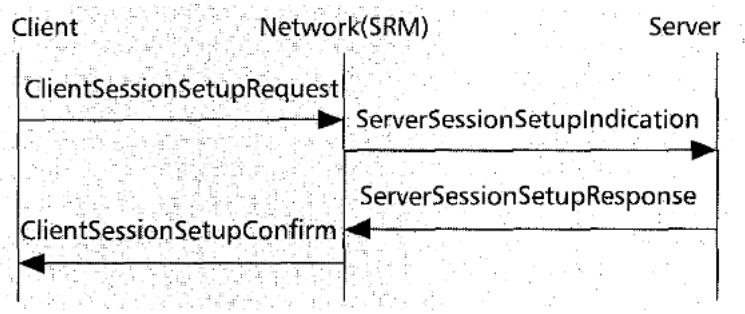

Figura 3.2: Setup de uma conexão U-N

3.5 Sessão de Rede e Controle de Recursos

3.5.1 Sessões

Um conceito chave para o DSM-CC é o de sessão. Sessão é definida como uma associação entre dois usuários, provendo a capacidade de agrupar recursos necessários para a instalação de um serviço. Quando um usuário acessa um serviço de "home shopping", por exemplo, uma sessão é criada e quando o serviço chega ao fim a sessão é encerrada.

3.5.2 Recursos Dentro das Sessões

O DSM-CC foi pensado para ser utilizado em uma rede onde os recursos não são necessariamente gratuitos e em quantidade, mas sim em uma rede onde os recursos são alocados quando necessários por um serviço e liberados quando este termina. Serviços diferentes precisam de recursos diferentes. A alocação de recursos está ligada a sessão. Existe um sessionID único e visível em toda a rede, identificando a sessão, permitindo que seja facilmente identificada. Isto é útil para a cobrança e administração da rede e também para a liberação de recursos utilizados pela sessão quando ela chega ao fim. O protocolo para criação de uma sessão é especificado para permitir autenticação de usuário pela rede e para garantir que o usuário tenha conhecimento de que uma sessão está sendo criada, assim, assumindo responsabilidade por ela.

Durante uma sessão, o servidor pode pedir a rede para adicionar ou remover recursos da sessão dependendo da fase do serviço. Isso é feito através do conjunto de mensagens AddResource, para adicionar, e DeleteResource, para remover. Nestas mensagens são descritos recursos que o servidor precisa da rede usando os descritores de recursos.

3.5.3 Conexões

As conexões para informação U-U são os principais recursos da rede para criar uma sessão. Em geral, uma sessão utilizará mais de uma conexão U-U. Frequentemente sessões possuem controles de fluxo para mensagens RPC e um fluxo MPEG-2. Deve-se observar que o lado do servidor da conexão, não necessariamente é o mesmo para todas as conexões, uma sessão pode receber dados de mais de uma fonte em uma dada sessão.

É possível separar o controle da sessão e dos recursos dos protocolos de controle da conexão. Uma rede pode usar um protocolo de controle de sessão e recursos mas implementar o controle de conexão usando protocolos das camadas de transporte.

3.5.4 Construindo Conexões Através de Recursos da Rede

Um recurso de rede é qualquer unidade de transporte que possa ser alocada. Para algumas redes isto é uma conexão fim-a-fim e os recursos estão ligados diretamente com as conexões. Entretanto o DSM-CC foi feito para trabalhar com conexões multirecursos. Uma conexão é um grupo de uma ou mais conexões e recursos que permitem que informações sejam transferidas entre dois ou mais pontos.

Descritores de recursos são estruturas de dados passadas entre a rede e os usuários através de

mensagens do tipo "resource allocation message"que descrevem os recursos. Estas mensagens possuem valores para atributos apropriados a cada tipo (endereço de origem e destino para ATM SVC ou mapa da tabela de programas para um MPEG TS). Também informa se os atributos são negociáveis e quem inicia a alocação dos recursos (cliente, servidor ou rede).

Descritores de recursos também possuem TAGs de associação, as quais possuem significância para fim-a-fim. Todos os recursos que contribuem para uma conexão possuem o mesmo TAG de associação. Quando há uma mudança de tecnologia na rede, é responsabilidade do SRM gerar novos descritores de recursos para a visão do cliente: servidores pedem recursos de acordo com sua visão da rede e clientes recebem recursos de acordo com sua visão.

Mesmo que em cada visão, os descritores de recursos tenham o mesmo TAG de associação, cada descritor de recurso contém um flag indicando a qual visão (cliente ou servidor) pertence. Esta habilidade de descrever conexões através de múltiplas tecnologias de rede é uma característica única do DSM-CC.

3.5.5 Notificação U-U Usando Mensagens PassThru

Toda sessão DSM-CC é iniciada por um cliente. Entretanto, há casos em que um servidor (ou outro cliente) precisa avisar um cliente que ele deveria iniciar uma sessão. Por exemplo, para audio ou video conferência, uma mensagem DSM-CC U-N *PassThruReceipt* pode ser utilizada para indicar a um usuário que uma chamada requer que ele inicie uma sessão DSM-CC para recebê-la. Se a mensagem *PassThruReceipt* não for respondida, pode-se inferir que o usuário destino não está presente ou rejeitou a chamada.

3.6 Configurando um Cliente

Em um cenário típico, um cliente (set-top box), ao iniciar usaria um conjunto de mensagens U-N Config para fazer auto-configuração de acordo com a rede onde esteja ligado.

A configuração U-N pode ser iniciada pelo usuário (mensagens UNConfigRequest, UNConfigConfirm) ou pela rede (UNConfigIndication, UNConfigResponse) ou como consequência de um usuário ouvindo um canal de transmissão conhecido.

A configuração U-N do DSM-CC é uma área de protocolo independente, qualquer aplicação do usuário que necessite configuração inicial da rede, pode usar esta parte do DSM-CC.

3.7 Fazendo Download para um Cliente

O protocolo de download do DSM-CC é leve e rápido para fazer o download de dados e software do servidor para um cliente. Quando um cliente cria uma sessão com o servidor, DSM-CC permite, mas não exige, que o servidor faça o download para o cliente de todo o ambiente operacional (incluindo pilhas RPC).

Para garantir que os dados enviados para o usuário funcionarão naquele dispositivo, informação sobre compatibilidade pode ser necessária para o servidor antes que o envio inicie. DSM-CC provê descritores genéricos de compatibilidade que permitem que um usuário descreva a si mesmo para o servidor.

Uma operação de download completa, transfere uma "imagem", que é divida em um ou mais "módulos", que por sua vez são divididos em "pacotes". O tamanho do bloco é negociado para atingir os requisitos de eficiência e detecção efetiva de erros e ainda leva em conta o mecanismo de transporte sendo utilizado.

O que é úncio no protocolo de download do DSM-CC é que permite tanto um controle de fluxo tradicional (interativo) como permite download em broadcast. Mensagens de controle de download podem ser usadas para trocar informações sobre o processo de download antes da transferência.

Mensages de dados de download são utilizadas para transmitir os dados sendo enviados e, para download interativo, indicar a transmissão de módulos. Um servidor pode utilizar download em broadcast para enviar dados a vários clientes de uma só vez. Neste caso os descritores de compatibilidade dentro das mensagens de informação de download direcionam o cliente a um canal de transmissão específico a uma taxa compatível.

Considerando que há um carrosel de dados, o DSM-CC garante que haja informação suficiente nas mensagens de controle para que o cliente possa receber informação fora de ordem.

A habilidade de enviar software para o cliente elimina custos associados a controle de versão e updates periódicos para o software residente no cliente. Ideal para dispostivos clientes de baixo custo e também simplifica a manutenção de serviços provedores.

3.8 Carrosel de Dados e Carrosel de Objetos U-U

O carrosel de dados faz uso de mensagens de download sem controle de fluxo (broadcast) para prover transmissão periódica dos dados a um conjunto de clientes. Um mensagem de

controle de download fornece uma lista de módulos disponíveis a partir de um determinado carrosel de dados, assim, cada cliente pode identificar quais módulos deseja obter. Como em modo de broadcast não há um canal de retorno do cliente para o servidor, assume-se que todos os parâmetros de transferência tenham sido acordados a priori e todos os clientes sabem em qual canal de transmissão devem iniciar a escuta para mensagens de controle de download e de dados.

Enquanto imagens simples podem ser distribuídas usando serviços genéricos de carrosel de dados, um uso mais ambicioso é provar um ambiente onde verdadeiros objetos U-U por tras dos serviços de usuários sejam fisicamente entregues aos clientes. Para suportar isso, o DSM-CC especifica um carrosel de Objetos U-U e um protocolo de interoperabilidade de transmissão (BIOP). Este protocolo provê uma maneira padrão para empacotar em carroseis referências que descrevem localizações reais de representações de objetos dentro do mesmo canal de transmissão.

Os tipos de objetos U-U suportados atualmente, são: *Name Space*, *File* e *Stream*, permitindo que um cliente use o carrosel de objetos U-U para identificar *streams* que o cliente queira ver. BIOP é compatível com o protocolo RPC, assim, uma aplicação pode mesclar serviços em broadcast e interativos.

3.9 BFS

Do ponto de vista da entrega de dados, a infraestrutura de TVD provê uma rede de distribuição digital, protocolos de transporte de dados e terminais digitais (STBs) nos usuários. Assim, provê uma poderosa plataforma para a entrega de informação e serviços de dados. O middleware, ambiente de aplicação no usuário, provê uma vasta gama de serviços multimedia, como televisão digital, transmissão de dados, acesso a internet, entre outros.

Mas para que isso seja possível, é necessário que haja o suporte dos protocolos e de um sistema de transmissão de dados. Um dos middlewares mais importantes no mundo, o MHP utiliza o Carrosel de Objetos do DSM-CC para fazer a entrega das aplicações interativas e dados relacionados. O carrosel de objetos suporta o transporte de grupos estruturados de objetos, do servidor para os clientes através de objetos de diretório, objetos de arquivos e objetos stream. Todos os objetos são ciclicamente transmitidos. Dessa forma, o carrosel de objetos age como um gerenciador de sistemas de arquivos de rede, com a diferença de que pode haver uma maior latência.

O problema de utilizar o carrosel de objetos como o gestor do sistema de arquivos, é sua

difícil implementação em dispositivos com recursos limitados, como STBs. Ao contrário, o carrosel de dados é muito mais simples e pode ser implementado em STBs. Entretanto, é preciso uma nova solução para que o carrosel de dados possa atuar como um carrosel de objetos.

No middleware TVD, o BFS é um sistema de arquivos de rede, somente leitura, que além de ser utilizado para armazenar conteudo como video e audio, é atraves dele que as aplicações interativas podem ser enviadas para os clientes.

Como o software para TVD está cada vez mais complexo e avançado, tratar armazenamento através de endereços físicos de memória é complexo demais para atender a demanda. Assim, o gerenciamento de arquivos do sistema operacional foi trazido para a solução de middleware, possibilitando que o sistema de arquivos real seja desconhecido da aplicação.

Para prover um gerenciamento uniforme, o DSM-CC se comunica com o BFS, isolando o sistema de arquivos real. O BFS faz o mapeamento das estruturas DSM-CC para o tipo de estrutura definida para um arquivo no sistema de arquivos sendo utilizado. Assim, o middleware pode controlar qualquer sistema de arquivos da mesma maneira.

Esta técnica foi apresentada e utilizada em (ZHANG et al.,), onde foi apresentada uma proposta de BFS, compatível com VFS (mecanismo que isola o sistema de arquivos real do sistema, atuando como interface entre a aplicação e o sistema de arquivos) através do módulo *DSM File System*.

4 Projeto

O BFS implementado visa permitir a reconstrução do sistema de arquivos sendo transmitido pelo servidor no cliente. Para isso, o BFS utiliza mensagens DSM-CC para enviar, além dos dados, informações sobre a estrutura do sistema de arquivos. As mensagens utilizadas são *DownloadServerInitiate (DSI)*, *DownloadInfoIndication (DII)* e *DownloadDataBlock (DDB)*.

Através destas mensagens, o cliente irá recriar a estrutura em árvore do sistema de arquivos existente no servidor. Nas mensagens DSI e DII o cliente obtém informações de controle, nome do arquivo, diretório, "posição"do diretório na estrutura em árvore. Já os dados são transmitidos na mensagens DDB. Após receber todas as mensagens, o cliente possuirá uma cópia fiel do sistema de arquivos existente no servidor.

No BFS o descritor de nomes (name_descriptor), é usado para eviar o nome dos arquivos e diretórios e indicando o caminho absoluto e relativo, respectivamente: "/raiz/diretorio/"e "diretorio/". Este descritor é enviado através das mensagens DSI ou DII. A mensagem DDB é definida no DSM-CC para o envio de 1 byte de dados por vez, o BFS utiliza esta mensagem para enviar os dados dos arquivos byte a byte.

Syntax	No. of bytes	Value
name_descriptor () {		
descriptor_tag	1	0x02
descriptor_length	1	
for (i = 0; i < N; i++) {		
text_char	1	Name of the module, e.g. "index"
}		
}		

Figura 4.1: Sintaxe do descritor de nomes.

No EPOS, o BFS interage com o sistema de arquivos através de alguns métodos desenvolvidos no trabalho (MARCONDES, 2004), tanto no serivdor quanto no cliente.

4.1 Servidor

Na parte do servidor, que irá fazer a transmissão dos arquivos, o BFS precisa coordenar o tempo de vida dos arquivos no carrosel. Cada arquivo que é colocado no carrosel, deve representar uma cópia fiel do arquivo original, existente no sistema de arquivos em uso no servidor, encapsulado no formato de mensagens.

A interação com o sistema de arquivos se dá de forma simples. O nome do arquivo é colocado em um *namedescriptor* para ser transmitido através de uma mensagem DSI ou DII. O nome do arquivo é transportado como um array de caracteres chamado *text_char* dentro da estrutura name_descriptor.

Já os bytes que formam o arquivo, são colocados em mensagens DownloadDataBlock (DDB). Nesta estrutura, há um campo *blockDataByte*, que é um array de bytes, onde os dados são transmitidos. O tamanho da mensagem é definido pelo protocolo do DSM-CC, sendo necessário ao BFS apenas utilizar o tamanho previamente definido.

As mensagens contendo nome do arquivo e seus bytes são então colocadas de um módulo que será diponibilizado no carrosel para transmissão, por um tempo determinado, definido pela aplicação.

Para fazer o controle do tempo de vida de um módulo, é setado um alarme que dispara quando o tempo de vida estourar. Ao disparo do alarme o módulo é retirado do carrosel e as mensagens são destruídas.

Syntax	No. of bytes
DownloadDataBlock () {	
dsmccDownloadDataHeader()	
moduleId	2
moduleVersion	1
reserved	8 bits
blockNumber	16 bits
for (i=0; i < N; i++) {	
blockDataByte	1
}	
}	

Figura 4.2: Sintaxe DownloadDataBlock.

4.2 Cliente

Este é o lado que faz a recepção dos módulos. O BFS deve receber cada módulo e extrair as mensagens que compõem os arquivos. Ao extrair a mensagem DII ou DSI, um novo arquivo é criado no sistema de arquivos local, com o nome contido no campo *text_char*. Este arquivo é inicialmente vazio e é setado um flag para indicar que ainda não está completo e não deve ser utilizado.

A cada mensagem DDB extraída, ela é bufferizada. Antes de retirar o conteúdo do campo *blockDataByte* é preciso garantir que as mensagens estão em ordem, uma vez que podem ser recebidas fora de ordem. Quando o número de mensagens sequenciais for suficiente para que seu conteúdo tenha tamanho igual a um bloco de dados do sistema de arquivos, retira-se o conteúdo de cada uma, bufferizando, para que seja escrito um bloco de dados por vez no sistema de arquivos. Esta sequência se repete até que o arquivo esteja completo e ao final, o flag anteriormente setado para avisar que o arquivo estava incompleto, é invertido, indicando que o arquivo agora está completo.

Na mesma frequência que a thread periódica no servidor verifica o tempo de vida dos módulos, uma thread periódica verifica este mesmo tempo nos módulos que foram recebidos. Se um módulo estoura seu tempo de vida, e o arquivo correspondente ainda está marcado como incompleto, ele é excluído do sistema de arquivos, isto porque será impossível completar o arquivos, visto que ele já foi removido do carrosel e não está mais sendo transmitido. É gerada uma mensagem de erro informativa, para que a aplicação saiba que este arquivo não existirá.

Assim, através de mensagens, o BFS consegue reproduzir arquivos existentes no servidor, no sistema de arquivos do cliente, permitindo até que seja feita uma cópia fiel de toda estrutura de arquivos do servidor.

4.3 Uma outra abordagem

Foi pensada em uma abordagem diferente, que utilizaria menos memória no cliente.

No cliente, ao invés de utilizarmos um buffer para armazenar as mensagens DDB contendo os bytes dos arquivos, podemos criar um arquivo através do método *File(FileSystem * fs, char * name, u32 size)*, mas desta vez passamos como parâmetro para size o valor referente a quantidade de bytes transmitidos em cada DDB. Toda vez que receber um DDB, o BFS redimensiona o arquivo através do método *u32 resize(u32 bytes)* e utiliza o método *u32 append(u8 * buffer, u32 bytes)* para escrever os bytes de dados no final do arquivo, como parâmetro para buffer

passamos os bytes de dados que queremos escrever. Assim há um consumo menor de memória, permitindo que equipamentos com memória muito restrita ainda possam rodar um BFS.

Infelizmente, esta abordagem torna-se impraticável, pois para cada mensagem DDB, deveríamos escrever um bloco inteiro no sistema de arquivos, tornando-se uma abordagem cara. Esta abordagem só seria possível, quando em cada mensagem DDB, houvessem bytes de dados suficientes para preencher ao menos um bloco do sistema de arquivos.

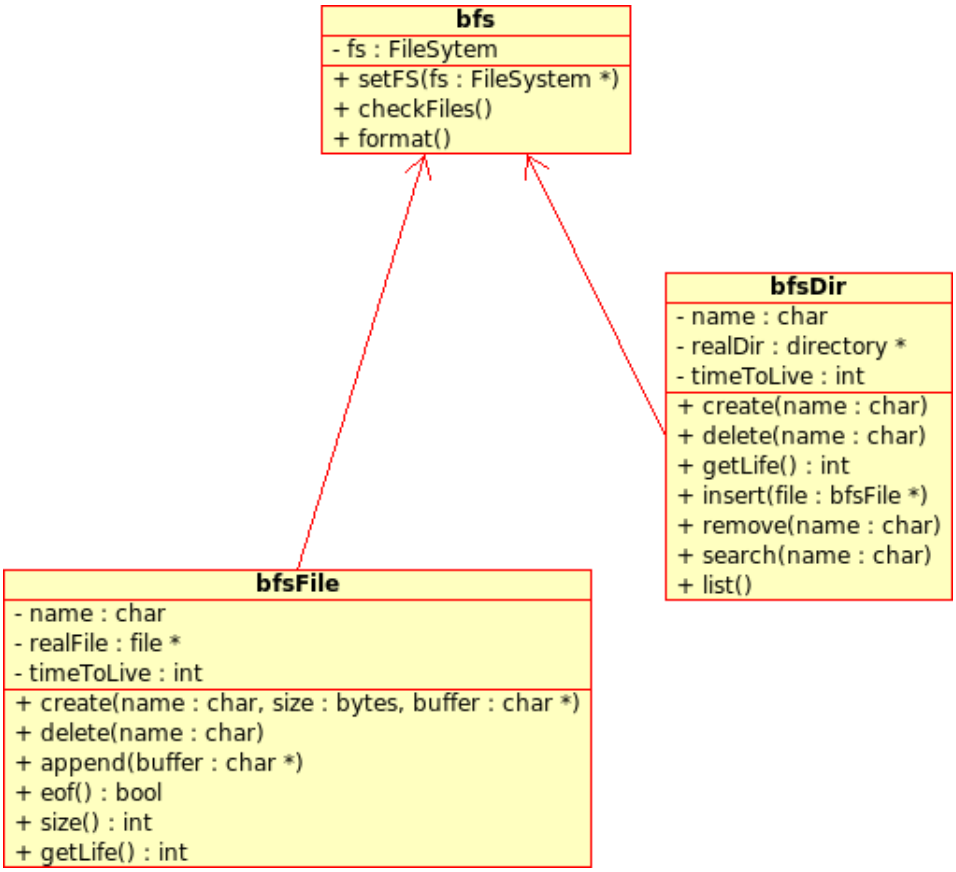

Figura 4.3: Diagrama de classes, representando o BFS.

5 *Conclusão*

Foi apresentado o conceito e a estrutura de um sistema de arquivos genérico e também o sistema de arquivos do EPOS, sistema operacional escolhido para implementação do BFS proposto neste trabalho. Também foram apresentados os conceitos do protocolo DSM-CC, que é utilizado mundialmente como protocolo de transporte na TV Digital. Este protocolo evoluiu de um "rascunho"em uma pequena parte do padrão MPEG-2. Também foi apresentado o carrosel de dados especificado no padrão DSM-CC, que aproveitando sua estrutura o BFS pode isolar a aplicação do sistema de arquivos utilizado pelo sistema operacional.

Com este trabalho, mostramos que é possível implementar um BFS em um sistema operacional eficaz, com baixo custo e nacional. Podendo ser utilizado equipamentos construídos a partir de sistemas simples e limitados em memória, processamento, armazenamento, como os STBs utilizados na TV digital. Pode ser utilizado tanto no servidor, fazendo a transmissão dos dados quanto no cliente que os recebe.

Referências Bibliográficas

BALABANIAN, V.; CASEY, L.; GREENE, N. Digital storage of media-command and control protocol applied to atm. In: *IEEE Journal on Selected Areas in Communications, pages 1162-1172, 1996*. [S.l.: s.n.].

FRÖHLICH, A. A. M. *Application-Oriented Operating Systems*. 1. ed. [S.l.]: GMD - Forschungszentrum Informationstechnik, 2001.

FRÖHLICH, A. A. M. *Epos - The Reference Manual*. [S.l.], 2002.

FRÖHLICH, A. A. M. *Notas de Aula - Dedicated Operational Systems*. [S.l.], 2003.

HIMONAS, S. D.; GELMAN, A. D. A digital storage media-command and control network. In: *Global Telecommunications Conference, pages 766-770, 1997*. [S.l.: s.n.].

JANG, S.-S. An implementation of dsm-cc user-to-network protocol in a davic vod system: The impress. In: *Protocols for Multimedia Systems - Multimedia Networking, pages 11-16, 1997*. [S.l.: s.n.].

MARCONDES, H. *Um Sistema de Arquivos para o EPOS*. [S.l.: s.n.], 2004.

MARCONDES, H. et al. Epos: Um sistema operacional portável para sistemas profundamente embarcados. In: *III Workshop de Sistemas Operacionais, Campo Grande, Brasil, 2006*. [S.l.: s.n.].

PARK, D.-H.; KU, T.-Y.; MOON, K.-D. Data broadcasting software architecture supporting real-time caching and monitoring in interactive tv. In: *Fourth Annual ACIS International Conference on Computer and Information Science, pages 593-597, 2005*. [S.l.: s.n.].

TANENBAUM, A. S.; WOODHULL, A. S. *Operational Systems - Design and Implementation*. 2. ed. [S.l.]: Prentice Hall, 1997.

ZHANG, H. et al. Design and implementation of broadcast file system based on dsm-cc data carousel protocol. In: *Consumer Electronics, IEEE Transactions, pages 929-933, 2004*. [S.l.: s.n.].