Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

"Projeto de um Servidor de Vídeo Sob Demanda Paralelo e Distribuído"

Carla Rodrigues Figueiredo Lara

Orientador: Hélio Crestana Guardia

São Carlos – SP
Abril/2003

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

L318ps

Lara, Carla Rodrigues Figueiredo.
Projeto de um servidor de vídeo sob demanda paralelo e distribuído / Carla Rodrigues Figueiredo Lara. -- São Carlos : UFSCar, 2003.
105 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2003.

1. Sistemas distribuídos. 2. Servidor de vídeo. 3. Vídeo sob demanda. 4. Arquivos paralelos. 5. MPEG. I. Título.

CDD: 005.43 (20ª)

Dedico este trabalho aos meus pais Raimundo e Maria das Graças, ao meu irmão Claudio, ao meu marido Gabriel e ao meu filho Daniel.

Agradecimentos

Em primeiro lugar à Deus, pela saúde, determinação e força para realizar este trabalho. Ao meu orientador, Prof. Dr. Hélio Crestana Guardia, sem o qual com certeza não conseguiria ter realizado este trabalho, sua confiança, apoio e amizade sempre me fortaleceram nos momentos mais difíceis. Sua presença diária acompanhando os meus passos fez muita diferença na realização do meu trabalho e na minha motivação.

Aos professores do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos. Aos funcionários do Departamento de Ciência da Computação, especialmente à Ofélia que me acolheu em sua casa e que cuidou de mim durante este período em São Carlos e à Vera que foi uma grande amiga nos bons momentos e também naqueles em que fiquei preocupada com o meu trabalho.

Aos vários amigos que fiz, especialmente aos companheiros do grupo de Sistemas Distribuídos e Redes, com os quais compartilhei o ambiente de trabalho e muitas coisas de minha vida. Especialmente ao Edilson e ao Eugeni que por trabalharem com assuntos relacionados ao meu, puderam me ajudar muito solucionando certas dúvidas.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro, com o qual pude me dedicar mais ao trabalho.

À família do meu marido, que me ajudou muito para que eu pudesse fazer este mestrado, compartilhando minhas angústias e temores, mudando as rotinas de suas vidas nos acolhendo para morar em sua casa e cuidando do meu filho com todo carinho, sem medir esforços.

Aos meus pais pelo apoio financeiro, pela voz sempre amiga ao telefone me incentivando e por serem compreensivos à minha falta de tempo para visitá-los.

E, por fim, ao meu marido Gabriel que entendeu a importância da realização deste trabalho para minha formação, que abriu mão de nossa convivência diária e sempre me apoiou em todas as minhas decisões. Pelo seu amor incondicional, que mesmo nos momentos mais difíceis falou mais alto. E ao Daniel, meu querido filho, que teve que aprender a ficar sem a minha presença e meus cuidados e mesmo assim não deixou de ser carinhoso e amoroso comigo.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho apresenta uma arquitetura para o desenvolvimento de um Servidor de Vídeo Sob Demanda (VoD) utilizando um Sistema de Arquivos Paralelos Distribuídos. No desenvolvimento deste trabalho vários tópicos relacionados à construção de servidores de VoD são considerados, entre eles: armazenamento e recuperação otimizados dos vídeos através da determinação de tamanhos apropriados para a unidade de distribuição e requisições de leitura e a coleta de informações dos discos e da rede para avaliação de desempenho do sistema proposto. As características do padrão utilizado para compressão dos vídeos, o MPEG, são estudadas e consideradas. Também são apresentadas discussões de como devem ser implementados mecanismos de *prefetching* em servidores de VoD. O conceito de paralelismo em aglomerados de computadores é explorado para que a distribuição eficiente dos dados forneça um tempo de resposta adequado e condizente com as exigências de QoS impostas pelos usuários da aplicação de VoD.

Abstract

This work presents an architecture for the development of a Video on Demand (Vod) Server using a Parallel Distributed File System. In the development of this work several aspects related to the project of VoD Servers are considered, including: optimized storing and retrieval of videos using appropriate striping unit and request sizes, and the monitoring of disk and network operation for the performance evaluation of the proposed system. The characteristics of the MPEG standard used for video compression are also studied and considered in the video transfers. We also discuss aspects related to the implementation of prefetching mechanisms in the VoD Server. The concept of cluster computing is also considered for efficient data distribution aiming at appropriate response time for satisfying the Quality of Service (QoS) require by the VoD users.

Capítulo 1 - Introdução

Neste capítulo são apresentados o contexto deste trabalho, os problemas identificados, as soluções já conhecidas e algumas das questões ainda não abordadas. Também são apresentadas a proposta deste trabalho de mestrado, suas motivações e principais objetivos.

1.1 - Introdução

Os recentes avanços na tecnologia de processamento e armazenamento tornaram possível a construção de aplicações que manipulem grandes volumes de dados. Um dos fatores determinantes neste processo foi o desenvolvimento de uma tecnologia de armazenamento baseada em discos rígidos cada vez mais baratos e com maior capacidade de armazenamento.

Em virtude destes avanços, implementações de aplicações multimídia passaram a ser mais viáveis. Multimídia envolve a combinação de texto, animações gráficas, vídeo e som, de forma que as informações são apresentadas de uma maneira mais interessante e fácil de compreender que apenas texto [FLY98].

As aplicações multimídia necessitam de mecanismos para armazenamento e distribuição de seus dados. Neste contexto surgem os servidores multimídia, que são os responsáveis por gerenciar o armazenamento, a recuperação e entrega eficiente de dados e também por tentar gerenciar da melhor forma possível os recursos do sistema (discos, rede, memória, etc).

A principal diferença entre os servidores utilizados para o armazenamento de dados comuns (textuais e numéricos) e os servidores multimídia é o fato de que dados multimídia impõem novas exigências, principalmente pelo fato de trabalharem com grandes volumes de dados e serem sensíveis ao tempo, sendo que atrasos na chegada dos dados devem ser minimizados e tratados.

Entre as diversas aplicações multimídia, existe um serviço denominado Vídeo sob Demanda. Um sistema de vídeo sob demanda (VoD - *Video on Demand*) é um sistema em que um servidor envia a um cliente dados de arquivos de vídeo solicitados, em tempo real, utilizando a infra-estrutura de uma rede de comunicação. Ou seja, um sistema de vídeo sob demanda consiste, em linhas gerais, de um conjunto de equipamentos e de software que permitem ao usuário, por intermédio de uma interface

instalada em um equipamento terminal (aparelho de televisão ou computador pessoal, por exemplo), enviar comandos a um equipamento servidor de vídeo, localizado nas instalações de uma empresa provedora do serviço, para estabelecer a programação que deseja assistir [LIM01].

As principais áreas de estudo relacionadas com sistemas de vídeo sob demanda são:

- **Aquisição** dos sinais de áudio e vídeo, o que envolve capturar, gravar e criar o conteúdo;
- **Compactação** do conteúdo adquirido, gerando um arquivo em um formato determinado, tentando eliminar informações que não são relevantes para a percepção humana, podendo assim gerar um arquivo menor e que ocupe menor largura de banda para a transmissão;
- **Armazenamento** adequado dos arquivos de vídeo, visando minimizar o espaço ocupado e facilitar a leitura dos dados;
- **Recuperação** dos arquivos de vídeo nos servidores, sendo que o principal objetivo é obter a informação o mais rápido possível;
- **Transmissão** dos vídeos através de uma rede, envolvendo o estudo de protocolos de rede específicos para vídeos e formas de aproveitar o mesmo fluxo para mais de um cliente;
- **Reprodução** dos vídeos nos clientes, abrangendo estudos para desenvolvimento de programas para exibição (*player*) mais eficientes e que utilizem menores tamanhos de *buffer*.

A implementação de servidores de vídeo sob demanda pode ser feita utilizando um único servidor ou diversos servidores trabalhando de maneira cooperativa numa rede, formando um aglomerado de computadores, também conhecido como *cluster*.

Quando apenas um único servidor é utilizado, o sistema de vídeo sob demanda fica limitado, tanto no que diz respeito à capacidade de processamento quanto para realizar operações de entrada/saída (E/S), ou seja, leitura e escrita dos dados.

Na tentativa de superar estas limitações uma outra abordagem para distribuir os vídeos foi desenvolvida, de forma que os vídeos são divididos em subconjuntos disjuntos e armazenados em servidores diferentes. Esta abordagem apresenta o

problema de falta de balanceamento de carga, que ocorre principalmente porque alguns vídeos são mais populares que os demais, fazendo com que alguns servidores fiquem sobrecarregados enquanto outros permanecem ociosos [LEE01].

Uma outra abordagem que tenta atender um grande número de usuários simultaneamente, é conhecida como *batching*. Nesses sistemas, as requisições de vídeos dos usuários permanecem numa fila até que a quantidade de requisições chegue a um número limite ou até que um contador de tempo expire. O objetivo neste caso é verificar a existência de solicitações para a mesma porção de um determinado vídeo, podendo assim atender diversas requisições através de um mesmo fluxo.

A eficiência dos sistemas que utilizam *batching* depende dos vídeos escolhidos, sendo melhor quando uma grande parte dos usuários assiste a uma pequena variedade dos filmes disponíveis. O desempenho depende também da escala do sistema, de forma que quanto maior o número de usuários simultâneos suportados, maiores podem ser as chances de que sejam escolhidos os mesmos vídeos por diferentes usuários. Mas o contrário também pode acontecer, quando muitos usuários poderão escolher vídeos diferentes.

Outra técnica que tem sido empregada na construção de servidores de VoD envolve o armazenamento utilizando sistemas de arquivos paralelos, denominada *striping,*. Sua forma de distribuir os vídeos consiste em dividir cada arquivo (vídeo) em unidades de um determinado tamanho e armazená-las em diferentes discos, que podem ser localizados em diferentes servidores.

A utilização de *striping* visa aumentar a eficiência de operações de E/S, pois diferentes partes de um arquivo podem ser acessadas ao mesmo tempo.

Usando controladores de discos especiais (RAID) [PAT88], muitos discos podem ser usados para tratar arquivos paralelos [CRO89] em uma única máquina. Em um aglomerado de estações, os discos presentes também podem ser combinados para conter arquivos paralelos. Neste caso, os servidores cooperam para armazenar e recuperar segmentos de arquivos paralelos em um sistema de arquivos local.

Arquivos paralelos tratam do particionamento e da distribuição dos dados entre diversos discos, buscando combinar suas operações simultâneas para obter maiores taxas de transferência e grande capacidade de armazenamento em operações de entrada e saída (E/S).

O mesmo conceito de paralelismo em aglomerados de estações pode ser explorado na implementação de servidores paralelos de Vídeo sob Demanda (VoD). Neste caso, a distribuição dos filmes visa prover um melhor tempo de resposta aos usuários e o suporte para um grande número de apresentações simultâneas, bem como o balanceamento de carga. Entretanto, essa distribuição é custosa, introduzindo novas complexidades à implementação destes servidores. É preciso gerenciar a distribuição dos dados, otimizar as operações de leitura e escrita, tratar as requisições que chegam dos clientes e realizar a junção dos dados que são recuperados dos discos num só fluxo de dados, procurando, ainda, balancear a carga entre os servidores.

Utilizando um Sistema de Arquivos Paralelos e Distribuídos (SAPD) esta complexidade gerada pela distribuição dos dados em diferentes servidores é minimizada. Um SAPD manipula arquivos de tal forma que estes sejam divididos, armazenados e lidos em diversos servidores interligados através de uma rede, de uma maneira mais amigável. Normalmente, cada um destes servidores possui o seu próprio disco local, que é compartilhado.

Exemplos de aplicações utilizando sistemas de arquivos paralelos distribuídos na implementação de servidores de VoD são vistos em sistemas como: *The SPIFFI Scalable Video-on-Demand System* [FRE95], *Design of the RIO (Randomized I/O) Storage Server* [SANT97], *Design and Implementation of the Parallel Multimedia File System Based on Message Distribution* [PAR00], *Design and Implementation of Linux Clustered VoD System* [PARK00], *GloVE: A Distributed Environment for Low Cost Scalable VoD Systems* [PIN02] e em *Design Considerations for the Symphony Integrated Multimedia File System* [SHE01].

Outros trabalhos discutem o uso da técnica de *striping* no projeto e na implementação de servidores VoD, mas não descrevem o uso de sistemas de arquivos paralelos [BOL96][LEE95][BER98][CAL00]. Além da discussão sobre armazenamento dos vídeos em sistema de VoD, outras propostas também são abordadas. Os principais tópicos apresentados são:

- tolerância a falhas [GOL01][LEE01][LEE97];
- confiabilidade, envolvendo redundância e paridade [GAF00][FRI96];
- controle de admissão [SHE01];
- Minimização de tráfego na rede [PIN02];
- Suporte a operações interativas [CHE96];

- Qualidade de Serviço (QoS) [TSA00][NG00].

Mecanismos de *prefetching* e a utilização da sintaxe da estrutura da padronização na qual os vídeos são codificados (MPEG) foram pouco abordados até o momento e são explorados neste trabalho.

Para tanto, o primeiro passo neste sentido foi estudar o funcionamento da padronização MPEG: sua sintaxe, sua estrutura interna, sua forma de sincronizar áudio e vídeo e o processo de decodificação. O passo seguinte foi utilizar estas informações de forma a propor novas abordagens no desenvolvimento de servidores de VoD que consigam melhorar o desempenho do sistema. Neste trabalho, quando é feita uma referência ao sistema de VoD estão sendo destacados todos os componentes do sistema: servidores, clientes e a rede. Quando se refere ao servidor de VoD estão sendo considerados apenas os servidores, que são os responsáveis pelo armazenamento e pela recuperação dos vídeos.

Utilizar o conhecimento da padronização MPEG na tentativa de melhorar o desempenho do sistema é apenas uma das propostas deste trabalho, sendo que outras discussões e propostas também são realizadas. Propõe-se que o controle de admissão e as garantias de QoS previstos utilizem dados coletados dinamicamente nos servidores e nos clientes, visando explorar ao máximo os recursos do sistema e assim poder maximizar o número de clientes simultâneos atendidos.

Outra questão bastante relevante na construção de servidores de VoD consiste em otimizar os acessos aos discos visando a maximizar o número de operações de entrada e saída, aumentando as taxas de operações dos discos e minimizando os tempos de resposta às requisições de leitura.

Neste sentido, este trabalho propõe a incorporação de mecanismos de *prefetching* ao servidor de VoD, que permitam explorar uma das principais características dos vídeos, que é o acesso seqüencial. Quando os vídeos estão sendo exibidos sem a realização de operações interativas (avanços, retrocessos), a aplicação do cliente realiza requisições de leitura seqüenciais que podem beneficiar-se da busca antecipada dos dados.

Duas formas de operação de *prefetching* são consideradas, uma utilizando *cache* nos servidores e outra *cache* nos próprios clientes. A primeira opção deve ser explorada quando a ocupação da rede está alta e não é vantajoso mandar os dados coletados

através da leitura antecipada imediatamente após a leitura do disco. Desta forma, propõe-se que quando a requisição de leitura vinda do cliente chegue aos servidores, estes poderão buscar os dados localmente no *cache* e enviá-los aos clientes. Uma operação de busca no *cache* é bem mais rápida que realizar uma leitura no disco. Ainda pode existir a opção de mandar estes dados previamente recuperados para os clientes caso a rede tenha sua ocupação reduzida. Neste caso, propõe-se que o cliente deve informar aos servidores as condições de ocupação da rede e a possibilidade de trocar a forma de operação do *prefetching*.

A segunda opção é prevista quando a ocupação da rede é baixa. Neste caso, os dados são recuperados e imediatamente enviados aos clientes.

Para controlar as informações sobre a ocupação da rede, taxa de operação dos discos, taxa de processamento dos dados nos clientes e outras informações de controle é proposta a utilização de um **Servidor de Controle**. Seu papel é ajustar o controle de admissão, parâmetros de QoS e *prefetching* em função das medidas coletadas.

Neste caso, os clientes devem enviar notificações para o **Servidor de Controle** em função de alterações significativas nas condições de rede e processamento locais.

Os vídeos que são utilizados no servidor de vídeo apresentado neste trabalho foram codificados utilizando a padronização MPEG-2, que será melhor detalhada no capítulo 3.

1.2 – Motivação e Objetivos

A capacidade de processamento dos computadores aumentou de forma significativa nas últimas décadas, enquanto os dispositivos responsáveis por realizar as operações de entrada e saída não acompanharam este aumento nas mesmas proporções. Este fato fez com que o "gargalo" das aplicações que necessitam de grande poder de processamento e manipulam grandes volumes de dados ou dados mantidos em unidades de armazenamento secundário deixasse de ser a capacidade de processamento e passasse a ser as operações de E/S.

A motivação para a realização deste trabalho surgiu a partir de estudos realizados sobre como prover operações de E/S de alto desempenho para aglomerados de computadores. Uma das formas que está sendo empregada atualmente é utilizar sistemas de arquivos paralelos, como por exemplo, o NPFS (*Network Parallel File System*) [GUA99].

Devido às exigências apresentadas como manipulação de grandes volumes de dados, necessidade de balanceamento de carga e freqüentes operações de leitura e escrita, apresentadas, aplicações multimídias e especialmente o serviço de vídeo sob demanda, este trabalho procurou estudar a adequação de SAP para o atendimento desses requisitos.

Outros trabalhos já fizeram discussões neste sentido, como apresentando anteriormente, mas muitas questões de projeto e implementação de servidores de VoD ainda estão em estudo e não foram exploradas.

De maneira geral, este trabalho procura determinar se as características impostas pelas informações multimídia podem ser atendidas de uma maneira natural pelos sistemas de arquivos paralelos, devido à correspondência existente entre as características da aplicação de VoD e as facilidades obtidas com o uso de sistemas de arquivos paralelos.

Sendo assim, os principais objetivos deste trabalho são estudar e discutir questões relacionadas ao projeto e implementação de servidores VoD que utilizam um sistema de arquivos paralelos. Mas, para tanto, é necessário entender como os diversos aspectos do sistema de arquivos paralelos estão relacionados com a aplicação, seja direta ou indiretamente. Isto envolve gerenciamento eficiente dos recursos do sistema, escolha de formas de armazenamento para os vídeos que consigam oferecer melhor desempenho no acesso aos dados, estudos sobre o padrão utilizado para codificação dos vídeos e, por fim, utilizar estes conhecimentos no desenvolvimento de propostas que mostrem os benefícios que podem ser obtidos através da adoção de certos procedimentos neste trabalho apresentados e validados através de testes experimentais.

1.3 – Estrutura da Dissertação

No capítulo 2 são apresentadas as principais características dos dados e servidores multimídia. Os principais aspectos envolvendo o projeto e implementação de servidores de vídeo sob demanda e os trabalhos relacionados são analisados.

No capítulo 3 é feito o estudo da padronização MPEG-2 que é largamente utilizada para a compressão e transmissão de vídeos.

O capítulo 4 descreve as características dos sistemas de arquivos paralelos de uma maneira geral e também o funcionamento do sistema de arquivos NPFS (*Network Parallel File System).*

No capítulo 5 são apresentadas a arquitetura e as questões de projeto relacionadas ao desenvolvimento do servidor de VoD proposto e implementado neste trabalho.

No capítulo 6 são exibidas as medidas coletadas através dos testes realizados e as discussões sobre os resultados obtidos.

O capítulo 7 apresenta as considerações finais, os trabalhos futuros e as conclusões.

Capítulo 2 – Servidores de VoD

Neste capítulo são apresentadas as classes de aplicações de mídia contínua, as questões de qualidade de serviço em servidores de vídeo sob demanda e outros tópicos relacionados ao projeto destes servidores, tais como: suporte a operações interativas, recuperação de falhas, implementação de *proxy* para remontagem dos fragmentos do vídeo, arquitetura do serviço de recuperação dos dados do disco, entre outros. Os trabalhos relacionados com estes tópicos também são apresentados.

2.1 – Multimídia e suas Aplicações

Não existe apenas uma definição de multimídia, mas é comum serem aceitas definições que relacionam multimídia com a combinação de vários tipos de mídias, tais como áudio, vídeo, animações gráficas e figuras. Devido a esta definição tão ampla, muitas aplicações podem ser definidas como "aplicações multimídias". Algumas das aplicações multimídias mais conhecidas são:

- **Vídeo sob Demanda:** é um serviço onde um ou mais servidores são responsáveis por armazenar e enviar aos seus clientes os vídeos solicitados sobre uma infra-estrutura de rede de comunicação.
- **Vídeo Conferência:** é uma forma de comunicação interativa que permite que duas ou mais pessoas que estejam em locais diferentes, se comuniquem com áudio e visualização de imagem em tempo real. Reuniões, cursos, conferências, debates, palestras são conduzidas como se todos os participantes estivessem juntos no mesmo local. Com os recursos da videoconferência, pode-se conversar com os participantes e, ao mesmo tempo visualizá-los na tela de um monitor (telão ou televisão, dependendo dos recursos utilizados), trocando informações como se fosse pessoalmente.
- **Ensino à distância:** é uma nova modalidade de ensino onde os estudantes não precisam estar no mesmo local para assistir as aulas. Pode combinar uma série de tipos de informações: apresentações multimídias armazenadas, aulas por vídeo conferência, aulas pela Internet. Ensino à distância envolve ensino não presencial e uso da multimídia.

- **Bibliotecas Digitais:** são uma extensão das bibliotecas convencionais; podem armazenar livros, imagens, áudio, vídeo, etc. A armazenagem em meios digitais custa bem menos e as informações em formato digital são mais fáceis de distribuir e de encontrar. O conteúdo digital tem ainda a vantagem de poder ser distribuído entre um número maior de pessoas, pela facilidade que tem de ser copiado.
- **Realidade Virtual:** é capaz de prover efeitos muito realísticos de visão e som e ao mesmo tempo permite ao usuário interagir com um mundo virtual. Por causa da interação que o usuário tem com o mundo virtual, os efeitos realísticos precisam ser criados em tempo real.
- **Telemedicina:** multimídia e telemedicina podem disponibilizar procedimentos médicos e de saúde de diversas formas. Informações digitais podem ser armazenadas centralmente e ficarem disponíveis simultaneamente para muitos locais. Médicos podem interagir com outros usando vídeo conferência, juntando assim os conhecimentos de muito peritos, podendo fazer um diagnóstico mais preciso. Multimídia pode ainda fornecer suporte para instruir pacientes e familiares.

Estes foram apenas alguns exemplos da grande variedade de aplicações que se beneficiam da utilização da multimídia, existem muitas outras formas de uso e a cada momentos novas aplicações são criadas.

2.2 – Classes de Aplicações de Mídia Contínua

De uma maneira bem ampla, as aplicações envolvendo conteúdo multimídia, também conhecidos como mídia contínua (*Continuous Media* – CM), podem ser divididas em três tipos:

- **Transmissão de áudio e vídeo pré-armazenados:** nestas aplicações, o conteúdo multimídia já foi codificado e armazenado. Exemplo: Vídeo sob demanda.
- **Transmissão de áudio e vídeo ao vivo:** neste tipo de aplicação, os dados multimídia não se encontram armazenados em um servidor, mas precisam ser capturados por equipamentos específicos e enviados através de uma rede. Exemplo: Rádio pela Internet ao vivo.

- **Transmissão de áudio e vídeo interativos em tempo real:** neste tipo de aplicação são possíveis a comunicação e a interação entre os usuários do sistema, sendo que as restrições de atraso são mais rigorosas. Exemplos: aplicações de voz e vídeo conferência.

2.3 – Características dos Servidores Multimídia

Servidores multimídia são constituídos basicamente por três componentes: um componente de **controle**, que é o responsável por atender as requisições dos clientes; um componente de **comunicação**, que move os dados do servidor para o cliente através da rede, e o **sistema de arquivos,** que gerencia o armazenamento e a recuperação de dados do disco [HAS98].

Os sistemas multimídia apresentam características particulares que são impostas aos seus servidores. Os principais desafios no projeto de sistemas multimídia são:

- Organização e gerenciamento dos dados armazenados, considerando a necessidade de altas taxas de transferências de dados e grande capacidade de armazenamento;
- Suporte a muitos clientes simultâneos;
- Garantia de QoS a todas as requisições dos clientes;
- Controle de admissão e em alguns casos, reserva de recursos em tempo real;
- Garantia de uma arquitetura escalável para o servidor;
- Suporte às operações de interação com o usuário, tais como pausa, avanço e retrocesso;
- Gerenciamento das informações, envolvendo indexação e recuperação de dados;
- Segurança em alguns tipos de aplicações, como por exemplo, aquelas disponíveis na Internet.

Para que uma transmissão multimídia seja realizada, alguns parâmetros devem ser observados; os principais deles são: a largura de banda (*bandwidth*), o atraso (*delay*) e a variação de atraso (*jitter*) [LU96].

Isto se deve principalmente ao fato de áudio e vídeo serem mídias contínuas, o que exige que as informações sejam apresentadas ao usuário em uma taxa constante.

Como normalmente os blocos de dados não chegam pela rede em intervalos constantes, é adotado o uso de *buffers* para um armazenamento local, possibilitando uma apresentação compatível com as necessidades dos usuários.

2.4 - Qualidade de Serviço em Sistemas Multimídia

Um problema significativo no projeto de um servidor de vídeo está na avaliação dinâmica dos seus recursos, para que este possa ser utilizado com eficiência máxima e, ao mesmo tempo, possa garantir a Qualidade de Serviço (QoS) aos clientes.

Os principais elementos necessários para prover garantias de QoS são:

- Mecanismos de especificação de QoS para as aplicações;
- Algoritmos de controle de admissão, para não afetar aplicações em andamento;
- Processos para negociação de QoS, para servir o maior número possível de aplicações;
- Mecanismos para alocação e agendamento de recursos;
- Políticas de policiamento do tráfego;
- Mecanismos de renegociação da QoS;
- Monitoramento da QoS fornecida às aplicações em andamento.

Para fornecer a QoS necessária, são utilizados algoritmos de controle de admissão para gerenciar o acesso ao servidor. Quando um novo cliente acessa o servidor, um módulo de controle de admissão verifica se existem recursos suficientes, tanto de disco como de rede, para satisfazer o pedido, mantendo um serviço ininterrupto aos demais clientes. Esses algoritmos podem ser determinísticos, que dão 100% de garantia de que a QoS vai ser mantida, ou estatísticos, que adotam estimativas para algumas variáveis do sistema; estes últimos não garantem a QoS, mas permitem uma maior utilização dos recursos do sistema.

Na maioria dos casos, o resultado da QoS é a qualidade percebida pelo usuário; em virtude disto, existem alguns trabalhos como o de [HAF95], que apresentam maneiras facilitadas para que o usuário informe os parâmetros desejados. Parâmetros como: resolução da imagem, qualidade de som para o áudio e tempo máximo previsto para a exibição, podem ser ajustados. Baseando-se nas escolhas do usuário, o sistema pode indicar se consegue ou não oferecer determinado vídeo exatamente com os parâmetros de QoS solicitados pelo usuário. Caso não seja possível, é comum apresentar

outras opções próximas dos valores desejados e neste caso, caberá ao usuário aceitar ou não alguma das configurações apresentadas, caracterizando um processo de negociação da qualidade do vídeo a ser exibido.

Outro fator importante em QoS é a capacidade do sistema em fazer uma renegociação mesmo depois da transmissão dos dados ter sido iniciada.

Um sistema VoD pode, ainda, ficar sobrecarregado; neste caso, uma solução é tentar adaptar-se para manter o nível de QoS o mais próximo do combinado. Quando isto ocorre, o servidor pode não conseguir garantir a qualidade do serviço e então existe a opção de garantia parcial. Existem, em geral, três níveis de garantias, que correspondem ao grau de compromisso dos servidores:

- *Hard* (Garantia Determinística);
- *Soft* (Garantia Estatística);
- *Best effort*.

Na garantia *hard,* a QoS especificada pelo usuário deve ser satisfeita totalmente. É a forma de garantia mais cara em termos de recursos, pois estes são reservados para o pior caso (estaticamente). Como desvantagem, tem-se que os recursos reservados, mesmo que não estejam sendo utilizados, não podem ser aproveitados por outras aplicações.

Na garantia *soft,* a QoS especificada pelo usuário deve ser satisfeita pelo menos para uma certa porcentagem dos parâmetros especificados. Normalmente, é a mais apropriada para mídia contínua e tem a vantagem de utilizar os recursos de forma mais eficiente (multiplexação estatística). O único problema é a difícil implementação devido à natureza dinâmica do tráfego e da utilização de recursos.

Na garantia *best effort,* nenhuma garantia é oferecida; ou seja, a aplicação é executada com os recursos disponíveis. Em termos de exigências do sistema é a implementação mais barata e, por isso, a maioria dos sistemas tradicionais opera dessa forma.

O tipo de garantia a ser utilizado vai depender muito do tipo de tráfego. Sistemas mais flexíveis permitem ao usuário determinar qual tipo de garantia este deseja utilizar. Existe ainda a possibilidade de uma mesma conexão utilizar níveis diferentes para cada parâmetro.

O usuário, ao fazer a especificação da qualidade que ele deseja numa apresentação de VoD, precisa levar em consideração que, quanto maior a qualidade exigida, maior o custo para o sistema realizá-lo.

Com o objetivo de facilitar as escolhas de parâmetros feitas pelos usuários, é desejável que se construam interfaces amigáveis onde o usuário possa entender claramente o que significa cada parâmetro ao qual ele está dando maior ou menor valor [HAF95].

2.4.1 – Qualidade de Serviço em Servidores de VoD

Os parâmetros de Qualidade de Serviço (QoS) desejáveis para os servidores de Vídeo sob Demanda possuem grande semelhança com aqueles apresentados pelos dados multimídia em geral, principalmente aqueles de natureza seqüencial e contínua (fluxos de áudio e vídeo, por exemplo). Alguns dos principais parâmetros a serem observados no projeto de um servidor VoD são:

- Armazenamento eficiente dos dados, de forma a facilitar a recuperação e a entrega dos mesmos aos clientes;
- Suporte ao maior número de clientes simultâneos possíveis;
- Dimensionamento adequado dos tamanhos dos *buffers* para que não exista desperdício e nem falte espaço para armazenamento;
- Exibição do vídeo de maneira ininterrupta e com qualidade satisfatória na apresentação ao usuário;
- Implementação de recursos que possibilitem ao usuário modificar a seqüência de apresentação do vídeo através de operações interativas;
- Suporte à realização de buscas de vídeos baseadas em seus conteúdos;
- Capacidade de oferecer diferentes níveis de QoS, dependendo das características de cada usuário e do hardware disponível.

2.5 - Servidor de Vídeo Sob Demanda (VoD)

O principal objetivo apresentado por sistemas de VoD é oferecer aos usuários acesso a vídeos/filmes armazenados em um servidor de vídeos [LEE97] [HAS98]. Para

tanto, várias abordagens são consideradas, cada uma apresentando suas vantagens e desvantagens.

Em [CHI97] é apresentada uma classificação para os sistemas de vídeo sob demanda existentes, organizando-os em três classes:

- ***Video-on-demand server*:** é a substituição da loja de aluguel de vídeos, operando numa WAN via cabo ou telefone. Os vídeos são longos (horas), o número de usuários simultâneos é enorme (milhares) e a latência pode ser relativamente grande (minutos). O usuário tem a possibilidade de escolher qual filme quer assistir a qualquer momento.
- ***LAN-Based enterprise video server:*** é um servidor estilo NFS (*Network File System*) que inclui suporte a arquivos de vídeo. Os vídeos são mais curtos e a latência esperada é baixa (segundos).
- ***Specialized video server:*** serve determinados mercados, tais como: hotéis e aviões. A carga máxima do sistema é fixa e a técnica *batching* é bastante utilizada.

A primeira classe de aplicações é bastante difundida nos Estados Unidos, normalmente é usada para fins comerciais. A segunda classe é mais utiliza no meio acadêmico, onde são estudados vários aspectos do serviço de VoD, tais como os que são apresentados neste trabalho.

A arquitetura geral de um sistema de vídeo sob demanda composto de vários servidores é apresentada na figura 1.

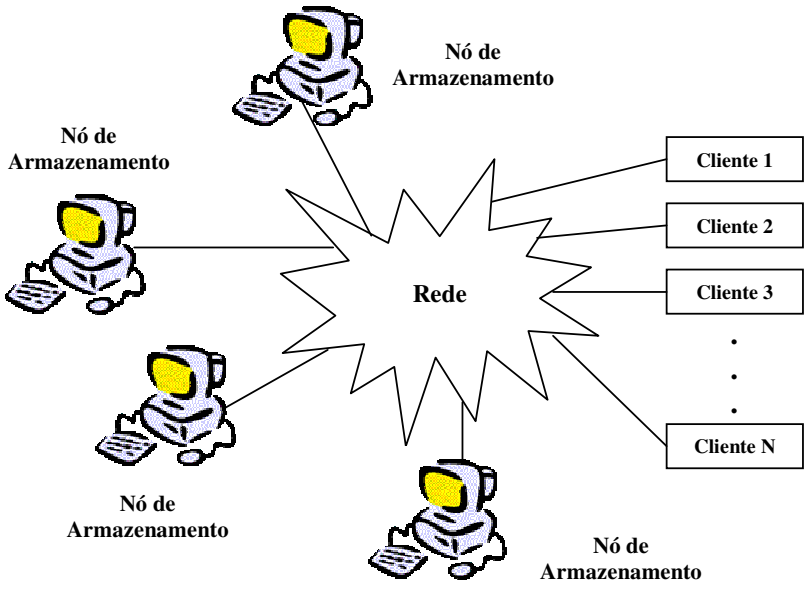

Figura 1: Arquitetura geral de um Sistema de VoD contendo vários Servidores

Um sistema de vídeo sob demanda funciona basicamente da seguinte forma: o cliente solicita ao servidor o vídeo desejado, o servidor recupera de seus discos a informação solicitada (parte do vídeo) e envia para o cliente.

Partindo deste ponto básico, novas funcionalidades e otimizações vêm sendo estudadas e adicionadas aos servidores propostos.

Os primeiros sistemas VoD criados utilizam uma arquitetura composta por apenas um servidor. Contudo, utilizar apenas um servidor traz grandes limitações ao sistema. A primeira delas é que a sua capacidade tanto de processamento quanto de realizar operações de E/S é limitada. Quando a demanda de carga no sistema alcança o limite da capacidade do servidor (tanto de processamento quanto de armazenamento), este precisa ser substituído por outro de maior capacidade, fazendo com que os dados sejam migrados para outros servidores; isto faz com que esta abordagem seja muito cara em termos financeiros, principalmente se tratando de dados multimídia, em que as capacidades de armazenamento e processamento exigidas são grandes.

Uma outra abordagem foi utilizada para tentar solucionar o problema citado anteriormente, dividindo os vídeos em subconjuntos disjuntos e armazenando-os em servidores diferentes.

Esta abordagem pode apresentar o problema de falta de balanceamento de carga, que ocorre principalmente porque alguns vídeos são mais populares que os demais, fazendo com que alguns servidores fiquem sobrecarregados enquanto outros permanecem ociosos [LEE01].

Uma segunda limitação apresentada por um sistema de VoD tradicional (composto por apenas um servidor) é não apresentar tolerância a falhas, pois quando o único servidor falha, todos os usuários do sistema são afetados. Para tentar contornar este problema, pode-se adotar um servidor de *backup* para melhorar a confiança do sistema ou colocar os vídeos em diferentes servidores. Contudo esta tentativa de resolver o problema de falta de tolerância a falhas pode ocorrer novamente o problema de falta de balanceamento de carga no sistema. Esta falta de balanceamento de carga surge em função do fato de alguns filmes serem mais populares que outros, como citado anteriormente.

Existe ainda uma outra abordagem que atende um grande número de usuários por vez, que é conhecida como *batching,* já discutida no capítulo 1. A eficiência dos

sistemas que utilizam *batching* depende do padrão de acesso aos vídeos e também do número de servidores e clientes do sistema. Esta técnica costuma ter um melhor aproveitamento quando uma grande parte dos usuários assiste a uma pequena quantidade de títulos [LEE98].

Por fim, existe ainda uma técnica utilizando sistemas de arquivos paralelos denominada *striping* para armazenamento de vídeos num sistema de VoD. Sua forma de distribuir os vídeos consiste em dividir cada arquivo (vídeo) em unidades de um determinado tamanho e armazená-las em diferentes discos, que podem ser de diferentes servidores.

Um dos primeiros trabalhos que utilizam esta forma de distribuir os vídeos é o "*Tiger Video Fileserver*" [BOL96]. Sua proposta é ser um servidor de arquivos distribuídos e com tolerância à falhas. Sua arquitetura é formada por um conjunto de computadores conectados por uma rede. Na arquitetura proposta cada servidor recebe o nome de "cubo". Cada servidor pode ser composto por vários discos, alguns discos do servidor são dedicados ao armazenamento dos dados no sistema de arquivos *Tiger* e um dos discos é usado para armazenar e realizar as tarefas do sistema operacional. Além dos cubos, o sistema *Tiger* possui uma máquina de controle central. Este controle central é usado como um ponto de contato para os clientes do sistema, como por exemplo, para um sistema de tempo global. Os dados dos vídeos não passam por este controle, o controle central não faz nenhuma ação até que o vídeo termine ou que o usuário requisite o fim da execução do vídeo. O *Tiger* possui um escalonador (*scheduler*) com o objetivo de explorar a distribuição dos dados nas operações de E/S. Uma de suas funções é atrasar levemente o atendimento de uma requisição de um determinado vídeo, de forma que esta requisição tenha um deslocamento em relação às outras em progresso, desta forma tenta minimizar a competição pelos recursos de *hardware* com os outros fluxos. A capacidade de trabalho do escalonador é dada em função das dimensões do sistema. Existe exatamente um *slot* (uma unidade operacional) a ser escalonado para cada fluxo simultâneo em potencial.

Quando um cliente faz uma requisição de serviço, a máquina de controle envia a requisição a um dos cubos acionando o primeiro bloco para ser enviado, enquanto adiciona este cliente no escalonador. Quando o disco que contém o primeiro bloco do vídeo processar o *slot* apropriado para o escalonamento os dados começam a ser

enviados ao cliente. Quando um cubo recebe uma nova requisição de serviço é selecionado um próximo *slot* ainda não usado.

O sistema de vídeos *Tiger* não usa informações sobre o conteúdo dos arquivos, apenas considera que todos os vídeos de um mesmo servidor possuem a mesma taxa de codificação (*bit rate*). O tamanho da *stripe* do *Tiger* varia de 64 KB to 1 MB e é o mesmo para cada arquivo, a escolha destes tamanhos foi baseada no tempo de execução do bloco.

As principais medidas coletadas pelo *Tiger* são: o número máximo de fluxos por disco (2 Gbyte) é de 4,7 e a latência inicial de cada fluxo, sendo que a menor obtida foi 1,3 segundos.

Pelo fato de ser um dos primeiros sistemas propostos não trata vários pontos importantes no projeto de servidores VoD, tais como: QoS, *prefetching*, controle de admissão, entre outros.

Uma proposta de implementação de servidor de vídeo utilizando o sistema de arquivos SPIFFI (*Scalable Parallel File System*) é feita por [FRE95]. Nele é apresentado um estudo, através de simulação, no qual são feitas variações nas configurações de hardware e testes usando algoritmos diferentes com o objetivo de verificar o efeito da variação de alguns parâmetros e alternativas de projeto para sistemas de VoD, determinando desta forma como um servidor deve ser projetado e configurado para minimizar custos e maximizar desempenho. São analisadas principalmente questões de *buffers* e escalonamento de discos.

Este trabalho critica o fato de que a maioria dos estudos analíticos usa algoritmos simples (por exemplo: *round-robin*) para gerenciamento dos discos, podendo assim fazer uma análise muito otimista ou pessimista e desta forma não explora ao máximo as possibilidades de utilização do *hardware*. Em contra partida, afirma que, utilizando um modelo de simulação detalhado, o desempenho do sistema pode ser corretamente previsto através do uso de modelos realísticos para os discos, a rede e CPU. Desta forma, o número máximo de usuários simultâneos que cada configuração do servidor pode suportar pode ser determinado e os efeitos da variação de cada parâmetro de projeto podem ser corretamente medidos. A principal métrica utilizada neste trabalho para verificar o desempenho de uma configuração particular é o número máximo de usuários que a configuração pode suportar enquanto provê um serviço de

vídeo contínuo, livre de interrupções (*glitch,* falha). Este método para avaliar sistemas de VoD é compatível com o método adotado em muitos estudos analíticos [FRE95].

O SPIFFI foi desenvolvido e implementado para funcionar em máquinas do tipo *Intel Paragon* e detalhes sobre o sistema de arquivos podem ser encontrados em [FRE96].

Entre os resultados obtidos por este trabalho, está a demonstração de que a distribuição dos vídeos usando *striping* eleva substancialmente o desempenho do sistema quando comparado à uma configuração que não utiliza *striping*. Para obter o número máximos clientes suportados a metodologia utilizada consiste em aumentar o número de clientes até que o número de falhas (*glitch*) comece a não ser zero. Para um tamanho de *stripe unit* de 512 KB, 1 GB de memória por servidor (processador) e 4 discos por servidor obteve um número máximo de clientes igual a 220. Sendo que o algoritmo usado para substituição de páginas na memória é o LRU, que substitui as páginas menos recentemente utilizadas.

São apresentados também resultados de testes relacionando algoritmos para gerenciamento de discos (*disk scheduling*) e tamanhos de *stripe unit*, que variaram de 128 a 1024 Kbytes. Os algoritmos considerados foram o *elevator*, o GSS (*Group Sweeping scheme*), *round-robin* e algoritmos de tempo real. O funcionamento destes algoritmos é explicado em [FRE95]. Os testes mostram que o desempenho dos algoritmos *elevator* e os de tempo real são muito parecidos para todos os tamanhos de *stripe unit*. O melhor desempenho foi obtido usando um algoritmo de tempo real e um tamanho de *stripe unit* de 512 KB, podendo suportar 225 clientes. Quando o tamanho da *stripe unit* diminui, os discos gastam mais tempo para achar um *byte* a ser lido e o desempenho decai. Desta forma, para a *stripe unit* de tamanho 1024 KB o desempenho cai substancialmente porque o tempo para completar cada leitura no disco é relativamente maior que a quantidade de vídeo armazenada em cada cliente. Assim, a escolha da *stripe unit* deve considerar a minimização de buscas de dados nos discos, o tempo de serviço para E/S e a memória disponível no cliente.

O sistema VIOLA (*Video on Local-Area-Networks*) [LEE95] utiliza o conceito de *"Redundant Array of Inexpensive Servers (RAIS)* [WON97]. Assim como a arquitetura RAID [PAT88], o sistema RAIS tenta dar suporte a escalabilidade e compartilhamento de carga entre os servidores. A distribuição dos vídeos entre os discos do sistema, assim como em outros trabalhos estudados, visa obter um

balanceamento de carga entre os discos e um sistema escalável. Utiliza paridade para obter suporte à falhas.

Para a implementação do VIOLA foram desenvolvidos protocolos para comunicação entre servidores e clientes, uma vez que um sistema de arquivos paralelos não é usado para tratar a distribuição dos seus vídeos. O VIOLA não faz duplicação de dados, mas utiliza paridade para tolerância à falhas.

A distribuição dos dados é feita num nível de objeto, ou seja, a distribuição é realizada em cada objeto individualmente, tal como uma página de texto, uma imagem, um áudio ou vídeo. Isto mostra que o sistema não foi projetado exclusivamente para um tipo de mídia. O esquema proposto para realizar a distribuição dos dados recebeu o nome de DOS (*Dynamic Object Striping*). No DOS todo armazenamento é dividido em pequenas *stripe units* (ex.: 1KB), denominadas micro-blocos, e os objetos podem ser representados como um múltiplo do micro-bloco. Usando *stripe units* pequenas, espera-se resolver o problema da fragmentação interna. Usando o DOS cada objeto possui três atributos: {START_UNIT, UNIT_SIZE, REDUNDANCY}. A política do DOS é criada quando um objeto é criado e armazenado nos discos dos servidores. Esta política pode ser trocada se necessário, mas os dados precisam ser redistribuídos entre os servidores. O primeiro atributo, START_UNIT guarda a *stripe unit* inicial de um objeto particular, UNIT_SIZE armazena o número de micro-blocos do objeto e REDUNDANCY armazena o nível de redundância empregado no armazenamento de cada objeto. Um valor maior em REDUNDANCY significa que um nível maior de redundância está sendo utilizado. Através destes parâmetros é possível armazenar objetos de tipos diferentes de maneiras diferentes.

Para medir a capacidade do sistema VIOLA foram usados um, dois e quatro servidores. Os vídeos foram distribuídos ao longo de todos os servidores e a capacidade de entregar dados aos clientes (*throughput*) foi medida para diferentes cargas de clientes. Os resultados mostram que para uma mesma quantidade de clientes, a carga por servidor é reduzida pela metade para dois servidores e de um quarto para quatro servidores. É demonstrado também que quando ocorre uma falha em um servidor o sistema é capaz de manter contínuo o serviço ao cliente.

Uma nova proposta para armazenamento é feita no servidor de armazenamento RIO (*Randomized I/O*) [SANT97]. O RIO foi desenvolvido para armazenamento de dados multimídia, portanto não é específico para vídeos, podendo armazenar conteúdos

de aplicações que não sejam puramente seqüenciais, como por exemplo, aplicações de realidade virtual. De uma maneira geral, conclui que a alocação aleatória é a melhor forma de distribuir os dados entre os discos. Para garantir o balanceamento de carga entre os servidores faz replicação parcial. Utiliza múltiplas *threads*, uma para cada novo cliente aceito.

Os resultados experimentais apresentados pelo RIO mostram que é possível obter uma baixa probabilidade de não cumprimento de prazos de entrega de dados e ao mesmo tempo uma alta utilização do disco, entre 70% a 95% da taxa efetiva. Usando replicação parcial a taxa de utilização do disco é na faixa de 60% a 80%, quando replicação total é usada permite uma quase completa utilização da largura de banda dos discos. Não possui mecanismos de *prefetching*.

O sistema **Glove** (*Global Vídeo Environment*) [PIN02] tem como proposta a implementação de um *cache* cooperativo, usando uma abordagem ponto-a-ponto e o sistema de armazenamento do RIO. A idéia básica é que os clientes ativos cooperem para criar um *cache* compartilhado de vídeos que é a primeira fonte de conteúdo de vídeos para subseqüentes requisições dos clientes. Desta forma, a largura de banda não é limitada ao número de usuários simultâneos. Uma vez estando no cache de vídeo cooperativo (CVC – *Cooperative Video Cache*) um vídeo pode ser diretamente transmitido, sem interferência do servidor VoD. É uma técnica que faz reuso dos dados já buscados dos discos dos servidores. Uma desvantagem desta abordagem é que a latência inicial é relativamente alta. Os testes mostraram uma redução próxima de 90% na utilização da largura de banda de um canal para entrega de um vídeo popular com intervalo de chegada entre as requisições de até 20 segundos, usando 16 MB de *buffer* no cliente. Os resultados mostraram ainda que, usando apenas um canal para um vídeo popular é possível manter uma taxa de requisições de 30 requisições/min, mesmo para um tamanho de *buffer* pequeno de 4 MB.

Uma outra arquitetura denominada DAVID (***D**istributed **A**rchitecture VIDEO on **D**emand*) é proposta em [CAL00]. Sua principal diferença em relação às demais é dividir os servidores em três grupos: Servidor de Acesso (*Access Server* - AS), Servidor Organizador (*Schedule Server* – SS) e Servidor de Disco (*Disk Server* – DS). O Servidor de Acesso é uma máquina responsável pelo acesso ao serviço de VoD e que gerencia apenas as requisições dos usuários. Os SS têm como principal papel gerenciar

o fluxo de dados durante a sessão inteira, e os DS são um grande subconjunto de servidores usando RAID e a técnica de *striping*.

A avaliação do sistema DAVID é feita usando um parâmetro req/sec (Rs) que denota a quantidade total de dados multimídia (expressa em unidades de 1,5 Mbps) requisitada por todos os usuários conectados ao sistema. Utiliza também o *throughput* (Tp) que significa a quantidade total de dados multimídia entregue pelo sistema, também expressa em Mbps. É feito então um relacionamento entre estas duas medidas Rs e Tp, de forma que a medida que aumenta o número de requisições aumenta também o *throughput*. No caso ideal, o valor de Tp deve ser igual ao valor de Rs (Tp=Rs), de forma que a curva do gráfico deveria ser uma diagonal. Os testes experimentais mostraram que em todas as configurações com mais de um DS, o sistema consegue ficar próximo da curva ideal quando está operando com até 40 fluxos ativos. Depois deste valor, o desempenho degrada rapidamente. Usando apenas um DS a carga de trabalho máxima (Rs) que pode ser processado em uma operação ótima é de 16 req/sec.

O achatamento apresentado pela curva que relaciona Tp com Rs ocorre essencialmente por causa de dois fatores: primeiro, pela capacidade disponível do *switch* (100 Mbps, na teoria) que impõe um limite no *throughput* do sistema. Os testes mostraram que para três ou quatros DS o *throughput* máximo obtido não ultrapassa 90 Mbps, que é bem próximo da capacidade do *switch.* Segundo, a exaustão dos recursos da CPU do SS quando apenas um é utilizado, contribui para a distância entre as curvas ideal e real.

Em [BER98] uma arquitetura diferente das já apresentadas é discutida. Esta proposta utiliza dois tipos de componentes, denominados *Drives* e *Merging Servers*. Os *Drives* são os responsáveis pelo armazenamento dos vídeos de maneira distribuída e em unidades de tamanho diferentes. Os *Drives* não se comunicam diretamente com os clientes. Os *Merging Servers* possuem duas funções: juntar e ordenar, se preciso, os fluxos que chegam dos diversos *Drives* em um único fluxo que é entregue ao cliente e armazenar os dados já recuperados. Um ponto vulnerável desta arquitetura é justamente os *Merging Servers* que podem falhar comprometendo várias seções e mesmo funcionando corretamente podem ser o "gargalo" deste sistema. Segundo descrito em [LEE98] o ideal é que os dados sejam remontados nos próprios clientes, as abordagens existentes para remontar os dados (*proxy*) ainda serão apresentadas neste capítulo. O

tamanho de *stripe unit* é variável, devido ao fato das estruturas dos MPEG (quadros) também ser variáveis.

Para distribuir os dados no DAVID foi utilizada uma granularidade grossa, com *stripe unit* variando de 64 KB a 1 MB. Estes tamanhos foram escolhidos para melhorar o desempenho através da redução do tráfego gerado pela busca dos dados nos discos (*seek*). Para medir o número máximo de clientes simultâneos suportados pelo sistema, foram usados além dos clientes normais um outro tipo de cliente denominado *dummy*. Este tipo de cliente não faz a decodificação e nem a exibição do vídeo, apenas gera tráfego no sistema, de forma que se pode obter o ponto onde ocorre a saturação do serviço. Resultados não são apresentados pelo trabalho.

Em [PAR00] é proposta a utilização de um sistema de arquivos denominado PMFS (*Parallel Multimedia File System*). Em suas discussões destaca as vantagens e desvantagens geradas pela utilização de um servidor para controlar as solicitações dos clientes, usando uma arquitetura com o nome de **arquitetura em duas camadas**. Esta arquitetura consiste de um servidor de controle e um grupo de servidores de armazenamento. Sua vantagem é prover um fácil balanceamento de carga entre os discos. Contudo, existe o tráfego gerado pela comunicação entre o servidor de controle e os servidores de armazenamento.

Para reduzir este tráfego, o sistema proposto transmite os dados dos servidores de armazenamento diretamente aos clientes, sem a intervenção do servidor de controle. O papel do servidor de controle em relação à entrega de dados se refere apenas à questão de gerenciamento de informações dos arquivos, sendo no trabalho referenciado como **metadados**. Na arquitetura proposta existem dois tipos de processos (*daemon*): o tipo que gerencia metadados (MGR – *Manager Daemon*) e o que gerencia operações de entrada e saída dos discos (IOD – *IO Daemon*). Apenas um MGR que fica no servidor de controle gerencia os metadados de todos os objetos de mídia e em cada servidor de armazenamento existe um IOD.

O sistema de vídeo PMFS utiliza uma técnica no trabalho denominada de distribuição de mensagens para realizar as operações de E/S. Sendo que estas operações são disponibilizadas através de uma biblioteca. Mais de uma política de disposição dos dados são utilizadas. Sua principal medida de desempenho é baseada na taxa de prazos de entrega não cumpridos (*deadline miss ratio*). Em termos de resultado, obteve melhor

desempenho (*throughput*) utilizando nas operações de leitura um bloco de tamanho 128 Kbytes.

Uma proposta de servidor de VoD utilizando um sistema de arquivos denominado **CrownFS** é apresentada em [PARK00]. A distribuição dos dados é semelhante às outras já apresentadas: os vídeos são distribuídos entre diversos discos e quando recuperados são enviados aos clientes. O CrownFS é organizado por dois tipos de processos: processo de trabalho (M-worker) e processo de administração do servidor (S-worker). As informações sobre a distribuição dos vídeos (metadados) são armazenadas em um servidor administrativo. O M-worker gerencia as requisições dos clientes e transfere as requisições de leitura/escrita para os S-workes. Os S-workers armazenam os arquivos de forma distribuída, e são responsáveis em realizar a operações desejada (leitura/escrita) e transferir os dados para o M-worker. Os S-workers também gerenciam *cache* e os discos, para maximizar o desempenho das operações de E/S. O M-worker tem a responsabilidade de montar as informações dos vídeos MPEG para formar o fluxo ordenado. Dependendo de como cada pedaço dos dados é coletado, a escalabilidade do CrownFS pode ser melhor ou pior. Devido ao fato de estarem logicamente separados, cada nó do sistema pode conter um M-worker e um S-worker ou somente um deles.

Para medir o desempenho foram usados arquivos com tamanhos de 16MB, 32MB, 64MB, 128MB e 256 MB, e *stripe unit* de tamanhos 64KB, 256 KB, 1 MB e 4 MB. O número máximo de clientes suportados por nó é 35. Quando o número de clientes é maior que este valor, começam a ocorrer variações do atraso na entrega dos dados (*jitter*). Usando arquivos MPEG-1 com taxa de codificação de 1,5 Mbps e *Fast Ethernet*, cada nó pode suportar 8 clientes normais e mais 30 clientes *dummy* (não exibem o vídeo). Considerando um sistema com 4 nós, verificou-se ser possível servir 150 clientes sem *jitter*.

Um dos sistemas de arquivos multimídia mais completos em termos de recursos disponíveis é o ***Symphony*** [SHE98][SHE01]. Entre esses recursos destacam-se: escanolador de discos que suporta requisições em tempo real e também que não sejam de tempo real, gerenciador de armazenamento que permite um controle na forma de dispor as informações, uma camada de tolerância a falhas, dois níveis de estruturas para metadados, o que facilita trabalhar com tipos de dados específicos, mecanismo para reserva de recursos e algoritmos para garantias de QoS, suporte tanto para a arquitetura

client-pull quanto *server-push*, suporte para utilização de tamanhos fixo e variável para os tamanhos de bloco de armazenamento e técnicas de *caching*.

Para suportar os múltiplos tipos de dados, o *Symphony* utiliza uma arquitetura em duas camadas. A camada mais baixa implementa um conjunto de mecanismos independente do tipo do dado, que fornece as funcionalidades de sistema de arquivos. A camada superior (camada específica para cada tipo de dados) consiste de um conjunto de módulos, um para cada tipo de dados, através da utilização dos mecanismos fornecidos pela camada inferior implementa técnicas específicas (forma de armazenamento, recuperação de falhas, *cache*) para cada tipo de dado. Esta arquitetura em duas camadas separa claramente os mecanismos que são independentes dos tipos de dados e as políticas que são específicas para cada tipo de dado. Isto permite que módulos possam ser adicionados para um novo tipo de dado ou que modificações sejam feitas em módulos já existentes.

No trabalho descrevendo o *Symphony* são apresentados alguns gráficos relacionando alguns parâmetros tais como tamanho da *stripe unit* e *throughput* por disco, tamanho da *stripe unit* e desbalanceamento de carga.

As maiores contribuições apresentadas no artigo [SHE01] são em relação às lições aprendidas no desenvolvimento do sistema. As principais vantagens e benefícios apresentados são:

- a **modularidade** que simplifica o projeto de um sistema de arquivos, no início o Symphony era monolítico e com o aumento da complexidade da implementação do sistema de arquivos, foi preciso rever o projeto e adotar o modelo de duas camadas.
- o projeto do sistema de arquivos para suporte a **múltiplas classes de serviço** é importante porque utilizando apenas uma classe de serviço (ex.: melhor esforço) que é otimizada para apenas um critério de desempenho, fica difícil atender a diversidade de carga de trabalho impostas pelos diversos tipos de aplicações.
- a utilização de **políticas específicas** para cada tipo de dado traz ganho de desempenho, por exemplo, usando o escalonador do *Symphony* foi possível obter um tempo de resposta 2,5 vezes melhor que utilizando algoritmos mais genéricos.

São apresentadas também as desvantagens com relação às escolhas de projeto feitas no *Symphony*, por exemplo, o fato da **generalidade** de alguns mecanismos comprometer o desempenho do sistema. Um caso onde isto pode ocorrer é no armazenamento, visto que o gerenciador de armazenamento permite que qualquer tamanho de bloco de dados seja utilizado para um arquivo, o que pode gerar fragmentação nos discos. Se os tamanhos de blocos forem restringidos a um conjunto de tamanhos é possível reduzir o problema de fragmentação do disco sem qualquer perda significativa de desempenho e flexibilidade. Outro problema é a utilização de **políticas de escalonamento complexas.** Devido ao fato de ter que multiplexar os recursos entre classes de aplicações diferentes, o sistema de arquivos explora formas complexas de escalonamento e políticas de gerenciamento de recursos. Isto faz com o projeto do sistema de arquivos fique muito complexo. E o último problema apresentado é a carência de *benchmarks* (medidas de desempenho) para sistemas de arquivos multimídia. Os existentes foram desenvolvidos para uma carga de trabalho textual e são inadequados para avaliar os sistemas de arquivos multimídia.

2.6 – Questões de Projeto em Servidores VoD

Nesta sessão são examinadas as principais questões relacionadas ao projeto de servidores VoD. São técnicas utilizadas no armazenamento, na leitura de dados dos discos, na recuperação de falhas, no controle de admissão e na remontagem dos vídeos.

2.6.1– Técnicas de Armazenamento

Para a determinação de qual a melhor política para armazenar os vídeos, muitos fatores devem ser considerados. Devido ao grande volume de dados que precisam ser armazenados nos servidores de VoD normalmente vários discos são utilizados, formando os *arrays* de discos. Para maximizar a capacidade de recuperar os dados (*throughput)*, a carga nos discos deve ser balanceada.

Para realizar o balanceamento de carga nos discos duas opções são disponibilizadas: técnicas de balanceamento estáticas e dinâmicas. A combinação dessas técnicas também pode ser utilizada.

No balanceamento de carga estático, o servidor precisa selecionar para cada fluxo de mídia (arquivo de vídeo) três parâmetros apropriados: o tamanho da *stripe unit*, o grau de *striping* e a quantidade de replicação a ser feita.

A escolha do tamanho da *stripe unit* depende do tipo da aplicação que usará os dados armazenados. Em sistemas de arquivos convencionais, o tamanho da *stripe unit* normalmente é escolhido para minimizar o tempo médio de resposta e maximizar o *throughput*. Para os servidores multimídia é escolhido um tamanho de *stripe unit* que minimize a variação no tempo de resposta e ao mesmo tempo maximize o *throughput*. A utilização de um tamanho pequeno para a *stripe unit* resulta em uma distribuição uniforme de carga entre os discos e diminui a variação no tempo de resposta, mas também aumenta o tráfego gerado pelas buscas no disco (*seek*) e a latência rotacional e por isto o *throughput* diminui. Por outro lado, utilizar grandes tamanhos para a *stripe unit* aumenta o *throughput*, mas pode causar o balanceamento de carga e aumentar a variação no tempo de resposta [LEE01].

Para maximizar o número de clientes servidos ao mesmo tempo, o servidor precisa selecionar um tamanho de *stripe unit* que seja capaz de se beneficiar do melhor de cada abordagem.

Os vídeos de um servidor de VoD podem ser distribuídos usando todos os servidores ou somente um subconjunto de servidores do sistema de arquivos paralelos. Enquanto a primeira abordagem é chamada de *wide striping,* a segunda é conhecida como *narrow striping*.

Cada uma destas abordagens apresenta vantagens e desvantagens. A principal desvantagem de se utilizar *wide striping* é que, se um servidor falha, todas as apresentações correntes são afetadas. Atingir o maior grau de balanceamento de carga é sem dúvida o maior benefício desta política.

Usando *narrow striping*, cada vídeo pode ser particionado entre diferentes grupos de servidores do sistema de arquivos. Deste modo, esta abordagem pode causar problemas de balanceamento de carga pelo fato de os vídeos não possuírem a mesma popularidade; com isso, enquanto alguns servidores podem permanecer sobrecarregados, enquanto outros podem ficar ociosos.

O grau de *striping* para fluxos de mídias é dependente do número de discos no sistema. Em sistemas contendo relativamente poucos discos, os fluxos de mídias devem ser distribuídos entre todos os discos (*wide striping*), produzindo balanceamento de

carga e maximizando *o throughput*. Em sistemas compostos por muitos discos, para maximizar o *throughput*, o servidor precisa distribuir os fluxos de mídias entre subconjuntos de discos e replicar os dados para obter balanceamento de carga [SHE01].

A quantidade de replicação depende da popularidade do vídeo, ou seja, depende da quantidade de vezes que um vídeo é requisitado em relação aos outros e também do total de espaço disponível para armazenamento.

Independentemente do grau de distribuição utilizado, a carga nos discos em um servidor multimídia pode ficar desbalanceada, mesmo que temporariamente, por causa do padrão de acesso das requisições que chegam. Para suavizar este problema, servidores multimídia podem utilizar técnicas de balanceamento de carga dinâmico. Se múltiplas réplicas do vídeo requisitado são armazenadas nos discos do sistema, então o servidor pode tentar balancear a carga nos discos servindo a requisição através do disco com menor carga e que contenha uma réplica. Adicionalmente, o servidor pode explorar a seqüencialidade na recuperação do áudio e vídeo através de *prefetching*, suavizando a variação de carga imposta por um vídeo.

2.6.2– Arquitetura de Recuperação (Leitura) dos Vídeos

Duas abordagens são comumente utilizadas para organizar a recuperação (leitura do disco) dos fluxos multimídia: *server-push* ou *client-pull*.

Quando *server-push* é a técnica escolhida, normalmente, os múltiplos fluxos são servidos em ciclos. Durante cada ciclo, o servidor recupera um número fixo de unidades da mídia por fluxo, por exemplo, cada cliente recebe cinco quadros por ciclo.

Para garantir a continuidade do vídeo, o número de unidades da mídia acessadas para cada fluxo deve ser o suficiente para sustentar a taxa de exibição e o tempo de serviço não deve exceder a duração de um ciclo. O tempo de serviço é o tempo total gasto para recuperar as unidades da mídia durante um ciclo.

Na arquitetura *client-pull*, o servidor recupera unidades da mídia para o cliente somente em resposta a uma requisição explícita de leitura. Enquanto o servidor precisa manter estados de clientes na abordagem *server-push*, a abordagem *client-pull* não precisa guardar este tipo de informação.

A primeira vantagem da arquitetura *server-push* é que ela otimiza a utilização dos discos do sistema nas operações de leitura e escrita. Além disto, fica mais fácil para

os servidores atender as garantias de QoS feitas aos clientes, principalmente porque o controle da recuperação dos dados é totalmente feito pelo servidor.

A abordagem *client-pull* possui a vantagem de não interferir no sistema de arquivos e também, caso o cliente queira interromper o recebimento do fluxo de dados, é necessário apenas parar com as requisições.

Em geral, os servidores multimídia usam a arquitetura *server-push* devido à natureza periódica e seqüencial dos dados utilizados na exibição das mídias contínuas. Mas nada impede que um servidor multimídia adote a outra abordagem.

Mesmo usando a abordagem *client-pull* é interessante que o servidor seja capaz de enviar dados aos clientes antes mesmo que estes façam a solicitação dos mesmos, uma vez que é fácil determinar qual será o próximo bloco de dados que o cliente irá solicitar, devido à natureza seqüencial apresentada por mídias continuas. Isto pode ser implementado na forma de *prefetching*.

2.6.3– Controle de Admissão

Para manter a qualidade dos serviços que são oferecidos, em ambas as arquiteturas, *client-pull* e *server-push*, é preciso utilizar algoritmos de controle de admissão para avaliar se os recursos necessários para atender uma nova requisição não irão afetar as exigências de tempo real dos outros fluxos que já começaram a ser servidos. Um algoritmo de controle de admissão determina os recursos exigidos por uma requisição através da estimativa da taxa de *bits* e do tempo de acesso aos discos. Dependendo da natureza desta estimativa, os algoritmos de controle de admissão podem ser classificados em três categorias:

- **Determinísticos**: os algoritmos deste tipo fazem estimativas considerando o pior caso para a taxa de *bits* e tempo de acesso ao disco para atender a requisição do usuário, são usados quando os clientes não toleram nenhuma violação de prazo para entrega dos dados;
- **Estatísticos**: os algoritmos deste tipo usam probabilidade para estimar a taxa de *bits* e as variações no tempo de acesso ao disco para atender a requisição de um novo usuário e garantem que os prazos de entregas dos dados podem ser cumpridos com uma certa probabilidade.

- **Baseados em Medidas**: este tipo de algoritmo controla as variações já sofridas na taxa de *bits* e no tempo de acesso ao disco e usa estas informações como indicador de futuras variações. Este tipo de controle aumenta a utilização do disco, mas fornece baixa garantia de atendimento aos prazos de entrega.

2.6.4 – Tolerância a Falhas

Sistemas compostos por muitos discos são altamente susceptíveis a falhas, e a possibilidade de ocorrer uma falha aumenta na medida em o número de discos vai crescendo. É desejável que o sistema de arquivos adotado pelo servidor de VoD utilize alguma técnica para recuperação de falhas que consiga manter o serviço ininterrupto aos clientes no momento em que uma falha ocorre.

A tarefa de recuperação de falhas de disco envolve duas tarefas: **reconstrução em tempo real (*on-line*)**, que envolve a recuperação do dado armazenado em um disco que falhou em resposta a uma requisição explícita a este dado; e **reconstrução total** (***rebuild***), que envolve criar uma réplica exata do disco que falhou em um disco substituto.

Em sistemas de arquivos desenvolvidos para dados textuais, as principais técnicas usadas para correção de erros são: replicação de dados em discos separados e códigos de correção de erro (como codificação de paridade). Mas estas abordagens podem aumentar significantemente a carga nos outros discos que não falharem caso ocorra uma falha, podendo resultar em violações de prazos na exibição dos vídeos. Para prevenir que isto ocorra quando estas técnicas convencionais de tolerância a falhas estão sendo utilizadas, os servidores precisam operar com baixas taxas de utilização dos discos, mesmo durante o período livre de falhas.

Servidores multimídia podem explorar as características dos fluxos de mídia para contornar este problema. Primeiro, os servidores podem explorar a seqüencialidade que ocorre no acesso ao áudio e ao vídeo para reduzir o tráfego de uma recuperação *on-line*. Calculando a informação de paridade sobre uma seqüência de blocos pertencentes a um mesmo fluxo de dados (vídeo), o servidor pode assegurar que os blocos da mídia recuperados para cobrir um bloco armazenado em um disco que falhou, serão requisitados pelo cliente em um futuro bem próximo. Através do armazenamento em *buffer* destes blocos e do uso destes dados para atender as requisições é possível

minimizar o tráfego gerado por um processo de recuperação de falha sem parar o sistema.

Segundo, uma vez que percepção humana é tolerante a pequenas distorções na exibição do vídeo, o servidor multimídia pode reduzir o tráfego gerado pela recuperação de uma falha através da utilização de uma aproximação do dado armazenado no disco que falhou, aproveitando as redundâncias espacial e temporal apresentadas pelos vídeos. Este método integra o processo de decodificação da mídia com o processo de recuperação de falhas e desta forma, distribui a tarefa de recuperação de falhas entre os clientes.

Segundo [LEE01] a maioria dos estudos no campo de tolerância a falhas em servidores de vídeo sob demanda é feita no nível do disco (no armazenamento); somente em alguns trabalhos encontrou os autores investigando tolerância a falhas no nível de servidores em sistemas de vídeo paralelo [BOL96][TEW96][WON97].

Através da leitura dos trabalhos apontados por [LEE01] foi possível observar que em [BOL96], a solução usada no servidor de vídeos *Tiger* para promover confiabilidade ao sistema consiste em manter duas cópias de cada *stripe* armazenada em servidores diferentes. Adicionalmente, propõe o uso de *declustering* (que é a determinação de quantos e quais discos serão utilizados no armazenamento) para organizar a disposição das *stripes*. A principal desvantagem do espelhamento (uma cópia dos dados em um outro disco) dos dados é a exigência do dobro de armazenamento.

Outro estudo é apresentado por [TEW96] em que diferentes arquiteturas são exploradas para a distribuição dos vídeos. Neste sistema, existem dois tipos de nós servidores, *back-end,* para armazenamento dos vídeos e *front-end,* para entrega dos dados. Os vídeos são divididos e armazenados entre os nós do tipo *back-end* e os nós do tipo *front-end* montam os dados recuperados dos discos (nós *back-end*) para serem entregues aos clientes.

Adicionalmente, outro trabalho [WON97] também propõe soluções para a questão de falhas em servidores de VoD. Seu enfoque é no uso de unidades de redundância (paridade) que são distribuídas ao longo dos diversos discos.

2.6.5– Suporte a Operações Interativas

Quanto ao grau de interatividade que oferecem aos usuários, as apresentações de vídeo sob demanda podem ser divididas em dois grupos: o primeiro grupo não apresenta opções que permitam modificações na apresentação corrente, e o segundo grupo que disponibiliza ao usuário a possibilidade de interferir no andamento da apresentação. Neste caso, devem ser consideradas as questões de QoS relacionadas à satisfação dos usuários, e por isso são disponibilizadas, através do servidor de VoD, certas funcionalidades ao dispositivo responsável pela exibição do vídeo.

A maioria dos projetos na área de sistemas de entrega de vídeos é focada no serviço de prover entrega contínua de vídeos armazenados em um servidor para um cliente através de uma rede. Apenas nos últimos anos tem sido dada ênfase a aplicações como vídeo na Internet, ensino à distância e suporte a operações interativas. Estas novas aplicações possuem um maior grau de exigência no que se refere à capacidade de processamento, motivando o uso de múltiplos servidores que podem, por exemplo, trabalhar de maneira cooperativa formando um aglomerado de estações de trabalho.

As interações dos usuários com a apresentação são realizadas através de operações como o avanço rápido (*fast-forward*), o retrocesso (*fast-backward*), pausas (*pause*) e o reinício da apresentação (*resume*).

Em [WU97], duas políticas são apresentadas: a primeira garante transmissão normal e operações interativas separadamente. Quando uma transmissão normal solicita uma operação destas, ela é tratada com uma nova requisição. Isto é chamado de política em dois estágios (*two-steps).* Neste caso, uma operação interativa pode ser rejeitada ainda que a apresentação normal tenha sido aceita.

A segunda política é chamada um estágio *(one-step).* Ela admite a transmissão normal e operações interativas simultaneamente. Isto é, quando uma requisição é aceita, ela pode trocar de apresentação normal para operação interativa, ou vice-versa, sem precisar de uma nova admissão.

Em [VER96] o padrão de compressão utilizado para codificar os vídeos é o MPEG-1. Como originalmente o mesmo não prevê em sua implementação a operação de *rewind* (retrocesso), foram propostas modificações no esquema de representação dos quadros, com a finalidade de dar suporte a tal operação. Para isto, é utilizado um arquivo extra, codificado especificamente para este fim.

Outra técnica também utilizada para dar suporte às operações interativas é apresentada em [CHE96]. Neste trabalho, utiliza-se a conversão dos quadros que formam o fluxo de dados. Desta forma, a dependência existente entre os quadros é eliminada.

Ainda em [CHE96], é apresentada uma abordagem denominada multi-resolução, que é uma outra forma freqüentemente utilizada para implementar operações interativas. Esta técnica também possibilita a criação de sistemas onde os usuários possuam características diferentes de capacidade tanto em termos de *software* quanto de *hardware*.

Armazenar múltiplas resoluções de uma seqüência de vídeo, contendo somente o mínimo de informações necessárias para a entrega no cliente, visa possibilitar que dependendo das exigências e capacidades do cliente, o fluxo de dados possa ser adaptado.

Existe, por fim, a opção de deixar a responsabilidade em coordenar as operações interativas para a aplicação do usuário, mais especificamente, para o visualizador (*player*).

2.6.6 - Reconstrução do Vídeo Utilizando Proxy

Quando um sistema de arquivos paralelos é utilizado, a aplicação do cliente recebe dados de diferentes servidores, é necessário um módulo capaz de receber estes dados, juntá-los de maneira ordenada e repassá-los ao visualizador (*player),* que será o responsável pela exibição do vídeo ao usuário. Este módulo, denominado *proxy* em [LEE98], refere-se à parte do sistema responsável por juntar os diversos fluxos de dados. Para tanto, o *proxy* deve conhecer a configuração do sistema, o que implica em saber os endereços de servidores, a localização dos dados, a política de *striping*, entre outras informações relevantes.

Existem três maneiras diferentes para se implementar *proxy*: no servidor (***proxy-at-server***), em um computador independente (***independent proxy***) e no computador do cliente (***proxy-at-client***).

Na arquitetura que implementa o *proxy* no servidor, cada servidor fornece o serviço de armazenamento e de *proxy*. Como cada servidor atende diversos clientes simultaneamente, o *proxy* também precisa fazer o mesmo. A vantagem de se usar esta

abordagem é que o trabalho realizado pelo *proxy* é totalmente transparente ao cliente. Possui, entretanto, a desvantagem de gerar um grande tráfego de processamento e de comunicação.

Na arquitetura que adota um computador independente, os servidores estão conectados aos *proxies* através de uma rede. Desta forma, continua sendo transparente ao usuário o trabalho do *proxy*, e ainda tem-se a vantagem de que a interferência entre os dois processos é eliminada. As implementações tanto do servidor quanto do *proxy* podem ser mais simples, mas isto requer mais processamento e largura de banda que a abordagem anterior. Ainda são necessários, *hardware* adicionais e *links* de rede extras para conectar os servidores aos *proxies*.

A terceira abordagem integra o *proxy* ao cliente e pode ser implementada através de um módulo de software no sistema operacional ou na própria aplicação. Neste caso, o *proxy* requisita os dados ao servidor que os envia diretamente ao cliente. Depois de processados, os dados vão direto para a aplicação.

Fazendo uma análise numérica dos dados, [LEE98] demonstrou que nesta última abordagem é necessária apenas a metade das transferências de dados usadas nas outras duas políticas, e não é necessário um *hardware* extra para o *proxy*. Ainda existe outra grande vantagem nesta proposta: se um *proxy* falha, estando nas duas abordagens anteriores, todos os clientes atendidos por ele padecem com esta falha, pois por razões econômicas geralmente um proxy atende diversos clientes simultaneamente; na abordagem que implementa o *proxy* no cliente, se um *proxy* falha, apenas um cliente pode ser afetado, pois cada *proxy* atende apenas a um cliente.

Um ponto negativo é a perda da transparência apresentada pelas outras abordagens, mas isto não chega a ser tão importante se comparado com os benefícios apresentados. Além do mais, o usuário final não consegue perceber a diferença entre a remontagem do vídeo ser local ou não, o que faz desta decisão uma questão conceitual de projeto.

Capítulo 3 – A padronização MPEG

Neste capítulo é feito um estudo sobre a padronização de áudio e vídeo MPEG. São apresentadas as características e o conteúdo interno do fluxo gerado pela codificação, a estrutura da camada de vídeo e as principais diferenças entre os três tipos de quadros existentes.

3.1 - A Padronização MPEG

Com a popularização de arquivos multimídia, principalmente os arquivos de áudio e vídeo, surgiu um novo campo de pesquisa nesta área, que envolve o estudo de formas eficientes de geração, compressão, armazenamento, recuperação e reprodução deste tipo específico de informação.

Neste contexto, um grupo que merece destaque é sem dúvidas o *Moving Picture Experts Group* (Grupo de Especialistas em Imagens com Movimento), ou simplesmente grupo MPEG, que trabalhou na criação de padrões tais como MPEG-1, MPEG-2, MPEG-4, MPEG-7, MPEG-21. Entre esses padrões, podem ser destacados o MPEG-1 e o MPEG-2, que tratam áudio e vídeo.

O MPEG-1 foi a primeira padronização criada por este grupo e trata do armazenamento de áudio e vídeo em mídias digitais, opera em taxas da ordem de 1,5 Mbps. Sua principal aplicação é a construção de CD-i (*Compact Disc interactive*), que são sistemas de CD-ROM que permitem que o usuário escolha qual caminho percorrer dentro do aplicativo nele armazenado.

O MPEG-2 é uma extensão do MPEG-1 e foi criado para ser utilizado em aplicações que exigem transmissão de dados, tais como serviço de transmissão por satélite, serviços de TV interativa e Internet.

As padronizações seguintes são o MPEG-4 que trabalha com transmissões de baixas taxas de bits e o MPEG-7 que é usado com propósitos de indexação, como por exemplo, para buscas baseadas em conteúdo. Outras padronizações estão sendo desenvolvidas e para obter informações sobre as padronizações acima citadas visitar a página do grupo em [MPE03].

Provavelmente, uma das vantagens mais significativas do MPEG é o conceito de codificação predicativa, que basicamente significa calcular uma imagem mudou em relação às anteriores e transmitir um código que indique esta diferença.

Um dos principais objetivos de se utilizar compressão de vídeos é a redução da largura de banda e da necessidade de armazenamento requeridas.

Os vídeos utilizados no servidor de vídeo proposto neste trabalho foram codificados utilizando as padronizações MPEG-1 e MPEG-2, que são as mais utilizadas para vídeo. As principais diferenças entre as duas padronizações são:

- No MPEG-1 existem apenas três tipos fluxos de dados: áudio, vídeo e o sistema, que interage com os outros dois. No MPEG-2 é possível utilizar um número ilimitado de fluxo de dados, abrindo assim um leque de possibilidades de utilização do sistema;
- O MPEG-1 trabalha com quatro tipos de quadros: I, P, B, D, enquanto o MPEG-2 não utiliza quadros do tipo D (DC-*Coded*). Os quadros do tipo D são usados para possibilitar imagens de baixa resolução em operações de avanço ou retrocesso rápido. Os demais quadros são idênticos nas duas padronizações e suas características são apresentadas ainda neste capítulo.
- O padrão MPEG-2 passou também a permitir a codificação de imagens entrelaçadas, de forma que a sua visualização pudesse ser acelerada. Isto não ocorre no MPEG-1 que possui apenas um nível de resolução (320x240 *pixels*), o MPEG-2 suporta 4 níveis: baixa resolução (352x240 *pixels*), resolução média (720x480 *pixels*), resolução alta-1440 (1440x1152 *pixels*) e resolução alta (1920x1080 *pixels*).

Por tratar-se da padronização mais recente e usual, este trabalho descreve com mais detalhes as características do MPEG-2.

3.2 – As Normas do Padrão MPEG-2

MPEG-2 é um padrão de compressão para áudio e vídeo. Foi desenvolvido para dar suporte a diversas aplicações de diferentes naturezas. É capaz de suportar taxas bem

superiores ao padrão anterior, o MPEG-1, podendo variar entre 3 a 30 Mbps, dependendo das características da aplicação.

Para atender à complexidade dos problemas associados à representação de sinais audiovisuais, a norma MPEG-2 foi organizada em várias partes, das quais as principais são:

- MPEG-2 *Systems* (ISO/IEC 13818-1) - Define a estrutura de multiplexação entre áudio, vídeo e dados, assim como a sincronização entre o codificador e o decodificador [ISO94].
- MPEG-2 *Video* (ISO/IEC 13818-2) - Define o sinal (representação codificada) do vídeo comprimido [ISO95].
- MPEG-2 *Audio* (ISO/IEC 13818-3) - Define o sinal do áudio comprimido.

A norma MPEG-2 *Video* define a sintaxe e a semântica do fluxo de dados do vídeo comprimido, especificando a estrutura deste fluxo que deve ser compatível com a norma, obedecendo a certos parâmetros tanto para a geração quanto para a obtenção das imagens no processo de decodificação.

Algoritmos e técnicas de codificação não são especificados pelas normas da padronização, mas pelos desenvolvedores ou fabricantes dos codificadores; a única regra imposta pela padronização é que o fluxo de dados resultante seja compatível com a norma. Desta forma, um vídeo pode ser codificado de várias maneiras e mesmo assim pertencer à padronização MPEG. Esta flexibilidade facilita o uso da padronização MPEG por aplicações com diferentes características e exigências.

3.3 – Quadros MPEG-2

A codificação MPEG-2 é baseada em estruturas denominadas quadros, que podem ser de três tipos: I, P e B.

Os quadros do tipo I (*Intra-codec*) são figuras codificadas usando o formato JPEG [MAT94]. Quadros deste tipo contêm imagens inteiras e são codificados/ decodificados de modo independente dos outros quadros e são usados como referência para os outros dois tipos de quadros. Quadros I são utilizados como ponto de acesso aleatório para começar a decodificar um fluxo de dados em andamento e para

sincronização após erros de transmissão. Oferecem um grau de compressão moderado em relação à imagem que representam.

Os quadros do tipo P (*Predictive*) são codificados usando predição simples (dependente diretamente de apenas um quadro), baseando-se na estimativa/compensação de movimento dos quadros anteriores do tipo I ou P. Em termos genéricos, pode-se dizer que a informação contida em um quadro tipo P consiste na diferença entre a imagem a codificar (imagem atual) e a imagem anterior (a última do tipo I ou P). Este tipo de quadro não pode ser codificado/decodificado de modo independente, pois depende da imagem de referência anterior e, por fim, pode servir de referência para outros quadros do tipo P e B. Na média, o tamanho dos quadros do tipo P é metade de um quadro I.

Usando uma predição composta (dependente diretamente de dois quadros), são formadas as imagens do tipo B (*Bi-directionally predicted)*. A estimativa/compensação realizada é baseada em duas imagens de referência: na anterior, que pode ser do tipo I ou P, e na seguinte que também pode ser I ou P. As imagens do tipo B não são usadas como referência por outras imagens. Quadros do tipo B tipo produzem o maior grau de compressão, mas possuem a "desvantagem" de possuir dependência dupla. Na média quadros do tipo B possuem um tamanho quatro vezes menor que um quadro do tipo I.

A interdependência existente entre os quadros MPEG-2 é ilustrada na figura 2.

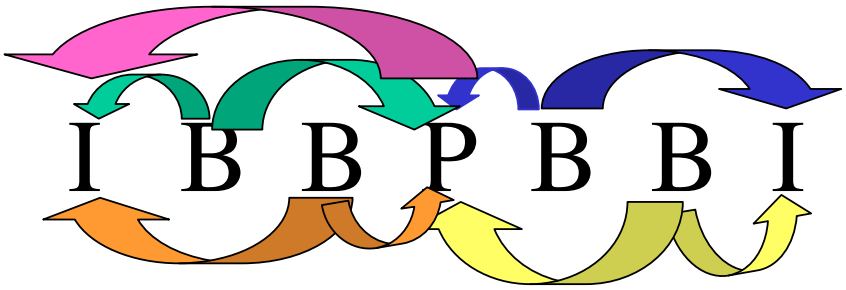

Figura 2: Interdependência entre os Quadros MPEG-2

3.3.1 – Transmissão e ordem de exibição

A dependência temporal de imagens futuras implica em reordenação da seqüência original das imagens. Isto se deve ao fato de os quadros B dependerem diretamente de dois outros quadros I ou P, um passado e um futuro. Evidentemente, como a informação futura ainda tem que ser transmitida, não estará disponível ao

decodificador. Para solucionar este problema a padronização MPEG envia os quadros numa ordem **errada**. Os quadros são enviados fora de seqüência e armazenados temporariamente de forma que um quadro futuro já estará no local de decodificação antes do início da decodificação do quadro B. A figura 3 mostra uma seqüência codificada e como ela é transmitida.

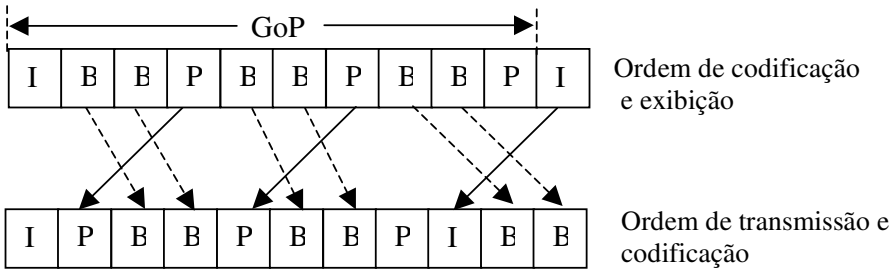

Figura 3: Mudanças na Seqüência de Quadros

A reordenação de quadros requer memória adicional no codificador e no decodificador, além de gerar atrasos em ambos no processo de retomar a ordem correta.

O número de quadros B codificados entre os quadros I e P deve ser restrito para reduzir gastos com processamento e minimizar os atrasos, uma vez que atrasos não são desejados.

3.4 - Sintaxe e Semântica do Vídeo Codificado

Um vídeo codificado consiste de um conjunto de fluxos de dados (*bitstreams*), denominados camadas. Se existe apenas uma camada, o vídeo codificado é denominado **fluxo de vídeo não escalável.** Se existem duas ou mais camadas, o vídeo é denominado **hierarquicamente escalável** [ISO95].

A primeira camada (camada base) sempre pode ser decodificada independentemente. As outras camadas (camadas de realce) só podem ser decodificadas juntamente com todas as outras camadas inferiores, iniciando pela camada base.

Um fluxo de dados codificado é organizado como uma seqüência, que é iniciada com um cabeçalho e termina com um código de fim de seqüência. Entre estas estruturas de início e fim deve existir uma seqüência de imagens, que podem ser adicionalmente subdivididas em grupos (*group of pictures* - GoP). Esta subdivisão opcional permite que certos parâmetros sejam modificados na seqüência, aplicando as alterações em grupos

de imagens. Outros parâmetros são definidos no cabeçalho da seqüência e aplicados a todas as imagens da seqüência. Um GoP sempre começa com um quadro do tipo I e contêm, normalmente, entre 12 e 15 quadros dos tipos P e B; seu tamanho é bastante flexível, mas é comum a utilização de um padrão fixo (ex.: I BB P BB P BB P BB) para o GoP . Um dos motivos da utilização de GoP's é fato que a codificação predicativa utilizada pelo MPEG não pode ser utilizada indefinidamente, pois isto tende a propagar erros. Como um novo GoP começa sempre com um quadro I, os erros não são propagados além de um GoP.

Alguns codificadores mais avançados tentam otimizar a disposição dos três tipos de quadros de acordo com as características locais da seqüência no contexto das características mais globais [ISO02], ou seja, a disposição dos quadros tenta facilitar o processo de decodificação. Uma situação onde isto pode ocorrer é com relação ao último quadro B de um GoP, que necessita do quadro I do próximo GoP para ser decodificado, desta forma o GoP não fica verdadeiramente independente como seria o ideal. Para superar este problema, os codificadores mais otimizados criam um GoP fechado que pode conter quadros B, mas terminam sempre com um quadro do tipo P.

Dentro do fluxo de dados codificado, certas informações chave (*start codes*) são identificadas por códigos de cabeçalho especiais. Estes códigos são caracterizados por uma quantidade relativamente grande de bits zero seguidos por um código específico que identifica o dado que vem logo depois. A tabela 1 resume estes códigos e seus significados.

Significado	**Código (hexadecimal)**
Cabeçalho de Seqüência	000001B3
Início de Grupo de Imagens	000001B8
Dados Estendidos	000001B5
Dados do Usuário	000001B2
Início de Imagem (Quadro)	00000100
Início de *Slice*	00000101-000001AF

Tabela 1 - Códigos de Início no Fluxo MPEG

Na figura 4 é apresentado um exemplo de como é um cabeçalho utilizado no MPEG-2. Serão descritos quais dados aparecem depois de cada código de início, os

conteúdos e os formatos dos mesmos. O exemplo apresentado é do cabeçalho de seqüência, representado pelo código 000001B3, em hexadecimal.

Continuando o exemplo, a seguir, é apresentada a definição e os valores aceitos para cada campo do cabeçalho.

Logo depois do código de cabeçalho de seqüência aparecem dois campos de 12 bits, que são os tamanhos horizontal e vertical que representam o tamanho da imagem em *pixels*. São permitidos valores de 1 a 4095.

O campo de *Aspect Ratio* é usado para selecionar um dentre os quatorze valores disponíveis de qualidade das imagens. O campo seguinte é o código correspondente à taxa de quadros, sendo que oito valores são definidos: 23.976, 24, 25, 29.97, 30, 50, 59.94, 60.

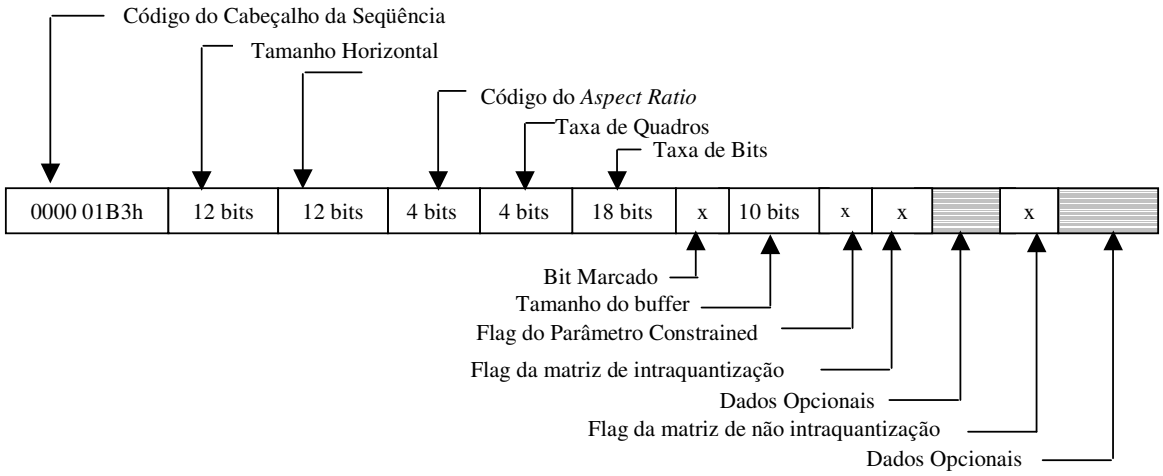

Figura 4: Organização do Cabeçalho de Seqüência MPEG

O campo de taxa de bits é um inteiro composto por 18 bits que quando multiplicado por 400 fornece a taxa de bits real do fluxo de dados. Quando se deseja indicar que o fluxo de dados irá operar utilizando uma taxa de *bits* variável, todos os 18 *bits* devem conter o valor 1.

Logo depois do campo que fornece a taxa de bits existe um campo composto por apenas um *bit*, denominado "bit marcado", que sempre recebe o valor 1. Seu objetivo é prevenir erros no momento de decodificação, pelo fato de todos os *start codes* iniciarem com uma longa seqüência de zeros.

O campo do tamanho de *buffer* é um número inteiro de 10 bits que, quando multiplicado por 16.384 (16 KB), fornece o tamanho mínimo requerido para o buffer de entrada do decodificador.

O *bit* do *flag* de parâmetro forçado (*constrained parameter flag bit*), quando habilitado em 1, indica que o fluxo de dados obedece aos requisitos de parâmetros "forçados" que tem o objetivo de promover a compatibilidade com outros decodificadores ou dispositivos que também obedeçam a certos parâmetros. Alguns destes parâmetros "forçados" são: tamanho máximo da imagem de 720X576 pixels, número máximo total de macro blocos igual a 396 e uma taxa máxima de 2.534.400 *pixels* por segundo.

O *bit* de *flag* da matriz de intraquantização, quando está com o valor 1, indica que um conjunto de 64 valores de 8-bits virá a seguir. Estes valores representam um conjunto 8 X 8 de coeficientes de intraquantização. Todos os 64 coeficientes devem ser diferentes de zero. Se o *bit* de *flag* está com o valor 0, a matriz de intraquantização deve ser ajustada usando uma tabela padrão.

O *bit* de *flag* da matriz de não intraquantização opera essencialmente da mesma forma, exceto que a matriz padrão tem todos os coeficientes preenchidos com o valor 16. E por fim, espaço para dados opcionais são disponibilizados.

Seguindo o mesmo modelo acima apresentado, os outros *start codes* também são seguidos por cabeçalhos compostos de vários campos. Estas informações são úteis para entender o funcionamento e os códigos dos programas usados para a decodificação.

A camada de vídeo do MPEG-2 é dividida em seis camadas, como relacionado abaixo e apresentado hierarquicamente na figura 5.

- Camada de Seqüência de Vídeo;
- Camada de Grupos de Imagens (GOP);
- Camada de Imagem;
- Camada de *Slice*;
- Camada de Macroblocos (área de 16 x 16 *pixel*);
- Camada de Blocos (área de 8 x 8 *pixel*).

Um cabeçalho de GoP, quando utilizado, deve sempre ser seguido por um quadro do tipo I, o que garante que o próximo quadro do tipo B poderá ser decodificado corretamente no caso de um acesso aleatório.

A ordem dos quadros no fluxo de dados codificados é a ordem na qual o decodificador os processa, mas não é necessariamente a ordem correta de exibição.

Cada imagem é subdividida em unidades chamadas *slices*. Desta forma, se ocorre um erro no fluxo de dados, isto afeta somente o *slice* em que ocorre e não se propaga pela imagem inteira. Um *slice* é composto de um ou mais macro blocos.

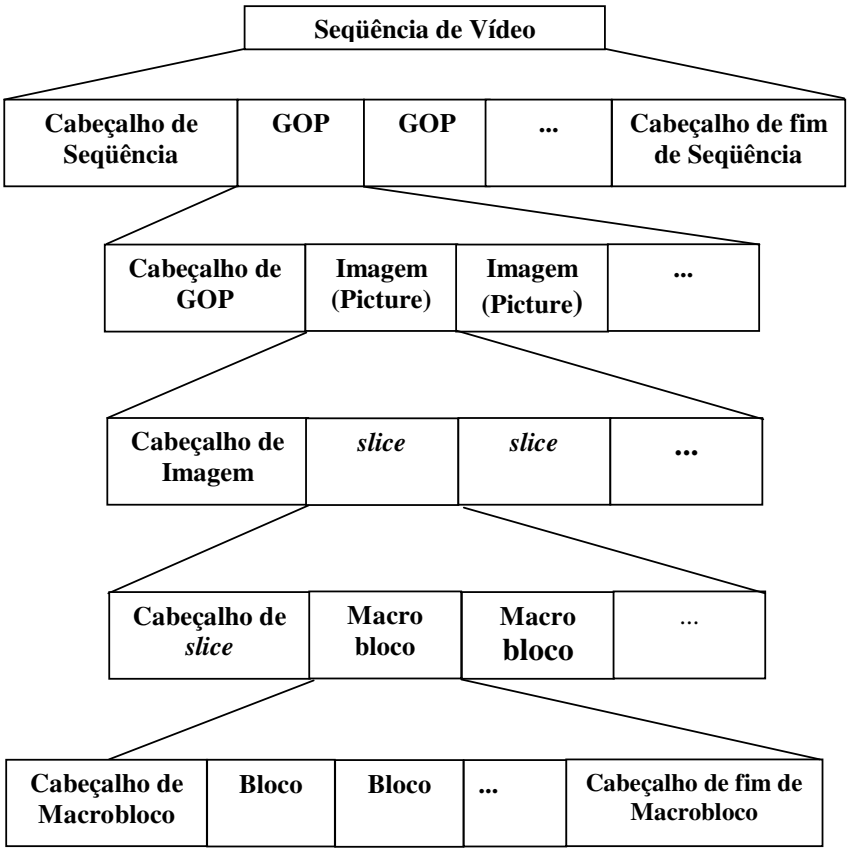

Figura 5: Estrutura da Camada de Vídeo MPEG-2

3.5 – Perfil e Nível

Levando-se em consideração a enorme variedade de aplicações que a norma MPEG-2 considera, é essencial que estas sejam organizadas de modo a não sobrecarregar desnecessariamente todos os recursos disponíveis. Assim, a norma MPEG-2 Vídeo define subconjuntos de especificações com flexibilidade e complexidade crescentes. Estes subconjuntos, identificados pelos conceitos de **Perfil** e **Nível**, garantem o ajuste na relação compressão-qualidade requerido por uma dada

classe de aplicações, limitando assim a complexidade e os custos associados aos sistemas codificadores e decodificadores.

O conceito de Perfil associa-se a um subconjunto de opções de codificação que se adaptam aos requisitos de uma classe de aplicações; por exemplo, um perfil de escalabilidade prevê a especificação adicional de ferramentas para codificação escalável, mas apenas deve ser adotado por aplicações que requerem essa facilidade. Um determinado perfil define a sintaxe do fluxo de *bits* por ele suportado.

O conceito de Nível estabelece, para cada perfil, os valores que certos parâmetros de codificações podem tomar; por exemplo, podem estabelecer um limite máximo para a resolução espacial do sinal a codificar ou para o ritmo binário de transmissão.

O padrão MPEG-2 suporta cinco tipos de perfis: **simples**, **principal**, **SNR escalável** (*Signal to Noise Ratio* – Taxa de Ruído), **espacial escalável** e **alto**. Cada perfil está relacionado a alguma área de aplicação. O perfil principal é para uso geral e o perfil simples é semelhante ao principal, exceto que exclui os quadros B, tornando a codificação/decodificação mais fácil.

Os perfis diferem em termos da **presença ou ausência de quadros B**, da **resolução de crominância** e da **escalabilidade do fluxo de *bits*** codificado para outros formatos.

A taxa de dados comprimidos é diferente para cada combinação de resolução e perfil. O intervalo é de 3 Mbps até 100 Mbps para HDTV (TV de alta definição), mas normalmente varia de 3 a 4 Mbps.

3.6 – Controle da Taxa de Dados no Formato MPEG-2

O número de *bits* transmitidos por unidade de tempo pode variar bastante. Uma descrição da taxa de *bits* é transmitida no cabeçalho de seqüência (*Sequence Header Code*) do vídeo codificado.

Para uma taxa de codificação de *bits* constante (CBR), o número de *bits* transmitidos por unidade de tempo em um canal deve ser é constante. Entretanto, a taxa de saída do codificador geralmente varia dependendo do conteúdo da imagem. O codificador deve regular esta taxa para que seja constante, o que pode ser feito

utilizando *buffers*, por exemplo. Nesta codificação CBR, a qualidade da imagem pode variar dependendo do seu conteúdo.

Uma outra forma que pode ser utilizada é a codificação com taxa variável de *bits* (VBR); neste caso, o número de *bits* transmitidos por unidade de tempo pode variar para um canal. A codificação VBR consegue oferecer uma codificação com qualidade constante.

3.7 – Obtenção de Compressão no MPEG-2

O algoritmo usado pelo MPEG para compressão causa perdas de informação, ou seja, um vídeo MPEG decodificado não é idêntico ao vídeo original antes da codificação. A taxa de compressão obtida é da ordem de 25 vezes, sendo bem melhor que os algoritmos que não perdem informações.

A compressão de vídeos pode ser obtida de diversas maneiras, algumas das mais utilizadas são:

- Eliminando a redundância de uma imagem individual (redundância espacial) e a redundância entre imagens sucessivas (redundância temporal) dos sinais de vídeo;
- Retirando do sinal de vídeo as componentes irrelevantes para o sistema visual humano;
- Representando com menos *bits* os elementos com maior ocorrência.

A redundância espacial é utilizada para simplificar a codificação de cada quadro separadamente usando JPEG. A redundância temporal tira vantagem do fato que quadros consecutivos são quase idênticos, de forma que ela pode ser mais explorada em seqüências de imagens onde muitas partes da imagem permanecem inalteradas durante muitos quadros. Por exemplo, na apresentação de um telejornal, onde o que muda são apenas os gestos e a expressão facial do apresentador e o fundo permanece o mesmo.

3.8 – Especificação do Sistema para Fluxos MPEG-2

O fato dos codificadores de áudio e vídeo trabalharem independentemente gera um problema: como coordenar os dois fluxos para que sejam síncronos. O sistema de multiplexação usado pelo MPEG-2 resolve este problema.

Como citado anteriormente, a padronização MPEG-2 foi dividida em várias partes. Uma destas partes é denominada Sistema *(Systems)* e seu objetivo é especificar como deve funcionar a multiplexação entre áudio, vídeo e outros dados, assim como a sincronização entre o codificador e o decodificador.

A codificação do sistema deve ser especificada de duas formas: **Fluxo de Transporte** (*Transport Stream-TS)* e **Fluxo de Programa** (*Program Stream-PS)*, sendo que cada uma destas formas é usada para um conjunto diferente de aplicações. Ambas são definidas de forma a prover uma sintaxe para a codificação que seja suficiente para sincronizar a decodificação e apresentação do áudio e vídeo, e para garantir que os *buffers* de dados do decodificador não fiquem vazios e nem que haja descarte de dados.

A abordagem básica para multiplexação de fluxos elementares individuais de áudio e vídeo é apresentada na figura 6.

Nesta figura, pode-se observar que as codificações de áudio e vídeo são realizadas de maneira independente, gerando assim dois fluxos codificados: um para o áudio e outro para o vídeo. Estes *fluxos* passam por um "empacotador" que tem como saída pacotes chamados **PES (Packetized Elementary Streams)** que, por definição, são o resultado do processo de empacotar fluxos elementares codificados. Estes pacotes podem ser multiplexados como fluxo de transporte (**TS)** ou como fluxo de programa (**PS)**, dependendo da aplicação na qual serão utilizados, conforme é discutido a seguir.

Um fluxo PS é o resultado da combinação de um ou mais fluxos de pacotes **PES** em um único fluxo, todos com o mesmo tempo base. Foi projetado para uso em meios relativamente livres de erros e é adequado para aplicações que podem envolver processamento de informações, como multimídia interativa. Pacotes **PS** podem ter tamanho variável e relativamente grande.

Um fluxo **TS** também combina um ou mais pacotes **PES** em um único fluxo, mas o tempo base de cada fluxo é independente. Foi projetado para ser usado em ambientes onde erros são prováveis, tais como armazenamento e transmissão em meios sujeitos a ruídos. Pacotes do tipo **TS** têm tamanho fixo de 188 *bytes*.

Os fluxos **PS** e **TS** foram criados para diferentes aplicações e suas definições não seguem um modelo estritamente em camadas. É possível converter de um para outro; contudo, um não é subconjunto ou superconjunto do outro.

O conceito de fluxo **TS** pode ser estendido para um modelo em camadas, que é projetado para facilitar a implementação de aplicações que exigem alta largura de banda.

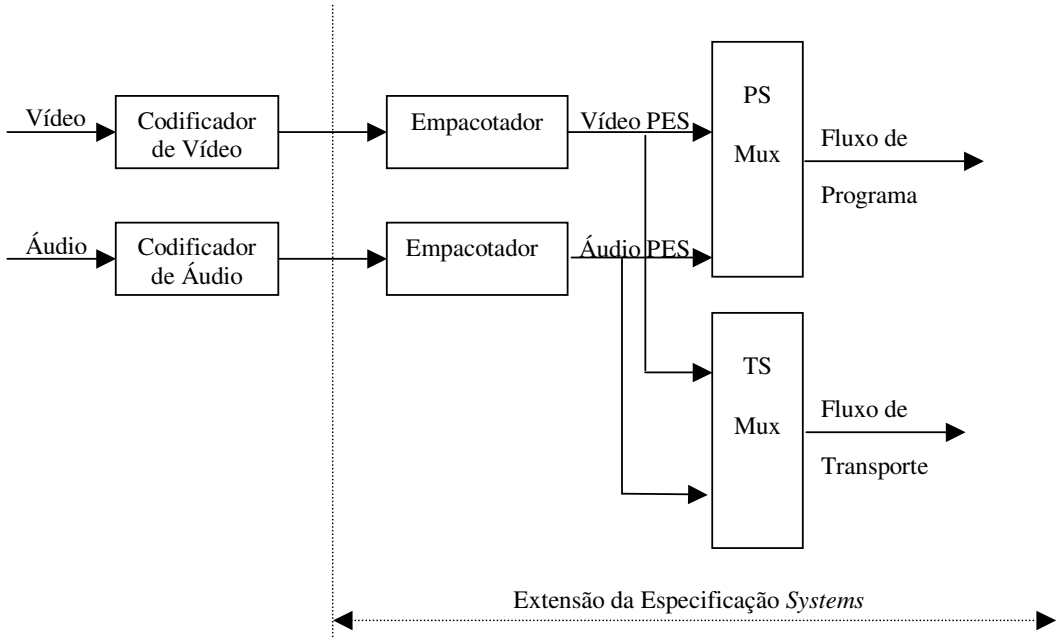

Figura 6: Multiplexação de Fluxos Elementares Individuais de Áudio e Vídeo

3.8.1 - Modelo Temporizado

Os fluxos de sistema, áudio e vídeo possuem um modelo de tempo em que o atraso fim-a-fim do sinal de entrada do decodificador e o sinal de saída do decodificador são constantes. Este atraso é a soma do tempo para codificação, b*ufferização* na codificação, multiplexação, comunicação e armazenamento, demultiplexação, *bufferização* na decodificação, decodificação e o atraso na apresentação.

Como parte deste modelo temporizado, todos os quadros do vídeo e amostras do áudio são apresentados exatamente uma vez (a não ser que o codificador especifique o contrário), e o intervalo entre quadros e a taxa de amostragem do áudio são os mesmos tanto no codificador quanto no decodificador. O sistema de codificação do fluxo de dados contém então informações de tempo que podem ser usadas para implementar sistemas incorporando um atraso fim-a-fim constante.

A informação de tempo é definida em termos de um sistema de um relógio comum, referenciado como *System Time Clock*. Nos fluxos do tipo **PS** este relógio deve

ter uma relação exatamente especificada para os relógios de áudio e vídeo, ou deve ter uma freqüência operacional que difira pouco da relação exata, e deve ainda prover com precisão uma temporização fim-a-fim, como explicado na próxima seção, para recuperação do relógio.

Nos fluxos do tipo **TS,** a freqüência do sistema de relógio é forçada a ter uma relação exatamente especificada para os relógios de áudio e vídeo em todos os tempos; o efeito de forçar isto é a simplificação na recuperação da taxa de amostragem nos decodificadores.

3.8.2 – Sincronização

A sincronização entre múltiplos fluxos elementares é realizada através de marcas de tempo para apresentação (*Presentation Time Stamps* – PTS). A decodificação de N fluxos elementares é sincronizada através de um tempo mestre comum que ajusta a decodificação de um fluxo de forma que combine com os outros fluxos. Este tempo mestre usado como referência pode ser baseado no relógio de um dos N decodificadores, pode ser do relógio da origem dos dados, ou pode ser de um relógio externo.

A sincronização fim-a-fim ocorre quando o codificador salva o *time stamp* no momento da codificação, quando o *time stamp* é propagado junto com os dados codificados a ele associado, e quando o decodificador usa este *time stamp* para organizar as apresentações.

Capítulo 4 – Sistema de Arquivos Paralelos Distribuídos

Conceitos importantes para o entendimento deste trabalho são apresentados neste capítulo. A definição de Sistema de Arquivos Paralelos Distribuídos (SAPD), e a descrição do seu funcionamento também são apresentados. O SAPD usado neste trabalho é o NPFS (*Network Parallel File System*), que utiliza discos presentes em servidores numa rede para distribuir seus dados. A descrição de sua arquitetura e forma de operação também faz parte deste capítulo.

4.1 - Sistema de Arquivos Paralelos Distribuído

Um sistema de arquivos paralelos e distribuídos manipula arquivos de tal forma que estes sejam divididos, armazenados e recuperados em diversos servidores interligados através de uma rede. Normalmente, cada um destes servidores possui o seu próprio disco local, que é compartilhado. O principal objetivo que leva à utilização deste tipo de sistema de arquivos é o aumento da capacidade de armazenamento e também da velocidade de acesso providos para as aplicações.

Este sistema também permite o acesso otimizado aos dados de aplicações que são inerentemente paralelas nas quais a manipulação de porções distintas dos dados de um arquivo compartilhado é realizada naturalmente. Normalmente, as operações de acesso a um mesmo arquivo não ocorrem exatamente na mesma parte do arquivo, o que torna viável o uso de arquivos paralelos.

4.2 – Arquivos Paralelos

Um dos principais problemas a ser enfrentado em relação a E/S diz respeito à necessidade de se prover suporte para o armazenamento de grandes volumes de dados e de se manipular suas transferências entre disco e memória em velocidades compatíveis com o processamento das aplicações.

De modo geral, observa-se que os arquivos em arquiteturas paralelas, predominantemente utilizadas na execução de aplicações de alto desempenho, diferem dos arquivos comuns pela necessidade de se maximizar as taxas de transferência e por permitirem o acesso concorrente a diversos processos. Além disso, aplicações

compostas de processos paralelos comumente acessam um arquivo de maneira intercalada, em que enquanto um manipula um conjunto fragmentado dos dados, outros acessam as partes restantes.

Para atender a essa demanda, a distribuição dos dados, que passam a ser manipulados de forma paralela, tem sido a solução adotada nos sistemas de alto desempenho, dando origem aos **arquivos paralelos**. Assim, entrada/saída paralela corresponde ao suporte provido por uma aplicação paralela, executada em diversos nós, que permite que os dados das aplicações, distribuídos fisicamente, sejam manipulados como se fossem um único arquivo lógico, chamado arquivo paralelo.

Resumindo, um arquivo paralelo corresponde a um único arquivo lógico, composto de um ou mais blocos de dados fisicamente disjuntos, conforme pode ser visto na figura 7. A determinação de quantos e quais discos serão utilizados no armazenamento é chamada ***clustering***, ou ***declustering***, sendo que a atribuição dos blocos de dados aos discos caracteriza a **distribuição**, ou o **padrão de distribuição** (***distribution pattern***) utilizado.

Um padrão comum é o chamado ***striping***. Neste padrão, as **unidades de distribuição** (***striping units***), que podem ser tanto da ordem de 1 bit (RAID nível 3) quanto igual ao tamanho do bloco do disco (RAID níveis 4 e 5) ou maiores, são atribuídas aos discos de forma circular (*round-robin*). O conjunto dos dados atribuídos a todos os discos numa rodada de distribuição constitui um ***stripe***.

Assim, o método de distribuição utilizado em um arquivo paralelo é definido pela unidade de distribuição e pelo padrão de distribuição dos dados sobre os discos.

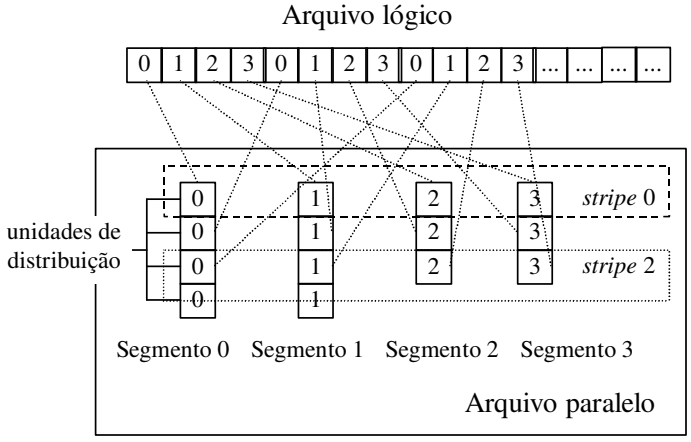

Figura 7: Exemplo de Segmentação e Distribuição de Dados de um Arquivo Paralelo

4.3 - O Sistema de Arquivos NPFS (*Network Parallel File System*)

O NPFS (*Network Parallel File System*) [GUA99] é um sistema de arquivos paralelos distribuído. Sua arquitetura consiste basicamente de um conjunto de estações de trabalho interligadas em rede, cada uma com seu próprio disco local, que pode ser compartilhado. De uma forma geral, seus objetivos consistem em prover primitivas/operações para a manipulação de um sistema de arquivos paralelos com suporte para distribuição de dados através da técnica de *striping*, discutida anteriormente.

A distribuição dos dados entre os servidores é determinada na criação dos arquivos, podendo-se selecionar o número de segmentos e o tamanho das unidades de distribuição. Além do aumento da capacidade de armazenamento e da velocidade de acesso provido para aplicações seqüenciais, o sistema permite o acesso otimizado aos dados de aplicações paralelas que manipulam porções distintas dos dados de um arquivo compartilhado [GUA99].

A estratégia utilizada no NPFS é o modelo cliente/servidor para a criação de servidores espalhados pela rede, que fornecem suporte para o acesso aos dados armazenados em seus sistemas de arquivos locais.

Três tipos de processos são definidos: **Mestre**, **Servidor** e **Cliente**. O funcionamento de cada um deles é apresentado a seguir e na figura 8 é apresentada a interação entre os mesmos:

- **Mestre**: responsável pelas funções de iniciação do sistema, cuida da ativação e manutenção dos servidores. Além disso, gerencia algumas operações que envolvem controle centralizado, como a abertura e o fechamento dos arquivos.
- **Servidor**: provê acesso aos segmentos dos arquivos que estão armazenados localmente. Uma cópia ativa deste processo existe em cada computador cujo disco local é compartilhado no sistema
- **Cliente**: é um processo de usuário. Sua interação com o restante do sistema se dá através da utilização das primitivas disponíveis em uma biblioteca de funções que incluem chamadas para os servidores e para o mestre, conforme o tipo da operação realizada.

A ativação do sistema inclui a iniciação do processo **Mestre**, especificando como parâmetro opcional o nome de um arquivo de configuração com a identificação

das estações servidoras. Fica a cargo do Mestre a ativação de um processo **Servidor** em cada uma das estações especificadas. Além disso, comandos são providos para permitir a ativação e a desativação dinâmica de servidores. A partir desta etapa, é possível a criação e a manipulação dos arquivos paralelos, podendo selecionar-se novas configurações ao longo do programa.

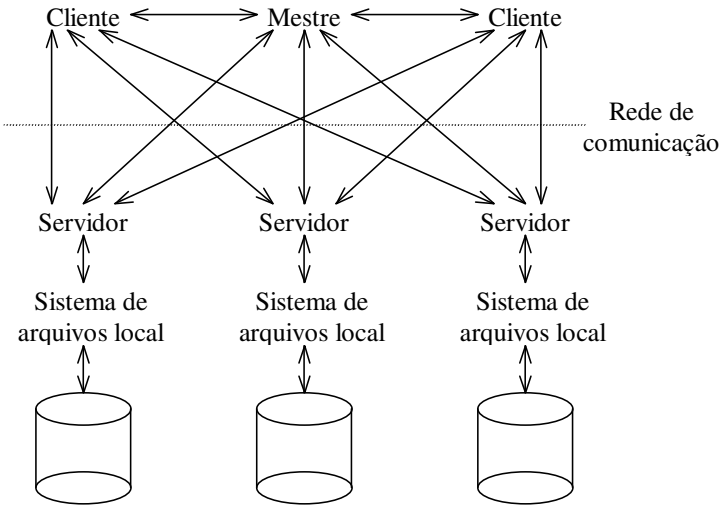

Figura 8: Interação dos Processos no NPFS

A manipulação dos arquivos paralelos é feita através de funções de uma biblioteca escrita em linguagem C, utilizando uma sintaxe parecida com a das funções para manipulação de arquivos comuns.

A seguir, são apresentadas algumas das principais primitivas existentes. Novas primitivas também podem ser desenvolvidas, principalmente para atender a uma aplicação específica, como por exemplo, ao servidor de vídeo sob demanda que está sendo proposto.

As primitivas a seguir permitem o controle dinâmico sobre os servidores envolvidos, fazendo suas ativações, desativações e consulta dos servidores ativos.

int **add_server** (*char* * host_name);
int **del_server** (*char* * host_name);
int **squery** (*struct sys_info* *, *char* **);

A manipulação dos arquivos paralelos se dá através de funções semelhantes às existentes para arquivos comuns, sendo possível abrir, fechar, ler, escrever e determinar a posição corrente do ponteiro para o arquivo.

int **p_open**(*char* *name, *int* flags, *int* mode, *int* view, *int* n_seg, *int* str_unit, *char* **servers);

O fechamento de um arquivo paralelo envolve a comunicação da aplicação diretamente com o processo Mestre, que se encarrega de fechar os segmentos em todos os servidores envolvidos.

int **p_close**(*int* fd, *int* ret);

O comando **p_read** lê até **count** bytes do arquivo e os copia para o *buffer* passado como parâmetro. Um valor inteiro de retorno indica o número de bytes efetivamente lidos, ou -1, em caso de erro. Quando os dados são lidos, o ponteiro do arquivo, que indica a posição para a próxima leitura ou escrita seqüencial, é incrementado com o valor apropriado.

O comando **p_write** escreve *count* bytes a partir do *buffer* para o arquivo especificado como parâmetro. O valor de retorno indica o número de bytes efetivamente escritos, ou -1 em caso de erro.

ssize_t **p_read**(*int* fd, *char* *buf, *size_t* count, *int* *retrans);
ssize_t **p_write**(*int* fd, *const char* *buf, *size_t* count, *int* *retrans);

Apresentando uma sintaxe semelhante ao comando *lseek* da linguagem C, **p_lseek** serve para reposicionar o ponteiro de leitura e escrita seqüencial no arquivo. Como retorno da função, tem-se o valor da posição corrente resultante, ou -1, em caso de erro.

off_t **p_lseek**(*int* fd, *off_t* offset, *int* whence);

Por razões de compatibilidade e portabilidade, as implementações do sistema são realizadas utilizando um serviço de comunicação no nível do protocolo de transporte nativo do sistema operacional. Para tanto, utiliza-se as primitivas de manipulação de *sockets*. Visando, contudo, minimizar a sobrecarga com as tarefas de comunicação, utilizou-se um protocolo não orientado a conexão (UDP) para as transferências de dados e trocas de informações de controle. Assim, a garantia do recebimento das informações foi implementada dentro das operações do NPFS.

Capítulo 5 – Servidor de VoD Paralelo e Distribuído

Neste capítulo são apresentados a proposta e o desenvolvimento deste trabalho de mestrado. Isto inclui a discussão sobre a implementação de um servidor de vídeo sob demanda utilizando arquivos paralelos e outros aspectos importantes no projeto do mesmo, tais como: determinação de quais são as melhores formas de armazenamento, como fazer o controle de admissão, escolha dos parâmetros de QoS, situações nas quais os mecanismos de *prefetching* devem ser usados e considerações sobre a sintaxe da padronização MPEG.

5.1 - Arquitetura para um Servidor de VoD Paralelo Distribuído

O armazenamento e o acesso em paralelo aos dados de um sistema de arquivos paralelos provêem naturalmente o balanceamento de cargas e a distribuição de acessos desejáveis em um servidor de VoD distribuído. Oferece ainda a vantagem de ser escalável de forma que para aumentar a capacidade do sistema, basta adicionar um novo servidor, desde que a rede não seja um gargalo.

Por estas e outras vantagens este trabalho apresenta uma abordagem para a construção de servidores VoD baseada principalmente no uso das potencialidades oferecidas por sistemas de arquivos paralelos. A disponibilidade do NPFS é aproveitada neste trabalho, já que aspectos de seu funcionamento interno podem ser estudados para o desenvolvimento do servidor VoD desejado.

Com o intuito de avaliar a utilização de arquivos paralelos em servidores VoD, este trabalho apresenta uma arquitetura para essa implementação.

Para tanto, a operação do modelo adotado e dos seus componentes foi estudada, visando identificar a viabilidade da implementação proposta.

Uma das primeiras questões a ser avaliada usando um servidor VoD remoto é se a adoção de apenas um servidor é capaz de utilizar toda a largura de banda oferecida pela rede. Para tanto, é preciso determinar se o ganho obtido no acesso paralelo aos dados alcançado pelo uso de mais servidores poderá ser repassado aos clientes.

Inicialmente, é preciso considerar o fluxo de dados que deve ser obtido pela aplicação que faz a apresentação do vídeo ao usuário. Para que o usuário possa enxergar o vídeo de forma satisfatória é necessário que os quadros sejam exibidos em uma taxa mínima de 30 fps (frames por segundo). Considerando que o tamanho médio de um

quadro MPEG ocupa aproximadamente 20 KB, a taxa de dados mínima no cliente deve ser em torno de 600 KBps.

Sendo assim, a capacidade da rede e dos discos deve ser considerada no projeto do servidor de VoD. Medindo experimentalmente, considerando as sobrecargas dos protocolos utilizados, observou-se que usando *Fast Ethernet* a capacidade da rede é de aproximadamente 10 MBps (Mega Bytes por segundo). No *Gigabit Ethernet*, esta capacidade aumenta para 100 MBps.

Através de um programa de teste que faz sucessivas leituras seqüenciais verificou-se que um disco **IDE** fornece uma taxa de leitura local aproximada de 5 MBps, que varia pouco para diferentes tamanhos de requisição de leitura. Desta forma, é possível concluir que usando apenas um servidor a largura de banda oferecida pela rede não é totalmente explorada. Usando discos **SCSI** a taxa de leitura medida usando o mesmo programa de teste ficou em torno de 33 MBps.

Deste modo, conclui-se que mesmo usando um disco com maior capacidade de leitura a largura de banda da rede não está sendo totalmente aproveitada. Usando uma rede de maior capacidade de transmissão (*Gigabit Ethernet*, por exemplo) fica clara a necessidade de otimização das operações de E/S.

A figura 9 apresenta como a utilização do sistema de arquivos paralelo distribuído pode conseguir taxas maiores considerando o somatório das taxas de cada disco do sistema. Uma vez que o acesso aos dados ocorre em paralelo, pode-se admitir que usando M servidores esta taxa de leitura é dada por M*5 MBps, considerando a utilização de discos IDE. O acesso paralelo obtido com o uso de sistemas de arquivos paralelos neste caso, visa a aproveitar melhor a largura da banda da rede e otimizar as operações de leitura e escrita.

Utilizando uma rede com uma largura de banda equivalente ao *Fast Ethernet*, o ganho no acesso paralelo aos discos não pode ser totalmente repassado as clientes, mas utilizando uma rede mais rápida, o ganho obtido com o paralelismo deve ser observado. Assim, o uso de um *switch* como apresentado na figura 9 não traz ganhos consideráveis quando um único servidor é utilizado, mas quando vários clientes e vários servidores estão conectados a este equipamento, cada cliente recebe um fluxo diretamente de cada servidor.

Caso um único servidor fosse utilizado, o acesso seqüencial ao seu sistema de arquivos representaria um fator limitante ao desempenho do sistema.

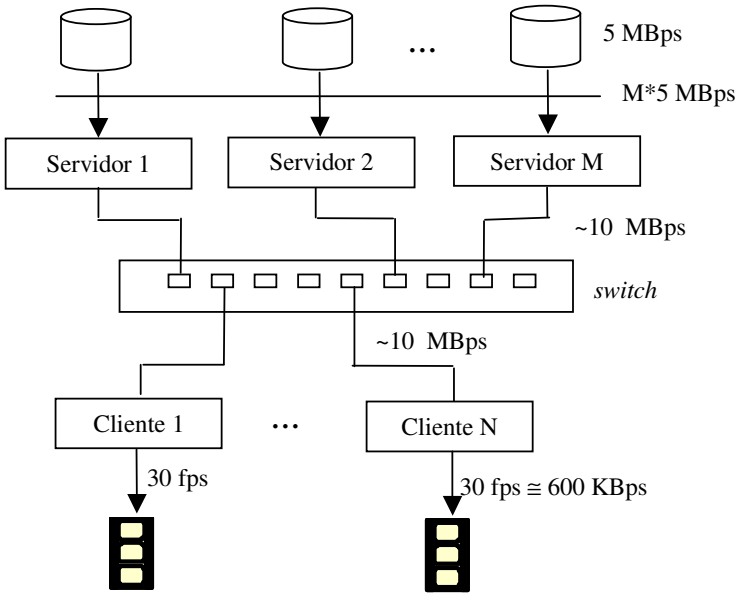

Figura 9: Esquema de Recuperação dos Vídeos

5.2 – Proposta do Servidor de Vídeo NPFS

Uma boa parte dos trabalhos desenvolvidos na área de vídeo sob demanda concentra seus esforços em encontrar formas eficientes de armazenamento dos vídeos, outros em como melhor recuperar e entregar os dados aos clientes; alguns propõem formas de resolver a questão de tolerância a falhas, e por fim, ainda existem outros em áreas variadas como: dimensionamento de *buffers*, controle de admissão, *cache* compartilhado, suporte a operações interativas. Contudo, certos aspectos podem ser estudados e discutidos produzindo assim novos resultados.

Considerando estes aspectos, este trabalho vem propor o desenvolvimento de um servidor de vídeo sob demanda, o **Servidor de Vídeo NPFS**, que utiliza o sistema de arquivos paralelos NPFS para o armazenamento dos vídeos. Os assuntos abordados e as discussões realizadas são generalistas e servem para qualquer sistema de arquivos paralelos. Entretanto, o uso do NPFS se deve ao fato da disponibilidade de seu código fonte e pela possibilidade de adaptação de suas funções às necessidades deste trabalho.

Muitas questões relacionadas ao projeto e à implementação de um servidor de VoD são discutidas e avaliadas. Os principais tópicos abordados neste trabalho são:

- Busca por melhores estratégias para armazenamento dos vídeos, considerando os fatores relacionados, como, por exemplo, tamanho de *stripe unit* que forneçam os melhores tempos de resposta e melhor sincronização entre áudio e vídeo;
- Pesquisa de formas eficientes de recuperação dos vídeos, através da investigação de quais são os melhores tamanhos de *stripe unit* a serem adotados, considerando, por exemplo, o tempo médio de acesso ao disco dos servidores e menores tempos de resposta;
- Estudo de como os mecanismos de *cache* e *prefetching* do sistema de arquivos podem influenciar o comportamento e o desempenho do sistema de VoD e influenciar também as escolhas de projeto (armazenamento, recuperação, controle de admissão, etc);
- Coleta de informações sobre a utilização da rede e dos discos e uso destas informações para a criação de um mecanismo de controle de admissão dinâmico;
- Adoção de certos parâmetros de Qualidade de Serviço (QoS) em função da qualidade percebida pela aplicação do cliente e dos parâmetros dos recursos do sistema, tais como, tempo de acesso aos discos dos servidores e ocupação da rede.

Para que os tópicos acima relacionados possam ser estudados, um novo sistema de Vídeo Sob Demanda é proposto, inicialmente chamado de **Servidor de Vídeo NPFS**. A arquitetura e o funcionamento do mesmo serão apresentadas nas seções seguintes.

5.3 - Arquitetura do Servidor de Vídeo NPFS

A arquitetura do **Servidor de Vídeo NPFS** é semelhante a outros sistemas de arquivos paralelos distribuídos já existentes, sendo composta basicamente de servidores, clientes e uma rede para interconexão de ambos.

Os servidores de armazenamento não são parte do sistema de VoD. O sistema proposto apenas usa os seus serviços. Da mesma forma, outros sistemas de arquivos paralelos poderiam ser utilizados.

Apenas um novo servidor é proposto, tendo recebido o nome de **Servidor de Controle**. Ele é o responsável por realizar o controle de admissão e por gerenciar os parâmetros de QoS. É sua função também manter a lista atualizada dos vídeos disponíveis.

Como descrito, o serviço de VoD utiliza o modelo cliente/servidor em que o cliente é uma aplicação do usuário. Sua interação com o sistema de VoD ocorre para a seleção do vídeo a ser assistido através do **Servidor de Controle** e depois diretamente com os servidores de dados do sistema de arquivos paralelos.

A aplicação do cliente recebe as partes dos vídeos que chegam dos diversos servidores, armazena estes dados em *buffer*, faz a decodificação e passa ao visualizador (*player*) que é o responsável por exibir o conteúdo do vídeo.

No sistema proposto a exibição do vídeo fica sob a responsabilidade de um programa denominado *MPlayer* [MPL03]. Este programa possui código fonte aberto, escrito na linguagem de programação C e foi escolhido por ser bastante eficiente e disponível.

Para tornar possível a comunicação entre estas duas entidades fundamentais do sistema VoD, clientes e servidores, é utilizada uma rede através da qual os dados dos vídeos e os dados de controle trafegam. Esta arquitetura é apresentada na figura 10.

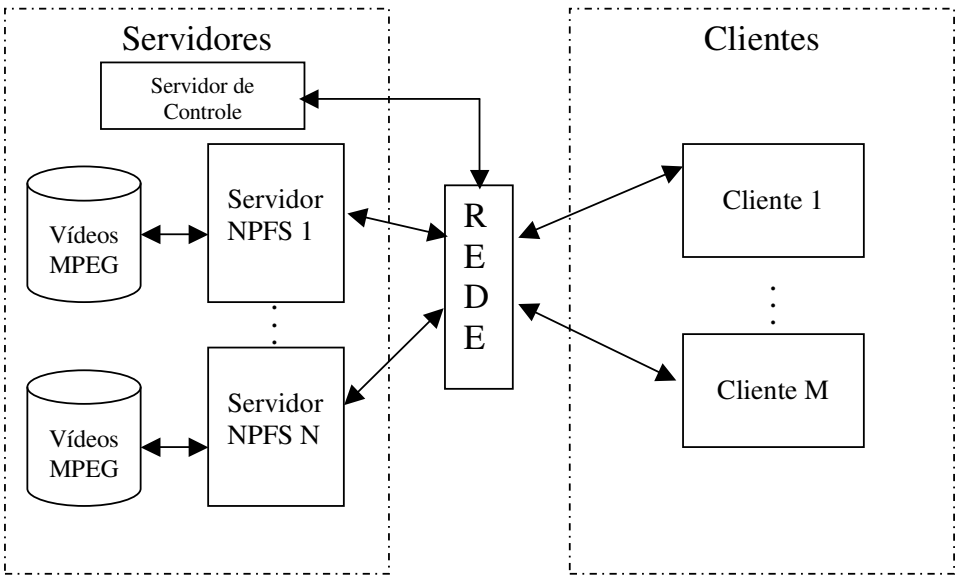

Figura 10: Arquitetura do Servidor de Vídeo NPFS

A arquitetura do sistema proposto pode ser vista como um conjunto de módulos que interagem e juntos fornecem um serviço de entrega de vídeos explorando o acesso paralelo aos mesmos. A interação existente entre estes módulos é apresentada na figura 11.

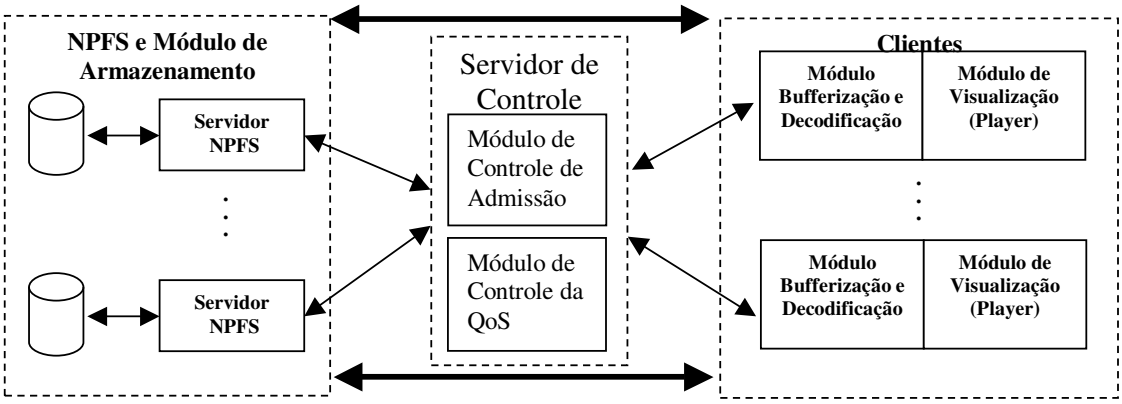

Figura 11: Arquitetura dos Módulos do Sistema

É importante destacar a presença das duas setas mais largas na figura. Elas representam a comunicação direta existente entre os clientes os servidores NPFS.

O primeiro módulo a compor esta arquitetura é o **módulo de armazenamento,** que deve ser integrado a qualquer sistema de arquivos paralelos para juntos realizarem o armazenamento dos vídeos. O sistema utilizado nesta proposta é o NPFS.

Outros dois módulos importantes nesta arquitetura são o **módulo de controle de admissão e módulo de controle da QoS**, implementados no **Servidor de Controle**. O **módulo de controle de admissão** é responsável por receber as requisições dos clientes, descobrir em quais servidores aquele arquivo requisitado se encontra e oferecer as informações necessárias para que os clientes possam solicitar a transmissão dos dados.

Depois de estabelecida a "conversação" direta entre servidores e clientes, o **módulo de admissão** não interfere na transmissão dos dados, a não ser quando o vídeo termina ou o cliente pede para encerrar a transmissão; se os dados tivessem que passar por ele, o paralelismo obtido no acesso estaria sendo perdido. Em conseqüência disto, suas tarefas se restringem àquelas de natureza centralizada, tais como negociação de parâmetros de qualidade e controle de admissão. Outra tarefa importante deste módulo de admissão é controlar todos os usuários que deixam o sistema; esta informação é importante para que o sistema decida sobre a aceitação da admissão de um novo usuário.

Quando os vídeos são armazenados utilizando a técnica de *striping* é preciso que as partes de cada arquivo sejam ordenadas e juntadas formando um único fluxo, no formato esperado pelo decodificador. Em [LEE98] o módulo do sistema responsável por fazer isto é denominado *proxy*, que pode ser implementado de três formas diferentes, conforme descrito no capítulo 2. Entretanto, como neste caso, a primitiva de leitura do sistema de arquivos NPFS é otimizada para entregar ao cliente os dados na ordem certa e no mesmo *buffer*, não é preciso que a aplicação do cliente se preocupe em implementar um módulo de *proxy*.

Também não é necessário implementar um *proxy* nos servidores, como é feito em [SAN97], pois este pode ser um ponto crítico do sistema. No referido trabalho, o desempenho do sistema utilizando *striping* se mostrou menos eficiente (em termos de clientes simultâneos), sobretudo porque utilizava uma abordagem na qual os dados vindos dos servidores eram montados em um dos servidores antes de serem enviados aos clientes, o que aumenta o custo com a comunicação.

Os **módulos de bufferização e decodificação** trabalham em conjunto com o **módulo de visualização**. Eles são implementados na aplicação do cliente e têm como função armazenar os dados antes da decodificação, decodificar os vídeos e apresentar (exibir) os vídeos.

Devido ao fato de já existirem alguns programas desenvolvidos para a decodificação e exibição de vídeos, este trabalho não precisou implementar este módulo completamente. Foi necessário apenas fazer uma adaptação de um programa já existente, chamado ***MPlayer*** [MPL03], para que o mesmo fosse capaz de utilizar as primitivas oferecidas pelo sistema de arquivos NPFS.

5.4 - O Funcionamento do Servidor de Vídeo NPFS

No **Servidor de Vídeo NPFS** o armazenamento é de responsabilidade do sistema de arquivos NPFS. Como já descrito anteriormente, o sistema de arquivos paralelo disponibiliza primitivas (que podem ser entendidas como funções) específicas para cada operação que ele pode realizar. Para o armazenamento, a primitiva de escrita, *p_write*, utiliza como parâmetros o número de servidores nos quais o arquivo deverá ser armazenado, o tamanho da *stripe unit* e o tamanho da requisição de escrita. Passados estes parâmetros, o NPFS se encarrega de armazenar corretamente os vídeos usando

uma política de *stripping round-robin*, explicada no capítulo 4. Estes parâmetros são fornecidos apenas uma vez durante o processo de criação do arquivo.

O **Servidor de Controle** é o responsável por manter a lista dos vídeos armazenados, e é sua função informar aos clientes quais são os vídeos disponíveis. Ele também guarda informações adicionais sobre cada vídeo e também sobre as condições de ocupação da rede, de forma a realizar um controle de admissão dinâmico e que possa utilizar toda a capacidade da rede. Determinar o melhor formato para armazenamento dos vídeos também é tarefa do **Servidor de Controle**.

Em sua operação, o cliente deve interagir com o servidor de controle para a seleção de um vídeo e ajuste de parâmetros de QoS. Com as informações obtidas do **Servidor de Controle** o cliente passa a interagir diretamente com os servidores de dados para a obtenção do fluxo desejado.

Para recuperar os vídeos, os clientes utilizam a primitiva de leitura *p_read*, que possui os mesmos parâmetros da primitiva de escrita, e opcionalmente um parâmetro de de *prefetching*. O mecanismo de *prefetching* será melhor explicado ainda neste capítulo. A primitiva de leitura funciona da seguinte forma: toda vez que um cliente quiser receber uma parte do arquivo (vídeo), ele precisa obrigatoriamente fazer uma requisição de leitura referente àquela porção desejada. Esta requisição é enviada diretamente a todos os servidores que recuperam em seus discos os dados solicitados e os enviam diretamente ao cliente, sem passar por nenhum outro servidor ou controle. Isto é feito implicitamente pela primitiva de leitura de dados.

Chegando no cliente, esta informação é armazenada em um *buffer* da aplicação e está pronta para ser utilizada. No caso específico dos vídeos, pronta para ser decodificada e em seguida apresentada.

5.5 – Detalhamento da Proposta Servidor de Vídeo NPFS

Com o objetivo de estudar e avaliar o armazenamento de vídeos no **Servidor de Vídeo NPFS** alguns parâmetros e medidas foram coletados através de testes experimentais. Uma discussão mais detalhada das propostas deste trabalho é feita nas seções seguintes.

5.5.1 - Armazenamento dos Vídeos

Quando um sistema de arquivos paralelos é usado para guardar os vídeos, a determinação do tamanho da *stripe unit* adequado para esta aplicação é fundamental para garantir o bom desempenho do sistema. Este parâmetro pode interferir na eficiência do balanceamento de carga do sistema, nas características dos *buffers* e na latência existente durante a exibição do vídeo.

Existem duas técnicas para fazer o particionamento dos vídeos, denominadas *time striping* (divisão por tempo) e *space striping* (divisão por espaço). Como os vídeos podem ser vistos como um conjunto de quadros, pode ser adotada uma unidade de divisão baseada em quadros como, por exemplo, um quadro. A segunda opção, *space striping*, divide o vídeo em unidades de tamanho fixo, em *bytes*. Usar esta abordagem simplifica o armazenamento de dados e também o gerenciamento dos *buffers*, porque a unidade de distribuição sempre terá o mesmo tamanho.

O objetivo de utilizar-se um sistema de arquivos paralelos é explorar as vantagens que já foram apresentadas anteriormente. Para isto, é preciso estudar adequadamente a natureza de cada aplicação para definir qual é o melhor padrão de distribuição e qual é o melhor tamanho da unidade de particionamento dos dados. Considerando a escolha pelo formato MPEG-2, feita neste trabalho, o entendimento do formato do fluxo MPEG-2, bem como as técnicas de codificação utilizadas é importante no momento de escolha dos tamanhos da *stripe unit,* bem como dos tamanhos das requisições.

Alguns trabalhos encontrados na literatura desenvolveram técnicas para determinar tamanhos adequados de *stripe unit* através de modelos analíticos e simulações para validação dos mesmos. Exemplos de trabalhos neste sentindo são [SHE97] e [SHE99], que além de tentar determinar o melhor tamanho para a *stripe unit* também fazem uma análise sobre o grau de distribuição dos dados (se os arquivos são particionados entre todos os servidores, ou não) e sua influência no desempenho do sistema. Entretanto, nestes trabalhos não é considerado o uso de sistemas de arquivos paralelos e os clientes são servidores em ciclos. Durante cada ciclo, o servidor recupera um número fixo de unidades da mídia por fluxo.

A determinação do tamanho ótimo para a *stripe unit* depende dos requisitos de QoS dos clientes e de vários parâmetros do sistema, tais como, número de clientes, taxa de exibição, número de discos, etc. [SHE97].

Observa-se que um tamanho de *stripe unit* muito grande não faz necessariamente o sistema ter um desempenho melhor, uma vez que pode causar a diminuição do número máximo de clientes simultâneos suportados pelo sistema. Aparentemente, isto deve acontecer porque um tamanho de *stripe unit* grande tende a aumentar o tempo de serviço de cada solicitação.

Considerando que o sistema proposto neste trabalho não é composto por um número muito grande de discos, devido às limitações do ambiente de teste, foi utilizada apenas a abordagem *wide striping,* que divide os vídeos entre todos os discos do sistema. A discussão sobre as opções de grau de *striping* e quando devem ser utilizadas foram discutidas no capítulo 2.

Com o objetivo de determinar qual a melhor forma de armazenar os vídeos no **Servidor de Vídeo NPFS** foram realizados testes variando o tamanho da *stripe unit,* pois este fator interfere diretamente no desempenho do sistema, como descrito por [SHE99]: "o desempenho de um sistema que utiliza a técnica de striping para distribuir seus dados é governado por dois parâmetros: o tamanho da unidade de particionamento (*stripe unit*), que denota a quantidade máxima de dados armazenados logicamente de maneira contínua em um único disco, e o grau de *striping*, que se refere ao número de discos no qual um dado arquivo é distribuído".

Determinar o que significa melhor desempenho em um sistema de vídeo sob demanda depende do ponto de vista que está sendo utilizado. Um sistema pode ser considerado muito bom se analisados alguns parâmetros de Qualidade de Serviço (QoS), mas por outro lado, pode ter um desempenho considerado não muito bom se outros parâmetros forem mais valorizados. Se um servidor de vídeo é projetado, por exemplo, para atender ao maior número de clientes possível, alguns parâmetros como tempo de resposta terão que receber um menor grau de importância. Em outro sistema pode ser que o número de clientes simultâneos não seja tão importante como o fornecimento de um tempo de resposta menor.

Isto é uma questão de escolha de projeto mas, de modo geral, espera-se que um sistema de VoD consiga atender e otimizar os principais parâmetros de QoS, tanto do cliente quanto do servidor. Seguindo esta filosofia, encontrar a melhor maneira de armazenar os vídeos não é uma tarefa tão simples, pois envolve encontrar uma combinação que atenda um maior número de parâmetros de QoS possíveis.

Os principais parâmetros de QoS estudados e considerados no projeto de **Servidor de Vídeo NPFS** que possuem relação com o armazenamento são: o tempo de resposta às requisições dos clientes e a variação no atraso (*jitter*) no processo de chegada de dados aos clientes. Estes parâmetros são influenciados diretamente pelo tamanho da *stripe unit* e podem, ainda, influenciar na taxa de processamento de dados pelo cliente.

Para tanto, foi realizado um estudo para verificar se o tamanho da *stripe unit* influencia o tempo total que o cliente gasta realizando operação de E/S e o tempo total gasto no processamento dos dados.

Considerando as duas principais formas de recuperar os dados armazenados em um servidor de VoD, denominadas *server-push* e *client-pull*, o sistema de arquivos NPFS pode ser classificado como *client-pull*. Devido ao fato do NPFS ter sido desenvolvido para armazenar qualquer tipo de informação, seja ela áudio, vídeo, texto ou número, sua abordagem é diferente daquela utilizada pela maioria dos sistemas de arquivos criados especialmente para armazenar informações multimídia. Por isso, sua maneira de recuperar os dados armazenados em seus discos é através da solicitação explícita de dados vindas dos clientes. Pelo fato de utilizar arquivos paralelos, cada servidor conhece apenas sua parte dos dados referentes a um arquivo paralelo, cuja estrutura é comandada pelos clientes. Uma forma de otimizar o acesso aos dados é utilizar mecanismos de *prefetching*, que podem recuperar os dados dos discos antes que uma solicitação explícita a estes dados tenha sido feita. Utilizar uma operação deste tipo tende a melhorar o tempo de resposta aos clientes, uma vez que a busca da informação em memória é bem mais rápida que buscá-la em disco. Este tipo de operação de leitura antecipada (*prefetching)* ainda está sendo adicionado ao NPFS, de forma que este trabalho inicialmente discute como sua utilização deve ser incorporada ao sistema de VoD.

5.5.2 – Controle de Admissão e Qualidade de Serviço

Um dos parâmetros usados para medir Qualidade de Serviço (QoS) é a realização de um controle de admissão que garanta que o aceite de um novo cliente ao sistema não irá interferir no atendimento aos outros clientes anteriormente admitidos.

Para realizar o controle de admissão no **Servidor de Vídeo NPFS** são usados o valor de RTT e o tempo médio de acesso ao disco, de forma que, ao atingir um valor limite (*threshold*) o sistema não mais poderá aceitar novos clientes sem que o atendimento dos outros clientes fique prejudicado. Usando as duas medidas espera-se obter uma informação bem próxima da realidade de ocupação do sistema.

Para tanto, parâmetros de QoS devem ser negociados entre o cliente e o **Servidor de Controle** durante o processo de solicitação de um vídeo. Com o valor adotado, cada cliente passa a ser responsável por alocar o mínimo de recursos suficientes para satisfazer os requisitos da QoS.

Observando os efeitos que a variação de alguns parâmetros do sistema têm sobre a qualidade do vídeo apresentado, foi possível determinar experimentalmente, três classes de Qualidade de Serviço. As classes foram formadas em função do atraso máximo permitido entre o áudio e o vídeo, pois é esta medida que indica quando e de quanto deve ser o descarte de quadros. As classes de QoS definidas para o **Servidor de Vídeo NPFS** são:

- **Qualidade ótima:** nesta classe de QoS nenhum descarte de quadros é feito. O atraso máximo permitido entre o áudio e o vídeo é de 0,2 segundos. Foi possível obter este valor através da observação da qualidade visual e da sincronização entre o áudio e o vídeo.
- **Qualidade boa:** nesta classe de QoS descartes de quadros são permitidos e o atraso entre áudio e vídeo pode chegar a 0,4 segundos. Este valor também foi obtido através da observação da exibição de alguns vídeos.
- **Qualidade ruim:** neste caso, muitos quadros são descartados e o atraso na sincronização pode chegar a vários segundos. Pode ser usado apenas para dar uma idéia do conteúdo de um vídeo, pois é improvável que um usuário queira assistir aos vídeos com tal qualidade.

A definição destas classes tem como primeiro objetivo determinar qual é a quantidade mínima de dados que precisa chegar ao cliente numa certa fração de tempo, ou seja, qual é a taxa mínima de dados que precisa chegar no cliente para que o vídeo não sofra interrupções na exibição e nem fique sem sincronização.

Após determinar qual é esta taxa mínima necessária para que o vídeo seja exibido com a QoS escolhida pelo usuário, é possível usar “folgas” que estejam ocorrendo em alguns clientes de forma a aumentar o número de clientes simultâneos suportados.

Para tanto, o **Servidor de Controle** coleta medidas de desempenho e realiza um controle de admissão dinâmico. Neste caso, o cliente é responsável por enviar notificações para o **Servidor de Controle** caso ocorra alguma mudança significativa em seus parâmetros de QoS. Com essas informações o **Servidor de Controle** decide quais procedimentos precisa realizar para ajustar os parâmetros de QoS.

A capacidade de renegociação dos parâmetros de QoS é uma característica importante em sistemas de VoD, pois possibilita uma melhor utilização dos recursos do sistema. Inicialmente, o sistema de VoD projetado e implementado neste trabalho não possui esta funcionalidade, mas é desejável que durante um processo de otimização do mesmo seja feito um estudo de como acrescentar ao sistema a capacidade de mudar os parâmetros de QoS previamente negociados de acordo com as necessidades atuais do sistema, que podem variar bastante em certas situações.

5.5.3 – *Buffers* e mecanismos de *prefetching*

A principal razão para a utilização de *buffers* na aplicação do cliente de VoD é prevenir que um congestionamento da rede ou sobrecargas nos servidores de armazenamento interrompam a exibição de um vídeo, que deve ser contínua. Mantendo dados suficientes em um *buffer* local, a aplicação pode corrigir o *jitter* (variação no atraso) que pode ocorrer durante a chegada de partes do vídeo.

Já a utilização de *prefetching* (leitura antecipada) tem como principal objetivo explorar o tipo de acesso realizado por aplicações de VoD. O acesso seqüencial usado para recuperar os vídeos facilita a adoção deste tipo de mecanismo.

O mecanismo de *prefetching* proposto neste trabalho prevê duas formas de operação, uma que recupera os dados dos discos e os armazena no *cache* do servidor e outra que, após a leitura dos dados dos discos, os envia diretamente ao *cache* dos clientes.

A proposta deste trabalho é que a primeira opção seja utilizada quando a ocupação da rede está alta e não é vantajoso mandar os dados coletados através da leitura antecipada imediatamente após a leitura do disco. Desta forma, quando a requisição de leitura vinda do cliente chegar aos servidores, estes poderão buscar os dados localmente no *cache* e enviá-los aos clientes. Uma operação de busca no *cache* é bem mais rápida que realizar uma leitura no disco. Ainda pode existir a opção de

mandar estes dados previamente recuperados para os clientes caso a rede tenha sua ocupação reduzida. Neste caso, o cliente toma a iniciativa de mudar o tipo de *prefetching*, uma vez que é ele quem comanda esta operação. A implementação deste tipo de *prefetching* pode utilizar "dicas" (*hints*) para determinar qual o próximo dado a ser recuperado.

A segunda opção deve ser utilizada quando a ocupação da rede é baixa. Neste caso, os dados são recuperados e imediatamente enviados aos clientes.

O controle das informações sobre a ocupação da rede, taxa de operação dos discos, taxa de processamento dos dados nos clientes e outras informações de controle usadas para as decisões sobre utilizar ou não *prefetching* são gerenciadas pelo **Servidor de Controle**.

O *prefetching* sempre parte do cliente e ele nem precisa consultar o **Servidor de Controle** para verificar a ocupação dos discos e da rede. Estas informações são coletadas na primitiva de leitura *p_read*, sendo preciso apenas determinar como estas informações devem ser utilizadas. Dependo dos valores apresentados deve-se escolher entre usar um tipo ou outro de *prefetching* ou até mesmo decidir que não é vantagem neste momento usar o mecanismo. No decorrer do tempo, o **Servidor de Controle** pode enviar informações de controle para o cliente para que ele mude de tipo de *prefetching* ou até pare de realizá-lo.

Um desafio é encontrado justamente no momento de decidir em quais situações o *prefetching* deve ser empregado ou não. Quando a carga no sistema é baixa, isto é, discos e rede estão operando muito aquém de sua capacidade total de carga, os dois tipos de *prefetching* podem com certeza ser usados. Mas em situações onde os discos estão sobrecarregados, atendendo a muitas requisições não é tão simples saber se a utilização do *prefetching* no servidor é viável.

Outro ponto analisado é como utilizar o mecanismo de *prefetching* sem comprometer o parâmetro de QoS relacionado ao número máximo de usuários simultâneos suportados. Um cliente não deve deixar de ser aceito porque outro cliente está ocupando os discos e a rede com operações de *prefetching*.

5.5.4 – Sintaxe do Conteúdo MPEG

Este trabalho também apresenta de que forma o conhecimento da estrutura da padronização MPEG pode ser usado na otimização do **Servidor de Vídeo NPFS.**

Através de estudos da documentação que padroniza a estrutura do fluxo MPEG-2, [ISO95], discutida no capítulo 3, observou-se que basicamente este fluxo é composto de: um cabeçalho da seqüência, seguido opcionalmente pelo cabeçalho do GoP e então um ou mais quadros codificados, e termina com um campo de fim de seqüência (***sequence_end_code***).

Conforme exposto em [NG00] muitos dos sistemas e modelos para VoD existentes ignoram as características do padrão de compressão utilizado nos vídeos. Em virtude disto, não incorporam as características do MPEG em seus projetos.

Utilizando-se o conhecimento de como é a estrutura dos quadros MPEG-2, principalmente o fato de que os quadros possuem tamanhos variados e tempos de processamentos variados, este trabalho propõe a utilização destas informações no processo de armazenamento e recuperação dos dados e também no gerenciamento do sistema de VoD.

Pelo fato dos quadros possuírem tamanhos variados foi considerada a possibilidade de armazenar os vídeos utilizando tamanhos diferentes de *stripe unit*. Como o sistema de arquivos NPFS não possui esta opção de armazenamento, para conseguir fazer o armazenamento desta forma seria necessário acrescentar esta funcionalidade. Uma tentativa foi realizada neste sentido, mas devido à complexidade associada a gerenciar este tipo de armazenamento, a idéia foi abandonada e outras formas de utilizar o conhecimento da estrutura do MPEG foram estudadas.

Uma das estruturas conhecidas e consideradas neste trabalho é o GoP, já apresentado no capítulo 3. Sua vantagem é ser formado por uma certa quantidade de quadros, possuindo uma variação de tamanho menor que a apresentada pelos quadros individualmente e outra vantagem é que toda dependência entre os quadros está dentro do mesmo GoP.

O tamanho de cada GoP é determinado de certa forma pela variabilidade das imagens dos vídeos, pois quanto mais as imagens a serem codificadas mudam, mais informações precisam ser guardadas. Esta é a idéia principal do MPEG, guardar apenas as variações nas imagens, podendo assim reduzir o espaço de armazenamento necessário.

5.5.5 – Testes e Obtenção de Medidas

Diversos testes foram realizados para avaliar quantativamente a influência que o tamanho da *stripe unit* tem sobre os parâmetros de QoS do sistema e os efeitos causados por cada forma de distribuir as porções dos vídeos.

A primeira informação coletada foi o tempo decorrido entre uma requisição de leitura (*p_read*) e a chegada dos dados solicitados no cliente. Duas medidas de tempo são realizadas, uma imediatamente antes da requisição de leitura e outra imediatamente depois desta requisição. Desta forma, foi possível coletar o tempo gasto para o atendimento de cada uma das requisições de leitura, fazendo uma subtração entre o tempo final e o tempo inicial. Este tempo obtido é comumente referenciado como **tempo de resposta**.

Em aplicações convencionais de texto é desejável que se utilize tamanhos de *stripe unit* que minimizem o tempo de resposta médio e ao mesmo tempo maximizem o *throughput*. Em servidores multimídia a abordagem utilizada costuma ser outra: é desejável que o tamanho escolhido de *stripe unit* minimize a variação no tempo de resposta (variação no atraso, conhecida como *jitter*) e que maximize o *throughput,* tentando desta forma diminuir a freqüência de descontinuidade na apresentação do vídeo. Esta diferença é considerada nesta proposta de forma que a variação nos tempos medidos é calculada através de medidas estatísticas, tais como o **desvio padrão**. Esta medida indica a variabilidade das informações coletadas; quanto maior ela for, maior é a diferença entre os valores utilizados.

Outra medida também coletada foi o tempo decorrido gasto para realizar o processamento de um GoP, neste trabalho denominado **tempo de processamento do GoP**. Para obter esta informação um procedimento muito parecido ao anterior foi utilizado. Um cálculo sobre como seria o tempo de processamento de cada quadro não foi realizado devido ao fato deste apresentar uma variabilidade muito grande de tamanho, e sendo assim este valor não traria nenhum acréscimo de conhecimento para ser usado na tentativa de obter melhor desempenho.

Ambas as medidas acima descritas foram coletadas na aplicação do cliente, uma vez que as requisições partem dela.

Também foram coletadas medidas no servidor. Uma delas é o tempo médio gasto para recuperar do disco uma requisição solicitada. Este cálculo é feito

dinamicamente usando como intervalo um tempo em segundos definido numa variável mantida no servidor, o cliente não interfere neste cálculo, a não ser pelo fato de ser o responsável pelas requisições.

De uma maneira resumida, os testes relacionados com armazenamento e tamanho de *stripe unit* podem ser divididos em três categorias:

- Baseado em informações extraídas dos vídeos através de um pré-processamento sobre os tamanhos dos quadros e sua média;
- Baseado em informações extraídas dos vídeos através de um pré-processamento sobre os tamanhos dos GoP's e sua media;
- Não baseado no conteúdo dos vídeos, usando tamanhos variados para *stripe unit*: na busca de valores que apresentem melhor desempenho, a princípio, melhor tempo de resposta.

A realização destes testes teve como principal objetivo estudar se existe uma relação entre a estrutura da padronização MPEG e tamanhos adequados para a *stripe unit*.

Nos testes realizados para determinar os melhores tamanhos de *stripe unit*, foram utilizadas requisições de leitura com tamanhos obtidos através da multiplicação do número de servidores pelo tamanho da *stripe unit*. Por exemplo, para um tamanho de *stripe unit* de 1024 *bytes*, usando quatro servidores, o tamanho da requisição usado foi de 4096 *bytes*. Desta forma o paralelismo pode ser explorado ao máximo.

Estes testes são apresentados em capítulo separado e demonstram quais são os melhores tamanhos de *stripe unit*, ou seja, aqueles que possibilitam um melhor aproveitamento da rede, atendem as requisições mais rapidamente e, conseqüentemente, podem suportar mais clientes simultaneamente.

Informações sobre a ocupação da rede também foram coletadas. Foi utilizada uma medida conhecida como RTT. Ela mede o atraso entre dois *hosts* e corresponde ao tempo total gasto para que uma requisição deixe uma máquina, atinja a outra e retorne. Esse tempo varia de acordo com o congestionamento da rede. No caso específico deste servidor VoD, é o tempo que uma requisição bem pequena, gasta para ir do servidor até o cliente e retornar ao servidor que enviou o pacote.

É possível relacionar o RTT com a ocupação da rede, de forma que quanto maior o tempo que a informação gasta para ir e voltar, maior é a possibilidade de alta

ocupação da rede neste dado momento. Neste caso, um problema que pode acontecer é que a informação demore um tempo considerado grande para voltar, não porque a rede está congestionada, mas sim porque o servidor estava ocupado e demorou para responder. Então, o ideal é utilizar também informações sobre o tempo de acesso ao disco e RTT para verificar como está a ocupação da rede e do sistema como um todo.

5.5.6 – O Servidor de Controle

As tarefas do Servidor de Controle foram apresentadas ao longo das sessões anteriores, em conjunto com a proposta, mas sem um tópico específico para descrevê-lo. Então, neste tópico é feito um levantamento de todas as funções desempenhadas pelo Servidor de Controle e como foram implementadas.

De forma bem resumida as principais tarefas do **Servidor de Controle** são:

- Manter uma lista atualizada dos vídeos disponíveis para escolha do usuário;
- Coletar informações sobre a utilização da rede e dos discos para realizar o controle de admissão de novos clientes;
- Gerenciar os parâmetros de QoS.

5.6 – Questões de Projeto no Sistema Implementado

Conforme apresentado na seção 2.6 existem diversos tópicos relacionados com as escolhas que devem ser realizadas durante a fase de projeto de um sistema de VoD. Um resumo das principais escolhas de projeto que foram realizadas na construção do sistema de VoD são apresentadas.

- **Armazenamento**: utiliza *space striping*, ou seja, os vídeos são distribuídos usando um tamanho fixo, em bytes.
- **Recuperação dos vídeos armazenados**: utiliza uma abordagem *client-pull*, na qual os dados são enviados aos clientes depois que os servidores recebem uma requisição explícita.
- **Controle de admissão**: é feito usando um algoritmo que pode ser classificado como baseado em medidas, tais como tempo de resposta, RTT, taxa de acesso aos discos.
- **Tolerância a falhas**: nenhum mecanismo foi implementado.

- **Suporte a operações interativas:** não foi implementado nenhum mecanismo específico, uma vez que o programa utilizado para decodificar e exibir os vídeos (mplayer) possui estas operações já implementadas independente da forma de armazenamento dos vídeos.
- **Reconstrução do fluxo de vídeo:** é feita no cliente, usando uma abordagem *proxy-at-client*, que possui as vantagens já discutidas.

Capítulo 6 – Avaliação, Medidas e Resultados

O objetivo deste capítulo é avaliar o sistema de VoD proposto, apresentar os testes realizados, comparar suas características de funcionamento e seu desempenho com outros sistemas anteriormente propostos.

6.1 - Avaliação do Servidor de VoD proposto

Uma vez concluída a fase de projeto e implementação do servidor de VoD procedeu-se a avaliação do sistema obtido. Foram realizados testes para medir o desempenho do sistema proposto e também verificar a influência que a variação de certos parâmetros têm sobre o sistema.

Antes da realização dos testes de desempenho, foi adaptado um programa decodificador baseado na biblioteca libmpeg2 [LIB03], gerando um *parser* capaz de extrair informações sobre a estrutura interna da padronização MPEG. Sua construção teve como objetivo obter informações sobre os tamanhos médios dos quadros e dos GoP's para alguns vídeos que são usados nos testes.

Os valores médios obtidos para tamanhos de quadros e GoP's para alguns dos vídeos testados são apresentados na tabela 2.

Nome do Vídeo	**Tipo MPEG**	**Tamanho Médio de Quadro (bytes)**	**Tamanho Médio de GoP (bytes)**
vídeo1.mpg	1	21.807	108.635
vídeo2.mpg	1	13.952	158.267
Vídeo3.mpg	1	15.402	221.737
01681.mpeg [OPE03]	2	15.406	221.912
vídeo4.mpeg	2	22.255	159.891

Tabela 2 - Tamanhos Médios de Quadro e GoP

Através da tabela 2 é possível verificar que existe uma grande variação nos tamanhos médios de quadros e de GoP's entre um vídeo e outro, tanto na padronização MPEG-1 quanto na MPEG-2.

Os vídeos apresentados na tabela 2 do tipo MPEG-1 sofrem uma variação muito mais rápida na imagem apresentada do que os do tipo MPEG-2. Isto ocorre porque os primeiros são desenhos animados e os outros são do tipo documentário, ou seja, não estão relacionados ao tipo do MPEG usado. O tipo do vídeo assistido pode ser um parâmetro usado nos controles de QoS.

6.1.1 – Ambiente de Teste

As configurações do *hardware* das máquinas que compõem o sistema de VoD são apresentadas na tabela 3.

Descrição Componente	Cliente	Servidor
Processador	1.6 GHz	233 MHz
Memória RAM	256 MB	128 MB
Disco Rígido	40 GBytes	6 GBytes
Rede *Fast Ethernet*	100 Mbps	100 Mbps

Tabela 3: Configuração dos Clientes e Servidores

O Sistema Operacional usado tanto nos clientes quantos nos servidores é o *Linux*, sendo que todos os servidores utilizaram a mesma distribuição (SUZE 7.3). Já os clientes usaram a distribuição Conectiva 8.0.

6.2 – Testes de Armazenamento e Recuperação (leitura) dos Vídeos

Testes foram realizados com o objetivo de verificar de que forma o tamanho da requisição de leitura pode influenciar no desempenho do sistema. A primeira parte dos testes refere-se ao armazenamento e à recuperação dos vídeos no servidor de VoD proposto.

Os tempos de resposta médios calculados foram coletados quando o sistema estava sendo utilizado por apenas um cliente, variando o tamanho da requisição de leitura de 1KB a 256 KB. Os resultados são apresentados na tabela 4 e na figura 12. Foram utilizados um vídeo MPEG-1 e outro MPEG-2.

	Vídeo MPEG-1		Vídeo MPEG-2	
Tamanho da Requisição (KB)	**Tempo de Resposta Médio (ms)**	**Taxa Lida (MB/s)**	**Tempo de Resposta Médio (ms)**	**Taxa Lida (MB/s)**
1	1,472	0,663426	1,561	0,625601
2	1,879	1,039449	2,606	0,749472
4	3,032	1,288341	1,975	1,977848
8	4,817	1,621860	3,734	2,092260
16	6,754	2,313444	6,026	2,592931
32	11,93	2,619447	30,56	1,022579
64	21,042	2,970250	24,219	2,580619
128	28,928	4,321073	29,558	4,228974
256	76,631	3,262387	49,362	5,064625
512	146,154	3,421049	88,831	5,628666

Tabela 4 - Tempos de Reposta Médios para dois vídeos diferentes

Este tempo de resposta refere-se ao tempo que uma requisição demora para ser atendida, ou seja, o tempo decorrido entre a realização da solicitação de dados pela aplicação do cliente (*p_read*) e a chegada destes dados no cliente. Esta medida reflete a capacidade de recuperação de dados nos discos dos servidores (*throughput*). Quando mais de um cliente estão usando o sistema, esta medida fornece um indicativo da ocupação da rede e da utilização dos discos, de forma que quanto maior o número de clientes maior deve ser tempo de resposta. Na figura 12 é apresentado um gráfico que mostra como fica o tempo de resposta quando mais de um cliente está utilizando o sistema.

Considerando os tamanhos médios de GoP's observados nos vídeos analisados (o tamanho médio máximo obtido foi de 222 KB, conforme apresentado na tabela 2), os testes realizados consideram requisições de até 512 KB, de forma que seriam suficientes para obter-se um GoP em uma única requisição. Como quase toda dependência entre os quadros está restrita à um GoP, tamanhos maiores de requisições não devem trazer ganhos de desempenho à aplicação.

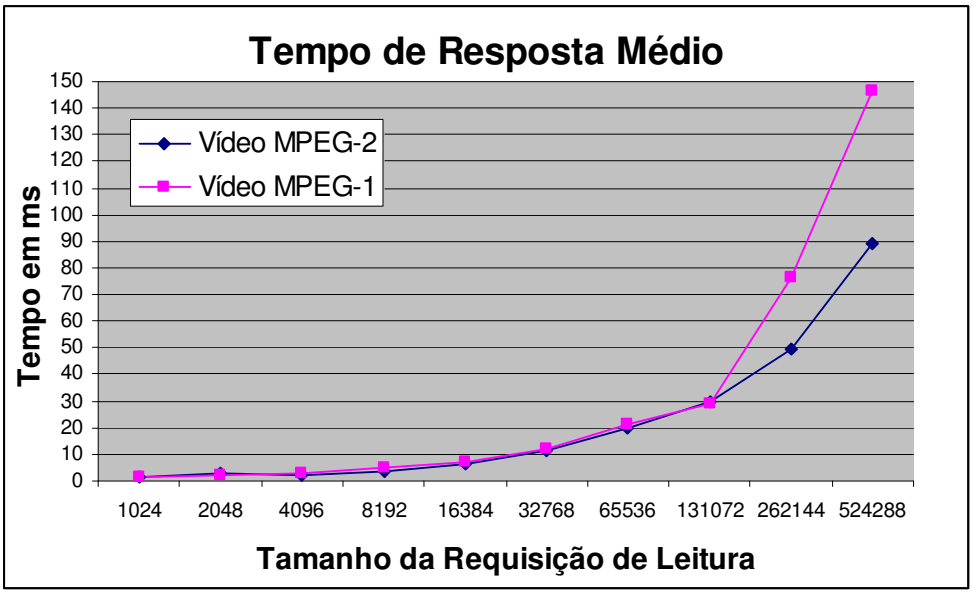

Figura 12: Gráfico dos Tempos de Resposta Médio para 4 Servidores e 1 Cliente

O tempo de resposta por si só, não diz muito sobre qual tamanho de requisição é o melhor para o sistema de VoD, é preciso considerar também a quantidade de dados que está sendo solicitada. Desta forma, a taxa de dados lida é uma medida mais objetiva. Esta taxa de dados lidos é obtida pela divisão entre o tamanho da requisição e o tempo de resposta para este tamanho e sua unidade é expressa em MB/s. Um gráfico mostrando a taxa de leitura média para os vídeos MPEG-1 e MPEG-2 é apresentado na figura 13.

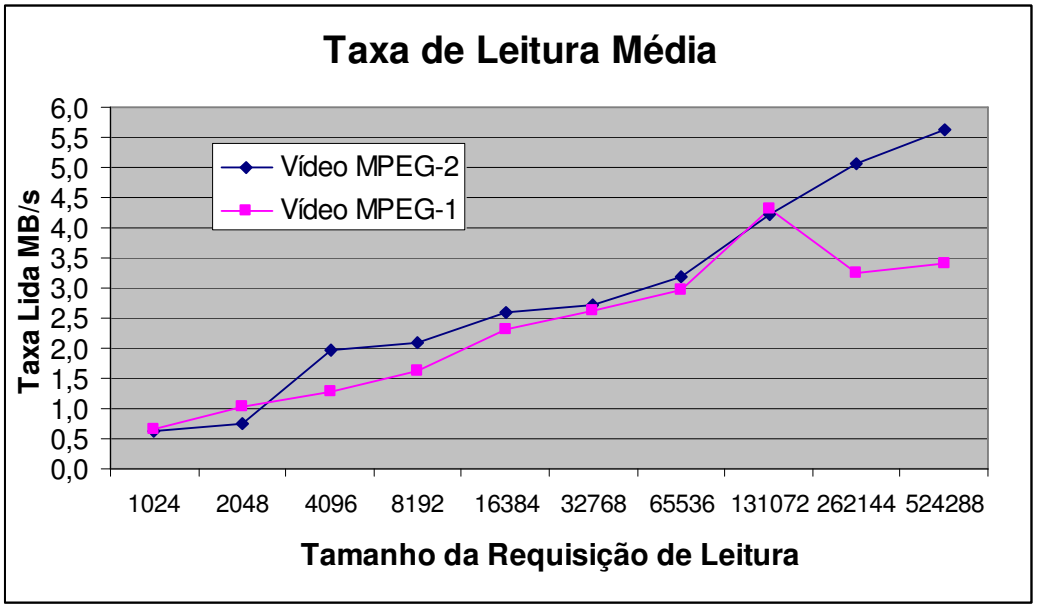

Figura 13: Taxa de Leitura Média para 4 Servidores e 1 Cliente

Através da análise da taxa de leitura é possível verificar que para o vídeo MPEG-1 o tamanho de requisição de leitura que obteve a melhor taxa de *bytes* recuperados foi o tamanho 128 KB, e para o vídeo MPEG-2 o tamanho com a melhor taxa de leitura foi o de 512 KB.

Uma situação que pode ser considerada razoável é um aumento do tempo de resposta igual ao aumento do tamanho da requisição, ou seja, dobrando o tamanho da requisição de leitura espera-se que o tempo de resposta no máximo também seja dobrado. Para o vídeo MPEG-1, durante a transição entre as requisições de 128 KB para 256 KB houve um aumento maior que o dobro no tempo de resposta e a partir deste ponto a taxa lida obtida foi sempre menor que aquela apresentada pela requisição de 128 KB. Para o vídeo MPEG-2 a mesma situação não ocorre, em nenhum caso o aumento do tempo de resposta foi maior que o dobro em relação ao tamanho de requisição anterior.

É importante salientar que este tempo de resposta apresentado para diferentes tamanhos de requisição de leitura não sofre influência do conteúdo do vídeo. Por outro lado, estes valores são influenciados pela combinação de fatores tais como: quantidade de dados requisitada (tamanho da requisição), taxa de leitura máxima suportada pelos discos do sistema, tamanho do bloco de dados utilizado pelo disco, uso do padrão de rede *Ethernet*, limitação do tamanho de *sockets* e pela ocupação da rede. A diferença ocorrida nos dados reflete possivelmente o efeito de alguma interferência na rede.

Até o momento foram apresentados os tempos de resposta e taxas de leitura em situações nas quais o sistema de VoD estava sendo utilizado apenas por um cliente. O comportamento do sistema quando está sendo utilizado por 1, 2 e 4 clientes é apresentado na figura 14 em função do tempo de resposta. Dois vídeos diferentes foram utilizados neste teste.

Analisando a figura 14 verifica-se que, de um modo geral, o comportamento do sistema foi bem estável e variou pouco para os diferentes números de clientes. Para apenas um cliente o tempo de resposta foi menor para a grande parte dos tamanhos de requisição de leitura testados devido à menor competição pelo meio de transmissão.

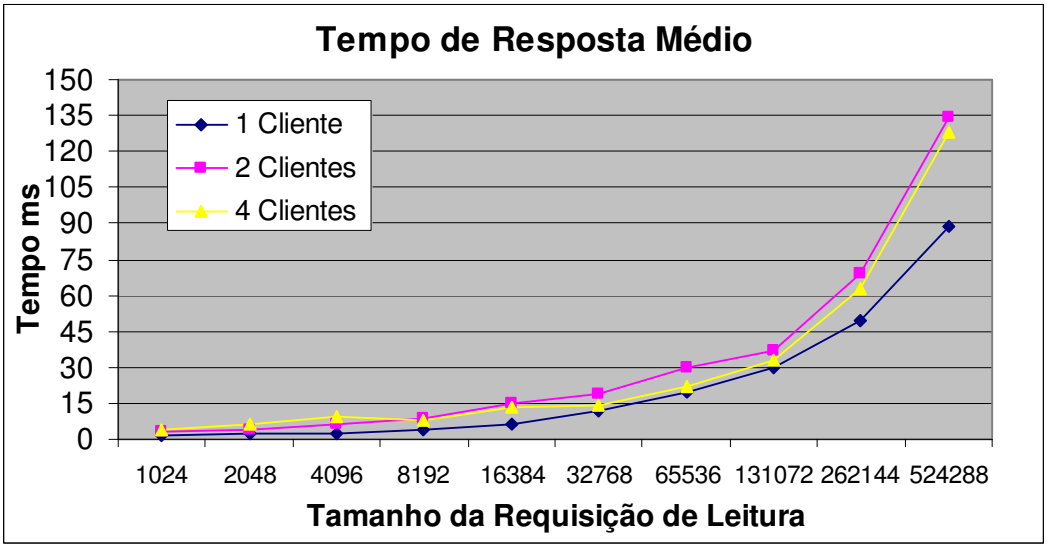

Figura 14: Tempo de Resposta para Múltiplos Clientes Simultâneos

Verificando o comportamento do sistema quando utilizado por quatro clientes observa-se que o tempo de resposta fica menor em relação aos tempos obtidos para 2 clientes. Uma possível explicação para estes resultados pode ser o fato de que nos testes envolvendo 2 e 4 clientes cada arquivo foi sempre requisitado pela mesma quantidade de clientes, ou seja, no teste de 2 clientes cada arquivo foi solicitado por um deles, no teste de 4 clientes cada vídeo foi solicitado 2 vezes. Desta forma, o tempo de resposta para quatro clientes deve ter sido melhor em função da operação de mecanismos de *cache* e *prefetching* do sistema operacional.

Isto ocorre porque uma vez lidos dos discos para um *buffer* do sistema de arquivos, os dados podem ser transmitidos para os clientes muito mais rapidamente. Usando um mesmo arquivo por cliente, a reutilização dos dados do *cache* tende a prover melhores resultados.

Para tamanhos de requisições maiores que 64 KB a fragmentação dos tamanhos de blocos solicitados aos servidores, devido ao tamanho dos *buffers* dos *sockets* prejudica o desempenho do sistema. Quando mais servidores usam a rede simultaneamente, a competição pelo meio diminui a eficiência da rede.

Conforme já discutido, é possível observar o desempenho de cada tamanho de requisição de leitura relacionando seu tamanho com o tempo de resposta correspondente. Desta forma, obtém-se um valor de taxa lida para cada tamanho. As taxas lidas para cada tamanho de requisição variando o número de clientes simultâneos são apresentadas na figura 15.

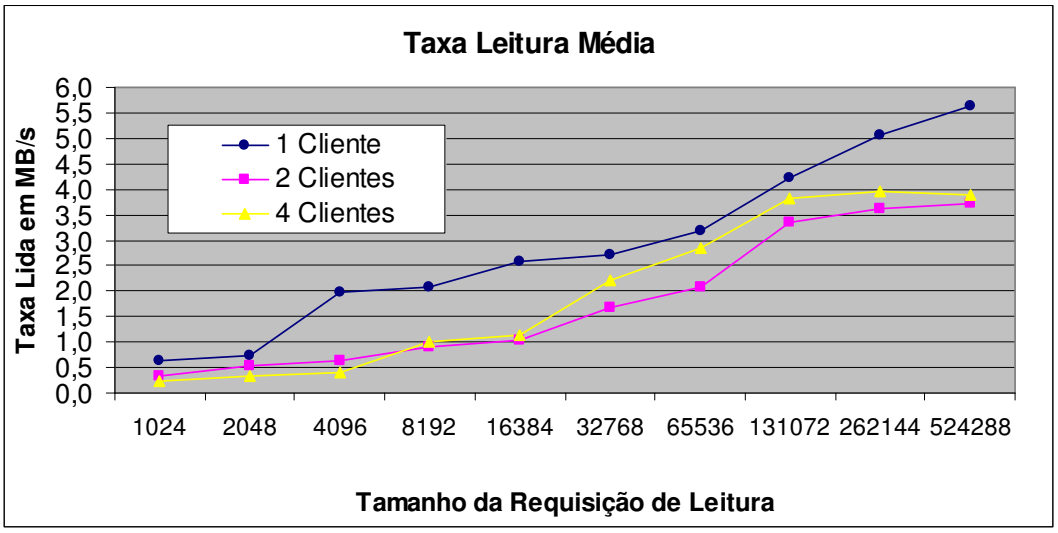

Figura 15: Taxa Lida obtida em função do Tempo de Resposta

Como era de se esperar, a taxa lida quando o sistema de VoD estava sendo usado por apenas um cliente é maior para todos os tamanhos de requisição, uma vez que todos os recursos do sistema estavam à disposição deste cliente.

Quando dois clientes estão utilizando o sistema, o cenário se modifica bastante, uma vez que agora os recursos do sistema precisam ser divididos entre estes dois clientes; além disso, neste caso, cada cliente está acessando um vídeo diferente. Nesta situação, o desempenho do sistema em termos de taxa lida fica bastante comprometido, uma vez que a maioria os valores obtidos para esta taxa são inferiores à metade da taxa obtida quando o sistema está atendendo apenas um cliente. Este cenário só se modifica para os tamanhos acima de 64KB quando a taxa lida consegue ser maior que a metade do valor da taxa obtida para um cliente. Em nenhum dos tamanhos de requisição de leitura testados a taxa lida para 2 clientes consegue ser maior que para um cliente.

Neste caso, como 4 servidores de arquivos NPFS foram utilizados, 126 KB correspondente ao valor limite para que não haja fragmentações nas requisições NPFS, uma vez que em cada requisição, cada servidor manipula 32 KB. Para requisições maiores, excede-se o limite de *buffer* dos *sockets*, o que gera fragmentação e redução de desempenho.

Quando o número de clientes simultâneos muda para quatro, o desempenho do sistema melhora consideravelmente, uma vez que um mesmo vídeo está sendo requisitado por dois clientes. Desta forma, a taxa lida para 4 clientes é maior para quase todos os tamanhos de requisição, sendo levemente menor até o tamanho 4KB e a partir

daí sempre maior. Novamente, estima-se a melhora obtida em função dos benefícios dos mecanismos de *cache* e *prefetching* do sistema operacional.

É possível analisar o desempenho do armazenamento dos vídeos através da observação de outros parâmetros do sistema como, por exemplo, o tempo gasto no acesso aos discos. Em aplicações de VoD o número de operações de leitura é muito maior que o número de operações de escrita, uma vez que o segundo tipo de operação só ocorre durante o armazenamento dos vídeos. Desta forma, para que o desempenho do sistema seja melhorado, as operações de leitura realizadas para o acesso aos vídeos são as que mais devem ser otimizadas. Entretanto, é importante que a escrita de dados dos vídeos no sistema de arquivos ocorra seguindo a melhor configuração para a leitura.

Sendo assim, os testes envolvendo acesso aos discos restringem-se à coleta de informações sobre as operações de leitura. Os tempos médios gastos na realização de operações de leitura envolvendo 1, 2 e 4 clientes são apresentados na figura 16.

As taxas de leitura apresentadas são bastante superiores à taxa máxima fornecida pelos discos em função da operação de *cache* realizada pelo sistema operacional. A taxa máxima dos discos do sistema de VoD em questão, sem utilizar o *cache* é de 7,77 MB/s e de 32 MB/s considerando a eficiência máxima do *cache*. (Dados obtidos utilizando o comando hdparm –tT /dev/hda.)

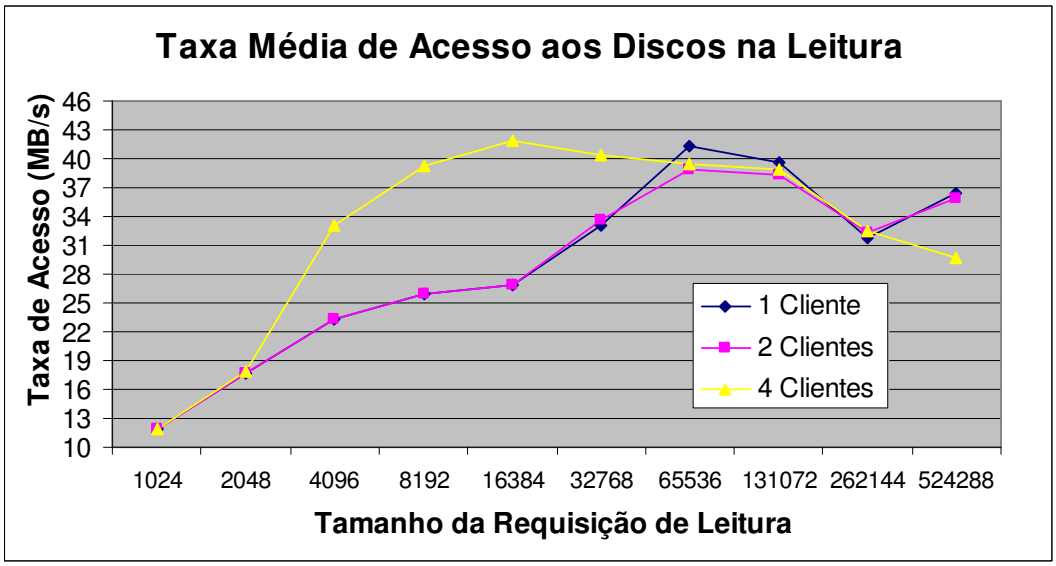

Figura 16: Taxa Média de Acesso aos Discos

Nos testes realizados para medir as taxas de leitura dos discos para diversos clientes simultâneos foram utilizados dois arquivos diferentes, sendo uma deles MPEG-

1 e o outro MPEG-2. Com os dados apresentados no gráfico da figura 16 tem-se uma indicação do melhor fator de operação dos discos na leitura de dados. Os resultados obtidos quando o sistema está servindo um e dois clientes são muito semelhantes porque nos dois casos um vídeo foi solicitado por apenas um cliente, ou seja, não existe efeito do cache.

Quando 4 clientes estão usando o sistema, uma boa parte das taxas são melhores em relação à 1 e 2 clientes pelo fato de que um mesmo vídeo está sendo requisitado por mais de um cliente, desta forma deve-se considerar a presença de dados em *buffers*.

O tempo médio de ida e volta de mensagens (RTT) na comunicação dos clientes com os servidores também foi medido e juntamente com o tempo de acesso aos discos fornece uma idéia da ocupação da rede. Os valores obtidos de RTT para um e vários clientes são apresentados na figura 17.

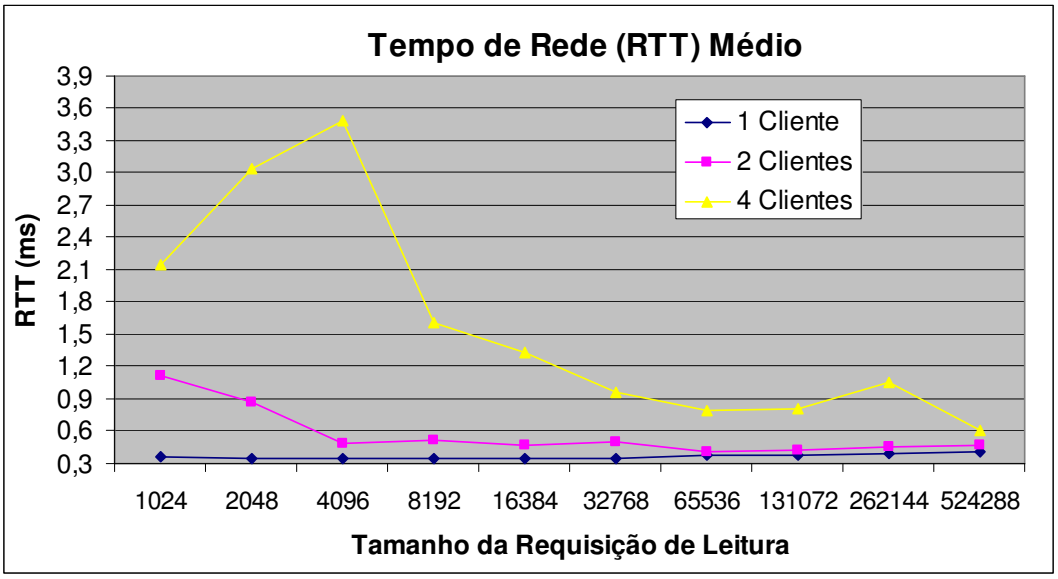

Figura 17: RTT para 1, 2 e 4 Clientes

O valor de RTT varia em função da competição pelo meio de transmissão. Para atender requisições de tamanhos e quantidades maiores é preciso transmitir um número maior de quadros, aumentando assim a competição. Sendo assim, para 1 e 2 clientes os valores de RTT não são muito altos comparados aos valores obtidos para 4 clientes, uma vez que a competição ao meio aumenta em função do aumento no número de clientes simultâneos. Além do aumento significativo apresentado para 4 clientes é importante ressaltar os picos que ocorreram. Isto se deve ao fato do protocolo utilizado

ser não determinístico, de forma que um certo servidor ou cliente possa ficar várias vezes sem conseguir transmitir enquanto outro consegue usar o meio de transmissão mais vezes. Este picos ocorreram sobretudo para 4 clientes que é justamente a situação na qual a competição ao meio se tornou maior.

6.3 – Medidas do *Player*

Medidas sobre o funcionamento do decodificador e do visualizador (*player*) foram coletadas com o objetivo de verificar se as situações nas quais o desempenho da aplicação do cliente são melhores coincidem com aquelas que fornecem os melhores fatores de operação do servidor de VoD. As principais medidas coletadas foram o número de quadros descartados, o atraso máximo e médio entre o áudio e o vídeo.

A figura 18 apresenta o descarte médio de quadros feitos pelo *player* para os diversos números de clientes.

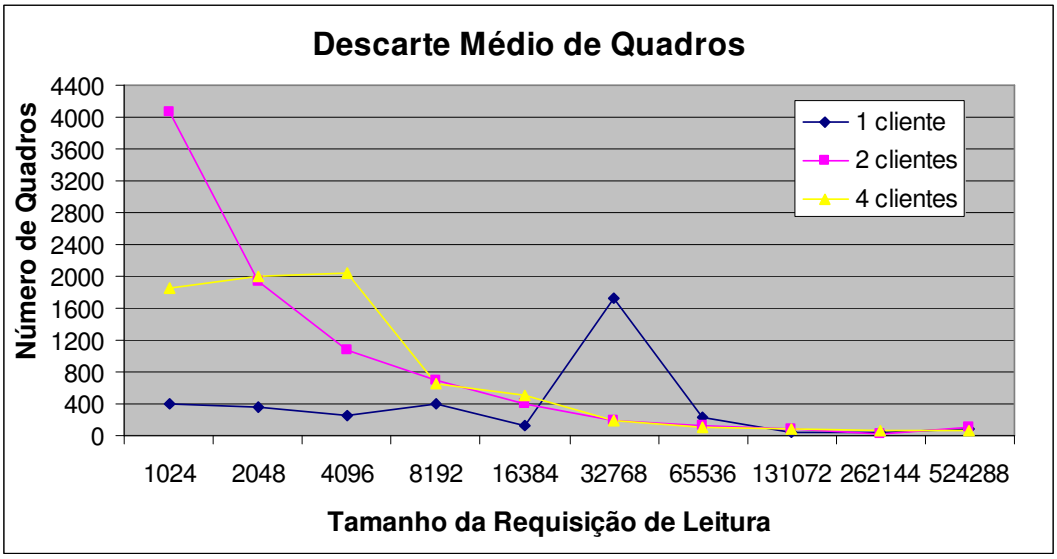

Figura 18: Média de Quadros Descartados

Analisando o gráfico da figura 18 pode-se perceber que os tamanhos de requisições de leitura (256KB e 512 KB) que apresentam os menores descarte coincidem com aqueles que apresentam as melhores taxas de leitura dos discos e, em conseqüência disto, os melhores tempos de resposta.

O descarte de quadros é feito em função da quantidade de atraso existente entre o *time stamp* do áudio e o do vídeo, com o objetivo de fazer uma correção neste atraso

evitando que cada vez mais o vídeo fique sem sincronia entre o áudio e o vídeo. Este atraso pode ser um valor tanto positivo quanto negativo, indicando assim se o fluxo de vídeo está adiantado ou atrasado em relação ao fluxo de áudio. Se o valor do atraso for positivo e maior que 10 ms é feito o descarte para correção; se o atraso for negativo não é feita correção.

Devido às correções que são feitas pelo decodificador e o *player* o atraso entre o áudio e o vídeo não cresce indefinidamente.

Características do vídeo também influenciaram a ocorrência de um atraso maior ou menor; como conseqüência, o vídeo MPEG-1 teve em média um número de descarte bem menor que o MPEG- 2 para todos os tamanhos de requisição. Não foi possível determinar se isto ocorreu em função da padronização utilizada ou do conteúdo do vídeo em si. Mais testes precisariam ser feitos para concluir algo a respeito disto.

Até um certo limite, os descartes não atrapalham a qualidade percebida pelo usuário do sistema de VoD, mas passando deste limite a Qualidade de Serviço é prejudicada e o usuário pode perceber a falta de sincronismo entre o áudio e o vídeo e em casos mais graves pode acontecer que o *player* comece a exibir o vídeo sem áudio ou somente o áudio. Estes problemas mais graves começam a aparecer quando o atraso entre o áudio e o vídeo aumenta a ponto de se tornar alguns segundos. Não existe um valor exato de atraso entre áudio e vídeo para o qual a qualidade visual e sonora do vídeo começa a decair, pois ela é dependente do conteúdo. Em vídeos mais rápidos, como os filmes de ação, com certeza os quadros descartados fazem muito mais falta que em filmes em que a seqüência de imagem muda bem mais lentamente.

Capítulo 7 - Conclusões

Neste capítulo são apresentadas as considerações finais, as conclusões sobre os resultados obtidos com a realização deste trabalho e os possíveis trabalhos futuros.

7.1 - Conclusões

Antes de apresentar as conclusões deste trabalho é necessário fazer algumas considerações finais. A primeira delas é salientar que as discussões realizadas por este trabalho tiveram como principal objetivo fazer uma análise sobre a utilização de sistemas de arquivos paralelos na construção de servidores de vídeo sob demanda e entender os diversos fatores relacionados. A utilização do NPFS se deu pela disponibilidade de seu código fonte e pela possibilidade de realizar adaptações que pudessem facilitar a coleta de medidas de desempenho. De maneira geral, acredita-se que as medições realizadas e os resultados obtidos possam ser extrapolados e válidos para a construção de servidores de VoD utilizando qualquer sistema de arquivos paralelos.

O número máximo de clientes simultâneos suportados é uma medida importante para caracterizar o desempenho de um sistema de VoD. O desempenho do sistema varia bastante em função do tamanho da requisição de leitura, de forma que o número de clientes simultâneos também sofre influências.

Devido às limitações do ambiente de testes utilizado neste trabalho, com relação ao número de computadores disponíveis para realizarem o papel de clientes, não foi possível utilizar mais que quatro clientes simultaneamente, de forma que o número máximo de clientes suportados não pôde ser obtido experimentalmente.

Considerando os valores obtidos nos testes realizados é possível concluir quais são os melhores tamanhos de requisição de leitura com relação aos diversos tópicos considerados e estudados. Sob o ponto de vista do armazenamento, os tamanhos de requisição de leitura que obtiveram um menor tempo de resposta foram as de tamanho grande como, por exemplo, 256 KB e 512 KB, que apresentaram melhores resultados para as diferentes quantidades de clientes testados.

Com relação à ocupação da rede, medida pelo RTT, para 1 cliente os tamanhos de requisição de leitura que apresentaram os valores mais baixos foram os menores,

entre 4KB e 16 KB, para 2 e 4 clientes os melhores tamanhos estão entre 128 KB e 512 KB. Extrapolando este resultado é possível prever que estes tamanhos maiores são os que deverão suportar um maior número de clientes simultâneos, mas isto deve ser comprovado experimentalmente usando mais clientes.

Verificando os valores obtidos para as taxas de leitura dos discos, os tamanhos de requisição de leitura que forneceram os melhores valores foram 64 KB e 128 KB para 1 e 2 clientes e 16KB e 32 KB para 4 clientes.

Considerando as medidas obtidas no *player*, principalmente com relação ao descarte de quadros, observa-se que apenas para tamanhos de requisição pequenos ocorre descarte de quadros, a partir da requisição de tamanho 64 KB o número de quadros descartados é praticamente nulo.

O conhecimento gerado através do estudo da padronização MPEG é de grande importância, uma vez que sua utilização é muito ampla e não se restringe apenas à aplicação de vídeo sob demanda, podendo ainda dentro do contexto da aplicação de VoD servir como ponto de partida para várias linhas de pesquisa, como Qualidade de Serviço, utilização de tamanhos de blocos para armazenamento dos vídeos em função dos tamanhos de quadros e GoP's, gerenciamento de operações interativas, entre outras.

O assunto proposto por este trabalho não está totalmente esgotado, muitas questões ainda podem ser discutidas, testadas e analisadas. O objetivo final deste trabalho foi obter um sistema de Vídeo sob Demanda onde diversos clientes usando vários servidores com vídeos armazenados segundo a melhor distribuição para cada um e com mecanismos de monitoramento da rede, dos discos e das folgas em cada cliente para manter a QoS e maximizar o número de clientes.

7.2 – Trabalhos Futuros

O assunto abordado neste trabalho é bastante amplo e algumas questões ainda podem ser aprofundadas. Para isto é necessário ainda a realização de algumas implementações e testes. Sem dúvida, é bastante interessante realizar os mesmos testes já realizados no *Fast Ethernet* usando *Gigabit Ethernet*, tornando possível comprovar que usando uma rede e discos mais rápidos os benefícios do uso de um sistema de arquivos paralelos distribuídos são maiores. Outro ponto que deve ser explorado é usar as medidas já coletadas efetivamente no controle de admissão dinâmico e também para as garantias de QoS. A possibilidade de renegociação da QoS contratada é também uma das

otimizações que pode ser realizada. E por fim, realizar testes com as duas formas previstas de operação do *prefetching*, podendo assim comprovar sua eficiência.

Glossário

Batching: É uma técnica que visa aumentar o número de usuários simultâneos suportados num sistema de Vídeo Sob Demanda. Sua idéia básica consiste em reter um certo número de requisições dos usuários e verificar a existência de solicitações para a mesma porção de um determinado vídeo, podendo assim atender diversas requisições através de um mesmo fluxo.

Client-pull: É uma arquitetura de serviço de entrega de dados na qual o servidor só envia dados aos clientes mediante requisição explícita do cliente para envio destes dados.

Crominância: É uma característica da imagem que é definida por dois valores: coloração e saturação, a coloração é a parte de luz refletida por um objeto. Este absorve luz e reflete apenas uma parte do espectro visível; a saturação define a proporção de branco que uma cor contém.

Group of Pictures **(GoP):** Grupo de imagens (quadros). É um dos níveis da hierarquia apresentada pela especificação MPEG-2 para a camada de vídeo. É formado por vários quadros dos três tipos existentes (I, P, B). Opcionalmente podem existir cabeçalhos para estes GoPs no fluxo MPEG-2.

JPEG ***(Joint Photographic Experts Group):*** este formato de arquivo (extensão .jpg ou .jpeg) é muito usado para imagens na Internet. Não está limitado a 256 cores e, por isso, pode utilizá-lo para apresentar fotografias de alta qualidade ou imagens que contenham milhões de cores. Uma vez que foi concebido como um formato de armazenamento de imagens, pode comprimir de forma eficiente fotografias de alta qualidade de grandes dimensões em arquivos muito compactos, o que o torna muito útil quando pretende enviar uma imagem grande numa mensagem de correio eletrônico. No entanto, quanto mais se reduz o tamanho do arquivo da imagem (ou modifica e salva novamente a imagem), mais informações da imagem são eliminadas e mais a qualidade diminui. Além disso, este formato não suporta transparência nem animação.

***Packetized Elementary Streams* (PES):** Os fluxos elementares de áudio e vídeo depois de codificados e antes de serem multiplexados passam por um processo de "empacotamento" através do qual são gerados pacotes que recebem o nome de **pacotes PES**. Estes pacotes PES podem ser de áudio ou de vídeo. Depois de multiplexados estes pacotes PES podem formar fluxos de programa (PS) ou fluxo de transporte (TS).

Pixel: É o menor ponto de luz cuja cor e luminosidade que podem ser controlados na tela.

***Prefetching*:** Técnica utilizada para minimizar o tempo de resposta às requisições feitas pelas aplicações dos usuários. Seu funcionamento é baseado na **leitura antecipada** dos dados. Quando a requisição chega no servidor, os dados requisitos já foram lidos do disco e estão na memória principal facilitando assim o envio destes ao usuário que fez a solicitação.

***Presentation Time Stamps* (PTS):** É responsável pela sincronização entre múltiplos fluxos elementares e faz parte do projeto do decodificador.

***Program Clock Reference* (PCR):** É usado na sincronização do sistema de decodificação nos fluxos de transporte.

***Program Stream* (PS):** O fluxo de programa é um conjunto de fluxos elementares MPEG (áudios, vídeos, informações auxiliares) que são relacionados entre si, e apresentam, desta forma, a mesma base de tempo.

Proxy: É um módulo responsável por juntar os diversos "pedaços" do vídeo que são recuperados do sistema de arquivos, quando arquivos paralelos são utilizados. Sua tarefa é recompor os dados formando um único fluxo de dados.

Qualidade de Serviço (QoS): pode ser definida como o conjunto de características de um sistema necessário para atingir uma determinada funcionalidade. Visa garantir o cumprimento dos parâmetros exigidos pelo usuário.

Quantização: O efeito de quantização é dado pela impossibilidade de se medir uma faixa infinita de valores de intensidade dos *pixels*. Portanto a quantização, ou

discretização é a codificação dos valores contínuos de um sinal em intervalos discretos. Em geral os equipamentos só medem 8 bits, correspondentes a 256 níveis de cinza, mas a tecnologia atual chega a 24 bits. O número de *bits* por *pixel* corresponde ao número de cores ou tons de cinza que podem ser representados.

Round Trip Time (**RTT**): Mede o atraso entre dois computadores de uma rede. Corresponde ao tempo total gasto para que uma requisição deixe uma máquina, atinja a outra e retorne.

Server-push: É uma arquitetura de serviço de entrega de dados na qual o servidor envia os dados aos clientes sem necessitar que o cliente faça requisições para cada parte dos dados a serem enviados. Uma vez iniciada a sessão, o servidor envia dados até que o cliente envie uma requisição pedindo para parar, ou até que o arquivo termine.

Stripe unit: É a unidade de distribuição usada no armazenamento de um arquivo paralelo. Denota a quantidade máxima de dados armazenados logicamente de maneira contínua em um único disco.

***System Clock Reference (*SCR):** É usado na sincronização do sistema de decodificação. Este campo aparece nos fluxos de programa. SCR e PCR são também "selos de tempo" codificando a temporização dos fluxos de *bits*.

System Time Clock **(STC):** É o relógio comum do sistema. Todo modelo de temporização do sistema é definido em termos deste relógio. Nos fluxos de transporte (**TS**), este relógio deve ter a mesma freqüência das amostras de vídeo e de áudio; nos fluxos de programa (**PS**) uma pequena variação é permitida.

Transport Stream (TS): O fluxo de transporte combina um ou mais programas com uma ou mais bases de tempo independentes em um único fluxo.

Vídeo sob Demanda (VoD): é o serviço no qual clientes solicitam aos servidores vídeos que foram previamente armazenados e os servidores recuperam e enviam as informações solicitadas.

Referências Bibliográficas

[BER98] BERZSENYI, M.,VAJK, I., ZHANG H., "Design and Implementation of a Video on-demand System", Computer Networks and ISDN Systems, N. 30, 1998, pp. 1467-1473.

[BOL96] BOLOSKY, W. et al "The Tiger Video Fileserver," in Proc. 6th International Workshop on Network and Operating System Suport for Digital Audio and Video, Zushi, Japan, April, 1996.

[CAL00] CALVAGNA, A., PULIAFITO, A. and VITA, L. "Design and Implementation of a Low-Cost/High-Performance Video on Demand Server". In IEEE Int. Conf. on Computer and Communication Networks (ICCCN'99), Boston, MA USA, October 1999.

[CHE94] CHEN, P. M., LEE, E. K., GIBSON, KATZ, R. H., PATTERSON, D. A., "RAID: High-Performance, Reliable Secondary Storage", ACM Computing Surveys, Vol.26, Nº 2, June 1994, pages 145-185.

[CHE96] CHEAN, M. and KANDLUR, D. D. "Stream Conversion to Support Interactive Playout of Videos in a Client Station". IEEE Multimedia, 3(2), Summer, 1996.

[CHI97] CHIUEH, T., VENKATRAMANI, C. and VERNICK M., "Design and Implementation of the Stony Brook Video Server" in Software - Practice and Experience, February, 1997.

[CRO89] CROCKETT, T.W. "File Concepts for Parallel I/O". Proceedings of Supercompunting'89, 1989, pp.574-579.

[FLY98] FLYNN, R. J. and TETZLAFF, W. H. "Multimedia – An introduction". IBM J. Res. Develp., Vol. 42, N. 2, March 1998.

[FRE95] FREEDMAN, C.S., BURGER, J. and DEWITT, D. J. "The SPIFFI Scalable Video-on-Demand System". Proceeding of the ACM SIGMOD International Conference on Management of data, pp.352-363, May 1995, San Jose, California, United States.

[FRE96] FREEDMAN, C.S., BURGER, J. and DEWITT, D. J. "SPIFFI-A Scalable Parallel File System for the Intel Paragon". IEEE Transactions on Parallel and Distributed Systems, Volume 7, N. 11, November 1996, pp.1185-1200.

[FRI96] FRIEDRICH, J.**,** GRUN T. and KELLER, J. "Video-on-Demand on the SB-PRAM. In Proc. of the 6th International Workshop on Network and Operating System Suport for Digital Audio and Video, NOSSDAV'96, Zushi, Japan, April, 1996, pages 105-111.

[GAF00] GAFSI, J. and BIERSACK, W. "Modeling and Peformance Comparison of Reliability Strategies for Distributed Video Servers". IEEE Transactions on Parallel and Distributed Systems, Vol.11, N.4, April 2000.

[GOL01] GOLUBCHIK, L., MUNTZ, R. R., CHOU, C. and BERSON, S. "Design of Fault-Tolerant Large-Scale VoD Servers: With Emphasis on High-Performance and Low-Cost". IEEE Transactions on Parallel and Distributed Systems, Vol.12, N.4, April 2001.

[GUA99] GUARDIA, H. "Considerações sobre as estratégias de um Sistema de Arquivos Paralelo integrado ao processamento distribuído". Tese de Doutorado EPUSP, 1999.

[HAF95] HAFID et al. "An Approach to Quality of Service Management in Distributed Multimedia Application: Design and an Implementation". In Proceedings of International Conference on Open Distributed Processing (ICODP'95), February 1995.

[HAS98] HASKIN, R. L. "Tiger Shark a scalable file system for multimedia". IBM Journal of Research and Development, 1998.

[ISO94] ISO/IEC 13818-1 | ITU-T H.222.0 - "Information Technology - Generic Coding of Moving Pictures and Associated Audio Information: System", 1994.

[ISO95] ISO/IEC 13818-2 | ITU- T H.262 - "Information Technology - Generic Coding of Moving Pictures and Associated Audio Information: Video", 1995.

[ISO02] ISOVIC, D., FOHLER, G. "Analysis of MPEG-2 Video Streams" MRTC Report (Mälardalen Real-Time Research Centre), Mälardalen University, August 2002

[LEE95] LEE, Y.B., and Wong, P.C. "VIOLA – Video on Local-Area-Networks in Proc. 2nd ISMM/IASTED Int. Conf. Multimedia Systems and Applications, Stanford University, Stanford, CA, Aug. 1995, pp. 101—104

[LEE97] LEE, Y.B., and Wong, P.C. "VIOLA - A Scalable and Fault-Tolerant Video-on-Demand System," Proc. Hong Kong International Computer Conference HKICC'97, October 1997.

[LEE98] LEE,Y.B. "Parallel Video Servers – A Tutorial"- IEEE Multimedia,vol.5(2), June 1998, pages 20-28.

[LEE01] LEE, Y.B. "Supporting Server-Level Fault Tolerance in Concurrent-Push-Based Parallel Video Servers". IEEE Transactions on Circuits and Systems for Video Technology, vol. 11 , N.1, January 2001.

[LIB03] Libmpeg2 Home Page "*libmpeg2 a free MPEG-2 video stream decoder*" – URL: http://libmpeg2.sourceforge.net/ Consultado em 15/04/2003.

[LIM01] LIMA, P. J. A., FILHO, G. L. S. and PAULA, V. C. C. "Especificação e implementação do DynaVideo VoD". Boletim bimestral sobre tecnologia de redes produzido e publicado pela RNP – Rede Nacional de Pesquisa, Julho 2001, Vol. 5, N. 4.

[LU96] LU, GUOJUN. "Communication and Computing for Distributed Multimedia Systems", Artech House, Inc, 1996.

[MAT94] MATTISON, P. E. "Practical Digital Video with Programming Examples in C", Wiley Professional Computing, 1994.

[MPE03] MPEG Home Page - URL: http://mpeg.telecomitalialab.com. Consultado em 07/04/2003.

[MPL03] MPlayer Home Page – The Movie Player for Linux – URL: http://www.mplayerhq.hu . Consultado em 07/04/2003.

[NG00] NG, J. K. "A Multi- Server Design for a Distributed Video System that Supports Arbitrary-Rate Playback". The Journal of Systems and Software, N .51, 2000, pp. 217-227.

[OPE03] "The Open Video Project" Home Page – URL: http://www.open-video.org/ Consultado em 15/04/2003.

[PAR00] PARK, S., PARK, Y., KIM, G. M. and CHUNG, K. D. "Design and Implementation of the Parallel Multimedia File System Based on Message Distribution". Proceedings of the eighth ACM international conference on Multimedia October 2000.

[PARK00] PARK, C., SON, Y., LEE, M. and KWON, O. "Design and Implementation of Linux Clustered VoD System". In Proceedings of the International Conference

on Parallel and Distributed Processing Techniques and Applications (PDPTA'2000), Las Vegas, Nevada, USA, June 2000.

[PAT88] PATTERSON, D. A., GIBSON, G. A. and KATZ, R. H. "A Case for Redundant Arrays of Inexpensive Disk (RAID)". In Proceedings of the ACM conference on management of data (SIGMOD'88), 1988, pp. 109-116.

[PIN02] PINHO, L. B., AMORIM, C. L. And ISHIKAWA, E. "GloVE: A Distributed Environment for Low Cost Scalable VoD Systems". In Proceedings of the 14th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'02)**,** Vitória, ES, Brazil, October 2002,

[RAO96] RAO, S. S., VIN, H.M. and TARAFDAR, A. "Comparative Evaluation of Server-push and Client-pull Architectures for Multimedia Servers"**,** In Proceedings of the 6th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'96), Zushi, Japan, April 1996.

[SAN97] SANDSTA, O., LANGORGEN, S., MIDTSTRAUM, R. "Video Server on an ATM Connected Cluster of Workstations" XVII International Conference of the Chilean Computer Science Society, SCCC'97, Valparaiso, Chile, November 1997.

[SANT97] SANTOS, J. R. and MUNTZ, R. "Design of the RIO (Randomized I/O) Storage Server". Technical Report #970032, UCLA Computer Science Department, May 1997.

[SHE97] SHENOY, P. J., VIN, H. M. "Efficient Striping Techniques for Multimedia File Servers". In Proceedings of the 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97), May 1997, pages 25-36.

[SHE98] SHENOY, P. J., GOYAL, P., RAO, S. and VIN, H.M. "Symphony: An Integrated Multimedia File System". In Proceedings of ACM/SPIE Multimedia Computing and Networking 1998 (MMCN'98), San Jose, Pages 124-138, January 1998.

[SHE99] SHENOY, P. J., VIN, H. M. “Efficient Striping Techniques for Variable bit rate Continuous Media File Servers”, Journal Performance Evaluation, N. 38, 1999, pages 175-199.

[SHE01] SHENOY, P. J., VIN, H. M. “Multimedia Storage Servers”. In Readings in Multimedia Computing, Kevin Jeffay et. al. (Editors), Morgan Kaufmann Publishers, 2001.

[TEW96] TEWARI, R., DIAS D.M. et al, “High Availability in Clustered Multimedia Servers”, In Proceedings of the IEEE International Conference on Data Engineering, New Orleans, February 1996, pp. 345-354.

[TSA00] TSAO, S., HUANG, Y and DING, J. “Performance Analysis of Video Storage Server under Initial Delay Bounds", Journal of Systems Architecture, Vol. 46, No. 2, 2000, pp. 163-179

[VER96] VERNICK, M., Venkatramani, C., Chiueh, T. “Adventures in Building the Stony Brook Video Server” in Proc. of ACM Multimedia '96, Boston, MA, 1996.

[WON97] WONG, P.C. and LEE, Y. B. “Redundant Array of Inexpensive Servers (RAIS) for On-demand Multimedia Services,” in Proc. ICC’97, Montreal, Canada, June 1997, vol.2 pp. 787-792

[WU97] WU, M.Y. and W. Shu. “Scheduling for Interactive Operations in Parallel Video Servers. Proc. of 97 IEEE Conf. on Multimedia Computing and Systems”, pages 178-185, Ottwa, Ontario, Canada, 1997.