PATRÍCIA SEREDA

SERVIDOR DE VÍDEO SVFSERVER

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Dr. Roberto A. Hexsel

CURITIBA
2003

Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, da aluna *Patrícia Sereda*, avaliamos o trabalho intitulado, "*Servidor de Vídeo SVFserver*", cuja defesa foi realizada no dia 27 de agosto de 2003, às quatorze horas, no Anfiteatro B da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação da candidata.

Curitiba, 27 de agosto de 2003.

Prof. Dr. Roberto André Hexsel
DINF/UFPR - Orientador

Prof. Dr. João Cesar Netto
INF/UFRGS - Membro Externo

Profª. Dra. Cristina Duarte Murta
DINF/UFPR – Membro Interno

Resumo

Este trabalho descreve a implementação do servidor de vídeo *SVFserver (Stored Video File server)* que envia fluxos de vídeo gerados a partir de arquivos armazenados em disco para qualquer programa cliente que reproduz vídeo. Este trabalho contém descrições e avaliações de desempenho da implementação de três métodos de envio de fluxos de vídeo que foram testados no servidor. É descrito o algoritmo de controle de banda de rede que utiliza uma forma determinística para estimar a banda utilizada por cada arquivo de vídeo. Neste algoritmo foram incluídos filtros que auxiliam na determinação da estimativa da banda utilizada, permitindo um aumento médio de até 40% no número de requisições aceitas pelo servidor e 50% na utilização da banda de rede sem perda de qualidade nos clientes. É descrito o algoritmo de controle de acesso a disco que determina a ocorrência de sobrecargas neste recurso. Este algoritmo foi implementado considerando a forma aleatória para o armazenamento dos arquivos de vídeo no disco, mas sem o uso de algoritmos específicos de escalonamento de leitura de arquivos de vídeo. Também é descrito o algoritmo de controle de admissão que recusa impede novas requisições quando algum recurso vier a se sobrecarregar.

Abstract

This dissertation describes the *SFVserver (Stored Video File server)*. This server transmits video streams to client programs, from video files stored on disk. Three algorithms for transmitting video streams are described and analysed with respect to performance and network capacity utilization. A deterministic algorithm for bandwidth control was implemented to estimate the effective video bandwidth of the video files. This algorithm employs a low-pass filter to improve the estimated bandwidth allowing for a 40% increase (average) in the number of requisitions accepted, and a 50% increase on network capacity utilization. An algorithm to control disk utilization was implemented and test results were poor because of the random placement of video files on the physical disks. An algorithm for addmission control was also implemented. It denies new requests if there is risk of loss of synchronization at the clients because of resource overload in the server.

Agradecimentos

Agradeço ao meu orientador prof. Roberto Hexsel que confiou na minha capacidade de realizar um trabalho de mestrado e a todos os professores que me apoiaram neste projeto.

Agradeço ao Gabriel, Willian e Cleverson que me ensinaram a trabalhar com o Linux e a programar em C. Agradeço também a todos os outros meninos do PET que sempre tiveram paciência e humildade em me ajudar.

Agradeço a todos os meus colegas de mestrado que me apoiaram e me incentivaram a superar as dificuldades encontradas no curso.

Agradeço aos meus pais que sempre me incentivaram a estudar.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O desenvolvimento de programas que utilizam multimídia, o aumento do interesse do público em geral em áudio e vídeo digitais e sua comercialização na Internet vem contribuindo para introdução de serviços que disponibilizam o envio de fluxos de áudio e vídeo por parte de provedores a clientes conectados em redes locais ou clientes conectados em redes de alta velocidade.

Atualmente o número de servidores que oferecem este tipo de serviço, que são de domínio público, e podem ser executados no sistema operacional Linux é pequeno. Encontram-se muitos servidores proprietários como o Windows Media Server [wms03] e o Clipstream Vídeo [cli03], e servidores de vídeo implementados diretamente em hardware como o Axis 2401 [axi03] e o Axonix VOD Server [axo03], mas são poucos os servidores de domínio público. Alguns servidores de domínio público disponibilizados estão ainda em fase de desenvolvimento, como por exemplo o servidor DV1494d [dv103]. Dentre os servidores disponíveis, muitos oferecem arquivos de áudio e vídeo em formatos específicos exigindo que a reprodução destes arquivos ocorra em programas clientes dependentes do formato codificado pelo servidor, limitando ainda mais a difusão deste tipo de serviço.

Considerando a pouca oferta destes servidores em código aberto e que são executados em Linux, no decorrer do trabalho aqui relatado foi implementado um servidor de vídeo que possibilita o envio de fluxos de vídeo de arquivos armazenados no disco a qualquer programa cliente que o reproduza sem a necessidade de armazenamento local. O desenvolvimento de um servidor de vídeo exige a implementação de funções de controle de recursos que limitam o número de requisições aceitas, para evitar que estes recursos sejam sobrecarregados. Um dos recursos que deve ser controlado é a banda de rede.

Nos servidores de vídeo que trabalham com arquivos de vídeo de banda variável, podem

Nos servidores de vídeo que trabalham com arquivos de vídeo de banda variável, podem ser implementados métodos determinísticos para estimar a banda de rede utilizada por cada requisição. Os métodos determinísticos estimam a banda utilizada pela maior quantidade de dados transmitida em um determinado intervalo de tempo, e este valor muitas vezes corresponde a uma quantia muito grande se comparado com a quantia média transmitida durante todo o período de reprodução do vídeo. A quantia de dados transmitida, se considerada como a banda utilizada por um determinado arquivo, contém rajadas de valores muito superiores ao valor médio transmitido. Os valores estimados através destas rajadas fazem com que o servidor super-estime o valor da banda consumida por cada requisição, diminuindo assim o número de requisições aceitas pelo servidor, piorando assim o seu aproveitamento.

Para aumentar a utilização do servidor neste trabalho é proposto um novo método de estimativa determinística da banda de rede consumida por arquivos de vídeo. Este método é baseado na utilização de filtros para melhor estimar o valor da banda de rede, grandeza que é utilizada no controle de sobrecarga deste recurso. Estes filtros levam em consideração a quantidade de dados transmitidos em intervalos de tempo contíguos ao calculado, possibilitando estimar a banda de rede através da distribuição da carga no decorrer do tempo, evitando que rajadas de dados isoladas influenciem o valor da banda estimada.

1.1 Servidor SVFserver

Este trabalho descreve o projeto e o desenvolvimento do servidor de vídeo chamado *SVFserver (Stored Video File server)* para rede local que serve fluxos de vídeo a partir de arquivos armazenados em disco. Os formatos destes vídeos são de domínio público e podem ser reproduzidos por qualquer programa cliente que possa decodificá-los. O servidor foi desenvolvido para ser executado no sistema operacional Linux e utiliza o protocolo *TCP/IP* para o transporte dos fluxos de vídeo.

No servidor *SVFserver* foi implementado um algoritmo de envio de fluxos de vídeo a clientes genéricos, considerando que estes não alocam espaço na memória principal para o armazenamento temporário dos fluxos recebidos do servidor. Foi incluída uma forma de tornar a execução do servidor confiável, através da implementação de algoritmos que controlam os recursos do sistema, permitindo que o maior número possível de requisições possa ser admitido sem que o serviço seja comprometido devido a sobrecargas.

Foi implementado no servidor um algoritmo que controla a banda de rede e indica se este recurso está sobrecarregado. Este algoritmo utiliza uma forma determinística para estimar a banda consumida. Para melhorar o aproveitamento do servidor foram estudados e implementados filtros que reduzem a banda alocada a cada requisição, aumentando assim o número de requisições aceitas pelo servidor.

Além do controle de utilização de banda de rede, foi implementado um algoritmo de controle de acesso a disco. Este algoritmo mede o tempo dos acessos a disco e informa se este recurso está sobrecarregado.

Com base na utilização da rede e do sub-sistema de discos, o servidor executa o algoritmo de controle de admissão, que decide se novas requisições podem ser admitidas pelo servidor, para garantir que os fluxos de vídeo sejam entregues aos clientes com um mínimo de degradação de qualidade.

1.2 Organização da Dissertação

Esta dissertação está dividida em 7 capítulos. O capítulo 2 descreve os conceitos necessários para a compreensão do funcionamento do servidor e inclui os trabalhos relacionados ao tema. O capítulo 3 expõe detalhadamente o funcionamento do servidor e descreve o método utilizado no envio de fluxos de vídeo. O quarto capítulo mostra as características dos arquivos de vídeo utilizados como carga de testes. O capítulo 5 descreve o algoritmo de controle de banda de rede e faz uma análise dos filtros utilizados para o cálculo da banda. O capítulo 6 descreve um algoritmo de controle de tempo de acasso a disco implementado no servidor e analisa os resultados obtidos com este algoritmo. O capítulo 7 conclui o trabalho.

Capítulo 2

Conceitos

2.1 Servidor de Vídeo

Um *Servidor de vídeo* é um programa que disponibiliza fluxos de vídeo a clientes que os reproduzem sem a necessidade de armazenamento local. As requisições podem ser emitidas a qualquer momento e podem ser referentes a qualquer arquivo de vídeo armazenado em disco ou a fluxos de vídeo transmitidos ao vivo.

A figura 2.1 mostra a arquitetura básica de um servidor de vídeo. Nesta figura observa-se que existem dois tipos de serviços que podem ser disponibilizados pelo servidor. O primeiro serviço é o acesso a sistemas de trasmissão de vídeo ao vivo e o segundo é o acesso a arquivos de vídeo armazenados em disco. No primeiro tipo de serviço a imagem e o som são codificados em formato de fluxos de vídeo e são enviados ao servidor de vídeo e/ou armazenados em disco. No segundo tipo de serviço, arquivos de vídeo disponibilizados pelo servidor podem estar armazenados diretamente em disco ou podem estar em armazenamento terciário como fitas e *compact discs*.

O procedimento de envio de fluxos de vídeo por parte do servidor depende do serviço que o cliente requisitou. Quando solicitado fluxos de vídeo ao vivo, o servidor repassa o fluxo correspondente recebido do codificador, que é armazenado temporariamente na memória principal, diretamente à interface de rede. No outro caso, sendo solicitado fluxos de um arquivo de vídeo armazenado, o servidor deve transferir os dados do disco, ou de outro tipo de armazenamento terciário, para a memória principal antes de encaminhá-lo ao cliente. Estando os dados na memória principal, estes são então transmitidos para a interface de rede. Neste caso o servidor deve ter o controle destes recursos para que a entrega ocorra sem atrasos.

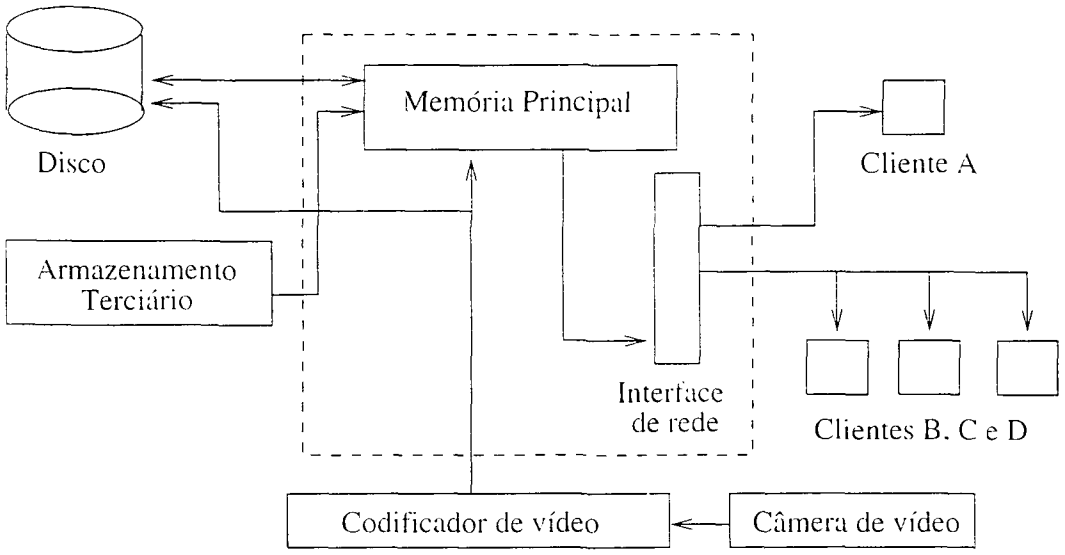


Figura 2.1: Arquitetura de um servidor de vídeo.

A transmissão de fluxos de vídeo pode ser feita através de dois tipos de conexões: *conexão unicast* ou *conexão multicast*. Uma *conexão unicast* corresponde a uma conexão na qual o fluxo de vídeo é enviado do servidor para um único endereço *IP* de destino, correspondendo ao endereço do cliente. Uma *conexão multicast* corresponde a uma conexão na qual o fluxo de vídeo é enviado do servidor a vários clientes simultaneamente, que utilizam um endereço *IP* do tipo *multicast*.

O serviço de envio de fluxos de vídeo ao vivo requer a utilização de conexões *multicast* para economizar largura de banda, pois os dados enviados aos clientes que solicitam o mesmo fluxo são idênticos. Isso não ocorre quando a solicitação é feita para arquivos de vídeo armazenados em disco, pois as requisições são feitas para arquivos distintos em tempos diferentes, dificultando a utilização de conexões *multicast* [XLP99, CT99, SHGCK98, DEZ01].

A principal característica que diferencia o servidor de vídeo de outros servidores é sua carga. Para transmitir vídeo digital o servidor deve ter controle de todos os recursos do sistema como: disco, memória principal, tempo de CPU, banda de rede, entre outros. A seção a seguir descreve as características e propriedades do vídeo digital que determina o modelo interno de um servidor de vídeo.

2.2 Vídeo Digital

Um *Vídeo Digital* é obtido pela composição de imagens estáticas sincronizados a um sinal de áudio. Uma imagem estática, ou *quadro*, é composta pelo arranjo de vários elementos de imagem

chamados *pixels*. A *resolução* de um quadro corresponde à quantidade de *pixels* descrito em termos de número de linhas por colunas.

Para que a reprodução de vários quadros consecutivos que possuem pequenas diferenças apresente a imagem de forma que o movimento apareça na tela tal como contínuo e natural, a freqüência de sua reprodução deve ser no mínimo de 24 quadros em um segundo. Esta freqüência é definida como taxa de quadros por segundo *(qps)* ou como *frames per second (fps)* [Gro89].

Cada *pixel* de um quadro de um vídeo digital é constituído de três informações:

1. No padrão *RGB*: 1 *pixel* = 8 bits de informação da cor vermelha *Cr-Red*, 8 bits de informação da cor verde *Cr-Green* e 8 bits de informação da cor azul *Cr-Blue*;

2. No padrão *YCbCr*: 1 *pixel* = 16 bits de informação de iluminância ou intensidade de luz *Y*, 8 bits de informação da cor azul *Cr-Blue* e 8 bits de informação da cor vermelha *Cr-Red*.

A qualidade de um vídeo digital é determinada pela resolução de cada quadro, pela quantidade de quadros reproduzidos por segundo e pela quantidade de bits que representa cada *pixel*. Quanto maior a qualidade de um vídeo, maior a quantidade de dados armazenados e a sua correspondente largura de banda durante a transmissão.

Para se ter uma idéia da quantidade de dados de um vídeo digital, considera-se um vídeo com 60 minutos de duração, com resolução de 640x480, na qual cada *pixel* é representado por 24 bits padrão RGB, a uma freqüência de 30 quadros por segundo. Este vídeo possui uma largura de banda 221 Mbps e utiliza 99,5 GB de disco, somente para a informação de imagem, desconsiderando áudio e outras informações. Para tornar viável o transporte e o armazenamento de vídeos digitais aos usuários existe a necessidade de comprimir as informações redundantes de um vídeo. A seção a seguir descreve um algoritmo de compressão de vídeo digital adotado como padrão internacional.

2.2.1 Compressão de Vídeo

O padrão de compressão de vídeo digital *Motion Picture Experts Group (MPEG)* foi desenvolvido no início da década de noventa com o objetivo de diminuir a largura de banda de fluxos de vídeo. Este padrão reduz a largura de banda de vídeos genéricos para o intervalo de 1.5 Mbps no padrão MPEG-1 e para 4.0 Mbps no padrão MPEG-2 [Gal91, CGM99].

O padrão de compressão MPEG diminui a largura de banda de um vídeo através da eliminação das informações redundantes da imagem e do som. No caso da imagem, com maior

quantidade de informações, dois tipos de redundância são eliminados: redundância espacial e redundância temporal.

A redundância espacial corresponde aos dados de crominância e luminância que preenchem áreas de uma determinada imagem com a mesma informação com a mesma representação de cor para vários *pixels* vizinhos. Para eliminar este tipo de redundância é aplicado uma seqüência de algoritmos de compressão de dados. Inicialmente cada quadro referente à informação de crominância ou iluminância é dividido em blocos de 8x8 *pixels* e dentro de cada bloco somente informações necessárias são codificadas. Para o padrão *YCbCr* as informações referentes à crominância são menos relevantes que a informação de luminância, desta forma cada grupo de 4 *pixels* de uma determinada crominância é codificado em um único *pixel*, aumentando assim o grau de compressão. Em cada bloco criado é aplicado o algoritmo *Discrete Cosine Transform (DCT)*, para a eliminação das redundâncias espaciais. Para uma descrição mais detalhada veja [Gal91].

A redundância temporal corresponde às informações repetidas em quadros consecutivos ao longo da reprodução de um vídeo. Para eliminar informações repetidas, ao invés de codificar inteiramente cada quadro em separado, são codificados somente as diferenças entre quadros adjacentes na seqüência de reprodução. O algoritmo de compensação de movimentos é utilizado para detectar movimentos de objetos em seqüências de quadros consecutivos. Este algoritmo transforma um quadro original em um de três tipos de quadros:

1. Quadro *Intracoded (I)*: este quadro é codificado com todas as informações do quadro original excluindo somente as redundâncias espaciais. Este quadro será utilizado como fonte de comparação de movimento de objetos para quadros anteriores e posteriores. Este deve ser o primeiro quadro a ser reproduzido no cliente e deve ser armazenado em um buffer para a reconstrução dos outros quadros;

2. Quadro *Predictive (P)*: este quadro é codificado somente com as diferenças detectadas no quadro atual com relação ao quadro-I ou quadro-P anterior. Ele também é utilizado como base de comparação;

3. Quadro *Bidirectional (B)*: este quadro é codificado somente com as diferenças detectadas no quadro atual com relação ao quadro-I ou quadro-P anterior e posterior ao mesmo tempo. Este quadro possui uma mescla de informações dos dois quadros, parecendo duas imagens sobrepostas. Este quadro possui o maior grau de compressão comparado com os outros

dois quadros.

A figura 2.2 mostra a relação entre os quadros resultantes da compressão. Observa-se que os quadros-B são codificados com as diferenças entre seu quadro original e os quadros-I e P anterior e/ou posterior, enquanto o quadro-P é codificado com as diferenças entre seu quadro original e o quadro-I anterior.

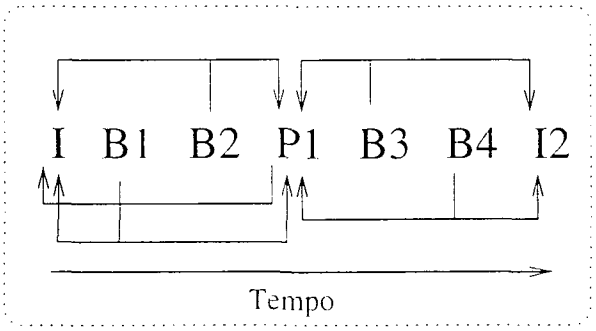


Figura 2.2: Relação entre quadros-I, P e B.

O quadro-I serve como base para a codificação e decodificação dos outros tipos de quadros, por isso é importante que este quadro apareça periodicamente na codificação de um vídeo digital. Quando a fonte do vídeo reproduzido não é local, a sua reprodução é feita através do recebimento de fluxos de vídeo através de uma rede. Neste caso, é necessário que a freqüência dos quadros seja alta e constante evitando que a sua reprodução seja interrompida quando um quadro-I chega com erro ou quando é perdido. Em um servidor de vídeo que envia fluxos de vídeo utilizando transmissão *multicast*, quando novos usuários são incluídos no grupo, estes usuários devem esperar um quadro-I para poder iniciar a reprodução do vídeo. Sendo assim, se a quantidade de quadros-I é pequena, o tempo para iniciar a reprodução será longo.

A quantidade e a seqüência de tipos de quadros são definidos como um *Group of Picture (GOP)*. Cada grupo inicia com um quadro-I e é seguido por quadros B e P. A quantidade de quadros de um *GOP* pode ser definida pelo usuário, assim como o número de quadros por segundo. Estes dois valores podem ser iguais mas são independentes. Quanto menor for o tamanho de um GOP, maior é a quantidade de quadros-I codificados, conseqüentemente maior é a largura de banda do vídeo.

A figura 2.3 mostra a codificação de dois *GOP* constituídos por 12 quadros cada, em um vídeo com 24 quadros por segundo. A ordem de codificação dos quadros é diferente da ordem de reprodução, devido à dependência que quadros-B possuem de quadros anteriores e posteriores para a sua codificação e decodificação. Observa-se que o quadro superior mostra a ordem real

de reprodução dos quadros e o quadro inferior mostra a ordem real de codificação e decodificação. Os quadros-P e os quadros-I devem estar armazenados na memória antes da codificação e decodifição dos quadros-B.

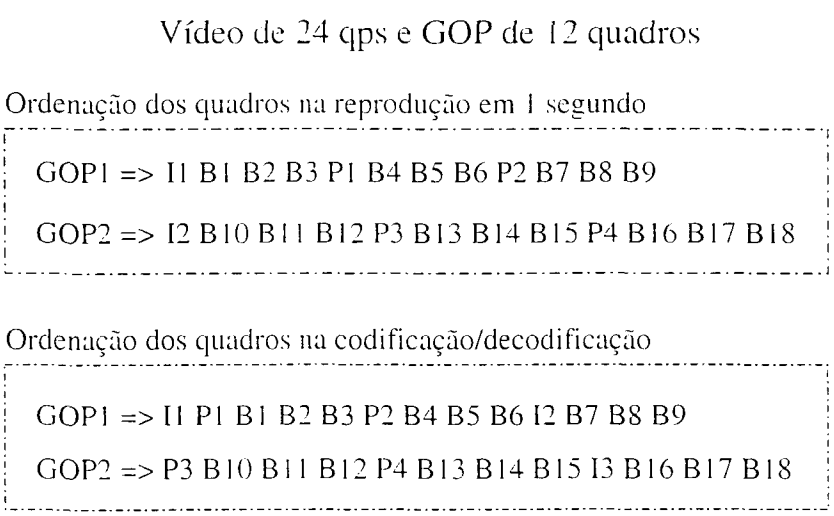


Figura 2.3: Ordenação dos quadros dentro de um GOP.

2.2.2 Compressão de Áudio

O padrão MPEG comprime a informação de áudio através da eliminação de informações acústicas irrelevantes. Este padrão considera a inabilidade da audição humana de registrar sinais de áudio codificados quando estes sofrem mascaramento. O fenômeno de mascaramento corresponde a uma propriedade da percepção do sistema de audição que ocorre quando um sinal de aúdio forte torna imperceptíveis os sinais que são vizinhos à sua freqüência e que tem baixa amplitude. Resultados empíricos mostram que o sistema de audição humana possui limitações no seu espectro. Estas limitações correspondem a larguras de banda menores que 100 Hz para freqüências baixas e maiores que 4 kHz para freqüências altas. Estas limitações correspondem à falta de capacidade do sistema de audição de traduzir fielmente o sinal recebido nestas freqüências. Analisando a amplitude do sinal e a sua freqüência, as informações não percebidas são descartadas pelo algoritmo de compressão de áudio [Pan95].

O primeiro padrão de compressão de áudio MPEG-1, pode somente representar o sinal de áudio em um canal mono ou em dois canais stereo. Este padrão possui três distintas camadas de compressão. A primeira camada (*Layer I*) corresponde ao algoritmo base de compressão, a segunda e terceira camadas (*Layer II*) e (*Layer III*) são melhorias nas técnicas utilizadas na primeira camada. A tabela 2.1 mostra a média do grau de compressão para vários tipos de

músicas [Nol99].

O segundo padrão de compressão de áudio, MPEG-2, incluiu a representação do áudio em multicanais. A *International Telecommunication Union (ITU-R)* e outros órgãos internacionais recomendam a configuração *3/2-stereo* para multicanais. Nesta configuração existe um canal esquerdo, um direito e um central (*L, R e C*) respectivamente, e mais dois canais adjacentes esquerdo e direito (*LS e LR*).

Camada	Banda [kbps]	Fator de Compressão
Camada I	384	4
Camada II	256	6
Camada III	192	8

Tabela 2.1: Grau de compressão de cada camada.

O sistema de áudio em multicanais dispõe também de canais com vários idiomas e canais com informações detalhadas referentes a imagens ou sinal de áudio prejudicados.

O padrão MPEG-2 permite a decodificação de áudio no formato mono ou stereo, também permite que decodificadores MPEG-1, que não reconhecem aúdio no formato de multicanal, possam reproduzir o sinal em somente dois canais, mono ou stereo.

2.2.3 Codificação VBR e CBR

Observa-se que a compressão de um sinal de áudio acoplado a um sinal de vídeo que possui muitas mudanças nas imagens de fundo e movimento de objetos produz um arquivo de vídeo compactado com variabilidade nos tamanhos dos quadros e nos pacotes que armazenam som. Como cada quadro possui um grau de compressão diferente, o resultado da compressão de um vídeo digital corresponde a uma largura de banda variável. Quando um vídeo possui largura de banda variável ele é denominado *Variable Bit Rate (VBR)*. Quando a compressão de um vídeo força a produção de quadros com o mesmo tamanho, o vídeo fica com a largura de banda constante e neste caso ele é denominado *Constant Bit Rate (CBR)*.

O processo de codificação na compressão de um vídeo digital varia de acordo com a escolha dos algoritmos implementados. Alguns aplicativos podem ter implementado codificadores MPEG com diversos graus de compressão de dados. Esta característica permite que o processo de codificação seja otimizado de acordo com as necessidades de largura de banda e da qualidade do vídeo desejados pelo usuário.

A codificação *CBR* é um tipo opcional de serviço disponibilizado pelo algoritmo de compressão de vídeo. Nesta codificação os quadros produzidos devem ter o mesmo tamanho. Se o resultado da compressão de um quadro é superior ao valor definido, o quadro sofre novas compressões até atingir o valor desejado. Se o tamanho do quadro é inferior ao valor determinado, a quantia que falta é preenchida com dados nulos.

Sistemas de vídeo digital devem controlar eficientemente os recursos de disco e de rede para reduzir custos e melhorar a sua utilização. Arquivos de vídeo codificados no formato *VBR* economizam banda de rede quando transmitidos de forma multiplexada e exigem menos espaço para o seu armazenamento quando comparados a vídeos codificados no formato *CBR*. A codificação *VBR* também garante que a qualidade da imagem e do som não seja prejudicada, enquanto a codificação *CBR* limita o tamanho dos quadros produzidos para tornar a banda mais constante possível.

Um arquivo de vídeo codificado no formato *VBR* exige menos espaço para o seu armazenamento que o mesmo arquivo de vídeo codificado no formato *CBR*, possuindo os dois arquivos o mesmo grau de qualidade [SGB98]. Um arquivo *VBR* possui o inconveniente de inserir rajadas de dados nos momentos em que são transmitidos pacotes com imagem ou som com baixo grau de compressão, ou seja, os pacotes maiores. Estas rajadas dificultam o controle sobre os recursos do sistema quando o controle é determinístico, pois o valor utilizado para estimar os recursos utilizados é o maior valor obtido durante toda a transminsmissão do vídeo, sub-utilizando assim os recursos controlados.

Os gráficos das figuras 2.4 e 2.5 mostram a variabilidade dos tamanhos dos quadros de dois filmes comprimidos em *MPEG*. Observa-se que o primeiro vídeo possui picos em torno de 100.000 Bytes e a média do tamanho dos quadros em torno de 8.000 Bytes, já no segundo filme o maior quadro é do tamanho de 110.882 Bytes, mas a média dos tamanhos dos quadros fica em torno de 10.000 Bytes.

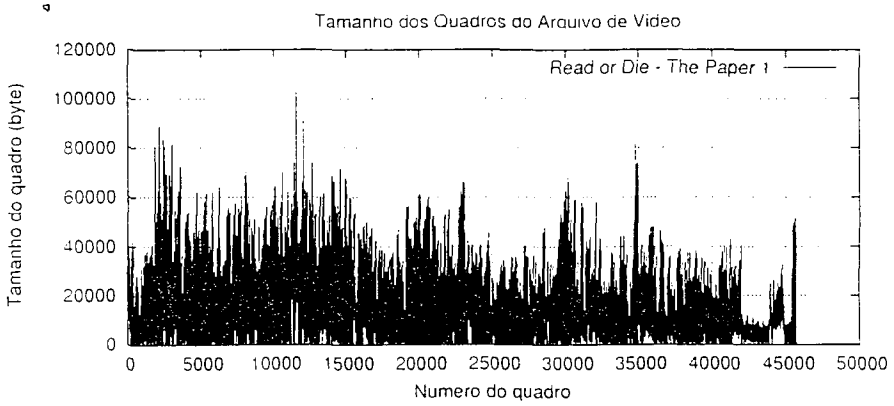


Figura 2.4: Tamanho dos quadros do desenho "Read or Die - The Paper 1".

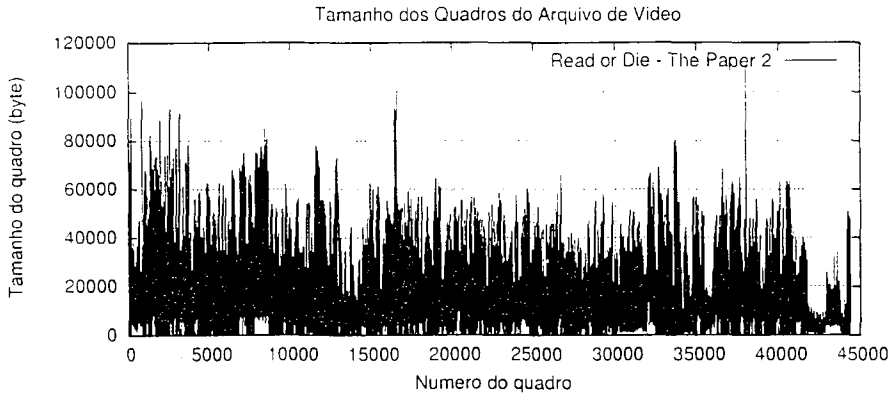


Figura 2.5: Tamanho dos quadros do desenho "Read or Die - The Paper 2".

As características de cada vídeo digital comprimido definem o seu processamento e a sua transmissão por um servidor de vídeo e conseqüentemente determinam a sua arquitetura interna. A próxima seção descreve detalhadamente o funcionamento de um servidor de vídeo.

2.3 Funcionamento do servidor de vídeo

O funcionamento de um servidor de vídeo que envia fluxos de vídeo ao vivo ou fluxos de vídeo de arquivos de vídeo armazenados em disco pode ser representado pelo diagrama da figura 2.6. O primeiro bloco representa o processamento de novas requisições que chegam no servidor e este bloco é responsável pela análise das requisições recebidas. Ao receber uma nova requisição o servidor reconhece qual é o vídeo requisitado, se este não corresponder a algum vídeo disponível o servidor envia uma resposta negando o serviço. Se o vídeo requisitado corresponder a algum dos disponíveis, então o servidor examina os recursos do servidor para certificar que não ocorrerá sobrecarga se a requisição for aceita. Sabendo que não ocorrerá sobrecarga se esta nova requisição

for aceita, o servidor envia uma resposta confirmando o envio dos fluxos de vídeo. Caso contrário, o servidor envia uma resposta negando o serviço. Após enviar a resposta confirmando aceitação, o servidor inicia o envio dos fluxos de vídeo.

Um servidor de vídeo envia uma determinada quantidade de dados a seus clientes em intervalos de tempo fixo. A cada intervalo são transmitidas quantias distintas de dados, suficientes para serem reproduzidos até o envio de novos dados. Desta forma o servidor deve manter sincronismo entre o período em que envia uma quantidade determinada de dados e o período de reprodução destes dados no cliente. A quantidade de dados enviados em cada período deve ser suficiente para que seja reproduzida durante todo o período sem que ocorra falta de dados e conseqüente interrupção na produção do vídeo no cliente, até o recebimento de novos dados. Este período é denominado *ciclo do servidor*. Os ciclos do servidor definem a quantidade de dados que é enviada a cada intervalo de tempo em que ocorre a transmissão de fluxos de vídeo para todos os clientes conectados no servidor. Por exemplo, um servidor pode definir o seu ciclo do servidor como sendo de 1 segundo, e neste período o servidor deve enviar fluxos de vídeo correspondentes aos dados que serão reproduzidos durante 1 segundo em todos os clientes.

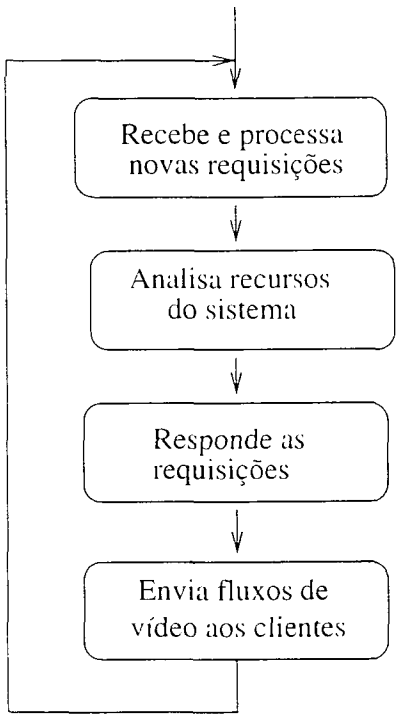


Figura 2.6: Diagrama de funcionamento do servidor.

A sincronização entre servidor e cliente ocorre da seguinte forma. O servidor transmite os dados que serão reproduzidos no próximo ciclo, enquanto o cliente reproduz os dados que

receben no ciclo anterior, conforme a figura 2.7 mostra. Nesta figura o tempo é representado pela variável T, o ciclo do servidor é de 1 segundo e a quantidade de dados enviada é representada pela variável D. No intervalo T, o servidor envia aos clientes os dados D2 que serão reproduzidos no intervalo T+1 enquanto os clientes reproduzem os dados D1 que receberam no intervalo T-1. Observa-se que se o filme é codificado como *VBR* as quantidades D1, D2, ... Dn são distintas, mas se o filme é codificado como *CBR* a quantidade de dados enviados em cada ciclo é constante.

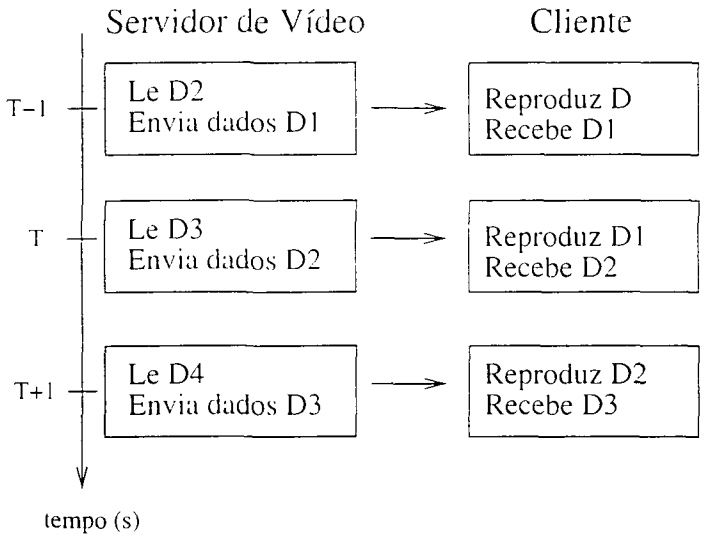


Figura 2.7: Sincronismo de leitura, transmissão e reprodução em alguns ciclos do servidor.

Em cada ciclo o servidor deve enviar os dados a todos os clientes conectados. Ao mesmo tempo em que o servidor transmite estes dados, deve adiantar a leitura dos dados que serão enviados no próximo ciclo. *No caso do serviço de envio de fluxos de vídeo de arquivos armazenados em disco*, o servidor deve ler estes dados do disco a tempo de poder enviá-los logo no início do próximo ciclo. A figura 2.7 mostra que no ciclo atual o servidor envia fluxos a todos os clientes e ao mesmo tempo lê dados do armazenamento secundário e/ou terciário que serão enviados no próximo ciclo.

Como o consumo dos dados no cliente depende do tempo de reprodução do vídeo, o servidor deve garantir a entrega dos fluxos de vídeo a tempo de não ocorrer a falta de quadros na reprodução.

Para assegurar a entrega dos fluxos no tempo certo, o servidor deve controlar os recursos do sistema, pois estes podem adicionar atrasos no processamento do servidor quando estão sobrecarregados. Os recursos que podem causar atrasos na entrega do fluxos de vídeo são:

1. espaço disponível na memória principal, *buffer* do servidor;

2. tempo de leitura dos dados do disco;

3. congestionamento na interface de rede;

4. tempo de CPU.

O controle de recursos no servidor é feito através da limitação de requisições atendidas, evitando assim sobrecargas. A limitação das requisições resulta do cálculo da quantidade do recurso utilizado para cada vídeo requisitado. Se a soma dos recursos utilizados para cada requisição ultrapassar a quantia disponível ao servidor, então não são aceitas mais requisições até a finalização de uma ou mais transmissões correntes.

A banda do vídeo é usada no cálculo de utilização de alguns recursos. Os vídeos que são codificados no formato CBR possuem a banda constante durante todo o período da sua reprodução, neste caso, o controle dos recursos é facilitado devido ao conhecimento prévio deste valor. No caso de vídeos codificados no formato VBR, a sua largura de banda é variável dificultando o cálculo da carga imposta aos recursos da máquina durante a sua reprodução.

O controle de recursos exige a implementação dos seguintes algoritmos:

1. Algoritmo de controle de memória principal: que determina se a memória principal está sobrecarregada;

2. Algoritmo de controle de banda de rede: que determina se a interface de rede está sobrecarregada;

3. Algoritmo de controle de tempo de leitura de disco: que verifica se o recurso de disco está sobrecarregado;

4. Algoritmo de controle de admissão: analisa os recursos do sistema segundo o resultado determinado pelos outros algoritmos e decide se podem ser admitidas novas requisições.

Os algoritmos de controle de recursos são objeto de estudo em vários projetos de servidores de vídeo. Na próxima seção são descritos alguns trabalhos que foram tomados como referência na implementação do servidor *SVFserver*.

2.4 Trabalhos Relacionados

Vários algoritmos de controle de recursos de um servidor de vídeo foram propostos. No caso do espaço de memória utilizado pelo servidor, [Ver96] apresenta um servidor de vídeo que uti-

liza alocação individual do *buffer* para cada requisição, mesmo método empregado no servidor *SVFserver*. Neste caso, o controle do *buffer* é realizado através da soma da quantia de memória reservada a cada cliente.

Em [BOS95, BOS96, CGM99, WFS97] são propostos algoritmos que controlam os *buffers* das requisições utilizando compartilhamento de memória entre cada requisição. Neste modelo o aproveitamento da memória principal é muito maior se comparando com a utilização de alocação separada, como é feito no servidor *SVFserver*. Este método permite o aumento do número de requisições aceitas pelo servidor, mas em compensação impõe um aumento no processamento do servidor para administrar a memória compartilhada. Nos trabalhos [KLY99a, KLY99b, KY00], o controle de memória inclui também o controle do número de faltas na memória cache.

O controle da banda de rede é realizado de forma diferenciada para cada tipo de codificação de vídeo. Em [MR96, SGB98] foram propostos algoritmos que controlam a banda da rede no servidor utilizando arquivos de vídeo no formato *CBR*. Estes trabalhos mostram a facilidade em administrar e calcular os recursos quando um servidor de vídeo trabalha com arquivos neste formato. Os métodos propostos nestes trabalhos foram desconsiderados na implementação do servidor *SVFserver*, pois o servidor trabalha com arquivos com banda de vídeo variável.

No caso da codificação *VBR* foram propostos vários métodos de estimativa de banda de rede como os descritos em [NGDR98, KZ97, KZ95, DMH99, FS95a, FS95b, MR95, MGT97]. Nestes trabalhos são propostos algoritmos que utilizam cálculos determinísticos e multiplexação estatística no cálculo dos recursos consumidos da rede. Todos estes servidores são conectados por clientes que alocam espaço na memória principal para armazenamento dos fluxos recebidos, desta forma permitindo que o servidor envie altas quantidades de dados de forma adiantada nos intervalos em qua a banda do vídeo é alta, eliminando assim as rajadas de dados e tornando o controle de recursos mais simples. Um dos trabalhos mencionados, o trabalho de [KZ97] foi utilizado no método de controle de banda de rede implementado no servidor.

Para controlar os recursos de armazenamento secundário e terciário vários estudos são descritos em [AMG98, CGM99, DMH97, TC95, RRS96]. Estes trabalhos relacionam algoritmos que controlam o recurso do disco tanto para arquivos de vídeo na codificação CBR quanto para a codificação VBR. Alguns destes trabalhos propõem a utilização de algoritmos específicos do sistema operacional que modificam a ordem de leitura através escalonadores específicos para leituras de arquivos de vídeo no disco. Também são propostos métodos específicos para armazenamento de arquivos de vídeo, de forma que o tempo de leitura destes arquivos seja reduzido.

Dentre os trabalhos listados que propõem controle ao acesso a disco, nenhum destes foi utilizado na implementação do algoritmo de controle de acesso a disco no servidor, devido à forma aleatória em que os arquivos são armazenados no disco e que não se utiliza de algoritmo de escalonamento de leitura de disco que seja específico para leitura de arquivos de vídeo.

Tendo controle sobre os recursos utilizados é imprescindível incluir no servidor um algoritmo de controle de admissão. O algoritmo de controle de admissão utiliza os resultados dos algoritmos que controlam os recursos para decidir se novas requisições podem ser aceitas pelo servidor.

Alguns dos trabalhos citados foram tomados como base para a implementação do servidor, e estes são descritos e analisados mais detalhadamente nos próximos capítulos. No próximo capítulo é descrita a arquitetura e os algoritmos implementados no servidor *SVFserver*.

Capítulo 3

Servidor SVFserver

Este capítulo descreve o funcionamento do servidor de vídeo *SVFserver* e contém uma análise dos algoritmos escolhidos para envio de fluxos de vídeo de arquivos armazenados em disco e os algoritmos de controle de recursos do servidor.

O servidor *SVFserver*, objeto deste estudo, extende a implementação do servidor *FFServer* [bel03], agregando a ele funções que permitem o envio de fluxos de vídeo de arquivos de vídeo armazenados no disco, incluindo um algoritmo de controle de banda de rede, um algorimo de controle de tempo de disco e um algoritmo de controle de admissão, cujo objetivo é o aumento da capacidade de transmissão através do controle de recursos.

O *FFServer* é um servidor de vídeo que trabalha em conjunto com um programa de codificação de vídeo chamado *FFMpeg*. O *FFMpeg* faz a codificação de fluxos de vídeo ao vivo e faz transcodificação de arquivos de vídeo armazenados em disco para codificações em diversos formatos. O *FFServer* disponibiliza dois tipos de serviço aos usuários. O primeiro serviço é o envio de fluxos de vídeo ao vivo aos clientes. Neste caso, o *FFMpeg* alimenta o servidor com fluxos de vídeo codificados em tempo real e o servidor repassa estes dados aos clientes conectados. O segundo serviço é o envio de fluxos de vídeo de arquivos armazenados em disco aos clientes. Neste caso o servidor necessita que o codificador *FFMpeg* transcodifique o arquivo desejado, para que o codificador envie o fluxo de vídeo assim produzido ao servidor e este o repasse aos clientes. Observa-se que o servidor depende do processamento do codificador *FFMpeg* mesmo quando o serviço requisitado não precise deste processamento, como no caso de arquivos armazenados em disco, o que aumenta consideravelmente a carga de processamento no sistema. Este modo de operação também dificulta o envio de fluxos de arquivos de vídeo diferentes, pois para cada tipo de arquivo enviado, o codificador deve fazer a transcodificação para alimentar o servidor.

A extensão da implementação do servidor *FFServer* inclui o envio de fluxos de vídeo de arquivos armazenados em disco sem a utilização do codificador *FFMpeg*, permitindo assim que o servidor possa enviar uma quantidade maior de fluxos de vídeo de arquivos de vídeo distintos armazenados no disco. Também são incluídos algoritmos que controlam os recursos de rede e de disco viabilizando a implementação de um algoritmo de controle de admissão. Estes algoritmos são empregados somente com o novo serviço de envio de fluxos de vídeo, implementado no servidor.

O *SVFserver*, como no *FFserver*, foi implementado para trabalhar com qualquer programa cliente que reproduza vídeo através do recebimento de fluxos de vídeo, pois o método de comunicação com os clientes é simples e generalizado. As requisições e as respostas utilizam o protocolo *Hiper Text Transfer Protocol (HTTP)* para informar somente o arquivo desejado, por parte do cliente, e determinar a resposta do servidor. Esta forma de comunicação evita que sejam incluídas nas requisições solicitações específicas de programas proprietários que possam reduzir a funcionalidade do servidor, isto implica na falta de informações referentes ao cliente que possam viabilizar um melhor desempenho do servidor, como será visto na seção 3.4.

O servidor *SVFserver* envia fluxos de vídeo de arquivos armazenados em disco, sem a necessidade de decodificação dos quadros de vídeo e das informações de áudio, diminuindo assim a quantidade de processamento do servidor, e impedindo que serviços que exijam decodificação dos quadros como serviços do tipo *VCR - Video Cassete Record* como *fast forward, fast rewind, pause, resume, begin*, sejam disponibilizados.

Um servidor de vídeo que envia fluxos de vídeo de arquivos armazenados em disco não necessita enviar ao programa cliente informações referentes ao sincronismo de reprodução dos quadros e nem informações sobre o sincronismo do sinal de áudio. Estas informações são incluidas durante o processo de compressão e codificação no arquivo de vídeo, fazendo com que o programa cliente seja responsável pela correta reprodução do vídeo.

Este servidor trabalha com arquivos de video nos formatos *Audio Video Interleaved (AVI)*, *Advanced Format System (ASF* versão 1.0) e *MPEG*, independentemente do tipo de codificação de compressão de áudio e vídeo, podendo estes serem *CBR* ou *VBR*.

Os algoritmos implementados são baseados na utilização da codificação *VBR*, pois os arquivos de vídeo que utilizam esta codificação apresentam melhor qualidade na imagem e no som.

3.1 Modelo do Servidor

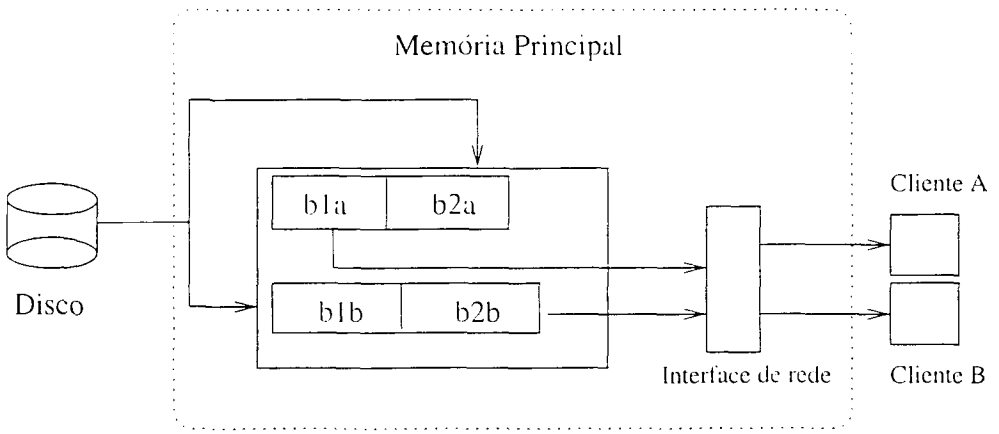
O servidor *SVFserver* foi implementado para ser executado em um computador pessoal convencional, utilizando o sistema operacional *Linux* interligado a uma rede local do tipo *Ethernet*.

Para facilitar a implementação do servidor foi escolhido o protocolo *TCP/IP* para o transporte dos fluxos de vídeo. Este protocolo garante que a entrega é feita ao cliente na ordem certa e sem erros.

Para facilitar a implementação do servidor foi decidido fazer reserva separada dos *buffers* de cada requisição aceita. Os *buffers* de cada requisição correspondem a dois espaços na memória principal que armazenam os dados que são lidos do disco e os dados que são enviados ao cliente. O tamanho de cada *buffer* é dererminado pelo período do ciclo do servidor, pois o *buffer* deve comportar a quantidade de dados a ser enviada e reproduzida no cliente durante todo o ciclo.

O ciclo do servidor no *SVFserver* corresponde a 1 segundo. Depois de testes com vários intervalos foi decidido pelo valor de 1 segundo porque ele contrabalanceia o tamanho do *buffer* exigido para comportar a maior quantidade de dados que é reproduzida em um ciclo com a freqüência das leituras de disco, sendo que estas ocorrem uma única vez a cada requisição para preenchimento completo do *buffer* em cada ciclo. Quanto menor o ciclo do servidor, menores são os *buffers* reservados a cada requisição. Em compensação, maior é a quantidade de chamadas de sistema para leituras de disco, o que interfere no desempenho do servidor quando este admitir uma quantidade grande de requisições de arquivos distintos, causando atrasos na leitura devido aos movimentos no braço do disco, podendo portanto atrasar a entrega dos fluxos de vídeo. No caso de o ciclo ser muito longo, menor é o número de leituras realizadas no disco durante todo o período de reprodução de cada arquivo de vídeo, e maiores são os *buffers* alocados a cada requisição, limitando assim o número de requisições aceitas pelo servidor por causa do tamanho finito da memória principal.

O *SVFserver* aloca espaços separados na memória para armazenar os dados lidos do disco e os dados transmitidos na interface de rede. A figura 3.1 mostra como são utilizados os *buffers* no servidor para duas requisições distintas. Durante um ciclo o *buffer1* da requisição A é utilizado para armazenar os dados lidos do disco, enquanto o *buffer2* armazena os dados que são transmitidos na interface de rede. No próximo ciclo a função de cada *buffer* é trocada, o *buffer1* armazena os dados que são enviados ao cliente enquanto o *buffer2* é preenchido com os dados lidos do disco.



b1a = buffer 1 da requisição do cliente A
b2a = buffer 2 da requisição do cliente A
b1b = buffer 1 da requisição do cliente B
b2b = buffer 2 da requisição do cliente B

Figura 3.1: Disposição da memória.

A determinação dos tamanhos dos *buffers* ocorre na inicialização do servidor. Quando o servidor é inicializado este analisa todos os arquivos de vídeo, para determinar qual é a maior quantidade de dados que será enviada durante um ciclo do servidor, para cada arquivo analisado. A maior quantia encontrada corresponde aos tamanhos dos *buffers* que são reservados às requisições feitas ao arquivo de vídeo correspondente.

Foi determinado que cada requisição possui o ciclo do servidor independente do ciclo das outrs requisições, pois se todas as requisições dependessem de um ciclo global, todas as requisições enviariam os fluxos de vídeo no inicio de cada ciclo, tornando os recursos sobrecarregados no início do ciclo e ociosos no final do ciclo. Desta forma, tendo os ciclos separados, os processamentos de cada requisição ficam distribuídos no decorrer dos segundos.

Para evitar que o tempo de leitura de disco interfira no desempenho do servidor, o processo de leitura de disco para preenchimento dos *buffers* foi codificado em uma *thread* separada.

A figura 3.2 mostra o modelo de execução do servidor *SVFserver*. Ao iniciar, o servidor lê um arquivo de configuração onde é indicado o diretório onde estão armazenados os arquivos de vídeo. Todos os vídeos são analisados e então determinados os tamanhos dos *buffers* requeridos para cada filme.

Após analisar os arquivos de vídeo, o servidor inicializa a *thread* que fará as leituras de disco. O servidor então entra em estado de espera de novas requisições. Recebendo uma nova requisição o servidor analisa o pedido, verifica se existem recursos disponíveis e responde confirmando ou não a aceitação da requisição. Se a requisição é aceita, o servidor reserva os *buffers* para a

requisição até a finalização do envio dos fluxos de vídeo. Um fluxo pode ser encerrado pelo cliente a qualquer momento, ou por parte do servidor quando chegar no final do arquivo.

Após reservar os *buffers*, o servidor inicia o envio de fluxos de vídeo ao cliente até a conclusão da requisição. Durante este período novas requisições podem ser aceitas.

Durante todo o período de execução do servidor a *thread* de leitura de disco verifica na lista de requisições se existe algum *buffer* vazio e, se encontrado, este é preenchido com os dados do disco. Pode ocorrer de a *thread* preencher os dois *buffers* de uma mesma requisição em um mesmo ciclo, isto acontece porque o *buffer* com os dados que estão sendo enviados ao cliente pode esvaziar antes do final do ciclo do servidor.

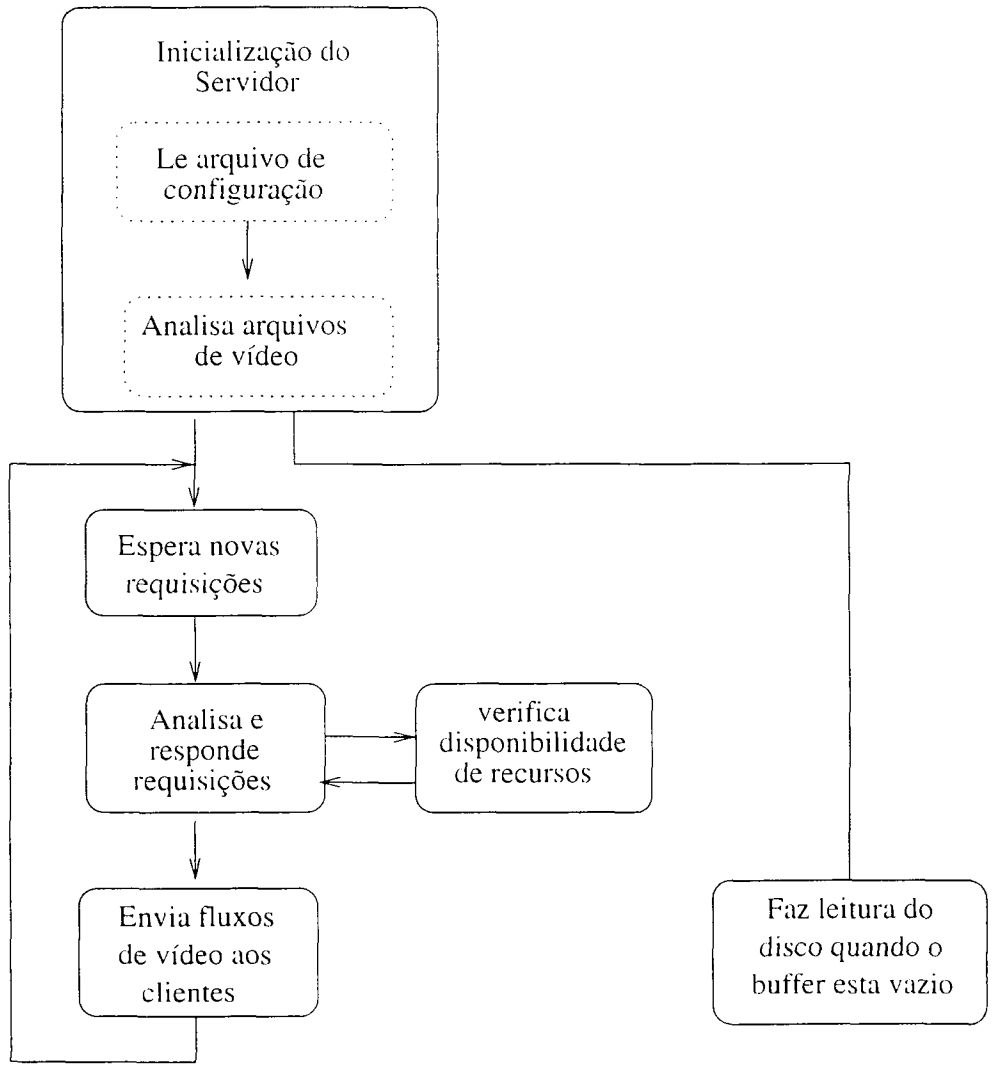


Figura 3.2: Diagrama de funcionamento do SVFserver.

O modelo empregado na entrega dos fluxos de vídeo por parte do servidor depende diretamente do modelo de funcionamento do programa cliente, que é descrito na próxima seção.

3.2 Cliente

Para evitar que pequenas variações no tempo de entrega dos fluxos interfiram na reprodução do vídeo, o programa cliente designa um espaço na memória para armazenar uma pequena quantidade de quadros recebidos. Este *buffer* deve ser preenchido antes de iniciar a reprodução do vídeo. O período correspondente ao tempo de reprodução dos quadros que ficam armazenados no *buffer* do cliente, equivale ao intervalo de tempo que assegura a chegada de quadros atrasados. Desta forma, quanto maior o *buffer* utilizado no cliente, menor é a probabilidade da reprodução sofrer interferências devido ao atraso no recebimento dos fluxos de vídeo.

Quando o servidor envia fluxos de arquivos de vídeo do tipo *CBR*, o consumo dos quadros ocorre na mesma proporção da chegada de novos quadros, como mostra o primeiro quadro da figura 3.3, permitindo que o buffer do cliente seja relativamente pequeno. Para arquivos de vídeo do tipo *VBR* a taxa de quadros reproduzidos não é a mesma da taxa de quadros recebidos, exigindo que o cliente reserve um *buffer* maior para comportar quadros grandes.

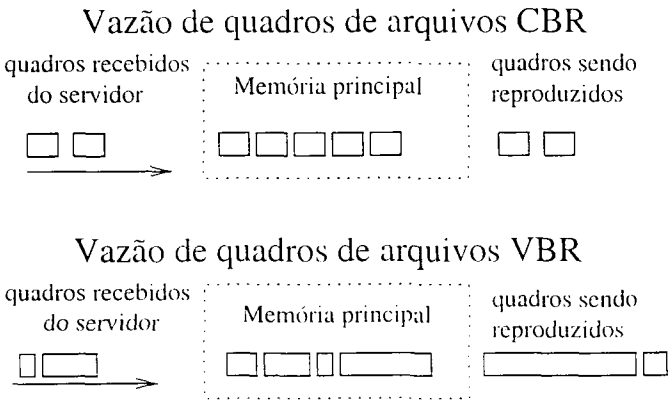


Figura 3.3: Vazão e consumo dos quadros no cliente.

Se o cliente possui um *buffer* para armazenamento temporário de alguns quadros, o servidor deve evitar que este *buffer* se esvazie ou transborde durante o tempo de reprodução do vídeo. A existência e o tamanho do *buffer* do cliente influência na escolha do método de envio dos fluxos de vídeo pelo servidor.

O *SVFserver* foi implementado para permitir conexão com vários tipos de cliente e, portanto, o método de comunicação com os clientes é o mais generalizado possível. Esta característica não permite que o servidor obtenha informações referentes à utilização de *buffer* por parte do cliente, desconsiderando assim a sua utilização. Desta forma o servidor envia fluxos de vídeo sob a suposição de que *buffers* não são utilizados pelos clientes.

3.3 Modelo de Envio de Segmentos de Fluxos de Vídeo

Dois modelos de envio de segmentos podem ser implementados em um servidor de vídeo. O primeiro modelo é caracterizado pela solicitação da quantidade de dados por parte do cliente, ou seja, de tempos em tempos o cliente informa ao servidor a quantidade de dados que aquele pode receber. Este modelo é denominado *Modelo Pull*. O segundo modelo é caracterizado pela passividade do cliente no envio dos dados, e todo o controle de transmissão é feito somente pelo servidor. Este modelo é denominado *Modelo Push* [AMG98].

Devido às características da comunicação do servidor *SVFserver* com os clientes que reproduzem vídeo, foi implementado o *Modelo Push*.

Alguns dos algoritmos propostos de envio de segmentos de fluxos de vídeo que empregam o *Modelo Push* foram implementados e analisados neste servidor, como os descritos em [Fen97], [WFS97], [FS95a], [FS95b], [MR96], [MR95]. Note-se, que neste modelo, a existência do *buffer* do cliente e a limitação da banda de rede influenciam a entrega dos segmentos aos clientes.

A banda de rede no servidor de vídeo fica limitada porque o tráfego é variável e os tamanhos dos quadros são variáveis. Analizando o comportamento variável da largura de banda dos vídeos, conclui-se que a limitação dos maiores picos faz-se necessária para garantir que não ocorram sobrecargas temporárias na rede.

Alocação pela Média (Avarage Allocation) O primeiro modelo de envio de segmentos de fluxos de vídeo implementado no servidor, foi baseado no algoritmo *Alocação pela Média* [FS95a]. Este algoritmo calcula a média dos tamanhos de todos os quadros do arquivo de vídeo, de acordo com a fórmula 3.1.

$$\overline{T} = 1/n \sum_{i=1}^{n} quadro_i \tag{3.1}$$

A quantidade de dados enviada a cada ciclo do servidor corresponde ao produto do tamanho médio calculado pelo número de *qps* obtido no cabeçalho do arquivo de vídeo. Para o arquivo de vídeo do gráfico mostrado na figura 3.4, que possui uma duração de 1905 segundos de reprodução a 24 quadros por segundo, o valor médio do tamanho dos quadros corresponde a 8.266 bytes. Este gráfico foi extraído do desenho Read or Die - The Paper 1.

Com este algoritmo, a cada ciclo do servidor são enviados 24 segmentos de dados do tamanho

de 8.266 bytes, como mostra a figura 3.5, totalizando a quantia de 198.384 bytes em cada ciclo do servidor.

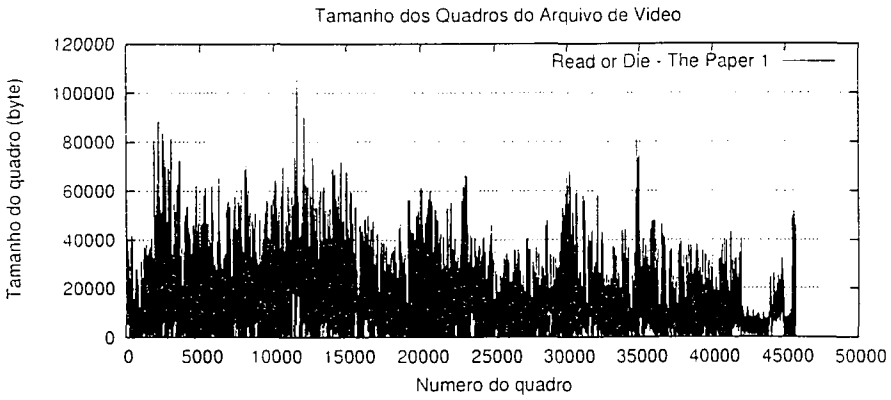


Figura 3.4: Tamanho dos quadros do desenho "Read or Die - The Paper 1".

Este algoritmo permite que o servidor transmita os segmentos de fluxos de vídeo a uma taxa constante. A vantagem deste algoritmo está no fato de se evitar que aumente drasticamente a banda utilizada nos momentos em que quadros muito grandes são transmitidos, sobrecarregando temporariamente a rede.

Para testar o algoritmo foi utilizado o programa cliente *Mplayer* [mpl03], que reproduz vídeo e informa se ocorrem falta de dados na reprodução. Este programa informa se a quantidade de dados enviada é insuficiente ou é excessiva. Através da reprodução dos fluxos de vídeo recebidos do servidor foi constatado que nos momentos em que o cliente reproduz os maiores quadros, o servidor não fornece todos os dados a tempo de sua reprodução, ocasionando a interrupção da reprodução do vídeo no cliente. Isto ocorre porque um quadro grande precisa de vários segmentos de um, ou mais de um, ciclos do servidor para ser transportado ao cliente, exigindo um longo intervalo de tempo para ser transportado inteiramente.

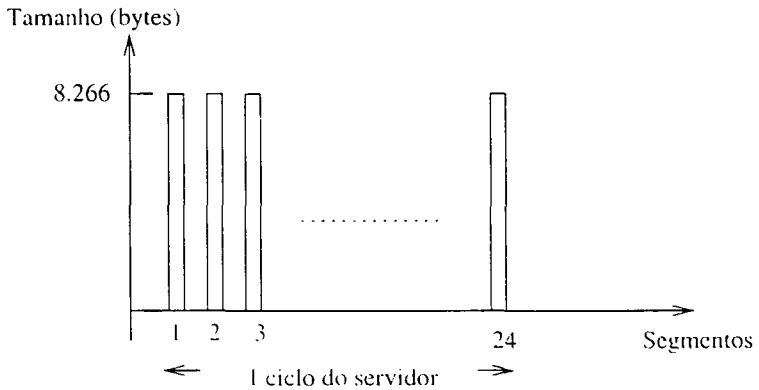


Figura 3.5: Tamanho dos segmentos em todos os ciclos do servidor.

Isto pode ser observado no gráfico da figura 3.6. O gráfico mostra que, do quadro 2.480 ao quadro 2.570, os tamanhos correspondentes estão todos acima de 10.000 bytes com alguns picos acima de 60.000 bytes. Note-se que os picos correspondem a quadros-I. Comparando estes valores com a linha que corresponde à quantidade enviada a cada ciclo, de 8.266 bytes, é observado que ocorre falta de dados no cliente pois a quantia enviada pelo servidor é muito menor do que a quantia necessária para reproduzir estes quadros, mesmo que uma determinada porção seja enviada de forma adiantada quando os quadros reproduzidos são bem menores que o valor da média calculada.

Uma solução para evitar este problema é enviar de forma adiantada todos os dados necessários em alguns segmentos anteriores, mas isto torna a taxa de transmissão variável, incluindo picos e podendo ocasionar instantes de sobrecarga na rede. O objetivo deste algoritmo é garantir o envio dos segmentos a uma taxa constante, desta forma os quadros são enviados de forma adiantada através do aumento da média e a correspondente quantidade de dados de todos os segmentos.

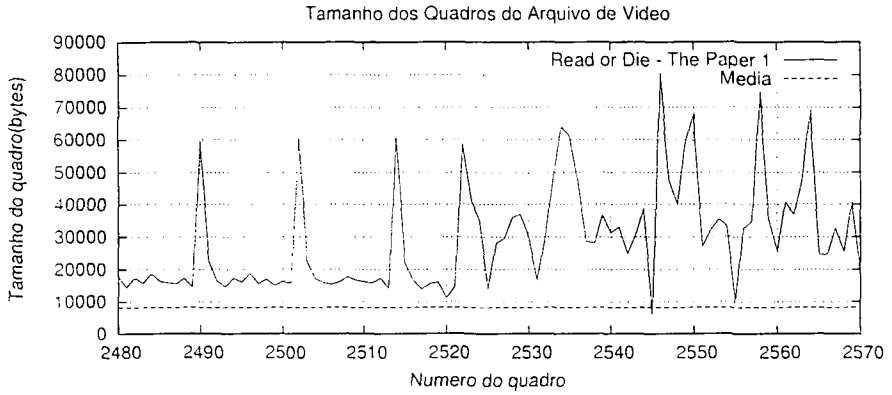


Figura 3.6: Tamanho dos quadros.

Alocação de Banda de Vídeo pela Média Máxima (Max Average Bandwidth)

Para evitar o problema de falta de dados no cliente nos momentos de pico, foi implementado o algoritmo *Alocação de Banda de Vídeo pela Média Máxima* proposto em [FS95a], [Fen97]. Este algoritmo calcula a média máxima dos tamanhos dos quadros do arquivo de vídeo. A quantidade de dados enviada em cada ciclo é o produto do número de *qps* do arquivo de vídeo pelo valor da média máxima calculada, e esta quantidade é constante a cada ciclo. O algoritmo que calcula a média máxima está descrito na figura 3.7.

Para o arquivo de vídeo "Read or Die - The Paper 1" analisado, a média máxima calculada

```
while (!fim_arquivo){
   quadro = busca_proximo_quadro( );
   soma += quadro;
   quantidade++;
   media = (soma + quantidade - 1)/quantidade;
   if (media > media_maxima)
      media_maxima = media;
}
```

Figura 3.7: Algoritmo *Alocação de Banda de Vídeo pela Média Máxima*

é de 14.744 bytes. Desta forma o servidor envia 24 segmentos de dados de 14.744 bytes em cada ciclo, conforme mostra a figura 3.8, totalizando 358.856 bytes em cada ciclo do servidor, enviando assim 155kbytes a mais em cada ciclo quando comparado com o algorimo anterior.

O problema com este algoritmo é o fato de que o valor da média máxima calculada pode ser um valor muito alto comparado com o valor médio dos quadros de todo o arquivo. Isto pode ocorrer quando os maiores quadros estão logo no início do arquivo, determinando um resultado alto no cálculo da média, impedindo que o resultado seja relacionado ao tamanho médio dos quadros. Desta forma o servidor transmite os fluxos de vídeo a uma taxa constante, calculada pelo valor de pico, ocorrendo a sobrecarga no cliente quando este reproduz quadros pequenos.

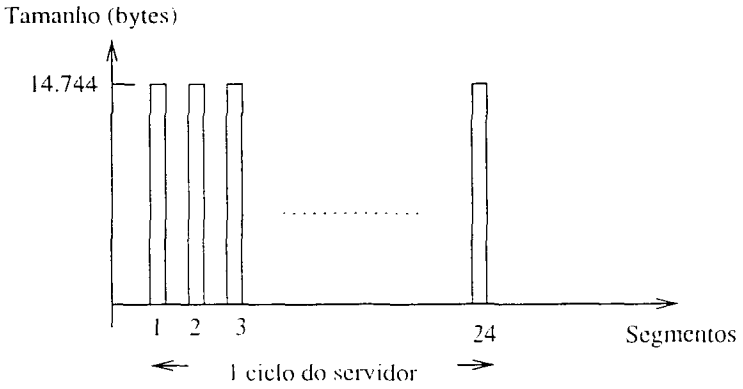


Figura 3.8: Tamanho dos segmentos de todos os ciclos do servidor.

Este problema pode ser observado quando o cliente reproduz os quadros do intervalo 830 ao 940 do gráfico na figura 3.9. Observa-se que neste gráfico a quantidade de dados de cada ciclo é muito pequena quando comparada com o valor calculado e enviado, que é de 5.000 bytes. Neste intervalo, o servidor envia muito mais informações do que o cliente necessita (média máxima = 14.744), ocasionando assim sobrecargas constantes no cliente. Isto é detectado através dos resultados obtidos quando o servidor escreve no *socket* da conexão. Quando a quantidade de dados enviada é excessiva, o servidor não consegue escrever os dados no *socket* e este retorna

um código indicando sobrecarga na conexão (*sockets BSD*).

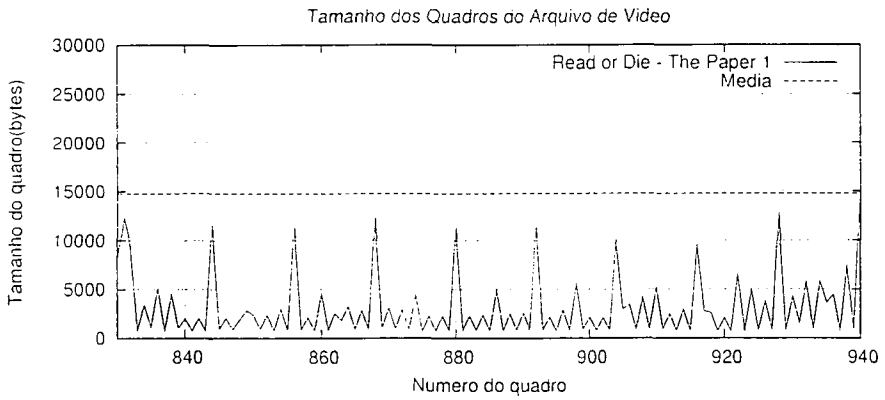


Figura 3.9: Tamanho dos quadros.

Após a implementação e testes dos dois algoritmos citados, chegou-se à conclusão que enviar os fluxos de vídeo a uma taxa constante prejudica a reprodução do vídeo em clientes que não alocam *buffers*, pois pode ocorrer tanto falta de dados na reprodução como sobrecarga na conexão do cliente. Para resolver os problemas encontrados nos algoritmos anteriores foi concluído que a quantidade de dados enviada a cada segmento deve ser proporcional ao tamanho de cada quadro.

Alocação de Banda de Vídeo por Limitação de Picos (Rate-Constrained Bandwidth Smoothing - RCBS) O terceiro algoritmo implementado no servidor é o algoritmo *Alocação de Banda de Vídeo por Limitação de Picos (RCBS)* [Fen97], [FS95a], [KY99]. Este algoritmo envia segmentos de dados proporcional ao tamanho dos quadros de vídeo, desde que estes sejam menores à média máxima dos tamanhos dos quadros do arquivo de vídeo calculada com o algoritmo anterior. Assim, se o quadro a ser transmitido possui tamanho superior ao da média máxima calculada, o servidor envia um segmento do tamanho da média calculada e a quantia não enviada é distribuída pelos segmentos vizinhos de tamanho inferior ao da média calculada. Para evitar atrasos, o excesso é enviado nos segmentos anteriores ao segmento que corresponde ao quadro de maior tamanho. Abaixo está descrito o algoritmo implementado.

Este algoritmo calcula os tamanhos dos segmentos na ordem inversa de reprodução dos quadros, ou seja, do último quadro para o primeiro quadro.

Como este algoritmo tenta enviar o tamanho original de cada quadro, ele evita que ocorra falta de dados para reprodução do vídeo, pois transmite a quantidade de dados necessária e suficiente para a reprodução do vídeo a cada ciclo do servidor. Ao mesmo tempo, este algoritmo evita a ocorrência de sobrecarga instantânea da rede, pois o tamanho do quadro fica limitado a

```
while( numero_quadro>=0 ){
   if (quadro[numero_quadro] > media_maxima){
      segmento[numero_quadro] = media_maxima;
      acumula += (quadro[numero_quadro] - media_maxima);
   } else {
      if ((acumula + quadro[numero_quadro]) <= media_maxima){
         segmento[numero_quadro] = acumula + quadro[numero_quadro];
         acumula = 0;
       } else {
         segmento[numero_quadro] = media_maxima;
         acumula = acumula + segmento[numero_quadro] - media_maxima;
       }
   }
   numero_quadro--;
}
```

Figura 3.10: Algoritmo *Alocação de Banda de Vídeo por Limitação de Picos*

um determinado valor, que é a média máxima calculada.

O algoritmo *RCSB* tem como objetivo manter o envio de segmentos de acordo com o tamanho médio dos quadros, e para que isto ocorra, deve haver mais de um valor de média máxima calculado, pois se houver um único valor da média máxima para todo o arquivo de vídeo este pode referir-se a um único caso isolado correspondendo a um quadro muito grande. Considerando-se somente um valor calculado, pode haver a liberação do envio de quadros de tamanho inferior ao valor da média máxima calculada, mas que ainda correspondem a quantias elevadas a ponto de ocasionar sobrecarga instantânea na rede. Quando o valor da média máxima é calculada para os primeiros quadros do arquivo, esta pode ser quase do tamanho original do quadro.

Para solucionar este problema o *SVFserver* computa a média máxima para cada ciclo do servidor. A implementação deste algoritmo exige dois passos na análise do arquivo de vídeo. No primeiro passo, o servidor examina todos os quadros e determina o valor da média máxima para cada ciclo do servidor em todo o arquivo. Esta análise ocorre na inicialização do servidor, quando este analisa todos os arquivos de vídeo disponíveis. Desta análise resulta um lista para cada arquivo com a quantidade de dados que deve ser lida do disco e enviada a cada ciclo do servidor e o valor da média máxima correspondente ao ciclo.

O segundo passo é executado quando o servidor utiliza a lista para ler do disco a quantidade de dados que deve ser enviada a cada ciclo. Com o valor da média máxima do ciclo obtido da lista, e após o servidor reconhecer todos os quadros que devem ser enviados, o algoritmo *RCBS* é empregado para calcular o tamanho de cada segmento do ciclo correspondente. Esta análise ocorre na ordem inversa de reprodução dos vídeos, permitindo que os dados excedentes sejam

enviados nos segmentos de forma adiantada. O tamanho de cada segmento corresponde ao valor da média máxima quando o quadro é maior que este valor, ou segmento corresponde ao valor original do quadro adicionado o valor excedente, se necessário.

A figura 3.11 mostra o gráfico original com os tamanhos dos quadros de um arquivo de vídeo e a figura 3.12 mostra o mesmo arquivo mas com os tamanhos dos segmentos calculados pelo algoritmo *RCBS*. Observa-se que o segundo gráfico possui menos picos, sendo estes de tamanho inferior comparado com o primeiro gráfico, porque a quantidade excedente dos picos é distribuída pelos segmentos adjacentes que possuem tamanhos inferiores ao valor da média calculada.

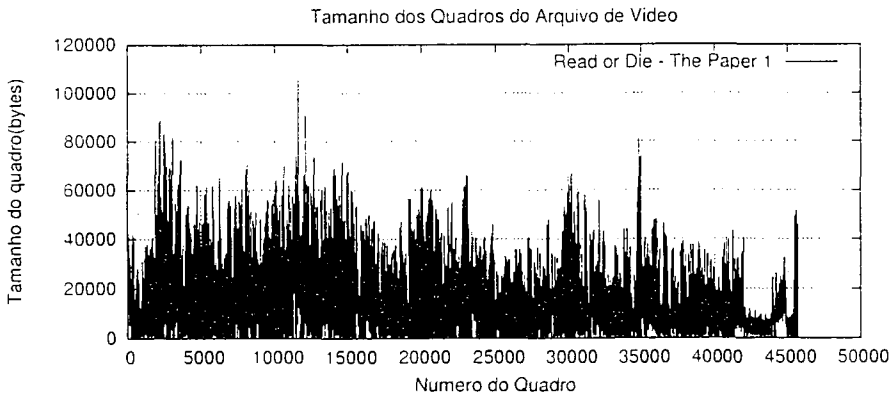


Figura 3.11: Tamanho dos quadros.

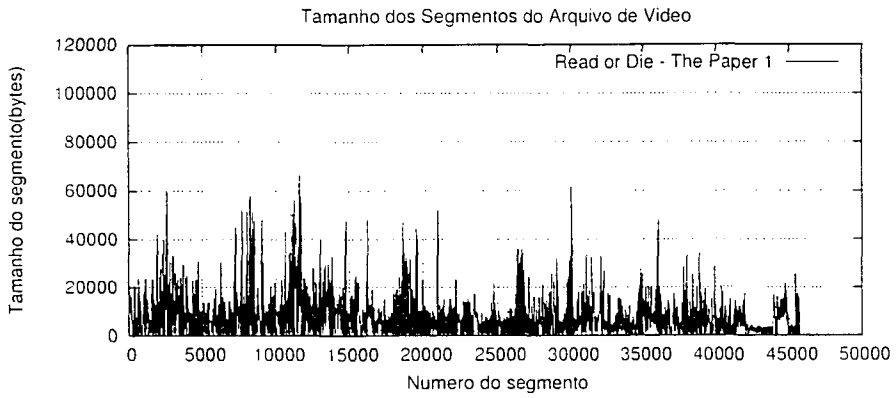


Figura 3.12: Tamanho dos segmentos.

A mesma análise vale para os gráficos das figuras 3.13 e 3.14. Nestes gráficos é mostrado o tamanho dos primeiros 1000 quadros e os correspondentes segmentos do mesmo arquivo de vídeo. Observa-se que no gráfico da figura 3.14 grande parte dos picos é eliminada. Nos dois gráficos é notável que a média dos segmentos enviados é muito menor da média do tamanho dos quadros dos arquivos de vídeo analisados.

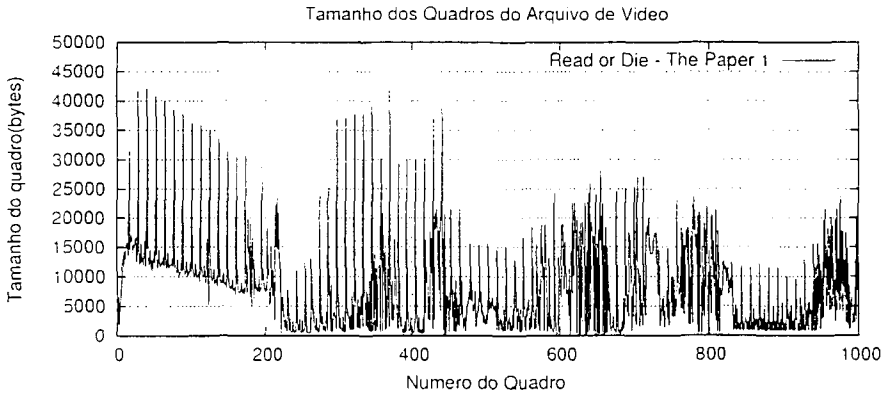


Figura 3.13: Tamanho dos quadros no intervalo 0 a 1000.

A utilização deste algoritmo garante que o servidor envie somente os dados necessários e suficientes ao cliente sem sobrecarregar a rede, permitindo que clientes que não utilizam *buffers* possam reproduzir os fluxos de vídeo sofrendo interferência somente dos atrasos imposta pelo tráfego na rede. Isto pode ser concluido através dos resultados obtidos pelo programa cliente *Mplayer*, que não indicou nenhuma falta de dados na reprodução do vídeo, e pelos resultados de escrita no *socket* que não indicaram conexão sobrecarregada.

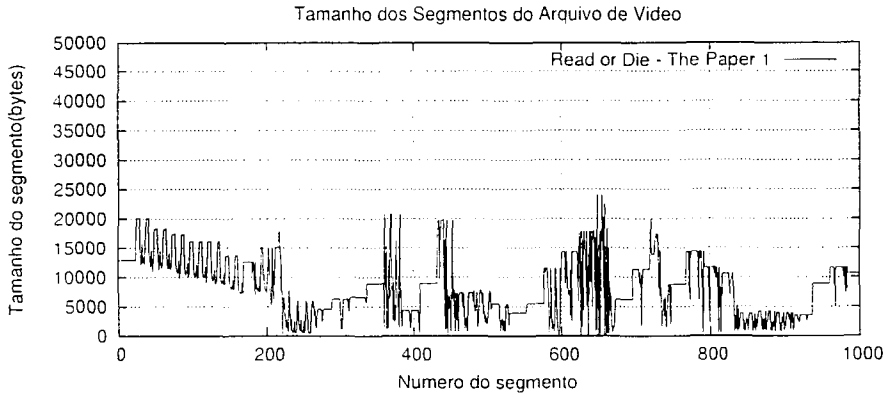


Figura 3.14: Tamanho dos segmentos no intervalo 0 a 1000.

Note-se nestes gráficos a quantidade de dados do gráfico que mostra os tamanhos dos quadros (3.139 é equivalente à quantidade de dados do gráfico que mostra o tamanho dos segmentos (3.14). A integral dos dois gráficos é de igual valor, pois a quantidade de dados representada nos picos do gráfico dos quadros é transferida para as depressões no gráfico dos segmentos.

Observa-se que, visualmente o gráfico 3.11 aparenta possuir maior quantidade de dados que o gráfico 3.12. Isto acontece porque a quantidade de picos representada é muito grande. A figura 3.15 e 3.16 mostram que os dados não são perdidos, mas sim transportados para ciclos vizinhos.

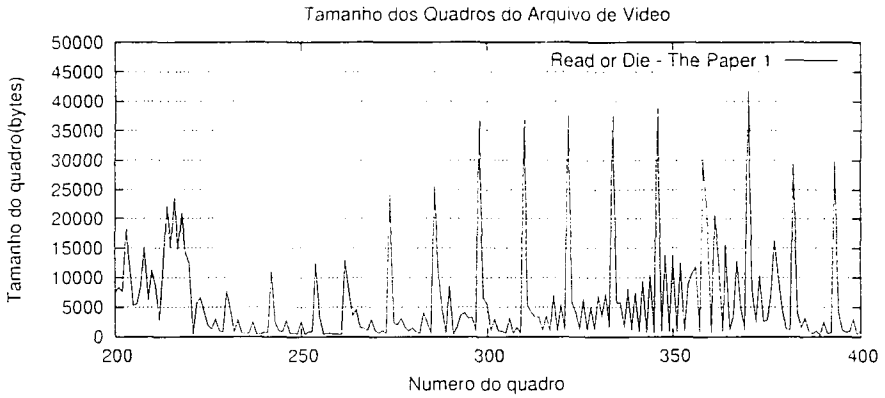


Figura 3.15: Tamanho dos quadros no intervalo 200 a 400.

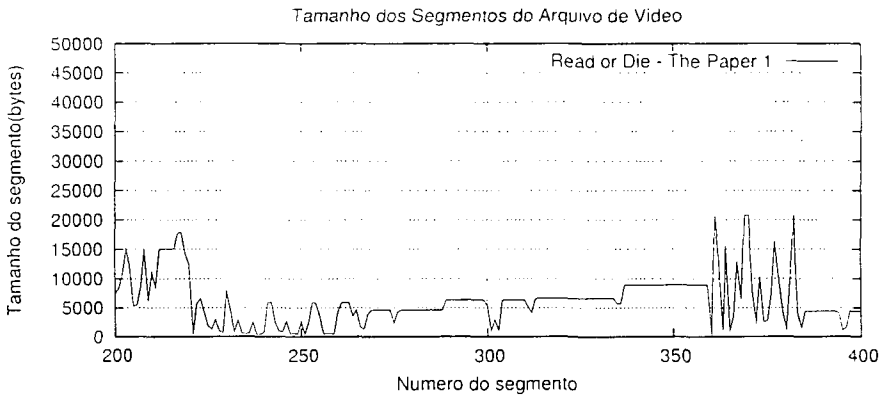


Figura 3.16: Tamanho dos segmentos no intervalo 200 a 400.

3.4 Controle de recursos

Servidores de vídeo são um tipo de serviço que pode ter o seu desempenho comprometido quando ocorrem carências de recursos no seu sistema, interferindo na qualidade do serviço oferecido aos clientes. Entende-se por *Qualidade de Serviço (QoS)* o conjunto de parâmetros (largura de banda, *buffer* utilizado, uso da CPU, tempo de acesso ao disco, dentre outros) que define a qualidade de um fluxo de dados específico [RBS94].

Em [FV90] são citados três níveis de *Qualidade de Serviço (QoS)* que podem ser aplicados a servidores de vídeo. Os três níveis são:

1. *Determinístico*: todos os recursos utilizados tais como tempo de execução e largura de banda são assegurados dentro dos valores previamente propostos. Neste nível, os algoritmos de controle de recursos assumem sempre o pior caso para o cálculo da quantia utilizada, sub-utilizando os recursos controlados;

2. *Estatístico*: a garantia provida no cálculo dos recursos não é de 100%. Neste nível, os algoritmos utilizam valores estatísticos para o cálculo dos recursos utilizados, permitindo que o sistema comporte mais usuários, ocasionando pequenas diminuições na qualidade do serviço prestado.

3. *Melhor Esforço (Best Effort)*: nenhuma garantia é dada. O sistema procura executar da melhor forma possível até tornar-se sobrecarregado.

Para evitar que ocorram sobrecargas no servidor e assegurar o *QoS*, é implementado um *Algoritmo de Controle de Admissão*. Este algoritmo tem como objetivo principal evitar que novas requisições sobrecarreguem o servidor. Quando uma nova requisição é recebida, o algoritmo de controle de admissão verifica se a aceitação desta nova requisição pode em algum momento sobrecarregar algum recurso do servidor, isto é feito através de consultas a algoritmos que mantém uma estimativa de quanto é consumido em cada recurso. A requisição é aceita se, e somente se, nenhum recurso venha a se sobrecarregar.

Vários trabalhos foram propostos na área de controle de recursos em servidores de vídeo para garantir a qualidade do serviço disponibilizado. No servidor *SVFserver* foram implementados algoritmos que controlam os recursos da banda de rede e o tempo de leitura do disco, além de um algoritmo de controle de admissão.

No servidor *SVFserver* o algoritmo de controle de admissão é o algoritmo que recebe os pedidos de novas requisições, utiliza as informações referentes aos recursos utilizados pelo servidor fornecidas pelos algoritmos de controle de recursos, e determina se novas requisições serão admitidas.

Nos próximos capítulos são explicados quais os algoritmos de controle foram implementados no servidor para controlar a banda de rede e o tempo de leitura do disco e é descrito o algoritmo de controle de admissão. Os arquivos de vídeo utilizados como carga nos testes do servidor de vídeo são descritos no início do próximo capítulo.

Capítulo 4

Resultados e Experimentos

A carga utilizada no servidor de vídeo SVFserver para efetuar testes consiste de arquivos de vídeo com diversos graus de compressão. Os arquivos de vídeo de codificação *VBR* possuem melhor qualidade mas impõem maiores demandas ao servidor. Foram escolhidos portanto dez arquivos de vídeo deste tipo de codificação como carga para os testes no servidor.

As características dos arquivos de vídeo utilizados estão descritos na tabela 4.1. Os gráficos com os tamanhos dos quadros e com a quantidade de dados enviada em cada ciclo do servidor encontram-se no Apêndice A, neste apêndice também encontram-se tabelas com informações detalhadas dos arquivos. Estes gráficos foram desenhados na mesma escala para que seja possível vizualizar e comparar o grau de compressão de cada vídeo.

A tabela 4.1 contém as características que determinam a qualidade do vídeo tais como resolução, número de quadros por segundo, tempo de duração e informa a quantidade máxima de dados enviados em um ciclo do servidor para o arquivo correspondente. Este valor corresponde ao tamanho dos *buffers* reservados para requisições feitas àquele arquivo de vídeo.

Para tornar os testes mais realistas foram utilizados arquivos que correspondem a vídeos de filmes, desenhos e shows musicais. Foram escolhidos para os testes vídeos que possuem diferentes graus de compressão e que são codificados com algoritmos de compressão distintos, isto pode ser observado comparando os gráficos do Apêndice A.

Nome	Tipo	Qps	Resolução	Duração[s]	Banda
O Senhor dos Anéis 1 (1)	filme	24	576x240	4623	474.028
O Senhor dos Anéis 1 (2)	filme	24	576x240	6151	467.292
Conan - O Bárbaro	filme	24	512x208	7822	437.396
Shrek	desenho	25	352x288	4057	100.540
Nightwish	show	25	720x400	4796	437.070
Read or Die - The Paper 1	desenho	24	640x480	1905	943.320
Read or Die - The Paper 2	desenho	24	640x480	1850	722.550
O Senhor dos Anéis 1	filme	24	576x240	2163 (36m iniciais)	494.593
O Senhor dos Anéis 2	filme	24	640x272	877 (15m iniciais)	488.816
From the Hell	filme	24	595x240	1755 (30m iniciais)	491.600

Tabela 4.1: Característica dos filmes utilizados nos testes.

Pode ser observado nos gráficos referentes à quantidade de dados enviada a cada ciclo do servidor (Apêndice A) que o vídeo de menor carga no teste corresponde ao desenho "Shrek", o vídeo com maior carga média é o vídeo do desenho "Read or Die - The Paper 2" e o vídeo com maiores picos é o desenho "Read or Die - The Paper 1". Foram utilizados nos testes arquivos de vídeo completos e arquivos de vídeo que correspondem a intervalos iniciais de alguns filmes. Estes últimos foram utilizados para reduzir a duração de testes preliminares.

Os testes realizados no servidor foram executados em uma rede do tipo *LAN - Ethernet* 100Mbps, composta pela máquina servidora e mais três máquinas clientes. Todas as máquinas empregadas nos testes são modelos similares da linha *Presario* da Compac.

As máquinas contém processador Pentium III de 933MHz, com memória RAM de 384Mbytes, um disco modelo Maxtor 36147H8 de 60GBytes, com 512kB de memória cache SDRAM, 5.400 rpm e tempo de busca abaixo de 9.5 ms, e uma placa de rede modelo Accton Technology Corporation SMC2-1211TX.

Nas próximas seções estão descritos os algoritmos de controle de recursos implementados no servidor incluindo a análise dos resultados obtidos nos testes destes algoritmos.

4.1 Controle de Banda de Rede

O algoritmo de controle de banda de rede é utilizado pelo servidor de vídeo para evitar que ocorram congestionamentos na interface de rede. Um dos grandes desafios enfrentados em projetos de servidores de vídeo é garantir o controle de recursos quando são utilizados arquivos de vídeos de largura de banda variável. Vários destes algoritmos foram propostos e são descritos em [DWL96, ZF94, KZ95, DMH99, EKZ95, KZ97].

No SVFserver foi implementado um método determinístico para controle da banda de rede e a implementação é baseada no trabalho de E. Knightly e H. Zhang [KZ97]. Neste trabalho é analisada a diferença entre a utilização do tamanho do maior quadro de um arquivo de vídeo para o cálculo da banda de rede, e a utilização da maior soma dos tamanhos de uma quantia de quadros consecutivos para o cálculo da banda de rede.

É possível considerar o agrupamento de quadros para cálculo da banda de rede devido à seqüência dos tipos de quadros de um arquivo de vídeo. Um quadro do tipo I, que é grande, sempre é seguido de alguns quadros do tipo P e B pequenos.

Para exemplificar, considera-se as seguintes características para um arquivo de vídeo. O maior quadro-I tem 50 kbytes, o número de quadros por segundo é de 25, os quatro quadros que seguem o quadro-I são um quadro-B de 1.1 kbytes, um quadro-B de 1.2 kbytes, um quadro-B de 1.0 kbytes e um quadro-P de 4.5 kbytes, e a soma destes com o quadro-I corresponde à maior soma de uma seqüência de cinco quadros de um arquivo de vídeo, que é de 57.8 kbytes. Para este arquivo, o valor da banda de rede estimada considerando somente o maior quadro é de 50.000 * 25 = 1.250.000 bytes/s. O valor da banda estimada considerando a maior seqüência de cinco quadros consecutivos é de 57.800 * 5 = 289.000 bytes/s.

Observa-se uma grande diferença nos resultados da estimativa para o recurso utilizado. Se o recurso for reservado com base no cálculo da banda de rede considerando somente o tamanho dos quadros-I isolados, haverá desperdício porque este tipo de quadro é infreqüente em arquivos de vídeo.

Foi implementado no SVFserver um algoritmo que administra a utilização da banda de rede, na qual a estimativa da banda utilizada para cada arquivo de vídeo é calculada através da determinação da maior soma de uma seqüência de quadros. A quantidade de quadros considerada na seqüência corresponde ao número de quadros reproduzidos em um segundo para o arquivo de vídeo. Desta forma, a estimativa equivale à maior quantidade enviada em um ciclo do servidor.

A figura 4.1 mostra a quantidade de dados que são enviados em cada ciclo do servidor para o arquivo "Read or Die - The Paper I". Nesta figura observa-se que alguns ciclos enviam quantias de dados acima de 900kBytes. Para este arquivo a banda estimada é de 943.320 bytes, o maior valor medido. Este valor é considerado como a quantidade máxima enviada durante todo o seu período de reprodução, e portanto deve ser considerado como a banda utilizada por este arquivo.

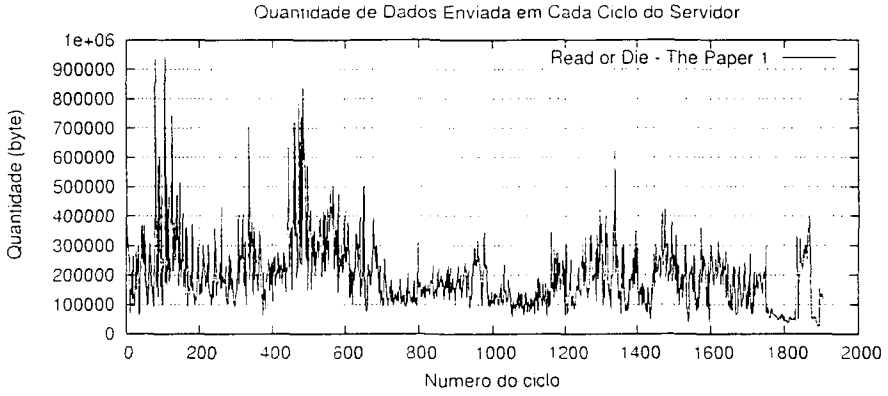


Figura 4.1: Quantidade de dados enviada a cada ciclo do servidor para o desenho "Read or Die - The Paper 1".

Observa-se que este valor corresponde ao tamanho dos *buffers* reservados a cada requisição para o arquivo de vídeo. A estimativa da banda consumida por todas as requisições sendo atendidas é obtida pela somatória dos valores estimados da banda de cada requisição de vídeo que foi aceita.

O algoritmo de controle de banda de rede do SVFserver mantém uma estimativa da banda utilizada pelos vídeos que estão sendo exibidos. Quando é recebida uma nova requisição, o algoritmo de controle de admissão do servidor verifica os resultados do algoritmo de controle de banda e decide se a nova requisição pode ser atendida. Sabendo quanto da banda de rede está sendo utilizada e adicionando a estimativa de banda da nova requisição, se o valor total não ultrapassar 80% da largura de banda disponível, a requisição é aceita.

Para verificar o funcionamento do algoritmo implementado foram realizados testes nos quais o tráfego na interface de rede é medido pelo programa *Xnetload*. Nestes testes o servidor aceita as requisições até o momento em que o algoritmo de controle de admissão detecta que a aceitação de uma próxima requisição acarretará em utilização acima de 80% da largura de banda disponível.

Após iniciado o teste e imposta uma sobrecarga à rede, é verificado se em algum momento o tráfego na interface de rede ultrapassa ao valor de 80% da banda disponível. Como os testes foram realizados em uma rede do tipo *Ethernet* de 100Mbps, o valor de banda disponível às requisições corresponde a 80Mbps.

4.1.1 Testes

Os programas que reproduzem vídeo exigem um nível de processamento alto inviabilizando a execução em paralelo de outros programas que reproduzem vídeo em uma mesma máquina.

Alguns destes programas alocam o driver de vídeo, não permitindo que sejam executados outros programas de reprodução de vídeo ao mesmo tempo. Perante esta dificuldade foi implementado um programa cliente que permite efetuar várias requisições ao servidor a partis de uma mesma máquina.

Este programa cliente possui funções específicas para os testes, ele efetua requisições aos arquivos de vídeo disponibilizados pelo servidor e verifica a quantidade de dados recebidos. Este programa possui a opção de armazenar os dados recebidos em um arquivo no disco, permitindo verificar se o arquivo recebido é idêntico ao arquivo disponibilizado pelo servidor.

Foram realizados três tipos de testes para comprovar a eficácia do método implementado, dois destes consistem de requisições ao servidor para o mesmo arquivo de vídeo, e o outro teste consiste de requisições feitas para arquivos diversos. Nos dois primeiros testes uma nova requisição é realizada a cada segundo, possibilitando a sobreposição dos momentos em que são transmitidos os ciclos com maior quantidade de dados. No terceiro teste as requisições são realizadas em intervalos de até 5 segundos.

Em cada teste é criado um arquivo de log do servidor que registra a quantidade de dados que deve ser enviada a cada ciclo do servidor. Este arquivo também informa se ocorreram atrasos na entrega de fluxos de vídeo ocasionados por sobrecarga na interface de rede. Os atrasos são detectados através da medição do número de ciclos do servidor necessários para enviar cada arquivo de vídeo, que deve se igualar ao número de segundos da reprodução do vídeo. O log informa se os *sockets* não transmitiram todos os dados injetados pelo servidor, indicando assim que em algum momento os *sockets* foram sobrecarregados. Isto é possível através da utilização de *sockets BSD*. Os gráficos que mostram a quantidade de dados enviada em cada ciclo do servidor, mostrados neste capítulo e do Apêndice B, foram plotados a partir destes registros.

A medição do tráfego da rede é realizada pelo programa *Xnetload*. Este programa informa ao final do teste qual a largura de banda máxima enviada através da interface de rede. Este valor é comparado com a quantia máxima injetada pelo servidor em um ciclo do servidor, conforme registrado no arquivo de log. Os dois valores obtidos informam a carga na interface de rede durante o teste e comprovam se o método utilizado para estimar a utilização da banda de rede é válido. Cada um dos testes foi repetido 5 vezes.

No primeiro conjunto de testes foram realizadas requisições ao arquivo "Read or Die - The Paper 1". Este arquivo possui a maior banda de rede calculada, e é o arquivo de vídeo que transmite a maior quantidade de dados em um ciclo do servidor, o que pode ser observado no

gráfico da figura A.12. Observa-se que o valor 943.320 (ver tabela 4.1) é utilizado no cálculo da banda utilizada. Como novas requisições são emitidas a cada segundo, a região de maior carga do arquivo irá sobrecarregar a interface de rede, porque o servidor transmite os ciclos desta região a todos os clientes praticamente ao mesmo tempo.

O resultado dos teste realizados encontra-se na tabela 4.2. Nesta tabela consta o valor máximo enviado em um ciclo do servidor registrado no arquivo de log. Na terceira coluna estão os valores medidos pelo *Xnetload*. O gráfico da figura 4.2 corresponde aos dados armazenados no arquivo de log do primeiro teste realizado (primeira linha da tabela), e estes correspondem à quantidade de dados enviada em cada ciclo do servidor. Os gráficos correspondentes às outras medições encontram-se no Apêndice B.

Observa-se que o valor medido pelo *Xnetload* é um pouco superior ao valor indicado pelo arquivo de log, isto ocorre porque o programa mede a quantia total transmitida na interface de rede incluindo o cabeçalho dos pacotes *TCP/IP*. Nestes testes o algoritmo de controle de admissão aceitou 10 requisições.

Teste	Banda Enviada[kB/s]	Banda Medida [kB/s]
1	6.138	6.305
2	6.137	6.281
3	6.143	6.305
4	5.721	6.319
5	5.845	6.594

Tabela 4.2: Quantidade de dados enviada pelo servidor sem a utilização de filtro (RoD-TP1).

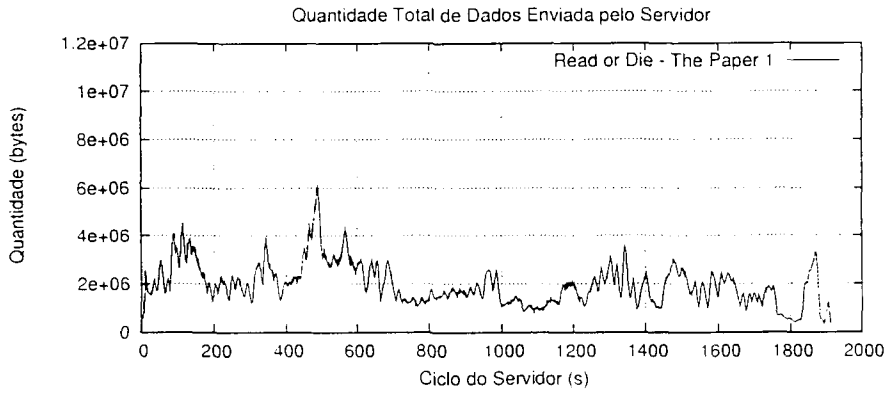


Figura 4.2: Quantidade de dados enviada em cada ciclo do servidor (RoD-TP1).

Observa-se nos resultados mostrados na tabela 4.2 que, apesar de o servidor estimar que 80%

da banda de rede está sendo consumida no teste, o tráfego máximo medido é de 6.594 kBytes/s e que o valor médio obtido nas medições é de 6.361kBytes/s.

O arquivo "Senhor dos Anéis - As Duas Torres" foi utilizado no segunda conjunto de testes, efetuados com o mesmo método dos testes anteriores. Este arquivo utiliza o valor de 488.816 bytes no cálculo da banda utilizada.

O resultado dos testes encontra-se na tabela 4.3. O gráfico da figura 4.3 corresponde aos dados armazenados no arquivo de log do primeiro teste realizado (primeira linha da tabela). O algoritmo de controle de admissão aceitou 20 requisições.

Teste	Banda Enviada[kB/s]	Banda Medida [kB/s]
1	6.120	6.371
2	5.846	6.303
3	5.892	6.186
4	5.853	6.064
5	6.069	6.390

Tabela 4.3: Quantidade de dados enviada pelo servidor sem a utilização de filtro (SdA2).

Observa-se que a banda máxima medida ficou em torno de 6.390 kBytes/s e que a banda média corresponde a 6.263 kbytes/s.

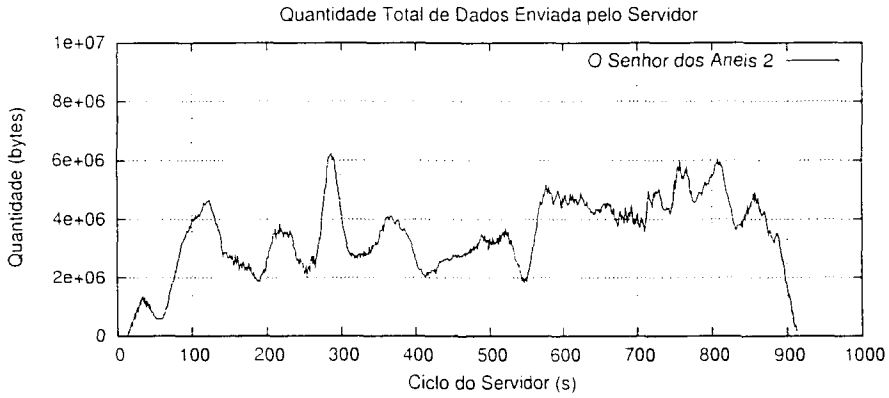


Figura 4.3: Quantidade de dados enviada em cada ciclo do servidor (SdA2).

No terceiro conjunto de testes foram realizadas requisições aos 4 arquivos de vídeo cujo tempo de reprodução está em torno de 30 minutos (36 minutos iniciais de "O Senhor dos Anéis 1", "Read or Die - The Paper 1", "Read or Die - The Paper 2" e "From the Hell"). Foram escolhidos estes arquivos porque o tempo de execução do teste fica em torno do tempo de reprodução dos arquivos, reduzindo assim o tempo de execução dos testes. Nestes testes foram geradas

requisições a cada 3 segundos de forma aleatória aos 4 arquivos disponibilizados, resultando em um número diferente de requisições aceitas para cada teste. Os valores utilizados no cálculo da banda correspondem aos valores mostrados nas tabelas do capítulo 4.

O resultado dos testes realizados encontram-se na tabela 4.4, e nesta tabela são informados a quantidade de requisições aceitas em cada teste. O gráfico da figura 4.4 corresponde aos dados armazenados no arquivo de log do primeiro teste realizado (primeira linha da tabela).

Teste	N. de req.	Banda Enviada[kB/s]	Banda Medida [kB/s]
1	16	4.158	4.229
2	15	4.146	4.410
3	13	4.298	4.434
4	16	4.983	5.147
5	14	5.056	5.182

Tabela 4.4: Quantidade de dados enviada pelo servidor sem a utilização de filtro (Vários).

Observa-se nestes testes que a maior banda medida chegou a 5.182kbytes/s e o valor médio obtido nas medições é de 4.680kBytes/s. Observa-se nos resultados da tabela 4.4 que os valores são muito menores do valor esperado de 80%, e os picos medidos são menores que os valores obtidos nos testes realizados com requisições a somente um arquivo de vídeo. Isto ocorre porque em cada arquivo de vídeo os ciclos que enviam as maiores quantidades de dados ocorrem em períodos distintos. Assim, se são enviados fluxos de diversos arquivos, geralmente os picos deste arquivos não são sobrepostos.

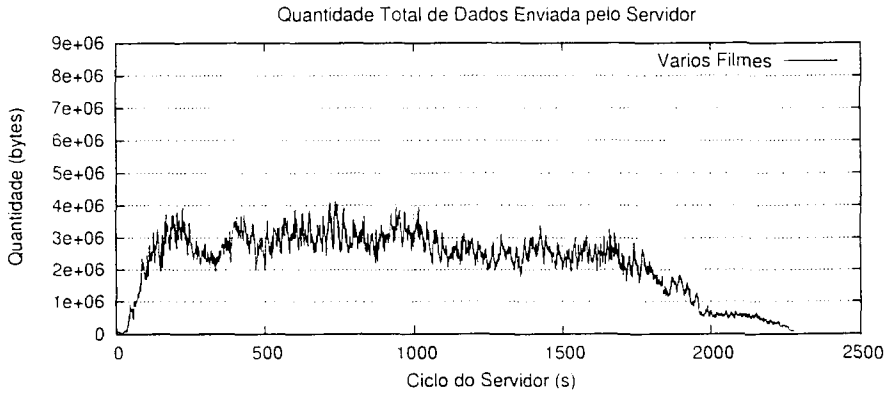


Figura 4.4: Quantidade de dados enviada em cada ciclo do servidor (Vários).

A tabela 4.5 mostra um resumo dos valores obtidos nos três conjuntos de testes realizados. Observa-se que a maior banda medida corresponde aos testes em que foram realizadas requisições

ao arquivo de vídeo com a maior banda estimada. A terceira linha mostra que a banda máxima medida é menor quando as requisições são feitas a arquivos diversos.

Conjunto	Banda Máxima[kB/s]	Banda Média [kB/s]
1	6.594	6.361
2	6.390	6.263
3	5.182	4.680

Tabela 4.5: Quantidade de dados medidos pelo *Xnetload*.

Após realizados testes no servidor com o algoritmo baseado no trabalho de E. Knightly e H. Zhang, foi comprovado que este impede a ocorrência de sobrecarga na rede, mas pode induzir a sua sub-utilização. Este fato ocorre porque a cada ciclo do servidor uma quantidade diferente de dados é enviada, podendo esta ser freqüentemente menor que a quantia utilizada na estimativa da banda. Isto pode ser observado nos gráficos que mostram a quantidade de dados enviados a cada ciclo do servidor do Apêndice B para os arquivos de vídeo empregados nos testes (figuras B.1, B.2 e B.3, por exemplo). Para aumentar a utilização da banda de rede é proposto um método que utiliza filtros no cálculo da estimativa da banda utilizada pelos arquivos de vídeo.

Filtros para a estimativa de banda Para diminuir o valor estimado da banda de rede, e portanto aumentar o número de requisições aceitas pelo servidor, é proposta a utilização de um filtro que determina o valor da banda de acordo com a quantidade de dados que deve ser transmitida durante um intervalo de vários ciclos contíguos.

Este filtro é necessário porque em alguns arquivos de vídeo a banda estimada é determinada através de picos isolados, como mostra a figura 4.5. Esta figura mostra a quantidade de dados enviada em cada ciclo do servidor para o filme "From the Hell". Observa-se no gráfico que existe um pico isolado próximo ao ciclo 900, e este corresponde ao valor de 491 kbytes. Portanto o valor 491 kBytes é considerado na estimativa da banda consumida por este arquivo. Mas este valor é muito superior ao valor médio transmitido nos outros ciclos, ocasionando assim desperdício de banda por parte do servidor, como demonstrado nos testes anteriores.

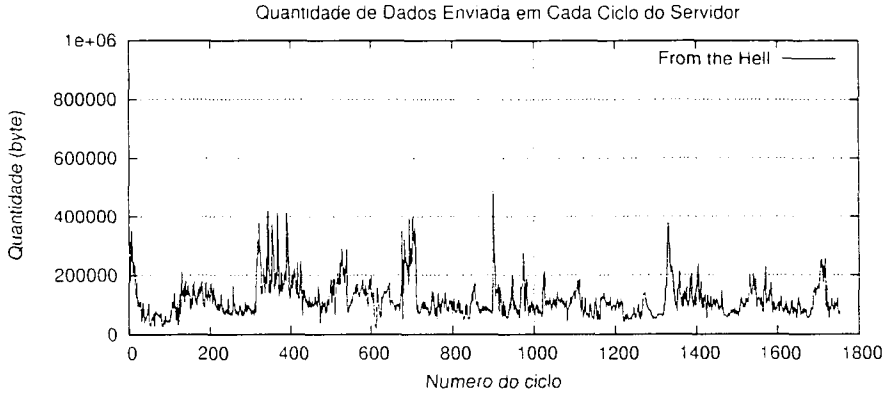


Figura 4.5: Quantidade de dados enviada a cada ciclo do servidor para o filme "From the Hell", 30 minutos iniciais.

O filtro é aplicado para calcular a média ponderada da quantidade de dados de um ciclo com relação aos ciclos vizinhos. Desta forma, o valor estimado é reduzido, evitando que picos isolados sejam utilizados no cálculo da estimativa e possibilitando que os intervalos em que muitos ciclos transmitem grandes quantidades de dados determinem a estimativa de banda consumida, como a região entre os ciclos 670 e 710 da figura 4.5.

O filtro funciona como um filtro passa-baixa que elimina o ruído (picos) de um sinal. Tais filtros não são citados na literatura nem nos artigos pesquisados que propõem algoritmos de controle de banda de rede deterministicos.

Os dois filtros definidos abaixo, *filtro 1*, definido pela Equação 4.1, e *filtro 2*, definido pela Equação 4.2, são utilizados para estimar novos valores de banda de rede utilizada pelos arquivos de vídeo.

$$banda1 = \sum_{-6 \leq n \leq -4} 0.05 \cdot ciclo_n + \sum_{-3 \leq n \leq +3} 0.1 \cdot ciclo_n + \sum_{+4 \leq n \leq +6} 0.05 \cdot ciclo_n \quad (4.1)$$

$$banda2 = 0.025 \cdot ciclo_{n-10} + \sum_{-9 \leq n \leq +9} 0.05 \cdot ciclo_n + 0.025 \cdot ciclo_{n+10} \quad (4.2)$$

A equação 4.1 calcula o valor da banda considerando as quantias enviadas dos 6 ciclos adjacentes anteriores e posteriores. A equação 4.2 calcula o valor da banda considerando as quantias enviadas dos 10 ciclos adjacentes anteriores e posteriores.

quantias enviadas dos 10 ciclos adjacentes anteriores e posteriores.

Outros filtros foram testados com diversas larguras de intervalo e verificou-se que ao aumentar a quantidade de ciclos adjacentes, a banda estimada diminui, tendendo ao valor da média da quantidade de dados enviada em cada ciclo do servidor.

Os filtros das equações 4.1 e 4.2 são filtros genéricos, e podem ser utilizados com qualquer arquivo de vídeo. Para demonstrar que os filtros não são viciados a determinados arquivos de vídeo, eles foram usados em todos os arquivos utilizados como cargas de testes neste servidor e citados no início deste capítulo. Observa-se que os arquivos utilizados como cargas de teste são vídeos de gêneros distintos (filmes, desenhos, shows), e possuem graus de compressão diferentes, como pode ser observado no Apêndice A.

A tabela 4.6 contém o valor estimado original da banda de cada filme e os novos valores calculados com as equações 4.1 e 4.2. Nesta tabela também são mostrados o percentual de redução dos novos valores com relação à estimativa inicial da banda.

Arquivo	Banda Inicial	Banda Eq1	Redução (%)	Banda Eq2	Redução (%)
O Senhor dos Anéis 1 - 1	474,0	326,0	31,2	309,0	34,8
O Senhor dos Anéis 1 - 2	467,3	295,5	36,8	279,7	40,1
Conan - O Bárbaro	437,4	259,8	40,6	210,5	51,9
Shrek	100,5	87,0	13,4	76,9	23,5
Nightwish	437,1	263,1	39,8	238,3	45,5
Read or Die - The Paper 1	943,3	606,7	35,7	570,0	39,6
Read os Die - The Paper 2	722,5	483,6	33,1	449,7	37,8
O Senhor dos Anéis 1 (36 m.)	494,6	372,8	24,6	344,8	30,3
O Senhor dos Anéis 2 (15 m.)	488,8	380,7	22,1	330,8	32,3
From the Hell (30 m.)	491,6	322,8	34,3	291,8	40,6

Tabela 4.6: Banda de rede estimadas segundo às equações 4.1 e 4.2 [kB/s].

Observa-se na tabela 4.6 que a diferença entre os valores calculados pelo filtro 4.1 e o filtro 4.2 é da ordem de 13%, se comparada com a diferença entre o resultado obtido com o filtro 1 e o valor original, que é cerca de 40%. Se fosse incluído um terceiro filtro com um intervalo maior de vizinhança, o resultado obtido provavelmente não seria muito menor que os valores obtidos com os outros filtros. Devido à variedade de tipos de arquivos utilizados nos testes, pode-se concluir que estes filtros são adequados também à outros arquivos de vídeo.

4.1.2 Testes Realizados utilizando o Filtro 1

Os testes descritos na seção 5.1 foram repetidos considerando novos valores estimados de banda de rede calculados pelo filtro da equação 4.1.

O resultado dos cincos testes realizados com o desenho "Read or Die - The Paper 1" encontram-se na tabela 4.7. O gráfico da figura 4.6 corresponde aos dados armazenados no arquivo de log do primeiro teste realizado (primeira linha da tabela). Nestes testes o algoritmo de controle de admissão aceitou 16 requisições, 6 requisições a mais que o teste anterior.

Teste	Banda Enviada[kB/s]	Banda Medida [kB/s]
1	9.126	9.927
2	9.961	10.023
3	8.364	8.840
4	9.599	10.089
5	8.817	9.109

Tabela 4.7: Quantidade de dados enviada pelo servidor utilizando o filtro 4.1 (RoD-TP1).

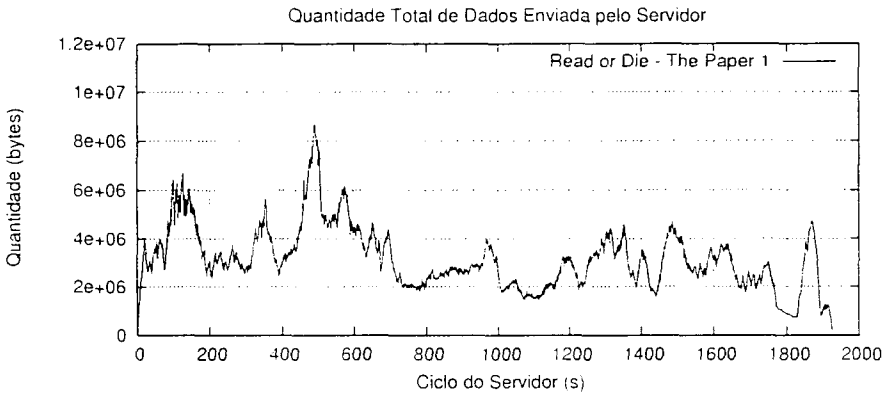


Figura 4.6: Quantidade de dados enviada em cada ciclo do servidor considerando a utilização do filtro 4.1(RoD-TP1).

Observa-se nos resultados obtidos, mostrados na tabela 4.7, que nas medições feitas pelo *Xnetload* o segundo e o quarto testes ultrapassaram o valor de 80% da banda de rede disponível. Nestes dois testes não ocorreram atrasos na entrega dos fluxos de vídeo, pois estes valores são muito próximos ao valor de 10Mbytes/s. O tráfego máximo medido é de 10.089kByte/s e o valor médio obtido nas medições é de 9.598kBytes/s.

O resultado dos cincos testes realizados com o trecho de 15 minutos iniciais do filme "O senhor dos Anéis - As Duas Torres" encontram-se na tabela 4.8. O gráfico da figura 4.7 corresponde

aos dados armazenados no arquivo de log do primeiro teste realizado (primeira linha da tabela). Nestes testes o algoritmo de controle de admissão aceitou 26 requisições, 6 requisições a mais que o obtido anteriormente.

Teste	Banda Enviada[kB/s]	Banda Medida [kB/s]
1	7.410	7.678
2	7.495	7.800
3	7.538	7.700
4	7.491	7.607
5	7.322	7.568

Tabela 4.8: Quantidade de dados enviada pelo servidor utilizando o filtro 4.1 (SdA2).

Observa-se nestes testes que a maior banda medida chegou a 7.800kbytes/s e o valor médio obtido nas medições é de 7.671kBytes/s, e que nos resultados da tabela 4.8 os valores são inferiores ao valor esperado de 80%. Neste teste ainda existe uma fração de banda de rede sendo desperdiçada.

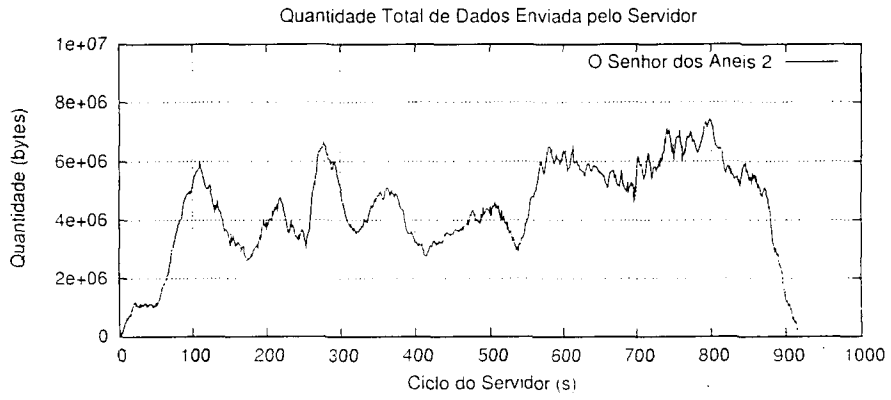


Figura 4.7: Quantidade de dados enviada em cada ciclo do servidor considerando a utilização do filtro 4.1 (SdA2).

No terceiro conjunto de testes foram realizadas requisições aos 4 arquivos de vídeo utilizados como carga, e neste conjunto foi obtido um número diferente de requisições aceitas para cada teste devido a aleatoriedade das requisições. Os valores utilizados no cálculo da estimativa da banda correspondem aos valores mostrados na tabela 4.6.

O resultado dos cincos testes realizados encontram-se na tabela 4.9. O gráfico da figura 4.8 corresponde aos dados armazenados no arquivo de log do primeiro teste realizado (primeira linha da tabela).

Teste	N. de req.	Banda Enviada[kB/s]	Banda Medida [kB/s]
1	22	5.934	5.969
2	24	6.485	6.600
3	19	6.795	6.851
4	26	6.514	6.630
5	21	6.525	6.653

Tabela 4.9: Quantidade de dados enviada pelo servidor utilizando o filtro 4.1 (Vários).

Observa-se nestes testes que a maior banda medida chegou a 6.851kbytes/s e o valor médio obtido nas medições é de 6.541kBytes/s. O maior valor de banda medido corresponde ao teste com menor número de requisições aceitas, porque neste teste ocorreram muitas requisições ao arquivo de vídeo de maior banda. Observa-se nos resultados da tabela 4.9 que os valores são inferiores ao valor esperado de 80%. Nestes testes ainda existe uma fração de banda de rede sendo mal utilizada porque a estimativa é exagerada.

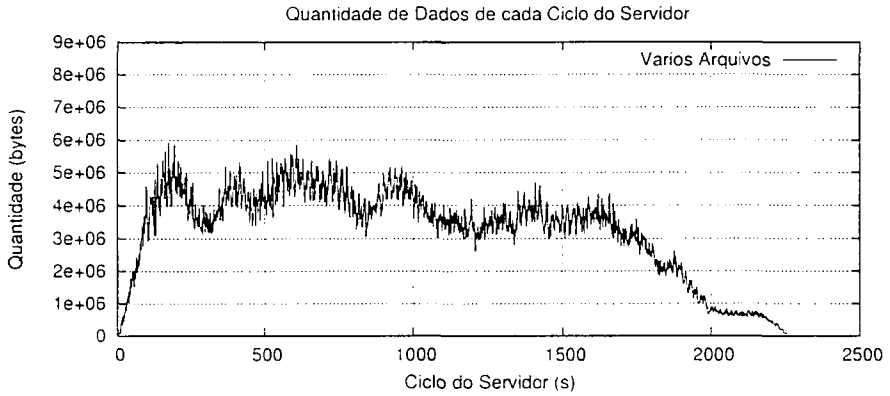


Figura 4.8: Quantidade de dados enviada em cada ciclo do servidor considerando a utilização do filtro 4.1 (Vários).

4.1.3 Testes Realizados com o Filtro 2

Os mesmos conjuntos de testes foram repetidos, desta vez, considerando os novos valores estimados de banda de rede utilizando no seu cálculo o filtro da equação 4.2.

O resultado dos cincos testes realizados com o desenho "Read or Die - The Paper 1" encontram-se na tabela 4.10. O gráfico da figura 4.9 corresponde aos dados armazenados no arquivo de log do primeiro teste realizado (primeira linha da tabela). Nestes testes o algoritmo de controle de admissão aceitou 17 requisições, 1 a mais que no teste com o filtro 4.1.

Teste	Banda Enviada[kB/s]	Banda Medida [kB/s]
1	9.775	9.997
2	10.708	10.561
3	9.951	10.053
4	9.515	9.578
5	10.236	10.299

Tabela 4.10: Quantidade de dados enviada pelo servidor utilizando o filtro 4.2 (RoD-TP1).

Observa-se nos resultados mostrados na tabela 4.10 que nas medições com o *Xnetload* o segundo, o terceiro e o quinto testes ultrapassaram o valor de 80% da banda de rede disponível, e o primeiro teste aproximou-se do valor limite. Em nenhum dos testes ocorreram atrasos na entrega dos fluxos de vídeo, nem mesmo no segundo teste que acusou medição do tráfego na rede inferior ao valor enviado pelo servidor. Isto ocorreu porque não existe um sincronismo entre o período em que sistema enviou os dados na interface de rede e o período em que o *Xnetload* mediu a interface de rede, e a quantidade não medida pelo *Xnetload* em uma medição é computada na próxima medição realizada.

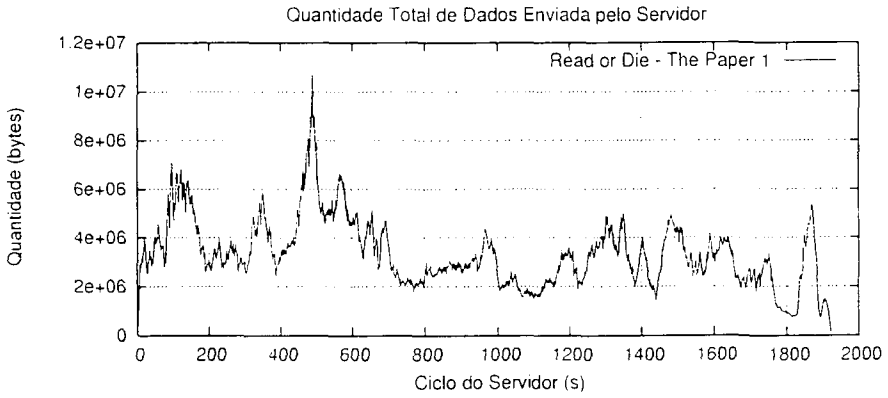


Figura 4.9: Quantidade de dados enviada em cada ciclo do servidor considerando a utilização do filtro 4.2 (RoD-TP1).

O tráfego máximo medido é de 10.561kByte/s e o valor médio obtido nas medições é de 10.098kBytes/s. Estes valores não são muito maiores que os valores obtidos no conjunto de testes com o filtro 4.1 porque este filtro permitiu que somente uma requisição a mais fosse satisfeita pelo servidor.

O resultado dos cincos testes realizados com o trecho de 15 minutos iniciais do filme "O Senhor dos Anéis - As Duas Torres" encontram-se na tabela 4.11. O gráfico da figura 4.10 mostra os dados registrados no arquivo de log do primeiro teste realizado (primeira linha da

tabela). Nestes testes o algoritmo de controle de admissão aceitou 30 requisições. 4 requisições a mais que no teste com o filtro 4.1.

Teste	Banda Enviada[kB/s]	Banda Medida [kB/s]
1	8.465	8.602
2	8.594	8.889
3	8.322	8.437
4	8.578	8.693
5	8.518	8.672

Tabela 4.11: Quantidade de dados enviada pelo servidor utilizando o filtro 4.2 (SdA2).

Observa-se nestes testes que a maior banda medida chegou a 8.889kbytes/s e o valor médio é de 8.859kBytes/s. Observa-se nos resultados da tabela 4.11 que os valores são inferiores ao valor esperado de 80%, o que mostra que uma fração da banda disponível é mail utilizada. Neste teste ainda existe uma quantidade de banda de rede sendo desperdiçada.

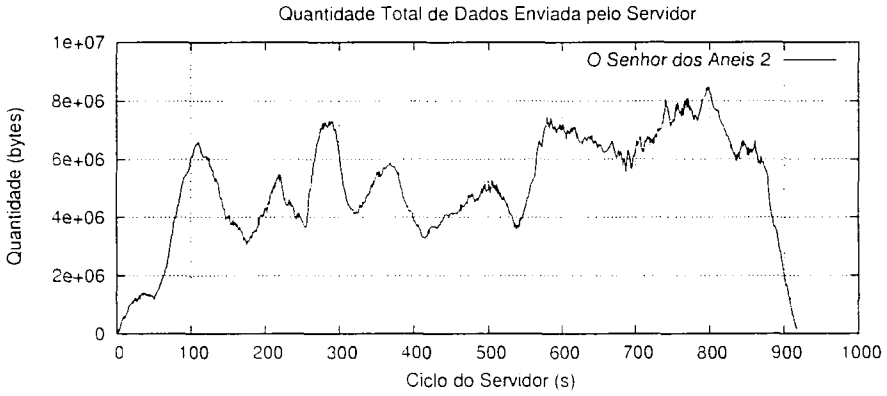


Figura 4.10: Quantidade de dados enviada em cada ciclo do servidor considerando a utilização do filtro 4.2 (SdA2).

No último conjunto de testes foram realizadas requisições aos 4 arquivos de vídeo utilizados como carga de testes. Este conjunto foi obtido um número diferente de requisições aceitas para cada teste devido a aleatoriedade das requisições. Os valores utilizados na estimativa do cálculo da banda correspondem aos valores mostrados na tabela 4.6.

O resultado dos cincos testes realizados encontram-se na tabela 4.12. O gráfico da figura 4.11 corresponde aos dados armazenados no arquivo de log do primeiro teste realizado (primeira linha da tabela). O número de requisições aceitas encontra-se na tabela 4.12.

Teste	N. de req.	Banda Enviada[kB/s]	Banda Medida [kB/s]
1	24	6.575	6.598
2	26	6.967	7.043
3	22	8.276	8.358
4	23	7.584	7.759
5	26	6.563	6.864

Tabela 4.12: Quantidade de dados enviada pelo servidor utilizando o filtro 4.2 (Vários).

Observa-se nestes testes que a maior banda medida chegou a 8.358kbytes/s e o valor médio obtido nas medições é de 7.324kBytes/s. Observa-se nos resultados da tabela 4.12 que os valores são um pouco inferiores ao valor esperado de 80%. Neste teste nota-se que há uma fração de banda de rede sendo mal utilizada.

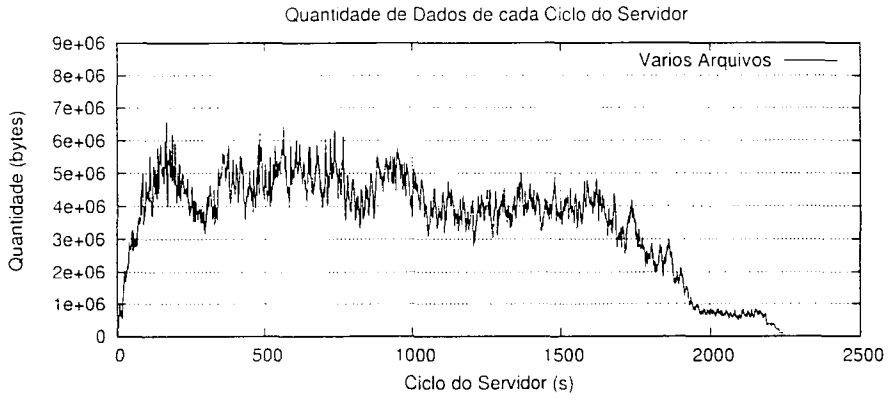


Figura 4.11: Quantidade de dados enviada em cada ciclo do servidor considerando a utilização do filtro 4.2 (Vários) .

Analisando os resultados obtidos nos testes conclui-se que o filtro 4.2 permite um aumento na utilização da banda de rede quando esta disponibiliza fluxos de vídeo de diversos arquivos. Na situação em que são feitas requisições a um mesmo arquivo de vídeo, e este possui intervalos em que são enviados quantidades muito grandes de dados se comparados com o valor médio, podem ocorrer períodos em que a quantidade enviada ultrapassa o valor limite do servidor, sem que tenham sido observadas interrupções na exibição dos vídeos.

Em uma rede local Ethernet de 100Mbps, o *SVFserver* admitiu 10 requisições feitas ao arquivo de vídeo "Read or Die - The Paper 1" , utilizando 60

Considerando-se os resultados dos testes aqui relatados, conclui-se que a utilização dos filtros 4.1 e 4.2 propostos no cálculo da estimativa de banda de rede alocada a cada requisição é adequada porque permite aumentar a utilização da banda de rede e obter um melhor aproveitamento

do servidor sem interferir negativamente no desempenho dos clientes.

Por conta destes resultados foi decidido pelo uso do filtro 4.2 no servidor. Se a banda de rede estiver sendo sub-utilizada, o responsável pela operação do servidor pode aumentar o valor limite da banda de rede, permitindo assim que um número maior de requisições sejam aceitas pelo servidor, já que é improvavel que requisições sejam feitas ao arquivo de vídeo com os maiores picos de transmissão ao mesmo tempo.

O método proposto por E. Knightly e H. Zhang para estimar a banda é válido mas sub-utiliza a banda de rede. Este método é eficaz quando é utilizado juntamente com um filtro que melhora a avaliação da banda de rede dos arquivos de vídeo, como aqueles propostos neste trabalho.

O próximo capítulo descreve o algoritmo para controlar o tempo de leitura de disco no servidor, e analisa os resultados obtidos nos testes realizados com este algoritmo.

4.2 Controle de Tempo de Leitura de Disco

O controle de banda de rede limita o número de requisições quando o servidor é executado em uma rede com banda limitada, como por exemplo uma rede de 100Mbps. Quando o servidor é executado em uma rede de 1Gbps, o serviço pode sofrer interferências devido a sobrecargas nos outros recursos, como por exemplo o recurso disco.

Além do controle da banda de rede, discutido no capítulo 5, o outro recurso controlado no servidor é o tempo de leitura de disco. Existem trabalhos publicados que consideram aproximações determinísticas que incluem no cálculo do tempo de leitura de dados do disco, o tempo de busca, o tempo de rotação e o tempo de transporte dos dados do disco à memória principal [RV91, PRR92, DAG92, RV93].

A escolha do método utilizado para o controle do tempo de leitura dos dados do disco depende da forma como os dados são armazenados. Os dados podem ser armazenados de três formas:

1. *Tempo Contante (Constant Time Length CTL)*: A quantidade de dados é armazenada em blocos correspondendo a um tempo fixo de reprodução. Se a codificação dos arquivos de vídeo é CBR, então todos os blocos possuem tamanhos iguais. Se a codificação dos arquivos de vídeo é VBR, então os blocos possuem tamanhos distintos;

2. *Tamanho Constante (Constant Data Length CDL)*: A quantidade de dados armazenada no disco corresponde a blocos com uma quantidade fixa de dados, neste caso o tempo de

reprodução para cada bloco será equivalente se a codificação dos arquivos é CBR e distinto se a codificação dos arquivos é VBR;

3. *Aleatoriamente*: Os dados são armazenados em qualquer posição do disco e em blocos de tamanhos variados, a decisão quanto ao local de escrita é tomada pelo sistema operacional.

Em [CZ94, CZ96] é comparado o desempenho de dois algoritmos determinísticos distintos para arquivos de vídeo com codificação *VBR* e que utilizam a forma *CTL* de armazenamento no disco. Em [HVG94a, HVG94b] são discutidos algoritmos de controle estatísticos considerando a forma *CTL* de armazenamento no disco.

Para controlar o tempo de leitura do disco no servidor *SVFserver* foi empregada uma forma de cálculo determinístico, onde a soma total dos tempos de leitura dos dados de todas as requisições é limitada em um determinado valor.

O servidor de vídeo *SVFserver* é normalmente executado em um sistema que permite que os arquivos de vídeo sejam armazenados de forma aleatória no disco, e os dados são gravados em qualquer posição e em blocos de tamanhos variados independentemente do tempo de reprodução, resultando assim em tempos de busca distintos, tanto para arquivos *CBR* como para arquivos *VBR*.

Em cada ciclo do servidor é preenchido um *buffer* com os dados lidos do disco para cada requisição. Desta forma, a soma dos tempos das leituras de todas as requisições deve ser menor que o ciclo do servidor, evitando assim atrasos na entrega dos fluxos de vídeo.

A chamada de sistema utilizada na leitura dos dados é a função *read()*. A implementação desta função no servidor viabiliza o preenchimento ordenado dos *buffers* de acordo com a ordem de chegada das requisições e de acordo com a ordem do seu esvaziamento. Quando se utiliza a função *read()*, o algoritmo de escalonamento de leitura de disco não modifica a ordem de preenchimento dos *buffers* no servidor.

Para implementar o controle determinístico nesta configuração é medido o tempo de leitura dos dados a cada *read()* executado pelo servidor. A medida de tempo é obtida com a chamada de sistema *gettimeofday()*, esta função permite uma medição com resolução de microssegundos. O servidor mede o tempo através de *gettimeofday()*, invoca a função de leitura dos dados do disco, e mede o tempo novamente. A duração do intervalo de leitura corresponde à diferença do último tempo medido com relação ao primeiro. Para evitar que outros processos interfiram na medição de leitura dos dados do disco, o servidor é executado com prioridade máxima.

O tempo total dispendido na leitura dos dados do disco em um ciclo do servidor corresponde à soma de todas as medições efetuadas no ciclo. Quando chega uma nova requisição, o algoritmo de controle de admissão verifica com o algoritmo de controle de acesso ao disco se a soma das medições realizadas em um ciclo do servidor ultrapassa ao valor de um tempo determinado. Se o valor for superior ao valor determinado o recurso de disco está sobrecarregado, e a nova requisição não é aceita. Os parâmetros de decisão de sobrecarga do estão descritos na poxima seção.

Para medir a carga na interface de disco foram realizados testes com as requisições dos clientes efetuadas na mesma máquina em que o servidor é executado, evitando assim que ocorram sobrecargas na interface de rede. Nestes testes foi utilizado somente o programa cliente implementado especificamente para os testes do servidor, o programa *Mplayer* não foi utilizado. Nestes testes foram criados arquivos de log que informam os tempos medidos pelo servidor e através destes arquivos são plotados gráficos com o tempo total de leitura em cada ciclo do servidor.

Testes Realizados O primeiro teste realizado tem o objetivo de mostrar qual é o comportamento quando 50 requisições são aceitas pelo servidor. As requisições são a arquivos distintos, sendo estes os 10 arquivos descritos no seção de cargas de testes. Efetuar requisições a arquivos distintos pode ocasionar um aumento no percentual do tempo de busca do disco no tempo final de leitura de cada requisição, pois cada arquivo é armazenado em locais diferentes do disco.

Através deste teste detectou-se que, em alguns ciclos, o tempo total de leitura ficou acima de 1 segundo. Isto ocorre por dois motivos: primeiro porque os arquivos são codificados no formato VBR podendo resultar em períodos longos para leitura dos dados nos ciclos que envolvem a transmissão de grande quantidade de dados; e, segundo, porque os dados são armazenados aleatoriamente no disco provocando tempos de busca relativamente longos devido à movimentação do braço do disco.

O gráfico da figura 4.12 mostra a soma dos tempos gastos nas leituras de disco em cada ciclo do servidor no teste realizado.

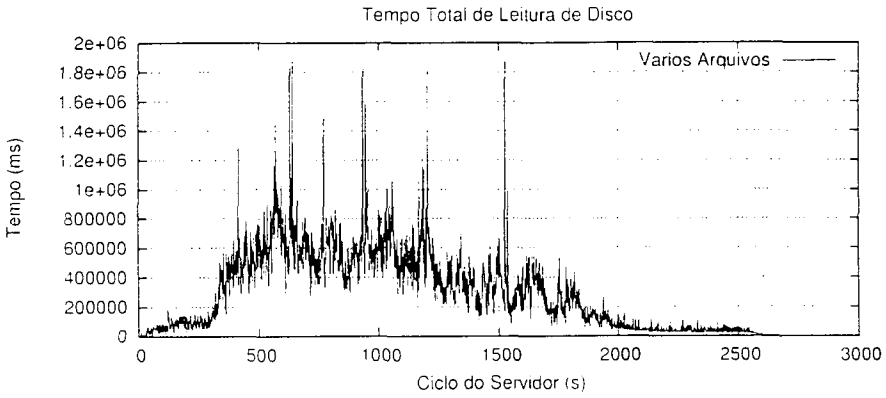


Figura 4.12: Tempo total de leitura dos dados do disco em cada ciclo do servidor.

Observa-se na figura 4.12 que em vários momentos o tempo total medido ultrapassou 1 segundo. Nestes momentos o servidor fica a espera de que os últimos *buffers* sejam preenchidos para poder iniciar o envio dos segmentos de vídeo aos respectivos clientes, ocasionando assim atrasos na sua entrega. Observa-se que o servidor dá tratamento independente às requisições e, se uma das requisições tem o envio do seu fluxo atrasado pelo servidor, o envio dos fluxos de vídeo das outras requisições não sofre qualquer interferência.

Perante o resultado do teste foi decidido implementar um algoritmo de controle de tempo de leitura de disco que informa ao algoritmo de controle de admissão se este recurso está sobrecarregado.

O algoritmo de controle de disco utiliza a soma dos tempos de leitura medidos para verificar se o recurso está sobrecarregado. Se alguma medição for superior a 850 ms, o algoritmo de controle de admissão somente aceita novas requisições se pelo menos uma das requisições correntes for concluída. Assim, a cada medida acima de 850 ms o algoritmo impede a admissão de novas requisições. Quando uma requisição é concluída o algoritmo libera a admissão de novas requisições até a detecção de novos valores acima do tempo determinado.

O valor de 850 ms foi escolhido porque foi observado que em algumas medições acima deste valor a finalização do preenchimento dos *buffers* ultrapassou o intervalo de um ciclo do servidor e o intervalo total para preenchimento de todos os *buffers* excedeu a de 1 segundo. O intervalo total para preenchimento de todos os *buffers* ultrapassa o intervalo de um ciclo do servidor quando o servidor possui muitas requisições e a liberação para o preenchimento de alguns destes *buffers* ocorre quase no final do ciclo.

Além do algoritmo de controle de disco impor um limite na admissão de requisições quando o tempo de leitura atinge o valor de 850ms, o algoritmo deve considerar também os momentos

quando a soma total dos tempos esteja aproximando-se do valor limite, como por exemplo 840 ms, permitindo mesmo assim que o servidor aceite novas requisições.

Para contornar esta situação foi implementado no servidor um mecanismo de controle que não admite novas requisições quando, em pelo menos uma das últimas medições efetuadas o tempo total de leitura esteve acima de um determinado valor considerado crítico, porém inferior a 850ms.

A versão inicial do mecanismo de controle de tempo de leitura de disco contém um vetor de 270 posições para armazenar as últimas 270 medições realizadas no servidor, e foi definido o valor de 700 ms como limiar de tempo de leitura crítico para impedir a admissão de novas requisições. Estes valores foram escolhidos arbitrariamente para observação dos resultados dos testes.

Foram realizados testes para verificar se este método evita que os tempos medidos ultrapassem o valor de 1 segundo. Os testes foram realizados da seguinte forma. Inicialmente foram feitas 30 requisições ao servidor para arquivos de vídeo distintos. Após 3 minutos foram efetuadas 5 novas requisições para os arquivos. Este processo é repetido de 3 em 3 minutos até a finalização do teste, que tem a duração de uma hora. Neste teste o algoritmo de controle de disco informa ao algoritmo de controle de admissão se algum dos valores armazenados no vetor é maior ou igual a 700ms. Se existir pelo menos um valor igual ou acima de 700ms o servidor não admite novas requisições.

Este método evita que novas requisições sobrecarreguem o servidor quando os tempos medidos se aproximam de valores críticos, e ao mesmo tempo permite que sejam aceitas novas requisições logo que os novos valores medidos sejam inferiores a 700ms, sem a necessidade de esperar que pelo menos 1 requisição seja concluída, aumentando assim a utilização do servidor.

A figura 4.13 mostra o resultado do teste. Observa-se nesta figura que o valor médio das medições fica abaixo de 400ms, mas ocorreram intervalos em que o tempo total de leitura do disco ultrapassou o valor de 1 segundo.

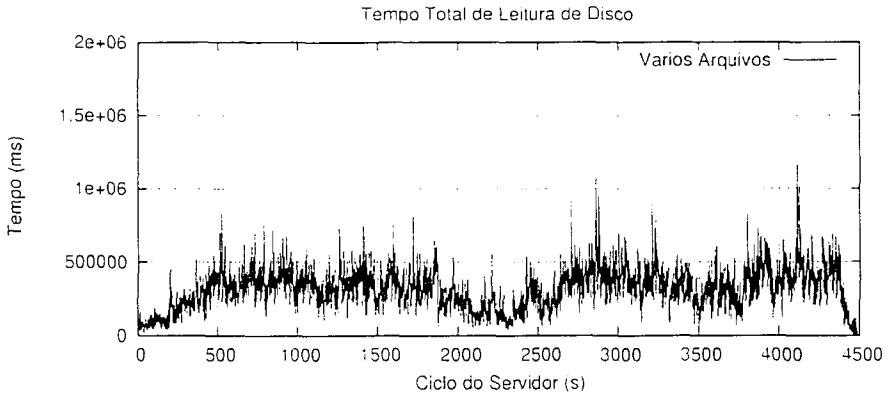


Figura 4.13: Tempo total de leitura dos dados do disco em cada ciclo do servidor (270 medidas armazenadas).

Para eliminar os picos que ultrapassaram 1 segundo foram realizados mais 2 testes com tamanho do vetor de 500 e de 1000 posições. Aumentando o tamanho do vetor, o servidor armazena por mais tempo as medições que ficaram no intervalo entre 700 e 850ms, limitando o número de requisições aceitas.

As figuras 4.14 e 4.15 mostram os resultados dos testes para vetores com 500 e 1000 posições, respectivamente. O gráfico da figura 4.14 mostra que foram obtidos valores acima de 1 segundo quando o vetor armazena as 500 últimas medições, mesmo que o valor médio tenha diminuido, comparando-se com o primeiro teste. O gráfico da figura 4.15 mostra que a utilização de um vetor de 1000 posições pode impedir a admissão de novas requisições por um período acima de 16 minutos. Isto pode ser observado entre os ciclos 1900 e 3000, quando foi obtido um único valor acima de 800ms. Observa-se que após este ciclo a utilização do servidor ficou muito pequena e que nenhuma nova requisição foi admitida. Mesmo assim, foram obtidas medições acima de 1 segundo, nos ciclos de número 3.666, 3.739, 3.741 e 3.742.

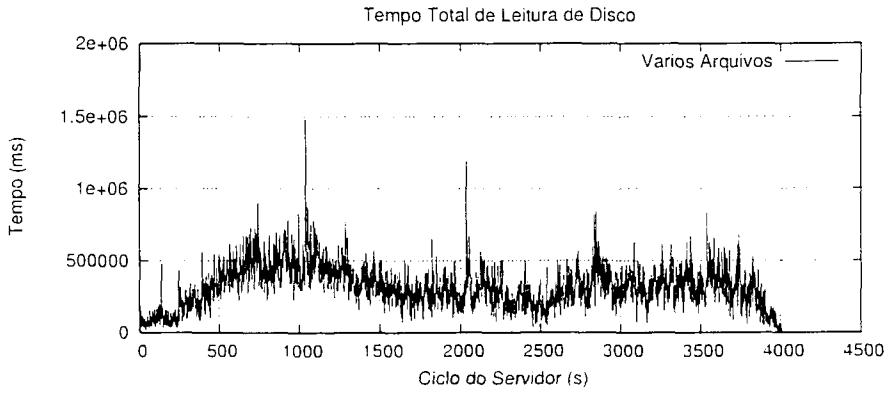


Figura 4.14: Tempo total de leitura dos dados do disco em cada ciclo do servidor (500 medidas armazenadas).

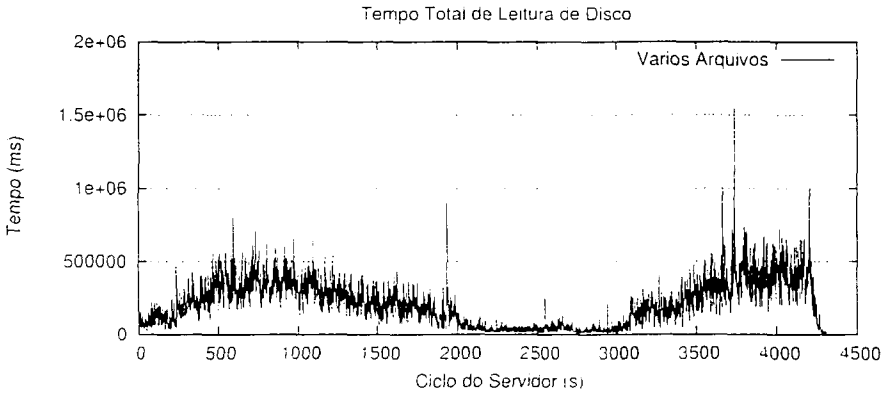


Figura 4.15: Tempo total de leitura dos dados do disco em cada ciclo do servidor (1000 medidas armazenadas).

A partir destes 3 testes foi concluído que o tamanho do vetor que contém o histórico das medições do servidor não é suficiente para evitar a ocorrência de picos acima de 1 segundo, mesmo que a utilização do servidor seja prejudicada. Para tentar resolver este problema foram realizados mais dois testes com o vetor de 270 posições nos quais os intervalos de tempo que impedem a admissão de novas requisições são de 600 a 850ms, e 500 a 850ms.

A figura 4.16 mostra o resultado do teste com o intervalo de 600 a 850ms. No gráfico observa-se que ocorreram algumas medições acima de 1 segundo. A figura 4.17 mostra o resultado do teste com intervalo de 500 a 850ms, observa-se também foram obtidas medições acima de 1 segundo.

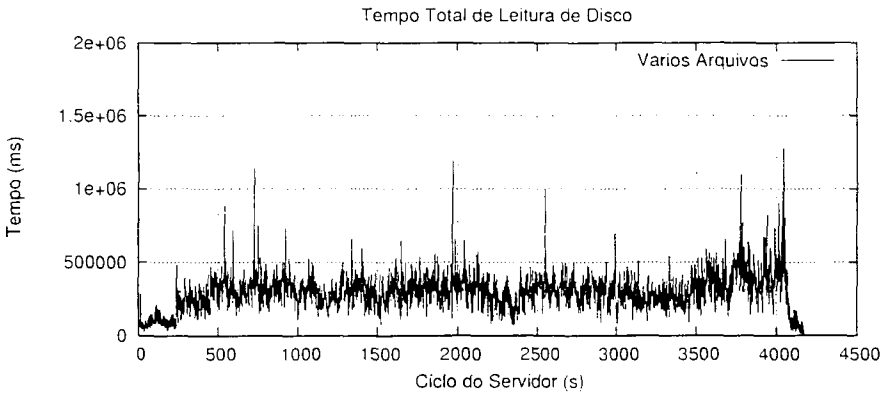


Figura 4.16: Tempo total de leitura dos dados do disco em cada ciclo do servidor (270 medidas armazenadas).

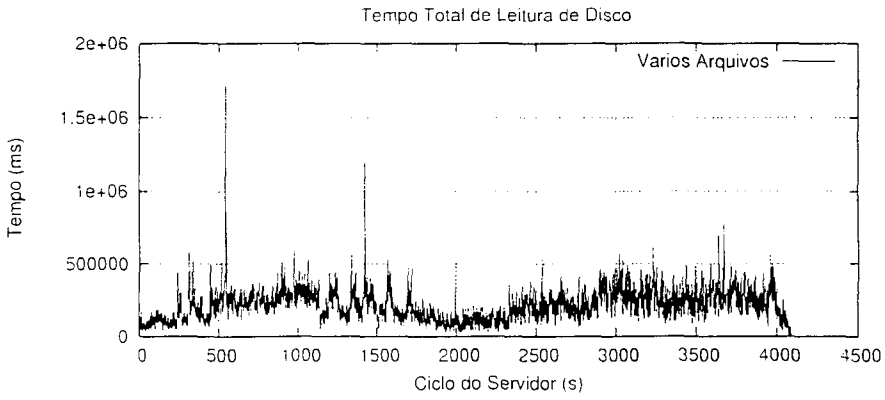


Figura 4.17: Tempo total de leitura dos dados do disco em cada ciclo do servidor (270 medidas armazenadas).

Estes dois últimos testes mostram que a expansão do intervalo que limita a admissão de novas requisições não é suficiente para eliminar as medições acima de 1 segundo. Estes intervalos acontecem devido à disposição aleatória dos arquivos de vídeo no disco, que pode ocasionar tempos de busca muito longos devido à movimentação do braço do disco.

Conclui-se então que o método de controle de tempo de leitura de disco proposto não é robusto e nem confiável, pois leva à sub-utilização do servidor e não evita que ocorram instantes de sobrecarga. A partir destes resultados conclui-se que é necessário o emprego de uma forma de armazenamento específico para os arquivos de vídeo diferente do método aleatório, para que o tempo de acesso ao disco seja menor, evitando que ocorram sobrecargas e seja possível obter um melhor desempenho.

Os gráficos da figura 4.18 e 4.19 mostram o número de requisições admitidas e a banda de rede estimada, respectivamente. Observa-se no gráfico 4.19 que o algoritmo de controle de tempo de disco permite aceitar um número de requisições que utiliza o dobro da banda de rede disponível.

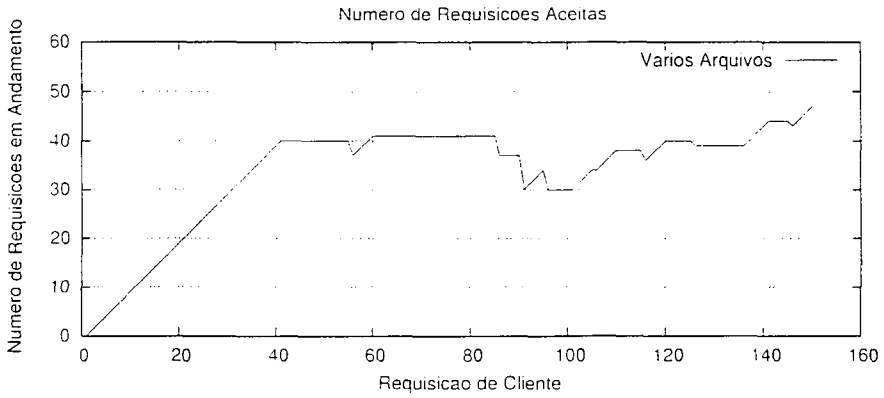


Figura 4.18: Quantidade de requisições em andamento quando chega nova requisição ao servidor.

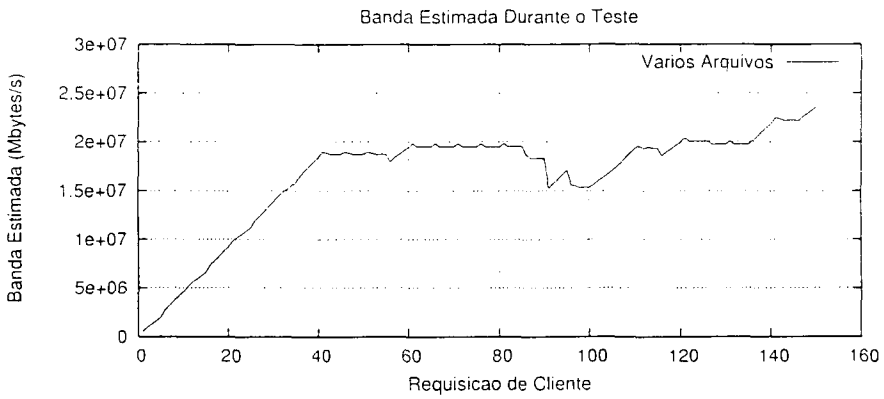


Figura 4.19: Banda estimada quando chega uma nova requisição ao servidor.

Com base nos dados obtidos no gráfico da figura 4.19 conclui-se que o servidor *SVFserver* pode ser usado em uma rede de 100Mbps mesmo não tendo um mecanismo válido para controlar o recurso do disco. A banda de rede é menor que a banda da interface de disco e permite que umas poucas requisições sejam aceitas sem sobrecarregar o subsistema de disco. Note-se que quando o servidor for instalado em uma rede com banda maior, esta situação poderá se inverter e o sistema de discos é que limitará o número de requisições que podem ser atendidas.

Capítulo 5

Conclusões e Trabalhos Futuros

Este trabalho descreve a implementação do servidor de vídeo *SVFserver*, servidor que disponibiliza fluxos de vídeo a partir de arquivos armazenados em disco, com codificação do tipo VBR e emprega filtros no cálculo da estimativa de banda de rede controlada pelo servidor. O servidor *SVFserver* contém algoritmos que controlam a utilização dos recursos da rede e de disco, e um algoritmo de controle de admissão.

No servidor *SVFserver* foram implementados algoritmos de envio de fluxos de vídeo sob a suposição de que o programa cliente não aloca espaço para os quadros na memória principal. A partir da análise do desempenho destes algoritmos conclui-se que o método que proporciona melhores resultados é o do algoritmo *Rate-Constrained Bandwidth Smoothing*.

O algoritmo de controle de banda de rede utiliza uma forma determinística para calcular a utilização da banda para cada arquivo de vídeo disponibilizado pelo servidor, sendo este algoritmo baseado no trabalho de E. Knightly e H. Zhang [KZ97]. Foi comprovado através de testes que este método é valido mas super-estima a banda de rede de todos os arquivos de vídeo testados, o que implica em baixa utilização da rede.

Para melhorar o aproveitamente do servidor, e permitir um maior número de requisições aceitas, é proposto um novo método que utiliza filtros para estimar a banda de rede dos arquivos de vídeo. Este método é eficaz porque estima o valor da banda de rede de acordo com a quantidade de dados enviados durante um intervalo de vários segundos. Este filtro foi proposto para melhorar a estimativa de banda consumida por uma requisição, aumentando assim a capacidade de transmissão do servidor.

Em uma rede local Ethernet de 100Mbps, o *SVFserver* admitiu 10 requisições feitas a um único arquivo de vídeo, utilizando 60% da banda de rede. Com a utilização do filtro, o servidor

admitiu 16 requisições e utilizou 95% da banda de rede. Isto significa um acréscimo de 60% no número de requisições aceitas para o arquivo de vídeo, e um aumento de 50% na utilização da banda de rede.

O algoritmo de controle de tempo de disco proposto neste trabalho utiliza uma forma de controle determinístico baseado na medição direta do tempo de leitura durante a execução do servidor. Este método foi escolhido devido à forma aleatória em que são armazenados os arquivos de vídeo. Através dos resultados obtidos nos testes com o algoritmo de controle de tempo de disco chegou-se à conclusão que existe a necessidade de se empregar um laioute específico para o armazenamento físico de arquivos de vídeo, pois a localização aleatória pode provocar intervalos muito longos para a movimentação do braço do disco, introduzindo assim atrasos na entrega dos fluxos de vídeo aos clientes.

O servidor *SVFserver* foi implementado para ser executado em uma rede do tipo *LAN Ethernet* de 100Mbps. O algoritmo de controle de banda de rede limita o número de requisições aceitas pelo servidor para servir ao maior número possível de requisições sem congestionar a rede, permitindo que o servidor seja executado sem um controle no recurso do disco. Se o servidor for instalado em uma rede de maior capacidade deverá ser empregada uma forma de armazenamento específico para os arquivos de vídeo e deverá ser implementado um algoritmo de controle de tempo de disco que administre o recurso de disco.

O algoritmo de controle de admissão implementado no servidor analisa o estado dos recursos controlados pelo algoritmo de controle de banda de rede e pelo algoritmo de controle de acesso a disco, e determina se novas requisições podem ser aceitas ou não. Este algoritmo evita que ocorram sobrecargas nos recursos do servidor através da limitação do número de requisições aceitas.

Vários trabalhos podem ser desenvolvidos a partir da implementação descrita aqui. Além de implementar um método robusto para controlar o recurso de disco, pode-se implementar funções que decodificam os quadros de vídeo viabilizando a execução de funções do tipo *VCR*. Outro trabalho que pode ser desenvolvido consiste em reprojetar a parte referente à transmissão de fluxos de vídeo ao vivo herdado do servidor *FFserver* e que tem baixo desempenho.

Bibliografia

[AMG98] J. Al-Marri and S. Ghandejarizadeh. An evaluation of alternative disk scheduling techniques in suport of variable bit rate continuous media. *Lecture Notes in Computer Science*, 1377:231–236, 1998.

[axi03] Microstorm.com, 2003. `http://www.microstorm.com`.

[axo03] Axonix Corporation, 2003. `http://www.axonix.com`.

[bel03] F. Bellard. 2003. `http://ffmpeg.sourceforge.org`.

[BOS95] R. Rastogi B. Ozden and A. Siberschatz. Demand paging for video-on-demand servers. *International Conference on Multimedia Computing and Systems*, pages 264 – 272, May 1995.

[BOS96] R. Rastogi B. Ozden and A. Siberschatz. Buffer replacement algorithms for multimedia storage systems. *International Conference on Multimedia Computing and Systems*, pages 172 180, 1996.

[CGM99] E. Chang and H. Garcia-Molina. Accounting for memory use, cost, throughput and latency in the design of a media server. Technical report, Stanford University Technical Report SIDL-WP-1998-0096, July 1999.

[cli03] Destiny Media Technologies, inc., 2003. `http://www.clipstream.com`.

[CT99] S. H. G. Chan and F. A. Tobagi. Providing distributed on-demand video services using multicasting and local caching. *Proceedings of IEEE Multimedia Applications, Services, and Technologies.*. June 1999.

[CZ94] E. Chang and A. Zakhor. Variable bit rate MPEG video storage on parallel disk arrays. *Proceedings of the 1st International Workshop on Community Networking Integrated Multimedia Services to the Home*, pages 127 – 137, july 1994.

[CZ96] E. Chang and A. Zakhor. Cost analyses for VBR video servers. *IEEE Multimedia (Winter 1996)*, pages 56 – 71, January 1996.

[DAG92] Y. Osawa D. Anderson and R. Govindan. A file systems for continuous media. *ACM Transactions on Computer Systems*, pages 311 – 377, November 1992.

[DEZ01] M. Vernon D. Eager and J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. *IEEE Transactions on Knowledge and Data Engineering, vol 13, number 5*, pages 742 – 757, October 2001.

[DMH97] G. Neufeld D. Makaroff and N. Hutchinson. An evaluation of VBR disk admission algorithms for continuos media file server. *Proceedings of the 5th ACM Multimedia Conference*, pages 143 – 154, may 1997.

[DMH99] G. Neufeld D. Makaroff and N. Hutchi. Network bandwidth allocation and admission control for a continuous media file server. *Interactive Distributed Multimedia Systems and Telecommunication Services*, pages 337 – 350, 1999.

[dv103] Dan Dennedy and Charles Yates. 2003. `http://dv1394d.sourceforge.net`.

[DWL96] H. Zhang D. Wrege, E. Knightly and J. Liebeherr. Deterministic delay bounds for VBR video in packet-switching networks: Fundamental limits and practical tradeoffs. *IEEE/ACM Transactions on Networking*, pages 352–362. 1996.

[EKZ95] J. Liebeherr E. Knightly, D. Wrege and H. Zhang. Fundamental limits and tradeoffs of providing deterministic guarantees to VBR video traffic. *Proceedings of ACM SIGMETRICS /PERFORMANCE '95 Conference*, pages 98 – 107, may 1995.

[Fen97] W. Feng. Rate-constrained bandwidth smoothing for the delivery of stored video. *Proceedings of the IS&T/SPIE Multimedia Networking and Computing*, pages 316 – 327, February 1997.

[FS95a] W. Feng and S. Sechrest. Critical bandwidth allocation for delivery of compressed video. *Computer Communications, vol. 18*, pages 709 – 717, October 1995.

[FS95b] W. Feng and S. Sechrest. Smoothing and buffering for delivery of prerecorded compressed video. *Proceedings of the IS&T/SPIE Symposium on Multimedia Computing and Networing*, pages 234 – 244, February 1995.

[FV90] D. Ferrari and D. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Jornal on Selected Areas in Communications, vol. 8*, pages 368 – 379, April 1990.

[Gal91] D. Gall. MPEG: A video compression standart for multimedia applications. *Communications of ACM*, 34(4):47 – 58, April 1991.

[Gro89] B. Grob. *Televisão e Sistema de Vídeo*. Editora Guanabara S.A., 1989.

[HVG94a] A. Goyal H. Vin and P. Goyal. An observation-based admission control algorithm for multimedia servers. *Proceedings of the IEEE International Conference on Multimedia Computing Systems (ICMCS'94)*, pages 234 – 243, may 1994.

[HVG94b] A. Goyal H. Vin and P. Goyal. A statistical admission control algorithm for multimedia servers. *In Proceedings of the ACM Multimedia*, pages 33 – 40, october 1994.

[KLY99a] J. B. Kwon K. Lee and H. Y. Yeom. Caching and concurrency control in mobile client/server database systems. *CODAS*, pages 228–239, 1999.

[KLY99b] J. B. Kwon K. Lee and H. Y. Yeom. Exploiting caching for realtime multimedia systems. *IEEE International Conference on Multimedia Computing and Systems (ICMCS'99)*, 1(1), June 1999.

[KY99] S. Kang and Heon Y. Yeom. Transmission of video streams with constant bandwidth allocation. *Computer Communications*, 22:173–180, 1999.

[KY00] S. Kang and H. Y. Yeom. Statistical admission control for soft real-time VOD servers. *Proceedings of the ACM Symposium on Applied Computing (SAC 2000)*, pages 579 – 584. March 2000.

[KZ95] E. Knightly and H. Zhang. Traffic characterization and switch utilization using a deterministic bounding interval dependent traffic model. *Proc. of IEEE INFOCOM*, pages 1137 – 1145, 1995.

[KZ97] E. Knightly and H. Zhang. D-bind: An accurate traffic model for providing QoS guarantees to VBR traffic. *IEEE/ACM Transactions on Network*, 5:219 – 231, April 1997.

[MGT97] S. Keshav M. Grossglauser and D. Tse. RCBR: A simple and efficient service for multiple time-scale traffic. *IEEE ACM Transactions on Networking*, 5(6):741 – 755. 1997.

[mpl03] Mplayerhq.hu. 2003. `http://www.MPlayerHQ.hu`.

[MR95] J. McManus and K. Ross. Prerecorded VBR sources in ATM networks: Piecewise constant -rate transmission and transport. Technical report, University of Pennsylvania. September 1995.

[MR96] J. McManus and K. Ross. Video on demand over ATM: Constante rate transmission and transport. *Proceedings of IEEE INFOCOM*, pages 1357 – 1362, March 1996.

[NGDR98] K. K. Ramakrishnan N. G. Duffield and A. R. Reibman. Save: An algorithm for smoothed adaptive video over explicit rate networks. *IEEE/ACM Transactions on Network*, 6(6):717 – 728, 1998.

[Nol99] P. Noll. Digital audio for multimedia. 1999. `http://citeseer.nj.nec.com/noll99digital.html`.

[Pan95] D. Pan. A tutorial on MPEG/audio compression. *IEEE Multimedia Journal*, 2(2):60 – 74, 1995.

[PRR92] H. Vin P. Rangan and S. Ramanathan. Designing an on-demand multimedia service. *IEEE Communications Magazine*, 30(7):56 – 64, 1992.

[RBS94] D. Clark R. Branden and S. Shender. Integrated services in the internet architecture: an overview. Technical report, RFC 1633, 1994.

[RRS96] B. Ozden R. Rastogi and A. Silberschatz. On the design of a low-cost video-on-demand storage systems. *Multimedia Systems Journal*, 4(1):40 – 54, February 1996.

[RV91] P. Rangan and H. Vin. Designing file system for digital video and audio. *In Proceedings of the 13th ACM Symposium on Operating Systems Principles*, pages 81 – 94, 1991.

[RV93] P. Rangan and H. Vin. Designing a multi-user HDTV storage server. *IEEE Jornal on Selected Areas in Communications*, 11(1):153 – 164, January 1993.

[SGB98] B. Khasnabish S. Gringeri, A. Lewis and B. Basch. Traffic shaping, bandwidth allocation, and quality assessment for MPEG video distribution over broadband networks. *IEEE Network*, 12(6):94 – 107, November/December 1998.

[SHGCK98] F. A. Tobagi S. H. G. Chan and T. M. Ko. Bandwidth planning in near video-on-demand. *Proceedings of the 1998 IEEE International Sysmposium on Circuits and Systems (ISCAS'98)*, pages 61 – 64, June 1998.

[TC95] F. A. Tobagi and S. G. Chan. Hierarchical storage systems for on-demand video servers. *Cin Proceedings of the SPIE (the International Society for Optical Engineering) High-Density Data Recording and Retrival Technologies*, pages 103 – 120, October 1995.

[Ver96] M. Vernick. The design, implementation and evolution of the Stony Brook video server. Technical report, State University of New York, NY, December 1996.

[WFS97] F. Jahanian W. Feng and S. Sechrest. Optimal buffering for the delivery of compressed prerecorded video. *Multimedia Systems*, 5(5):297 – 309, January 1997.

[wms03] Microsoft Corporation, 2003. `http://www.microsoft.com`.

[XLP99] M. H. Ammar X. Li and S. Paul. Video multicast over the internet. *IEEE Network*, 13(2):46 – 60, March/April 1999.

[ZF94] H. Zhang and D. Ferrari. Improving utilization for deterministic service in multimedia communication. *IEEE International Conference on Multimedia Computing and Systems*, pages 295 – 304, may 1994.

Apêndice A

Características dos Arquivos de Vídeo Utilizados nos Testes

Este apêndice contém informações referentes às características dos arquivos de vídeo utilizados nos testes realizados no servidor. As tabelas possuem características específicas referentes à codificação dos arquivos. Também neste apêndice estão os gráficos que mostram o tamanho dos quadros e a quantidade de dados enviada a cada ciclo do servidor dos arquivos. Estes gráficos foram desenhados na mesma escala para que seja possível vizualizar e comparar o grau de compressão de cada vídeo. As correspondentes características dos arquivos de vídeo encontram-se no capítulo 4.

O Senhor dos Anéis 1 - parte 1

Características	Dados
Nome	Filme: O Senhor dos Anéis 1 - parte 1
Qps / Resolução	24, 576x240
Codificação vídeo	msmpeg4
Codificação áudio	mp3 48kHz stereo
Banda ou *Buffer*	474.028
Tempo de reprodução	4623 s

Tabela A.1: Características da primeira parte do filme "Senhor dos Anéis - parte 1".

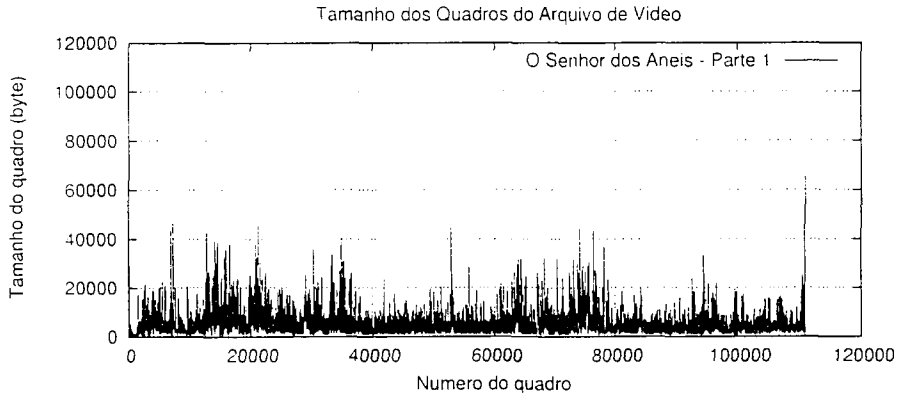


Figura A.1: Tamanho dos quadros do filme "Senhor dos Anéis - parte 1".

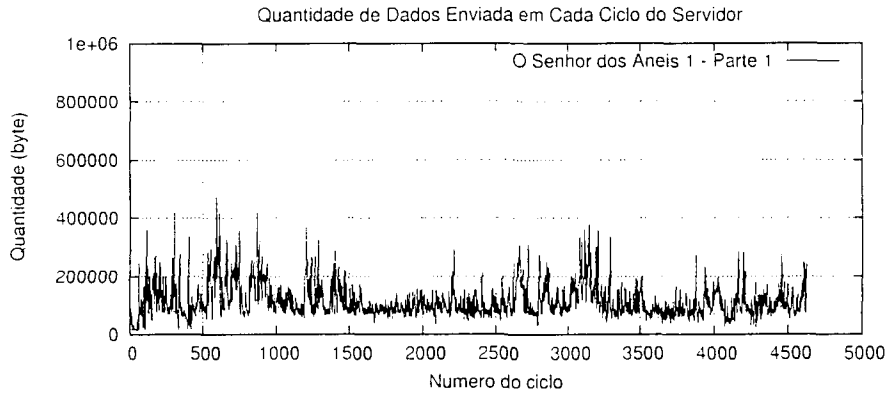


Figura A.2: Quantidade de dados enviada a cada ciclo do servidor para o filme "Senhor dos Anéis - parte 1".

O Senhor dos Anéis 1 - parte 2

Características	Dados
Nome	Filme: O Senhor dos Anéis 1 - parte 2
Qps / Resolução	24, 576x240
Codificação vídeo	msmpeg4
Codificação áudio	mp3 48kHz stereo
Banda ou *Buffer*	467.292
Tempo de reprodução	6151 s

Tabela A.2: Características da segunda parte do filme "Senhor dos Anéis - parte 2".

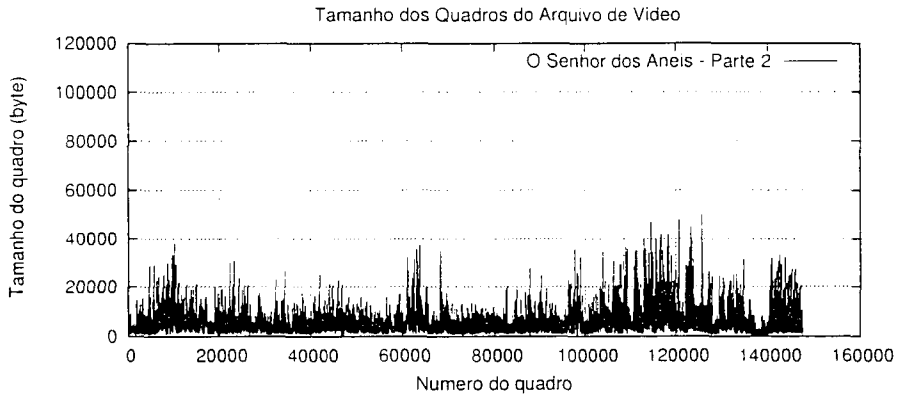


Figura A.3: Tamanho dos quadros do filme "O Senhor dos Anéis - parte 2".

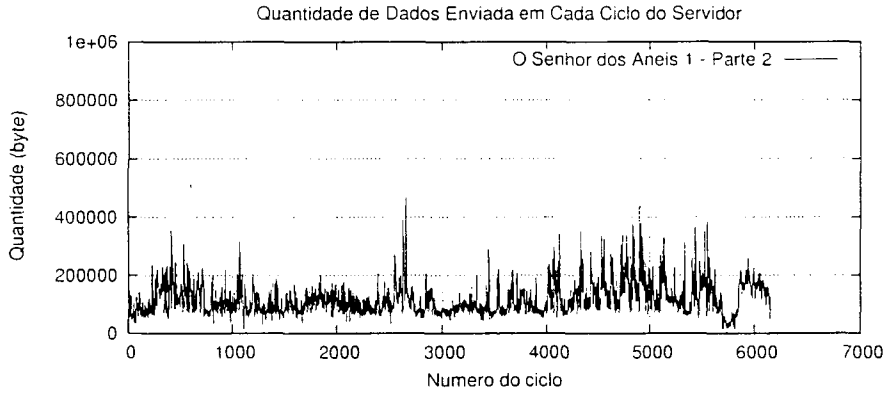


Figura A.4: Quantidade de dados enviada a cada ciclo do servidor para o filme "O Senhor dos Anéis - parte 2".

Conan - O Bárbaro

Características	Dados
Nome	Filme: Conan - O Bárbaro
Qps / Resolução	24, 512x208
Codificação vídeo	msmpeg4
Codificação áudio	mp3 44.1kHz stereo
Banda ou *Buffer*	437.396
Tempo de reprodução	7822

Tabela A.3: Características do filme "Conan O Bárbaro".

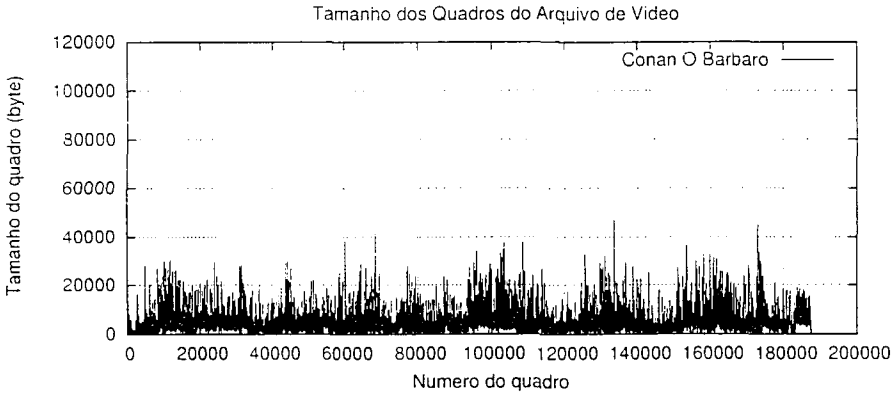


Figura A.5: Tamanho dos quadros do filme "Conan O Bárbaro".

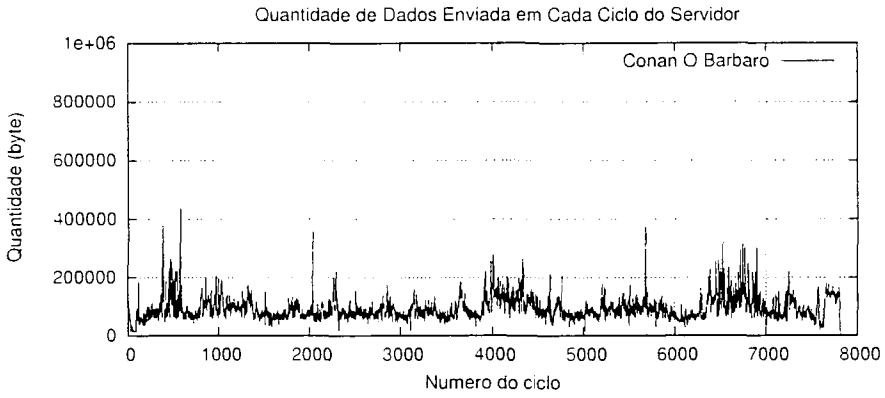


Figura A.6: Quantidade de dados enviada a cada ciclo do servidor para o filme "Conan O Bárbaro".

Shrek

Características	Dados
Nome	Desenho: Shrek
Qps / Resolução	25. 352x288
Codificação vídeo	msmpeg4
Codificação áudio	mp3 22.05kHz srereo
Banda ou *Buffer*	100.540
Tempo de reprodução	4057 s

Tabela A.4: Características do desenho "Shrek".

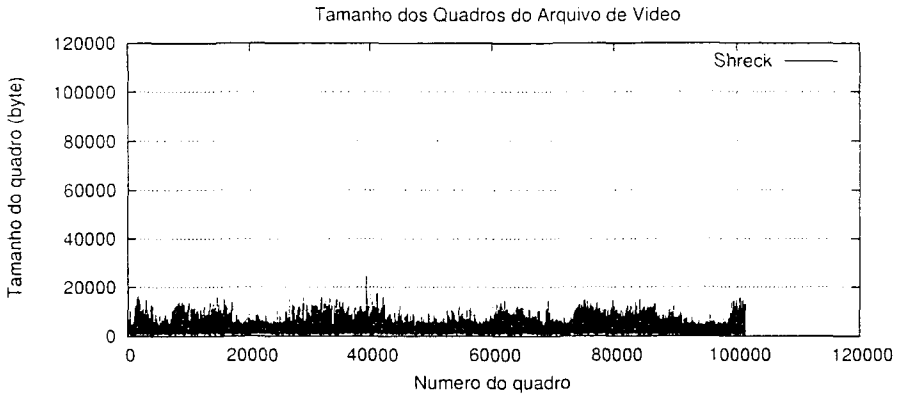


Figura A.7: Tamanho dos quadros do desenho "Shrek".

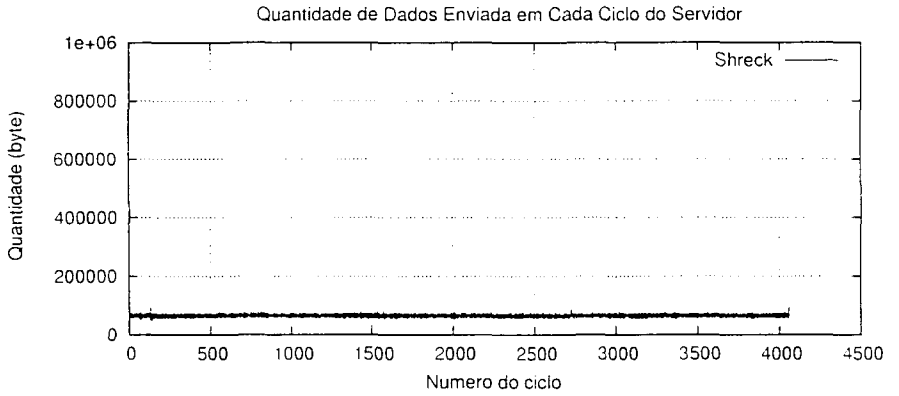


Figura A.8: Quantidade de dados enviada a cada ciclo do servidor para o desenho "Shrek".

Nightwish 1

Características	Dados
Nome	Show: Nightwish
Qps / Resolução	25, 720x400
Codificação vídeo	msmpeg4
Codificação áudio	mp3 48kHz stereo
Banda ou *Buffer*	437.070
Tempo de reprodução	4796 s

Tabela A.5: Características do show "Nightwish".

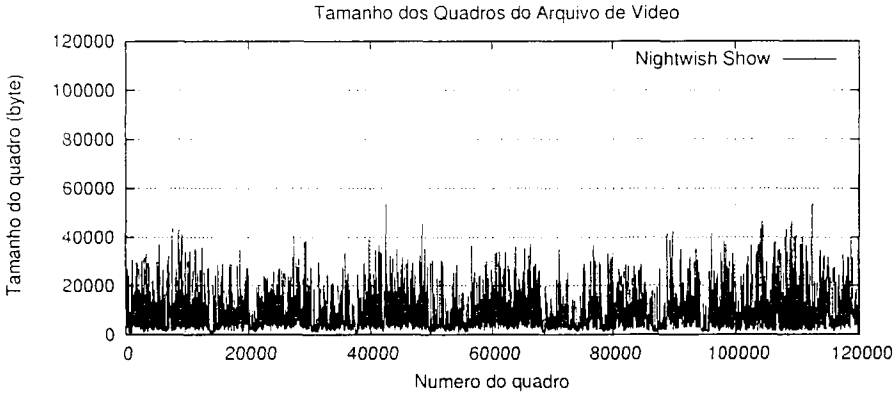


Figura A.9: Tamanho dos quadros do show "Nightwish".

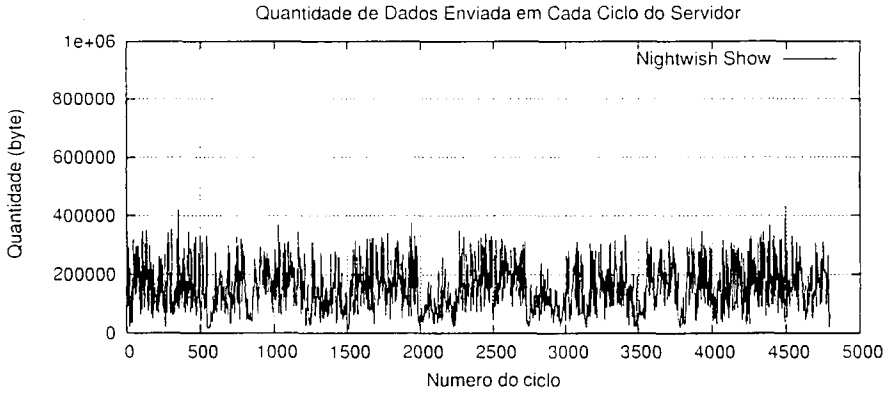


Figura A.10: Quantidade de dados enviada a cada ciclo do servidor para o show "Nightwish".

Read or Die - The Paper 1

Características	Dados
Nome	Desenho: Read or Die - The Paper 1
Qps / Resolução	24, 640x480
Codificação vídeo	mpegvideo
Codificação áudio	mp2 48kHz stereo
Banda ou *Buffer*	943.320
Tempo de reprodução	1905 s

Tabela A.6: Características do desenho "Read or Die - The Paper 1".

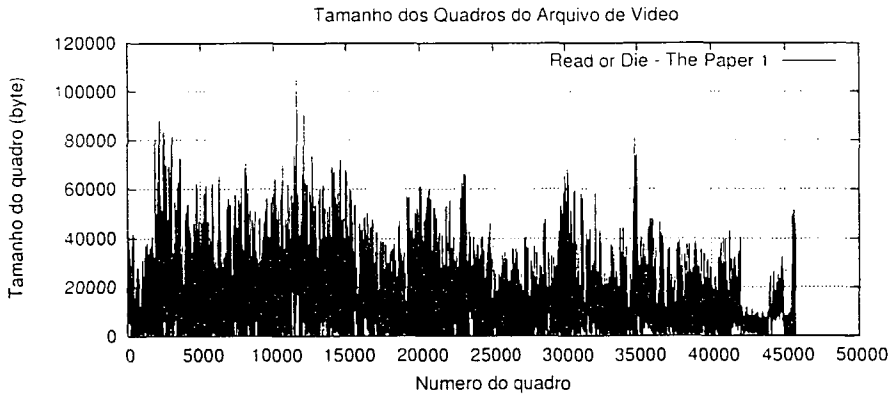


Figura A.11: Tamanho dos quadros do desenho "Read or Die - The Paper 1".

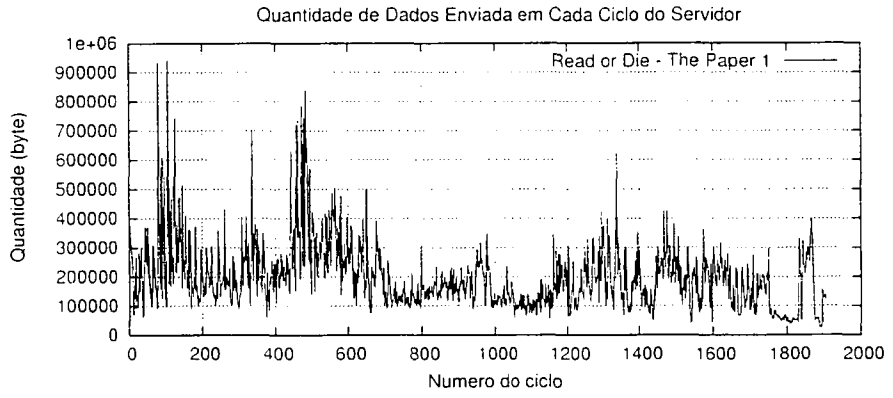


Figura A.12: Quantidade de dados enviada a cada ciclo do servidor para o desenho "Read or Die - The Paper 1".

Read os Die - The Paper 2

Características	Dados
Nome	Desenho: Read os Die - The Paper 2
Qps / Resolução	24, 640x480
Codificação vídeo	mpegvideo
Codificação áudio	mp2 48kHz stereo
Banda ou *Buffer*	722.550
Tempo de reprodução	1850 s

Tabela A.7: Caracterticas do desenho "Read or Die - The Paper 2".

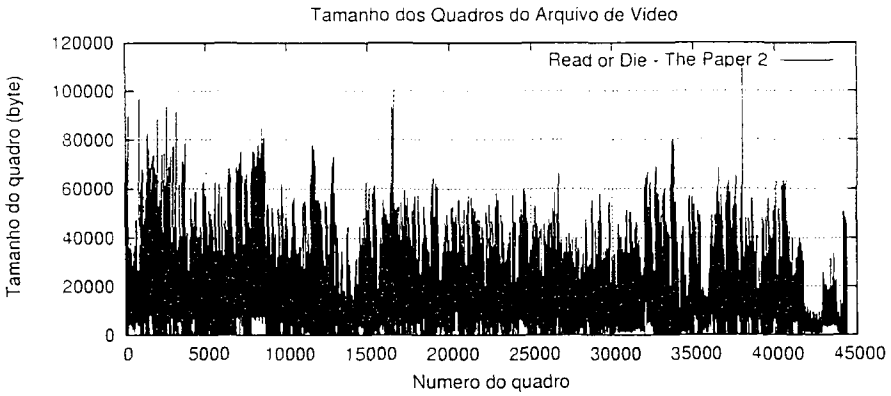


Figura A.13: Tamanho dos quadros do desenho "Read or Die - The Paper 2".

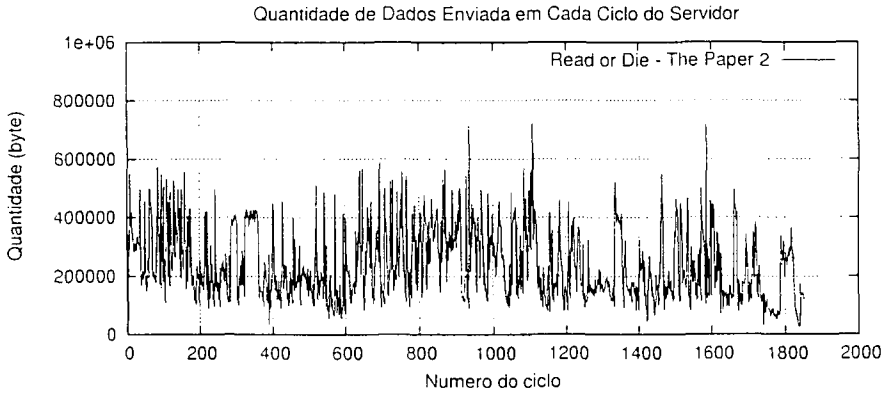


Figura A.14: Quantidade de dados enviada a cada ciclo do servidor para o desenho "Read or Die - The Paper 2".

O Senhor dos Anéis 1 (36 m. iniciais)

Características	Dados
Nome	Filme: O Senhor dos Anéis 1 (36 m. iniciais)
Qps / Resolução	24. 576x240
Codificação vídeo	mpegvideo
Codificação áudio	mp2 48kHz stereo
Banda ou *Buffer*	494.593
Tempo de reprodução	2163 s

Tabela A.8: Caracterísicas do filme "O Senhor dos Anéis - parte 1", 36 minutos iniciais.

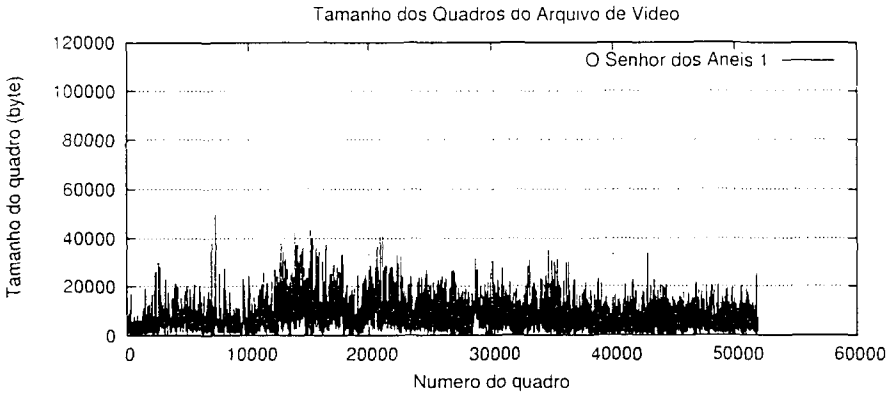


Figura A.15: Tamanho dos quadros do filme "O Senhor dos Anéis - parte 1", 36 minutos iniciais.

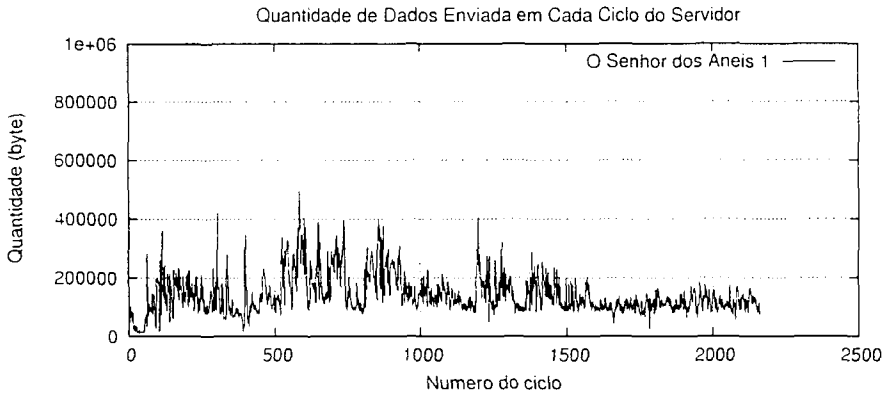


Figura A.16: Quantidade de dados enviada a cada ciclo do servidor para o filme "O Senhor dos Anéis parte 1", 36 minutos iniciais.

O Senhor dos Anéis - As Duas Torres (15 m. iniciais)

Características	Dados
Nome	Filme: O Senhor dos Anéis - As Duas Torres (15 m. iniciais)
Qps / Resolução	24, 640x272
Codificação vídeo	mpegvideo
Codificação áudio	mp2 48kHz stereo
Banda ou *Buffer*	488.816
Tempo de reprodução	877 s

Tabela A.9: Características do filme "O Senhor dos Anéis - As Duas Torres", 15 minutos iniciais.

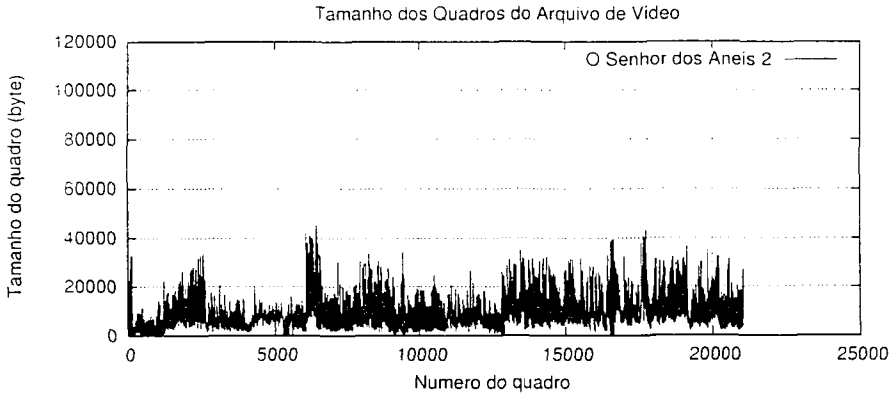


Figura A.17: Tamanho dos quadros do filme "O Senhor dos Anéis - As Duas Torres", 15 minutos iniciais.

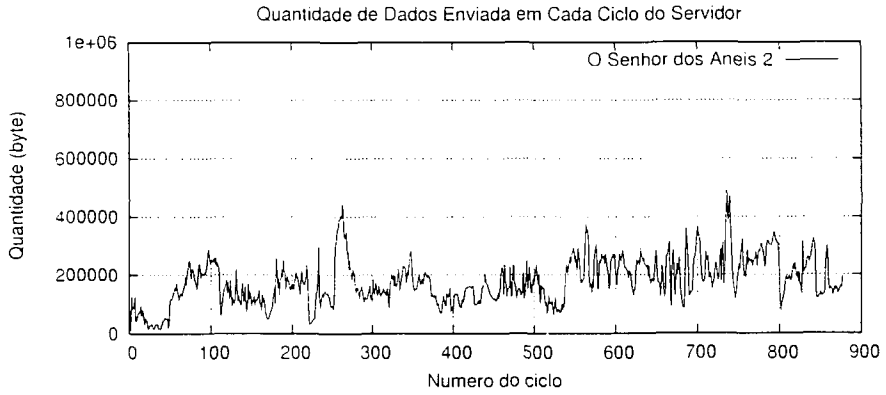


Figura A.18: Quantidade de dados enviada a cada ciclo do servidor para o filme "O Senhor dos Anéis - As Duas Torres", 15 minutos iniciais.

From The Hell

Características	Dados
Nome	Filme: From the Hell (30 minutos iniciais)
Qps / Resolução	24. 595x240
Codificação vídeo	mpegvideo
Codificação áudio	mp2 48kHz stereo
Banda ou *Buffer*	491.600
Tempo de reprodução	1755 s

Tabela A.10: Característica do filme "From the Hell".

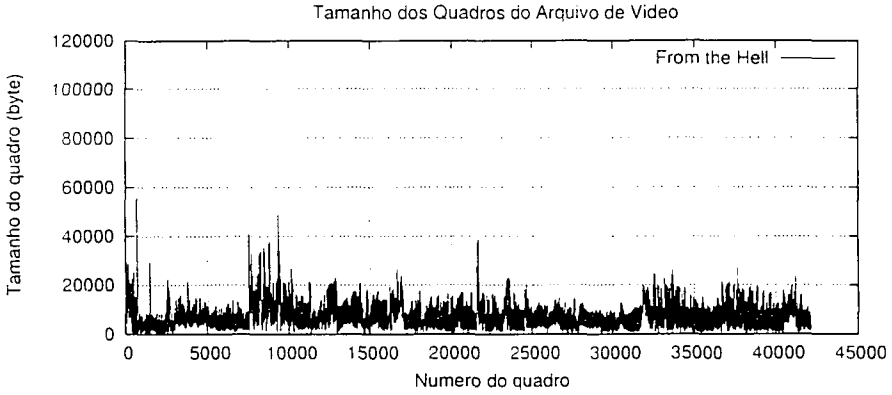


Figura A.19: Tamanho dos quadros do filme "From the Hell", 30 minutos iniciais.

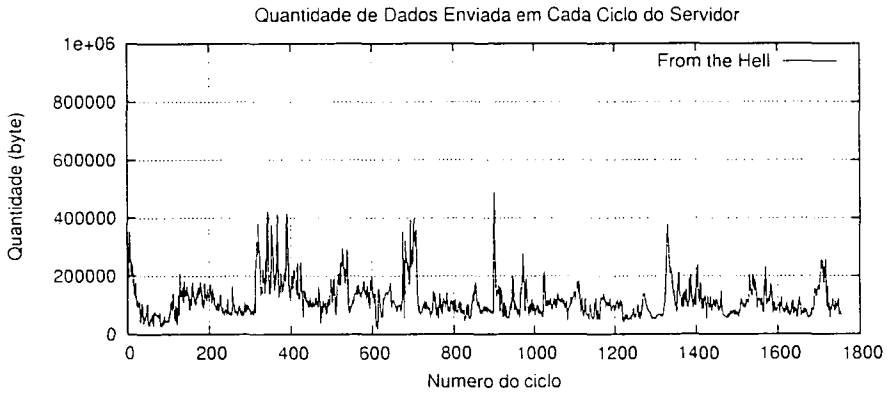


Figura A.20: Quantidade de dados enviada a cada ciclo do servidor para o filme "From the Hell", 30 minutos iniciais.

Apêndice B

Resultados Obtidos nos Testes do Algoritmo de Controle de Banda de Rede

As próximas páginas contém gráficos com as medições de quantidade de dados enviados em cada ciclo do servidor, obtidos nos testes de utilização de banda de rede discutidos no Capítulo 5. Cada figura mostra os gráficos de quatro dos cinco testes de cada conjunto de testes.

Gráficos do arquivo de log do servidor para os testes realizados com o arquivo "Read or Die - The Paper 1" considerando o valor da banda original.

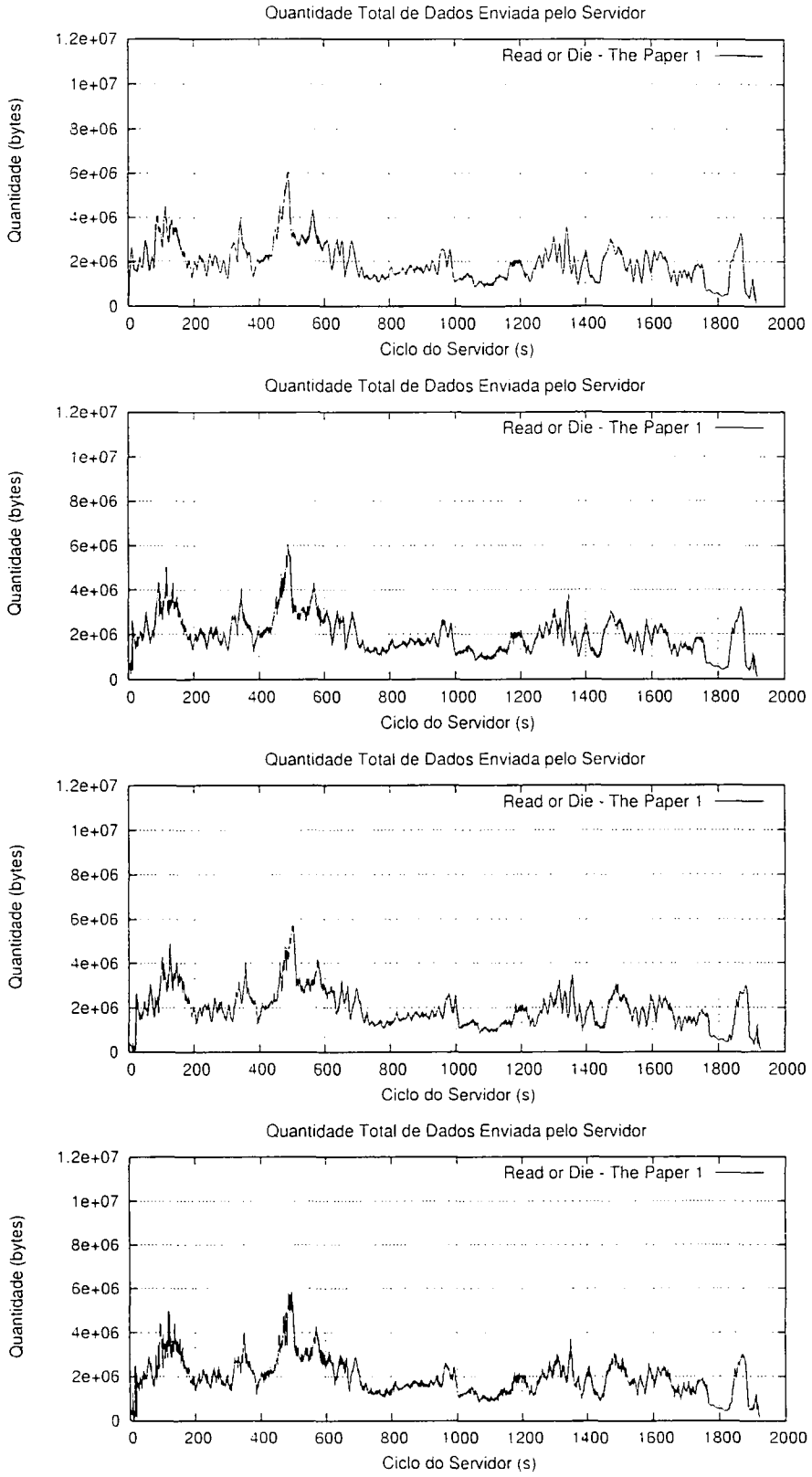


Figura B.1: Quantidade total de dados enviados em cada ciclo do servidor para requisições ao arquivo "Read or Die - The Paper 1".

Gráficos do arquivo de log do servidor para os testes realizados com o arquivo "Senhor dos Anéis - As Duas Torres", primeiros 15 minutos, considerando o valor da banda original.

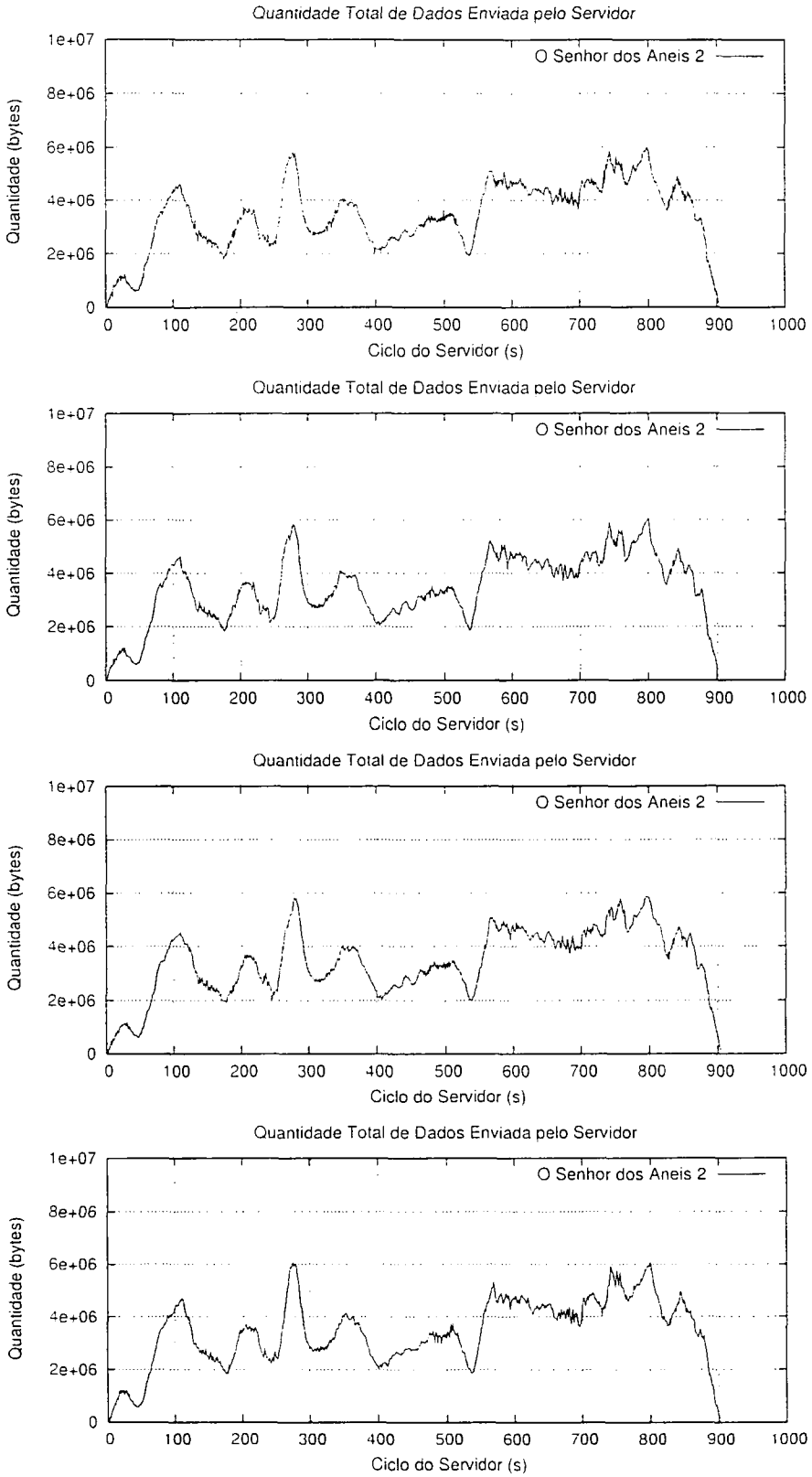


Figura B.2: Quantidade total de dados enviados em cada ciclo do servidor para requisições ao arquivo "Senhor dos Anéis - As Duas Torres", primeiros 15 minutos.

Gráficos do arquivo de log do servidor para os testes realizados com 4 arquivos distintos, considerando o valor da banda original.

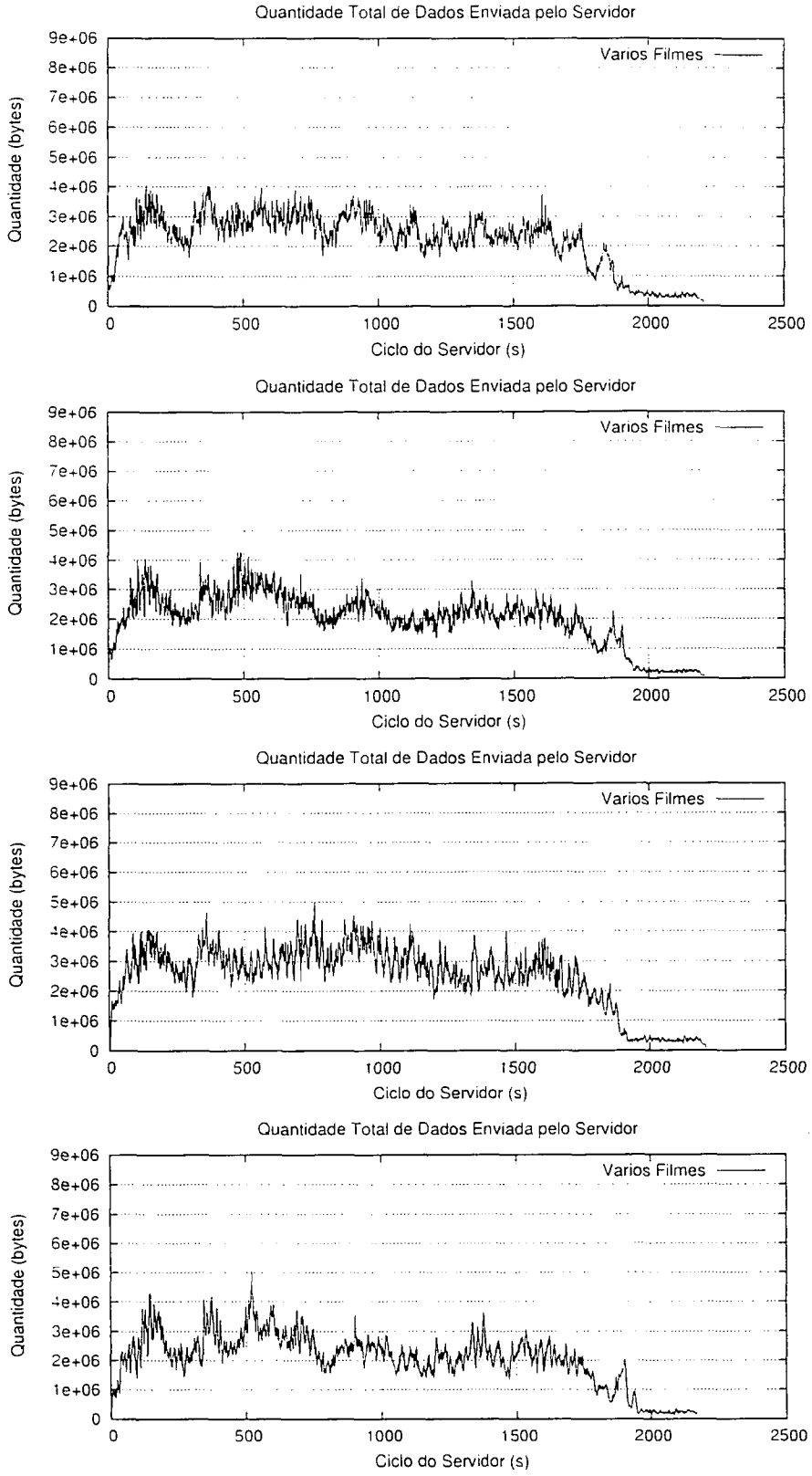


Figura B.3: Quantidade total de dados enviados em cada ciclo do servidor para requisições a arquivos diversos.

Gráficos do arquivo de log do servidor para os testes realizados com o arquivo "Read or Die - The Paper 1" considerando o valor da banda calculada segundo a equação 4.1.

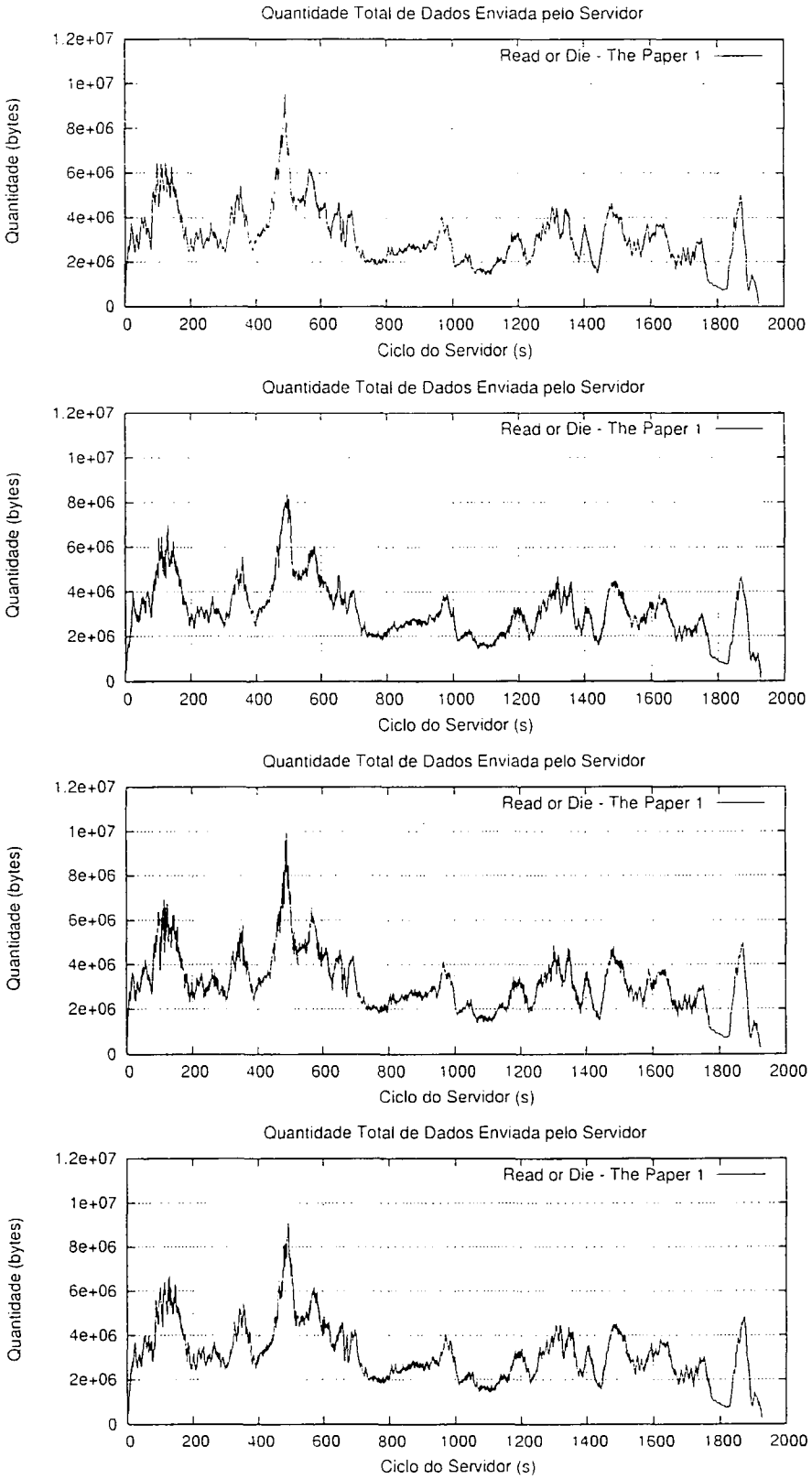


Figura B.4: Quantidade total de dados enviados em cada ciclo do servidor para requisições ao arquivo "Read or Die - The Paper 1", utilizando o filtro 4.1.

Gráficos do arquivo de log para testes realizados com o arquivo "Senhor dos Anéis - As Duas Torres". 15 minutos iniciais, considerando o valor da banda calculado pela a equação 4.1.

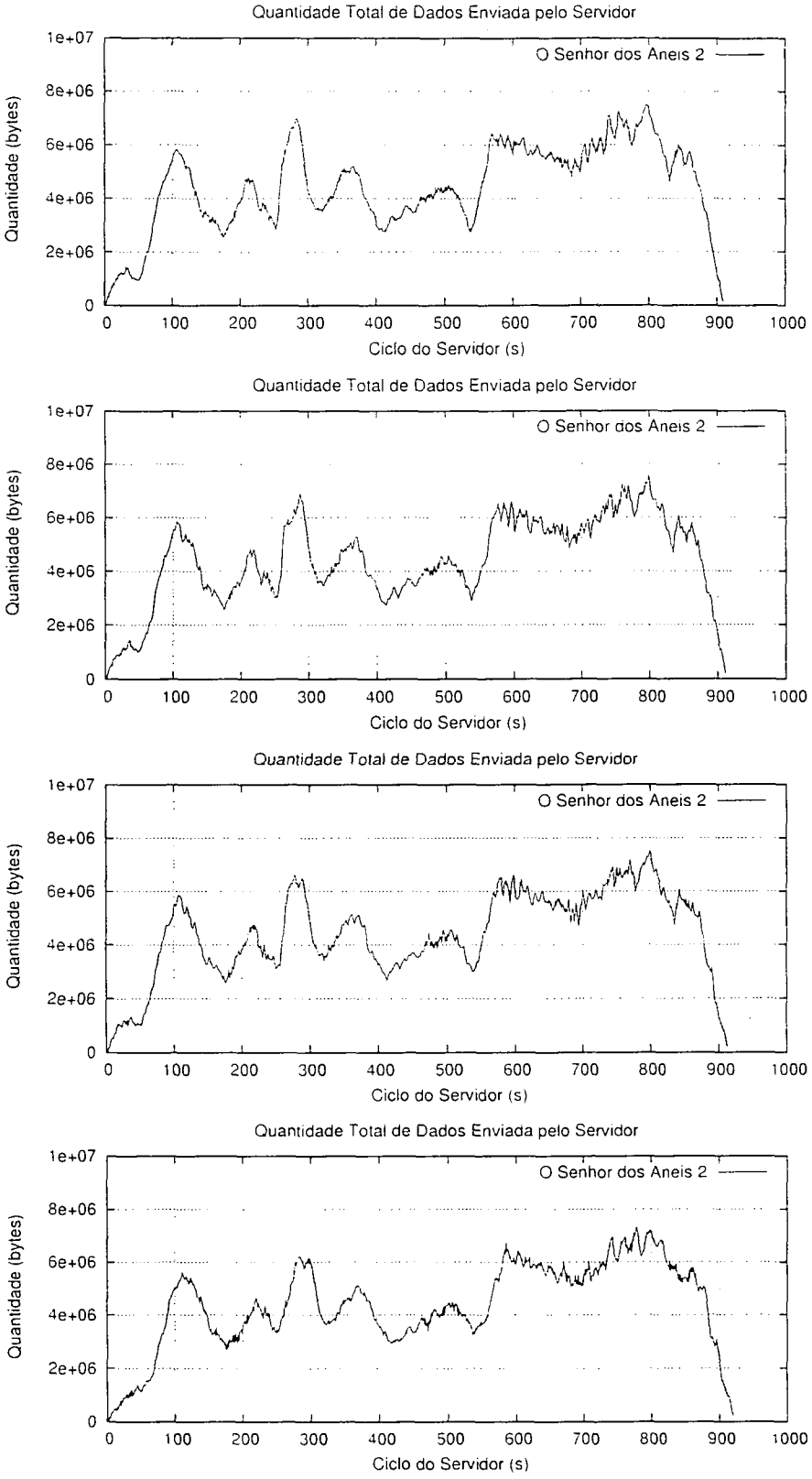


Figura B.5: Quantidade total de dados enviados em cada ciclo do servidor para requisições ao arquivo "Senhor dos Anéis - As Duas Torres", utilizando o filtro 4.1.

Gráficos do arquivo de log do servidor para os testes realizados com 4 arquivos distintos, considerando o valor da banda calculada com a equação 4.1.

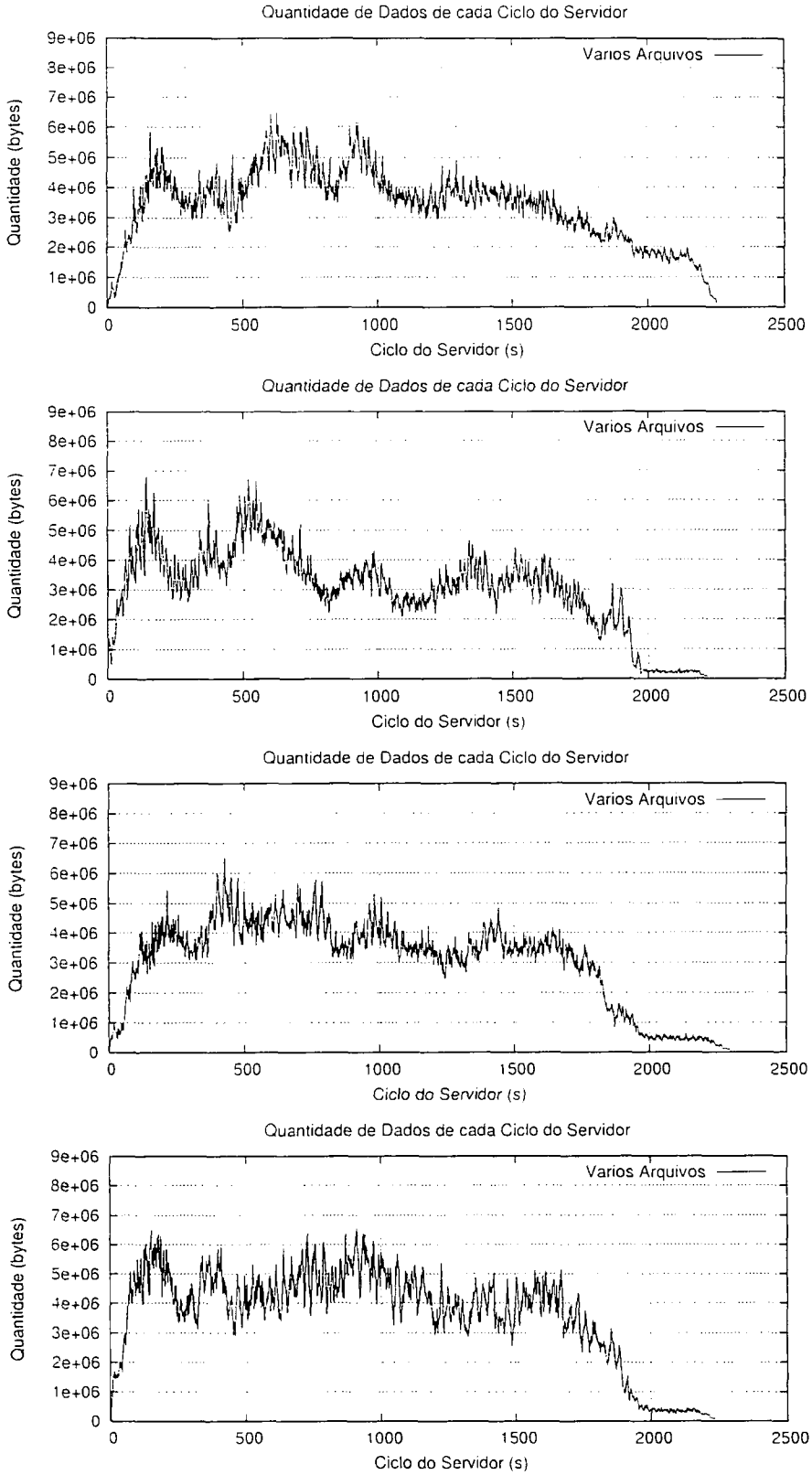


Figura B.6: Quantidade total de dados enviados em cada ciclo do servidor para requisições a diversos arquivos, utilizando o filtro 4.1.

Gráficos do arquivo de log do servidor para os testes realizados com o arquivo "Read or Die - The Paper 1" considerando o valor da banda calculada segundo a equação 4.2.

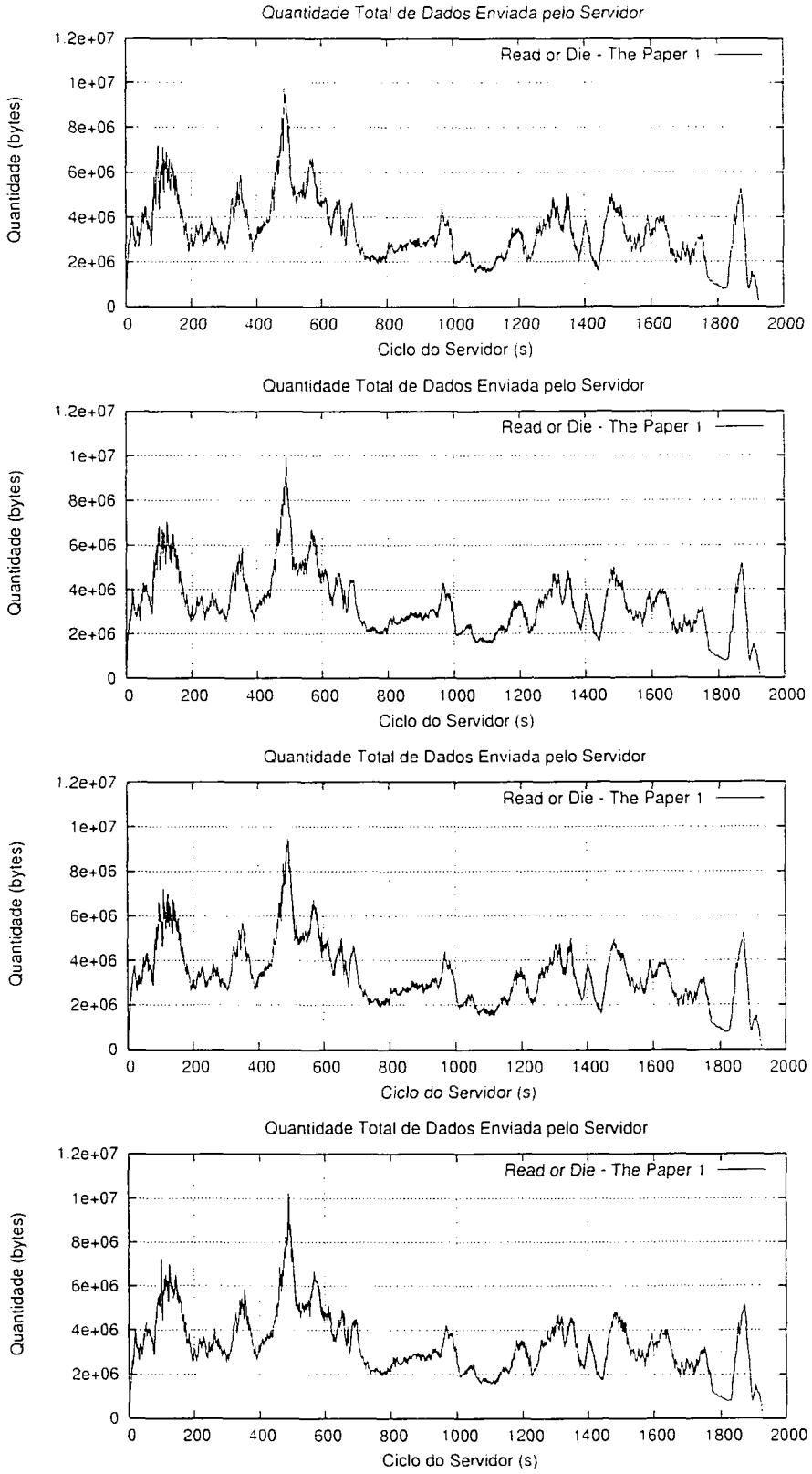


Figura B.7: Quantidade total de dados enviados em cada ciclo do servidor para requisições ao arquivo "Read or Die - The Paper 1", utilizando o filtro 4.2.

Gráficos do arquivo de log para testes realizados com o arquivo "Senhor dos Anéis - As Duas Torres". 15 minutos iniciais, considerando o valor da banda calculado pela a equação 4.2.

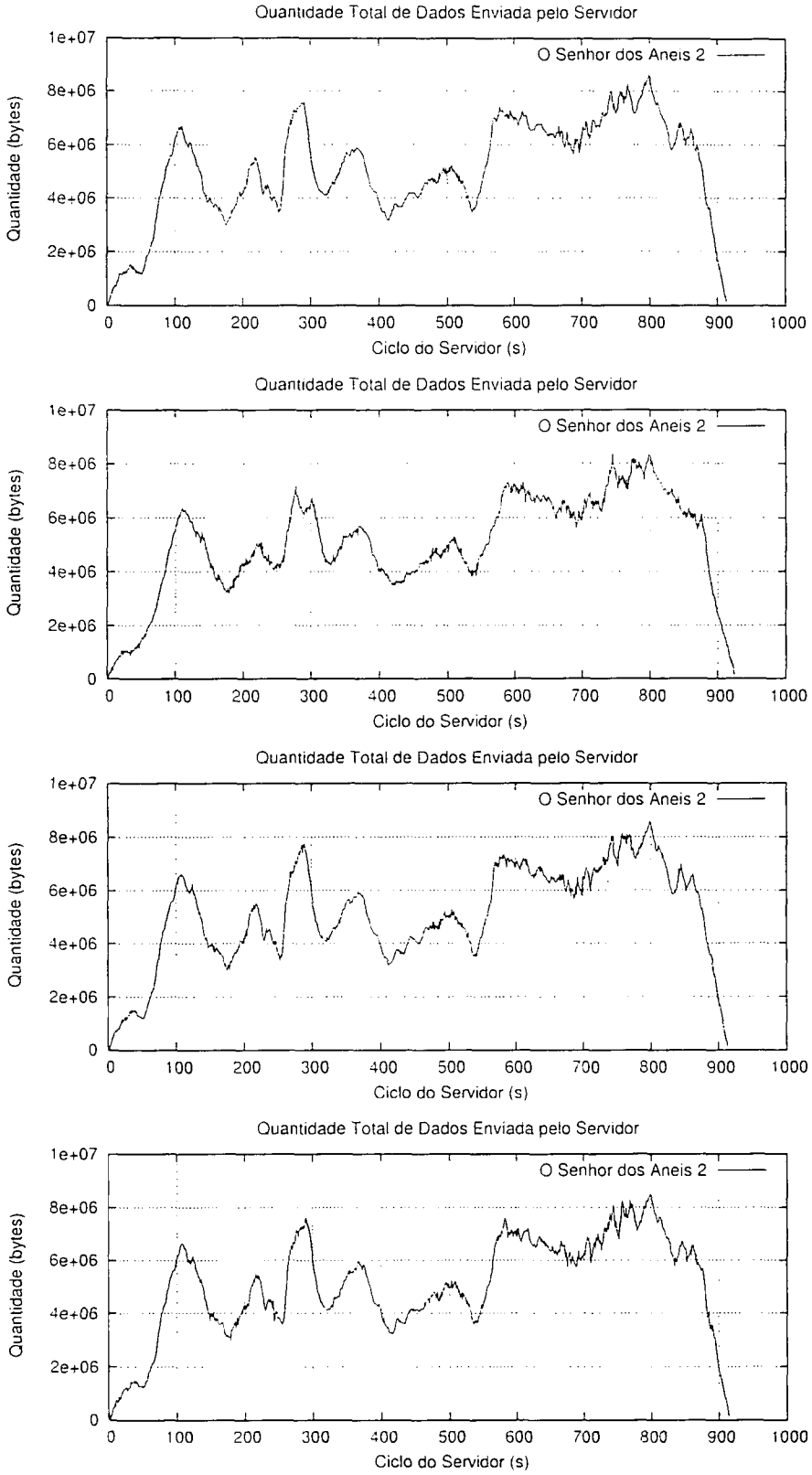


Figura B.8: Quantidade total de dados enviados em cada ciclo do servidor para requisições ao arquivo "Senhor dos Anéis - As Duas Torres", utilizando o filtro 4.2.

Gráficos do arquivo de log do servidor para os testes realizados com 4 arquivos distintos, considerando o valor da banda calculada com a equação 4.2.

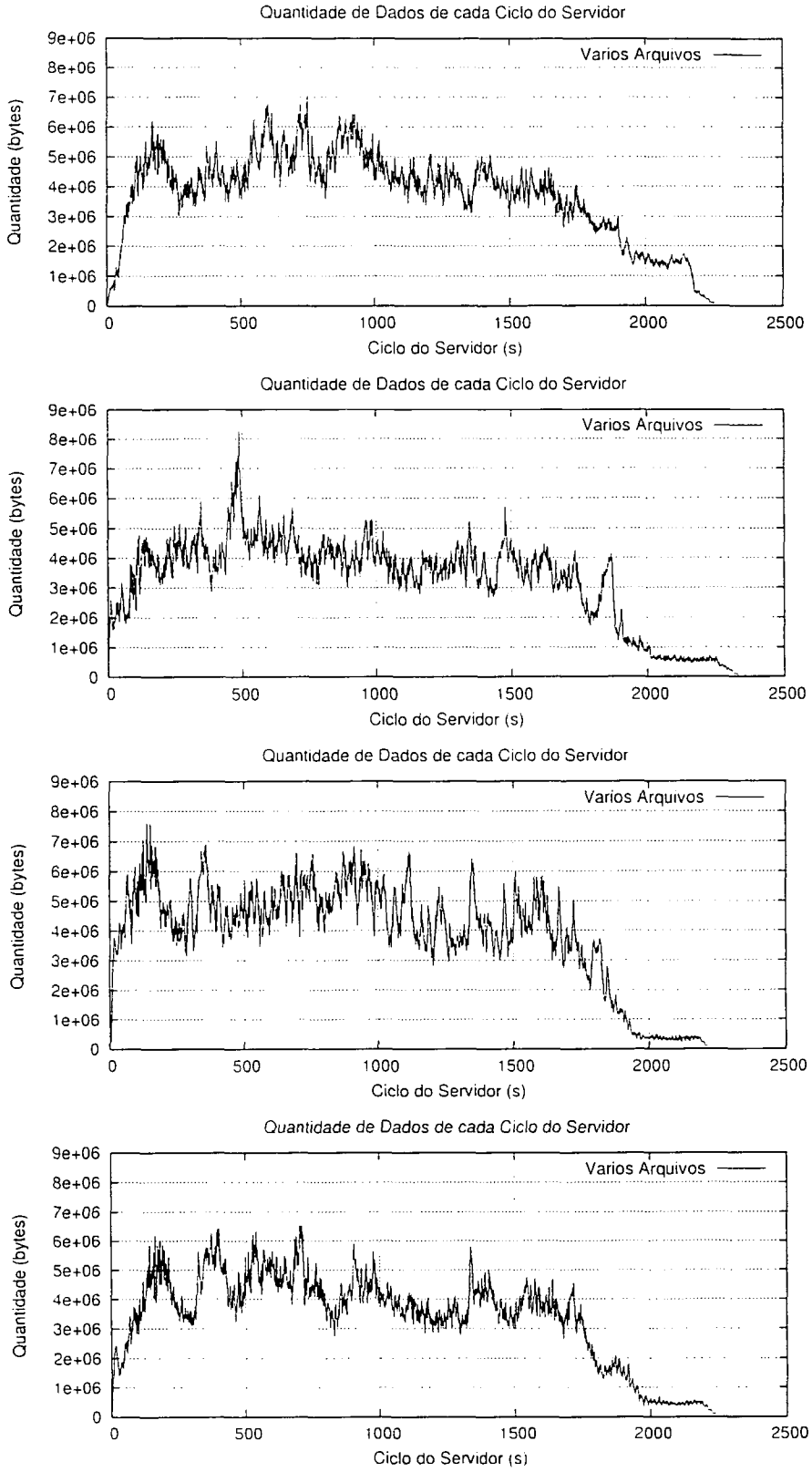


Figura B.9: Quantidade total de dados enviados em cada ciclo do servidor para requisições a diversos arquivos, utilizando o filtro 4.2.