Servidor de Vídeo SVFserver

Patrícia Sereda[1], **Roberto André Hexsel**[1]

[1] Departamento de Informática
Universidade Federal do Paraná
Centro Politécnico, C Postal 19081 – 81531-990 Curitiba, PR

patricia@inf.ufpr.br, roberto@inf.ufpr.br

Abstract. *This article describes the design and implementation of a vídeo server that is capable of transmitting video streams to any model of client. The video streams are generated from files stored on disk. The algorithms for transmitting the streams, and for admission control are described.*

Resumo. *Este artigo descreve a implementação do servidor de vídeo SVFserver. Este servidor é geral e disponibiliza fluxos de vídeo de arquivos de vídeo armazenados no disco a qualquer programa que possa reproduzí-los. São descritos o algoritmo responsável pelo envio de fluxos de vídeo aos clientes conectados e o algoritmo de controle de admissão que controla os recursos de rede disponíveis ao servidor.*

INTRODUÇÃO

O avanço da tecnologia na área de computadores e na área de redes vem permitindo o uso de aplicações que utilizam mais recursos do sistema. Em particular, aplicações que usam dados do tipo áudio e vídeo estão tornando-se cada vez mais populares. Atualmente existem servidores de vídeo conectados em redes internas e/ou locais que disponibilizam fluxos de arquivos de vídeo ao acesso dos usuários da rede.

Um *servidor de vídeo* é um programa que disponibiliza fluxos de vídeo a clientes que reproduzem o vídeo sem a necessidade de armazenamento local. As requisições podem ser solicitadas a qualquer momento e podem ser referentes a qualquer vídeo disponibilizado pelo servidor. Este tipo de serviço é conhecido como Video sob Demanda [Ver96].

Para que estes servidores possam disponibilizar vídeos a um ou vários usuários, aqueles devem ter controle de todos os recursos utilizados. Controlar os recursos para cargas como vídeo digital é um dos grandes desafios desta área. Este artigo descreve um servidor de vídeo que otimiza a utilização da capacidade da rede.

O texto está organizado em 5 seções. A seção 1 contém conceitos referentes a vídeo digital e a servidores de vídeo. A seção 2 descreve o método implementado no servidor para envio dos fluxos de vídeo, e a seção 3 o algoritmo de controle de admissão. A seção 4 contém informações das cargas utilizadas e os resultados obtidos nos testes. A seção 5 conclui o trabalho e propõe trabalhos futuros.

1. CONCEITUAÇÃO

Vídeo digital é a composição de uma seqüência de imagens estáticas, sincronizadas a um sinal de áudio. Cada imagem estática é chamada de *quadro*. Para se obter a aparência de movimento das imagens, vários quadros com pequenas diferenças entre si são organizados e reproduzidos em seqüência a uma taxa de, no mínimo, 24 quadros por segundo para se obter o efeito de persistência visual [Gro89].

Cada quadro de um vídeo digital é composto por elementos de imagem chamados *pixels*. Em uma imagem digital cada *pixel* é composto por três tipos de informações. No padrão *RGB*, cada *pixel* é composto de informações correspondentes às três crominâncias *Cr-Red*, *Cg-Green* e *Cb-Blue*, sendo cada informação representada em geral por 8 bits. No padrão *YCrCb*, cada *pixel* é composto de informações correspondentes à luminância *Y* e às crominâncias *Cr-Red* e *Cb-Blue*, sendo a luminância representada por 16 bits e cada crominância representada por 8 bits.

Cada quadro é representado por uma quantidade muito grande de informações, demandando assim uma quantidade de dados ainda maior para o armazenamento e transporte de vídeos. Para reduzir esta quantidade e tornar viável a utilização de vídeos digitais, foi desenvolvido um padrão de compressão para vídeo digital denominado *Motion Picture Experts Group (MPEG)* [Gal91].

O padrão *MPEG* reduz drasticamente a quantidade de dados de um vídeo através da eliminação de redundâncias. As redundâncias eliminadas são de 3 tipos: redundância espacial, redundância temporal e redundâncias de áudio.

A redundância espacial corresponde aos dados de elementos de imagem que preenchem áreas de uma imagem com informações repetidas. A redundância temporal corresponde às informações repetidas em quadros consecutivos ao longo da reprodução de um vídeo. A eliminação da redundância temporal produz três tipos de quadros: quadro *Intracoded (I)*, possui informações do quadro original eliminando somente a redundância espacial; quadro *Predictive (P)*, possui informações que diferenciam o quadro atual com relação a um quadro *I* ou *P* anterior; e quadro *Bidirectional (B)*, possui informações referente a diferenças do quadro atual com os quadros anterior e posterior *I* ou *P*, ao mesmo tempo.

As redundâncias espacial e temporal são variáveis a cada quadro e desta forma o fluxo de dados resultante do processo de compressão tem largura de banda variável no tempo. Quando o vídeo possui largura de banda variável no tempo ele é denominado *Variable Bit Rate (VBR)*. No caso inverso, quando a compressão resulta em quadros de tamanho constante, o vídeo possui então largura de banda constante no tempo e é denominado *Constant Bit Rate (CBR)* [Tan97].

Servidor de Vídeo

A figura 1 mostra a arquitetura básica de um servidor de vídeo. O servidor lê as informações do arquivo de vídeo do disco, armazena estas informações na memória principal e no tempo determinado, envia os fluxos de vídeo ao cliente.

Para que um cliente conectado a um servidor de vídeo possa reproduzir o vídeo respeitando o seu sincronismo, o servidor deve garantir que todos os fluxos de áudio e de vídeo sejam enviados sem atraso. Para que isso ocorra é imprescindível que o servidor de

vídeo controle todos os recursos utilizados, como o disco, a memória principal, a banda de rede e tempo de cpu. Para vídeos *CBR*, o controle dos recursos é facilitado, pois a quantidade de dados processados é previamente conhecido e sempre constante. No caso de vídeos *VBR*, o controle dos recursos do servidor é dificultado devido à variabilidade da largura de banda ao longo do tempo.

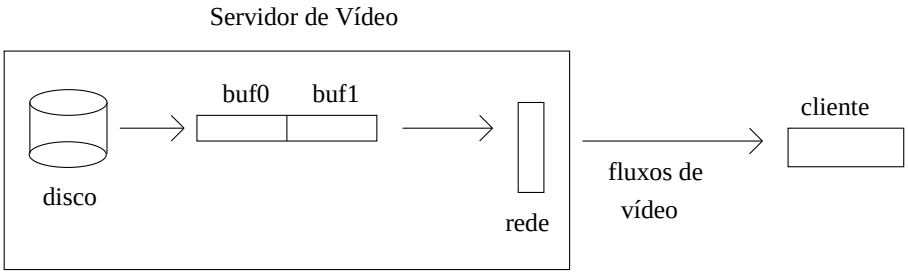

Figura 1: Arquitetura do servidor de vídeo.

Além de controlar os fluxos de dados que devem ser enviados a todos os clientes conectados, o servidor não pode permitir que novas requisições interfiram no seu desempenho. Novas requisições podem provocar sobrecarga em algum recurso, prejudicando assim, o envio dos fluxos de vídeo das requisições já atendidas. Para impedir que novas requisições ocasionem atrasos na entrega dos fluxos de vídeo, deve ser utilizado um algoritmo de controle de admissão.

2. MODELO DO SERVIDOR DE VÍDEO

O servidor *SVFserver* 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 e um algoritmo de controle de admissão. O *FFServer* é um servidor de vídeo que trabalha em conjunto com um programa de codificação de vídeo chamado *FFMpeg*. O *FFMpeg* alimenta o servidor com fluxos de vídeo codificados e o servidor repassa estes dados aos clientes conectados. O *FFServer* não é capaz de disponibilizar fluxos de vídeo diretamente a partir de arquivos de vídeo armazenados em disco.

O *SVFserver*, como o *FFserver*, foi implementado para trabalhar com qualquer programa que reproduza vídeo através do recebimento de fluxos de vídeo. O protocolo de interação com os clientes é sucinto e as características e comportamento dos clientes que podem interferir no desempenho do servidor são desconsideradas. O servidor *SVFserver* exerce controle o de sincronismo no envio dos fluxos de vídeo e é capaz de controlar os recursos internos do servidor.

As funções de controle de recursos do servidor são utilizadas somente para o envio de fluxos de vídeo de arquivos armazenados em disco. Não foram implementadas funções de decodificação das informações de áudio e vídeo no servidor, inviabilizando a implementação de serviços do tipo *Video Cassete Record (VCR)* como *fast forward, fast rewind, pause, resume, begin*.

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*.

2.1. Modelo do Servidor

A implementação das funções de envio de fluxos de vídeo e do algoritmo de controle de admissão do *SVFserver* basearam-se nas características de transmissão de arquivos de vídeo do tipo *VBR*, porque esta codificação produz vídeos de melhor qualidade, e esta é a codificação dos filmes disponíveis.

O processo de execução do servidor ocorre da seguinte forma. O servidor, ao ser inicializado, analisa cada arquivo de vídeo disponível em uma pasta específica de arquivos de vídeo, e obtém informações que serão utilizados nos algoritmos que controlam recursos do servidor. Após análise dos arquivos, o servidor fica em estado de espera, aguardando requisições. Ao receber uma requisição, o servidor utiliza o algoritmo de controle de admissão para verificar se existem recursos suficientes para acrescentar o fluxo de vídeo requisitado ao conjunto de fluxos das requisições previamente aceitas. Este algoritmo analisa a quantia exigida de cada recurso para o arquivo requisitado e a compara com a quantidade ainda disponível. Se a requisição não pode ser aceita, o servidor envia uma mensagem ao cliente recusando a requisição. Se a requisição pode ser aceita, o servidor envia uma mensagem confirmando a aceitação, reservando então recursos de banda de rede, tempo de leitura de disco e espaço na memória principal, necessários para satisfazer a esta nova requisição.

Estes recursos são reservados até a finalização do envio dos fluxos de vídeo, que corresponde à finalização da requisição por parte do cliente, podendo esta ser feita a qualquer momento que o cliente desejar, ou a finalização por parte do servidor, ao final da leitura do arquivo de vídeo.

Um *segmento* de fluxo de vídeo corresponde à quantidade de informações que deve ser enviada em um *ciclo*. Os dados que são enviados em um segmento de fluxo de vídeo são divididos e enviados em *pacotes*. As quantidades de dados enviados pelo servidor dependem de uma grandeza denominada *ciclo do servidor*. O ciclo do servidor corresponde ao intervalo de tempo decorrido entre o envio de dois segmentos de fluxo de vídeo contíguos, ou seja, o intervalo entre o início do envio de um segmento para um cliente e o início do próximo segmento enviado ao mesmo cliente. No caso deste servidor o ciclo corresponde a 1 segundo, e a cada segundo o servidor envia um segmento a cada cliente conectado, como mostra a figura 2. Nesta figura são enviados 4 segmentos a 4 clientes, 1 segmento para cada cliente.

Os segmentos enviados no ciclo atual correspondem aos dados que serão reproduzidos no cliente no próximo ciclo. Sendo assim, o servidor não pode atrasar a entrega do segmento atual, porque então causaria a interrupção na reprodução do vídeo por falta de dados no cliente *(starvation)*.

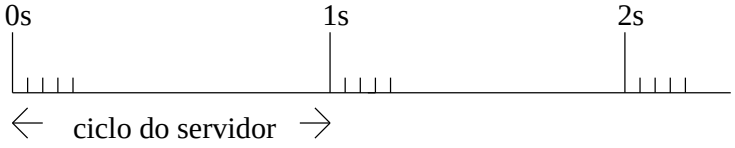

Figura 2: Ciclo do servidor.

Enquanto o servidor envia no ciclo atual um segmento ao cliente, ele deve no mesmo ciclo adiantar a leitura do disco, buscando os dados do segmento a ser enviado no próximo ciclo. Para permitir o procedimento de envio do segmento de fluxo de vídeo e a leitura do disco ao mesmo tempo, o servidor reserva duas áreas na memória principal, *buffer0* e *buffer1*, para cada requisição aceita, um para armazenar os dados que são lidos do disco e outro com os dados que serão enviados ao cliente, como mostra a figura 1.

O tamanho dos *buffers* alocados para cada requisição é calculado durante a análise do arquivo de vídeo, na inicialização do servidor. Como o ciclo do servidor corresponde a 1 segundo, o *buffer* deve comportar a maior quantidade a ser transmitida em um segundo.

Para melhorar o desempenho do servidor, são usados duas *threads*, uma responsável pela transmissão dos fluxos através da rede e outra responsável por efetuar a leitura do disco.

2.2. Modelos de 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* é proporcional ao intervalo de tempo disponível para a chegada de fluxos atrasados, e assim, 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 vídeo do tipo *CBR*, o consumo dos quadros ocorre na mesma proporção da chegada de novos quadros, permitindo que o buffer do cliente seja pequeno. Para arquivos de vídeo do tipo *VBR* a proporção dos quadros reproduzidos não é a mesma da proporção dos quadros recebidos, exigindo que o cliente reserve um *buffer* maior.

Tendo o cliente 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 influencia a escolha do método de envio dos fluxos de vídeo pelo servidor. Alguns métodos controlam o envio dos fluxos através do conhecimento da quantidade de memória disponível a cada ciclo do servidor.

Por razões de aplicabilidade deste trabalho, o *SVFserver* foi implementado para permitir conexão com qualquer cliente que possa reproduzir os arquivos de vídeo disponibilizados pelo servidor, através do recebimento de fluxos de vídeo. O método de comunicação com os clientes é o mais geral possível, não impondo qualquer restrição aos clientes conectados. Esta característica dispensa o servidor de obter informações referentes à utilização de *buffer* por parte do cliente, enviando assim fluxos de vídeo considerando que *buffers* não são empregados.

2.3. Modelo de Envio de Segmentos de Fluxos de Video

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 pode receber. Este modelo é denominado *Pull Model*. O segundo modelo é caracterizado pela passividade do cliente no envio dos dados, ou seja, todo o controle é feito pelo servidor. Este modelo é denominado *Push Model*[AMG98]. Devido às características da comunicação do servidor *SVFserver* com os clientes, foi implementado o *Push Model*.

Alguns dos algoritmos propostos de envio de segmentos de fluxos de vídeo que utilizam o modelo *Push Model* foram implementados e analisados durante a construção deste servidor [Fen97, WFS95, FS95a, FS95b, MR96, MR95]. 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 disponível ao servidor de vídeo fica reduzida porque o tamanho dos quadros é variável e a quantidade de dados enviados a cada ciclo do servidor não é constante, conforme mostram o gráfico da figura 3. Esta figura mostra que a quantidade de dados dos ciclos do servidor está em torno de 100Kbytes, e mostra também que em determinados ciclos isolados a quantidade chega a cerca de 450kbytes. Estes ciclos são responsáveis por possíveis congestionamentos na rede e conseqüentemente sobrecargas temporárias no servidor.

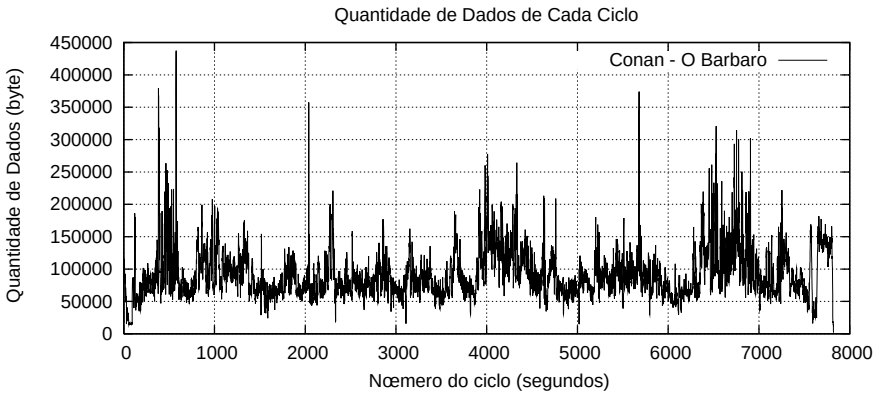

Figura 3: Quantidade de dados processados em cada ciclo do servidor.

Considerando-se que a banda de rede disponível é limitada, e que a demanda imposta pela transmissão dos vídeos é próxima de seu valor médio na maior parte do tempo, é possível atender a um maior número de requisições se a banda disponível ao servidor for alocada pelo valor médio da banda de cada vídeo. Contudo, esta forma simples de alocação pode causar tráfego excessivo na rede. A seguir são discutidos este e outros mecanismos mais sofisticados de alocação de banda.

Average Allocation O primeiro modelo de envio de segmentos de fluxos de vídeo implementado no servidor foi baseado no algoritmo *Average Allocation* [FS95a]. Este algoritmo calcula a média total dos tamanhos dos quadros do arquivo de vídeo. A quantidade de dados enviada a cada ciclo do servidor corresponde ao produto do tamanho médio calculado pelo número de quadros por segundo (*qps*) obtido do cabeçalho do arquivo de vídeo. Este algoritmo permite que o servidor remeta os segmentos de fluxos de vídeo a uma taxa constante, ou seja, a quantidade de dados enviados a cada ciclo do servidor é sempre a mesma. A vantagem deste algoritmo está no fato de se evitar que aumente drasticamente a banda utilizada nos momentos em que os quadros muito grandes são transmitidos, sobrecarregando temporariamente a rede.

Foi observado que nos momentos em que o cliente reproduz os maiores quadros, o servidor não fornecia toda a quantidade de dados a tempo de sua reprodução, ocasionando a interrupção da reprodução do vídeo no cliente. Isto ocorre porque um quadro grande precisa de vários pacotes de um ou mais ciclos do servidor para ser enviado ao cliente, exigindo um intervalo de tempo demasiadamente longo para ser transmitido. Uma solução para evitar este problema é enviar os dados de forma adiantada. Como o principal objetivo deste algoritmo é garantir o envio dos segmentos a uma taxa constante, para que os quadros fossem enviados adiantadamente o cliente deve dispor de um *buffer* para armazenar os quadros que chegam adiantados. Esta solução foi desconsiderada pois o servidor não possui conhecimento da existência de *buffer* no cliente.

Max Average Bandwidth Para evitar o problema de esvaziamento do *buffer* do cliente nos momentos de pico foi implementado o algoritmo *Max Average Bandwidth* [FS95a, Fen97]. Este algoritmo calcula a média dos tamanhos dos quadros ciclo a ciclo e emprega a maior destas como a *média máxima* 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, sendo então constante a cada ciclo.

O problema com este algoritmo é o fato de que transmitindo os fluxos de vídeo a uma taxa constante e calculada de acordo com o valor de pico, que corresponde à média máxima calculada na análise do arquivo de vídeo, pode ocorrer sobrecarga do envio dos dados no cliente, porque o servidor inunda o cliente com 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 um *buffer*. Para resolver os problemas encontrados com os algoritmos anteriores o servidor deveria enviar os segmentos dos fluxos de vídeo de acordo com o tamanho original de cada quadro, ou seja, os pacotes dos segmentos dos fluxos de vídeo enviados devem corresponder aos tamanhos dos quadros de vídeo. Partindo desta idéia foi implementado um terceiro algoritmo.

Rate-Constrained Bandwidth Smoothing O terceiro algoritmo implementado no servidor é o algoritmo *Rate-Constrained Bandwidth Smoothing (RCBS)* [Fen97, FS95a, KY99]. Este algoritmo calcula a média máxima dos tamanhos dos quadros do arquivo de vídeo e envia os pacotes dos segmentos dos fluxos de vídeo de acordo com os tamanhos dos quadros de vídeo originais, desde que sejam inferiores ao valor da média máxima calculada. Se o quadro a ser transmitido possui tamanho superior ao da média, o servidor envia a quantia da média calculada e distribui o valor em excesso nos quadros precedentes. Se o quadro tem tamanho inferior da média, o servidor envia o quadro original incluindo, se necessário uma parcela de um quadro muito grande que está para ser enviado.

Como este algoritmo envia 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 um determinado valor, que no caso é a média máxima calculada.

Este algoritmo deve considerar 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 este valor pode referir-se a um único quadro isolado que pode ser muito maior que os demais. Considerando-se somente um valor da média máxima, pode haver a liberação do envio de quadros de tamanho inferior ao valor da média máxima calculada mas que correspondem a valores muito grandes a ponto de ocasionar sobrecarga instantânea na rede.

Para solucionar este problema foi implementado no *SVFserver* um algoritmo em que a média máxima é calculada em cada ciclo do servidor. A implementação deste algoritmo exige dois passos de 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. No segundo passo, o servidor verifica quais quadros excedem ao valor da média calculada e quais quadros carregariam, se necessário, o valor excedente. Isso implica em que o servidor deve armazenar essas informações antes de receber requisições para o arquivo de vídeo analisado.

Como foi citado, o algoritmo *RCBS* deve determinar mais de um valor de média máxima para um arquivo de vídeo, para permitir que uma maior quantidade de quadros grandes tenham o seu valor dispersado em quadros menores, tornando a taxa do envio dos fluxos de vídeo mais constante possível, evitando intervalos de sobrecarga na rede.

O método de cálculo empregado no SVFserver é baseado no algoritmo de implementação do servidor. Quando o servidor, na fase de análise de cada arquivo de vídeo, determina a quantidade de dados que deve ser enviado a cada ciclo do servidor, é calculada também a média máxima de cada ciclo do arquivo de vídeo analisado, conforme mostra a figura 4.

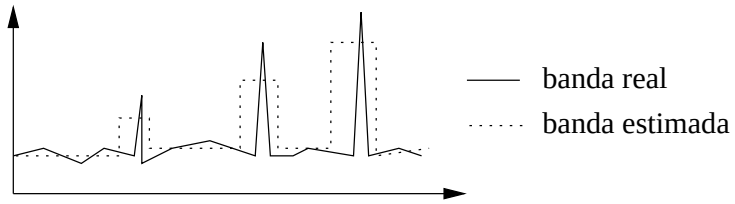

Figura 4: Ciclo do servidor.

O valor da quantidade que deve ser enviada e o valor da média máxima de cada ciclo é armazenado em uma lista. Esta lista é utilizada para informar a quantidade de dados que devem ser lidos do disco a cada ciclo. Quando o servidor estiver enviando fluxos de vídeo a clientes, toda vez que ele fizer a leitura da quantidade designada na lista, o servidor determina o tamanho dos quadros e compara-os com o valor da média máxima calculada para o ciclo em questão. Os quadros que possuem valor acima do valor da média máxima do ciclo tem sua quantidade em excesso transferida para quadros precedentes. A utilização deste método garante que o servidor envia 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.

Definido o melhor método para enviar os segmentos de fluxos de vídeo aos clientes, o servidor deve garantir que novas requisições não interfiram nas requisições atendidas. Tendo o servidor conhecimento prévio da quantidade de dados que são enviados aos clientes, torna-se possível avaliar e regular a utilização dos recursos de rede utilizados

pelo servidor. Para viabilizar este controle foi implementado um algoritmo de controle de admissão.

3. ALGORITMO DE CONTROLE DE ADMISSÃO

O algoritmo de controle de admissão de novas requisições é o algoritmo utilizado pelo servidor de vídeo para gerenciar a banda de rede, o tempo de acesso ao disco, tempo de cpu e ocupação da memória principal. Um dos grandes desafios encontrados no projeto de servidores de vídeo é garantir o controle de recursos quando são utilizados arquivos de vídeos de largura de banda variável. Vários algoritmos foram propostos e são descritos em [DWL96, ZF94, KZ95, DMH99, EKZ95, ZK97]. Este artigo descreve somente o controle de utilização de rede.

Neste servidor foi implementado um método determinístico para controle da banda de rede. A implementação é baseada no trabalho de E. Knightly e H. Zhang [ZK97]. Este trabalho analisa a diferença entre a utilização do tamanho do maior quadro do arquivo de vídeo para o cálculo da banda de rede, e a utilização de um grupo de quadros que possua a maior quantidade de dados para o cálculo da banda de rede.

O agrupamento de quadros para cálculo da banda de rede é possível devido à seqüência dos tipos de quadros de um arquivo de vídeo. Um quadro grande do tipo *I* sempre é seguido de alguns quadros *P* e *B* pequenos, e portanto não é necessário calcular a banda considerando somente o tamanho dos quadros *I*, pois estes estão separados por vários quadros *P* e *B*.

Foi implementado no SVFserver um algoritmo de controle de admissão que administra a utilização da banda de rede usando uma estimativa de largura de banda de um vídeo que é a maior soma dos tamanhos dos quadros a serem transmitidos em um ciclo. O controle de admissão mantém uma estimativa da banda utilizada pelos vídeos que estão sendo exibidos. A cada nova requisição, a banda desse vídeo é adicionada à estimativa e caso o novo valor estimado não ultrapassar 80% da capacidade da rede, a requisição é atendida.

Após realizados testes no servidor com este método, foi comprovado que ele não produz sobrecarga na rede, mas pode induzir a sub-utilização da rede. Para diminuir a estimativa de banda, e portanto aumentar o número de requisições aceitas pelo servidor, é usado 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 aplicado para calcular a média ponderada dos tamanhos de um ciclo com relação ao ciclos vizinhos. Desta forma, se a quantidade de dados a serem enviados em um ciclo for muito grande comparada com os ciclos vizinhos, seu valor estimado é reduzido pela aplicação do filtro. Os dois filtros, *filtro 1* e *filtro 2*, definidos pelas equações 1 e 2 respectivamente, são utilizadas para estimar novos valores de banda de rede utilizada pelos arquivos de vídeo.

$$\text{banda}_{f1} = \sum^{-4<n<0} 0.1 \cdot ciclo_n + 0.2 \cdot ciclo_0 + \sum^{0<n<4} 0.1 \cdot ciclo_n \quad (1)$$

$$\text{banda}_{f2} = 0.025 \cdot ciclo_{n\pm10} + \sum^{-10<n<10} 0.05 \cdot ciclo_n \quad (2)$$

4. RESULTADOS EXPERIMENTAIS

O algoritmo de controle de admissão soma as quantias de banda reservada a cada requisição e compara com o valor correspondente à 80% da capacidade disponível ao servidor, que corresponde a 10Mbytes/s, em uma Ethernet de 100Mbps. Adicionando o valor utilizado à demanda imposta por nova requisição, o algoritmo confirma a aceitação se o resultado da soma for inferior ao valor da banda disponível.

A medição da utilização de banda de rede durante os testes foi realizada com o *Xnetload*. Este programa analisa a interface de rede e mostra a quantidade de dados e de pacotes enviados e recebidos durante o período em que é executado. Os testes foram executados da seguinte forma. Ao iniciar o servidor, o *Xnetload* é também iniciado, e várias requisições são geradas para um mesmo arquivo de vídeo durante um período curto. Desta forma é possível sobrepor os ciclos em que são enviadas grandes quantidades de dados. As requisições são geradas até o algoritmo de controle de admissão recusar novas requisições por causa de possível sobrecarga no servidor.

Carga Utilizada Os testes realizados com o servidor utilizaram os dois filmes descritos na tabela 1. Os gráficos das figuras 5 e 6 mostram a quantidade de dados enviados a cada ciclo do servidor, e a grande variabilidade da largura de banda dos filmes utilizados nos testes.

título	dur [s]	qps	codificação	resolução	banda [kB/s]
Conan O Bárbaro	7822	24	msmpeg4, mp3†	512x208	437.396
Conan O Destruidor	6054	24	msmpeg4, mp3‡	640x272	391.560

† stereo a 44.1KHz; ‡ stereo a 48KHz

Tabela 1: Caracterização da carga.

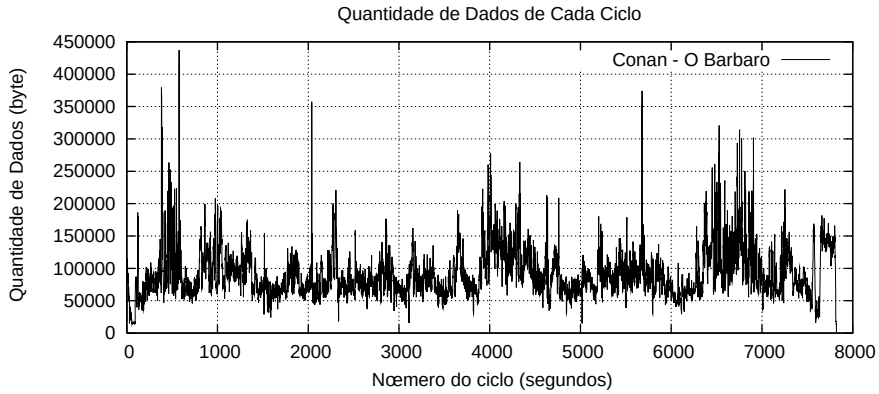

Figura 5: Quantidade de dados processados em cada ciclo do servidor.

Os testes foram executados em uma rede local Ethernet de 100Mbps, com clientes e servidores interligados por um comutador 3com SuperStack3-4226T. O servidor foi instalado em um computador com processador Pentium III de 866MHz, 256Mbytes de RAM, e disco de 40Gbytes, executando GNU/Linux, distribuição Debian.

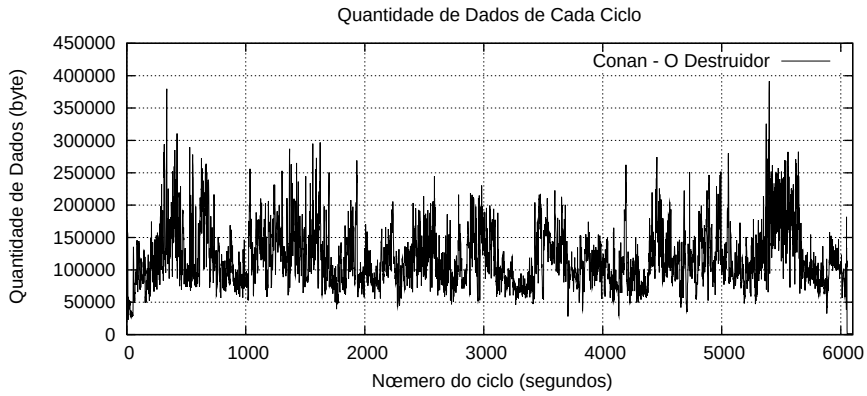

Figura 6: Quantidade de dados processados em cada ciclo do servidor.

Testes para o valor original da banda reservada

Estes dois testes comprovam que o uso da soma do tamanho dos quadros no cálculo da banda reservada a um arquivo de vídeo (método de Zhang [ZK97]), não causa *starvation* nos clientes, mas mostram que o servidor sub-utiliza a rede pois os resultados obtidos do *Xnetload* indicam que a rede não atinge 50% da sua capacidade.

título	req	pico [kB/s]
Conan O Bárbaro	22	4.164
Conan O Destruidor	25	5.760

Tabela 2: Requisições aceitas e tráfego induzido.

Testes para o valor da banda obtido do filtro 1

O valor da banda calculado com o filtro 1 (equação 1) corresponde à media ponderada dos tamanhos de 8 ciclos contíguos, com o ciclo corrente com peso de 20%. Utilizando este filtro para cálculo da banda de rede foram obtidos os resultados na tabela 3. Os gráficos das figuras 7 e 8 mostram a quantidade de dados enviados em cada ciclo.

título	req	pico [kB/s]
Conan O Bárbaro	36	6.541
Conan O Destruidor	34	7.588

Tabela 3: Requisições aceitas e tráfego induzido com filtro 1.

A tabela 3 mostra as medidas obtidas com o *Xnetload*. Estas medidas são comparadas às quantidades de dados medidos pelo servidor em tempo de execução. Para o filme *Conan - O Destruidor* o filtro determina uma banda de 288.003 bytes/s. O gráfico da figura 7 mostra a quantidade de dados enviados pelo servidor a cada ciclo. Nele observa-se que é enviada a quantia de 7.456.596 bytes/s (aproximadamente 5.500 segundos após a inicialização do teste) ao atender as 34 requisições, valor $1,73\%$ inferior ao medido pelo *Xnetload* que corresponde a 7.588 kBytes/s.

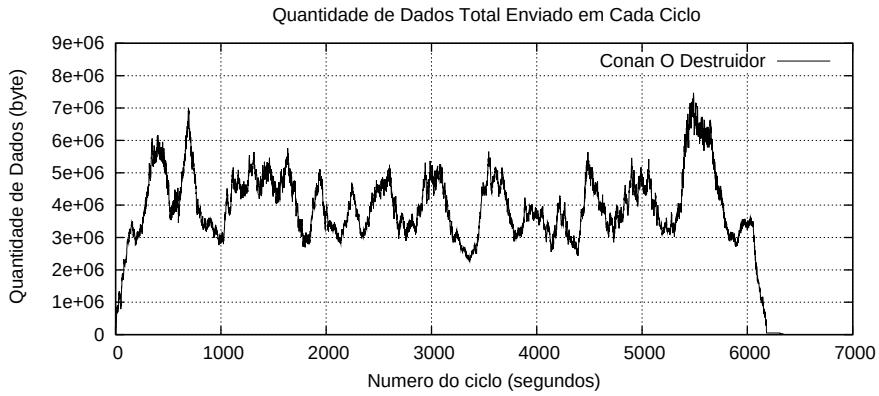

Figura 7: Quantia total enviada em cada ciclo do servidor – filtro 1.

Para o filme *Conan - O Bárbaro* o filtro determina uma banda de 272.282 bytes/s. O gráfico da figura 8 mostra a quantidade de dados enviado pelo servidor a cada ciclo, nele observa-se que é enviada a quantia de 6.458.672 bytes/s (aproximadamente 6.500 segundos após a inicialização do teste) ao atender as 36 requisições, valor $1,26\%$ inferior ao medido pelo *Xnetload* que corresponde a 6.541 kBytes/s.

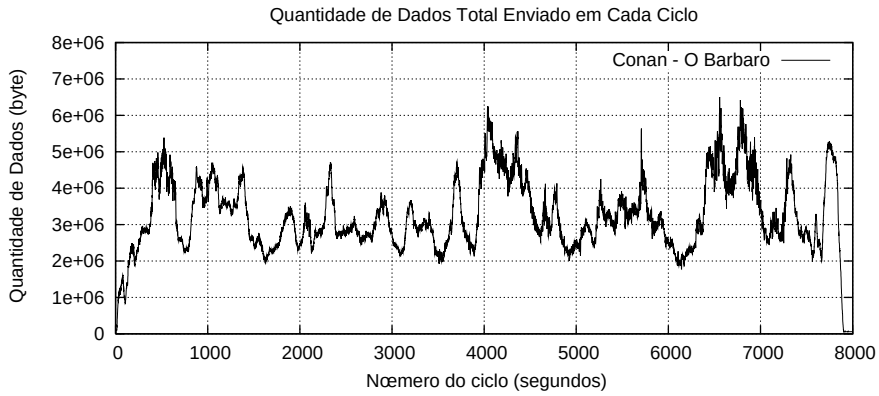

Figura 8: Quantia total enviada em cada ciclo do servidor – filtro 1.

Os dois testes realizados com o filtro 1 mostram que a demanda imposta ao servidor chega a apenas 61% da capacidade da banda de rede para o filme *Conan O Destruidor*, que é o pior caso. Observa-se que os valores medidos pelo *Xnetload* são um pouco superiores ao valor acusado pelo servidor. Isto ocorre porque o servidor calcula somente os bytes de dados enviados enquanto a ferramenta de medição de tráfego de rede mede todo o tráfego, incluindo os bytes enviados e o cabeçalho dos pacotes.

Testes para o valor da banda obtido do filtro 2

O valor da banda calculada com o filtro 2 (equação 2) corresponde à média ponderada dos tamanhos de 19 ciclos contíguos incluindo o ciclo corrente com peso de 5%, e 2,5% aos ciclo mais distantes em cada lado. Utilizando este filtro no cálculo da banda de rede foram obtidos os resultados da tabela 4. Os gráficos das figuras 9 e 10 mostram a quantidade de dados enviados em cada ciclo.

título	req	pico [kB/s]
Conan O Bárbaro	47	8.241
Conan O Destruidor	41	9.265

Tabela 4: Requisições aceitas e tráfego induzido com filtro 2.

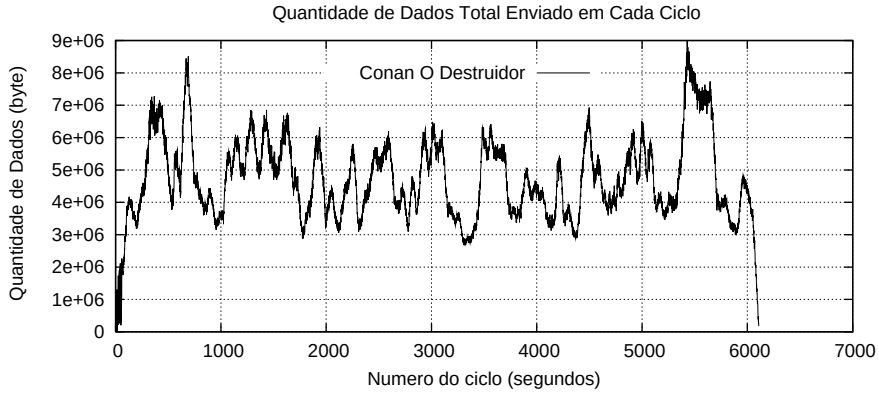

Figura 9: Quantia total enviada em cada ciclo do servidor – filtro 2.

A tabela 4 mostra as medidas efetuadas pelo *Xnetload*. Estas medidas são comparadas às quantidades de dados medidos pelo servidor em tempo de execução. Para o filme *Conan - O Destruidor* o filtro estimou uma banda de 241.408 bytes/s. O gráfico da figura 9 mostra a quantidade de dados enviado pelo servidor a cada ciclo. Nele observa-se que é enviada a quantia de 9.083.372 bytes/s (aproximadamente 5.500 segundos após a inicialização do teste) ao atender 41 requisições, valor $1,96\%$ inferior ao acusado pelo *Xnetload*, que corresponde a 9.265 kBytes/s.

Para o filme *Conan - O Bárbaro,* o filtro estimou uma banda de 210.523 bytes/s. O gráfico da figura 10 mostra a quantidade de dados enviado pelo servidor a cada ciclo, nele observa-se que é enviado a quantia de 7.991.844 bytes/s (aproximadamente 6.500 segundos após a inicialização do teste) ao atender as 47 requisições, este valor $3,02\%$ inferior ao acusado pelo *xnetload* que corresponde a 8.241 kBytes/s.

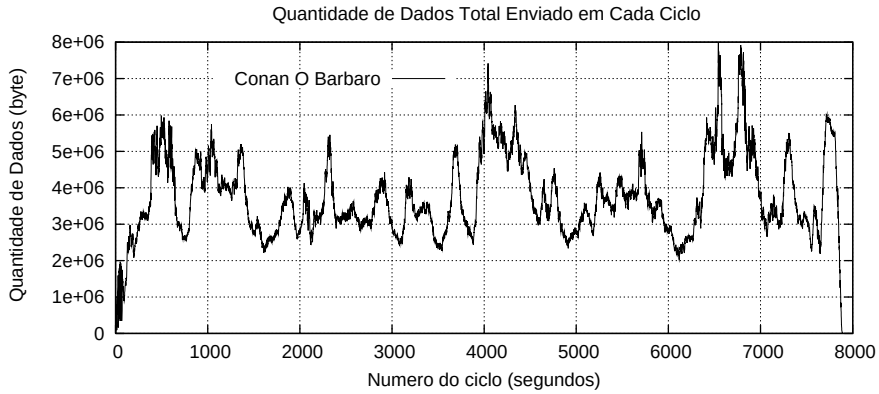

Figura 10: Quantia total enviada em cada ciclo do servidor – filtro 2.

Estes testes mostram que a aplicação do segundo filtro aumenta a utilização da rede para até 75% da largura de banda. Este valor aproxima-se ao valor determinado pelo algoritmo de controle de admissão que limita a banda a 10 MBps ou 80% da sua capacidade. Para aumentar ainda mais a utilização da rede deve-se alterar o limite definido no algoritmo de controle de admissão para um valor acima de 10MBytes/s e utilizar-se o filtro 2 testado, que garante uma melhor aproximação da banda consumida com a banda reservada.

5. CONCLUSÕES

Este trabalho descreve a implementação de um servidor de vídeo e os algoritmos utilizados no envio de fluxos de vídeo e no controle de recursos de rede. Este servidor foi implementado com o objetivo de disponibilizar fluxos de vídeo a qualquer programa que reproduza vídeo através do recebimento destes fluxos. Testes foram executados para determinar qual a melhor estimativa para o cálculo da banda de rede consumida pelo servidor e a segunda estimativa produziu a melhor utilização da rede sem causar atrasos na recepção pelos clientes.

Referências

[AMG98] J. Al-Marri and S. Ghandejarizadeh. An evaluation of alternative disk scheduling techniques in suport of variable bit rate continuos media. *International Conf. on Extending Database Technology - EDTB*, April 1998.

[Bel03] F. Bellard. FFmpeg Multimidia System, 2003. `http://ffmpeg.sourceforge.net`.

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

[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*, 1995.

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

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

[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.

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

[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.

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

[Tan97] A. S. Tanenbaum. *Redes de Computadores*. Editora Campus, 1997.

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

[WFS95] F. Jahanian W. Feng and S. Sechrest. Optimal buffering for the delivery of compressed prerecorded video. *Proc. of the IASTED/ISMM International Conf. on Networks*, January 1995.

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

[ZK97] H. Zhang and E. Knightly. D-BIND: An accurate traffic model for providing QoS guarantees to VBR traffic. *IEEE/ACM Transactions on Networking*, april 1997.