Suporte para Transmissão de Mídia entre Servidor e Clientes Cooperativos

Marcio Akio Shimoda, Luis Carlos Trevelin
Departamento de Computação – Universidade Federal de São Carlos (UFSCar)
shimoda@gsdr.dc.ufscar.br, trevelin@dc.ufscar.br

Abstract

In this research we address the problem of the distribution of multimedia content for a large number of computers. Specifically, we consider the problem of the overload of the server, that occurs due to the fast increase of the volume of solicitations to receive content. We present as solution a communication model to complement the traditional client-server. It is based on the cooperative networks, in which the customers cooperates to distribute the media, where they act as a kind of temporary server, until the server come back to the normal one.

1. Introdução

É inegável a contribuição que a transmissão multimídia traz. Isso se deve ao fato de que áudio, vídeo e imagens são um meio natural para a comunicação humana, além de serem ricas em informação. Contudo, o tradicional modelo cliente-servidor se mostra inadequado para a transmissão desse tipo de dados. Isso se deve, principalmente, à alta largura de banda requerida, o que acaba por limitar o número de clientes que podem ser servidos.

Muito têm se pesquisado para criar uma forma do modelo cliente-servidor melhor acomodar o tráfego multimídia. Alguns defendem que nenhuma alteração nos protocolos e serviços é necessária, devendo apenas adicionar mais banda. Outros se opõem observando que em breve surgiriam novas aplicações que consumiriam a banda, gerando assim um ciclo vicioso de disponibilidade e exaustão de largura de banda.

Uma visão mais interessante é a que pequenas mudanças em certos pontos do modelo cliente-servidor são suficientes para que este acomode a transmissão multimídia. Mudanças que permitam ao sistema se adaptar às alterações nas condições de transferência de dados, para conseguir um melhor aproveitamento da banda. É essa abordagem que utilizamos.

2. Trabalhos Relacionados

A disponibilidade de computadores multimídia e redes de alta velocidade tem feito crescer a demanda por aplicações distribuídas multimídia. Por isso, muitos esforços têm sido feitos no sentido de se criar novas abordagens para a distribuição de conteúdo. Os novos trabalhos nessa área se dividem basicamente em três grupos: a distribuição centralizada de conteúdo, a distribuição de conteúdo baseada em infraestrutura e o peer-to-peer.

Na distribuição centralizada, utilizando o modelo cliente-servidor (ver sessão 3.1), um computador é configurado para servir mídia para aqueles que a ele requisitarem conteúdo. Para melhorar a capacidade de armazenamento e atendimento pode-se usar clusters e servidores espelho, mas ainda assim ele apresenta problemas de escalabilidade.

As redes de distribuição de conteúdo (CDN - Content Distribuition Network ou Content Delivery Network) [1, 2] são baseadas em infra-estrutura e empregam um conjunto de máquinas dedicadas a armazenar e ajudar o servidor a distribuir conteúdo para os clientes. Os nós de uma CDN cooperam entre si para atender o usuário de forma satisfatória, movendo transparentemente o recurso para outras localizações de forma a otimizar o processo de distribuição. Garante alta disponibilidade e um bom desempenho, porém seu custo é auto.

No peer-to-peer (sessão 3.2) a proposta é colocar os clientes para armazenar e distribuir o conteúdo para outros clientes. O SpreadIt [3] foi um dos primeiros trabalhos de peer-to-peer com essa finalidade. Nessa arquitetura, uma abordagem voltada às transmissões ao-vivo, o stream é transmitido aos clientes através de uma árvore multicast implementada ao nível de aplicação. Na árvore, cada nó possui uma camada de peering responsável pela manutenção da árvore. Contudo, descobriu-se que a QoS é degradada de acordo com o número de clientes [4].

Também voltado para a transmissão ao vivo, o Zigzag [5] organiza seus peers em uma hierarquia de clusters e constrói uma árvore multicast acima dela. Em cada cluster um peer líder (leader ou head) é responsável por gerenciar a entrada de novos peers e um peer co-líder (coleader ou associate-head) tem a responsabilidade de transmitir os dados para os outros membros. O Zigzag possui esse nome porque o líder não encaminha dados para os demais no mesmo cluster, apenas para aqueles em um cluster diferente.

Ao contrário do que acontece no SpreadIt e no Zigzag, o GloVE (Global Vídeo Environment) [6], é uma abordagem estritamente sob-demanda. Ele propõe a utilização da técnica de reuso de fluxo denominada Cache de Vídeo Cooperativa (CVC) para adicionar maior escalabilidade a servidores de vídeo sob-demanda. Os buffers locais dos clientes integram uma memória global distribuída, usada como fonte de conteúdo. Ao chegar a primeira requisição, o servidor inicia um novo fluxo para o cliente. Quando uma segunda requisição chegar, o gerente da CVC procura em sua tabela por um cliente, chamado provedor, que possua a parte inicial do vídeo em seu buffer local. Caso seja encontrado, o cliente é inserido no grupo de multicast servido pelo provedor.

Um trabalho semelhante ao GloVE é o P2VoD [7]. Trata-se de uma arquitetura P2P, focada na transmissão sob-demanda, onde os clientes são agrupados em uma hierarquia de gerações. Pertencem a uma mesma geração aqueles que estão reproduzindo o mesmo trecho (ou um instante muito próximo do vídeo). Os peers da primeira geração são conectados diretamente ao servidor, os da segunda geração conectam-se a peers da primeira, e assim por diante. O nó pai é escolhido na geração anterior através de algoritmos de round-robin, menor atraso ou maior proximidade.

O Cooperative Networking (CoopNet) [8, 9] suporta tanto a transmissão ao vivo quanto a sob-demanda. Nascido como uma proposta para aliviar o volume de requisições à servidores web utilizando o poder de upload de alguns de seus clientes para servir outros, CoopNet verifica quais os outros clientes que acessaram a mesma URL (página web ou mídia) recentemente, monta uma lista com seus endereços e a envia para o cliente que fez o pedido, este ao receber a mensagem requisita conteúdo para os clientes na lista.

Nos trabalhos descritos acima, o servidor precisa ter conhecimento da rede. No GnuStream [10] e na arquitetura proposta em [4], um algoritmo de inundação é usado para a descoberta de recursos. Contudo, esse método apresenta problemas de escalabilidade quando se deseja alcançar todos os peers [11].

3. Modelos de Comunicação

Quando há diversas entidades que desejam se comunicar, torna-se necessária adoção de um protocolo comum de comunicação. É o que acontece num debate televisivo, onde um moderador que dá a palavra a cada um dos participantes através de um conjunto de regras pré-estabelecidas, ou numa conversa via rádio, em que cada parte fala alternadamente, ou numa entrevista, em que um interlocutor coloca questões e outro limita-se a responder. Todas essas estruturas se baseiam em um mecanismo de pergunta e resposta.

Os modelos para troca de dados em uma rede seguem essa estrutura entrevistador-entrevistado. A comunicação ocorre apenas entre duas entidades. Há sempre uma entidade que toma a iniciativa, enquanto a outra espera pelo pedido. A entidade que recebe o pedido responde a um conjunto perguntas pré-definidas, e que constituem a interface do entrevistado.

3.1 Cliente-Servidor

No modelo de comunicação cliente-servidor cada computador pode ser configurado como cliente ou servidor. Os servidores são máquinas poderosas dedicadas a oferecer algum tipo de serviço, tais como armazenamento de arquivos (servidores de arquivos), impressão (servidor de impressão) ou de acesso à rede (servidores de rede). Clientes são computadores nos quais os usuários executam seus programas e solicitam serviços aos servidores.

Comparando com a estrutura entrevistador-entrevistado, o entrevistador pode ser visto como a entidade que presta serviço, sendo que para cada pedido há uma resposta definida, dai a designação servidor. O entrevistado é o cliente, que usa os serviços.

O modelo cliente-servidor é baseado em um protocolo bem simples, sem conexão, do tipo solicitação/resposta. Primeiramente o cliente envia uma mensagem contendo a solicitação de um serviço do servidor, como por exemplo, o valor de um determinado campo no banco de dados do sistema descrito anteriormente. Em seguida, o servidor executa o serviço associado ao pedido do cliente. Por fim, o servidor devolve ao cliente uma mensagem contendo a resposta ao pedido (o resultado do serviço requerido pelo cliente).

Neste modelo o cliente é quem sempre toma a iniciativa, enquanto que, o servidor, fica passivamente esperando pelos pedidos. Dois fatos devem ser ressaltados: um é que o modelo não se preocupa com o

modo como a comunicação é executada, outro é que os pedidos permitidos (serviços oferecidos) são descritos pela interface do servidor.

3.2 Peer-to-peer

Popularizado pela utilização em redes de compartilhamento de arquivos como Napster [12], Gnutella [13] e BitTorrent [14], o peer-to-peer, freqüentemente referido como P2P, é um modelo de comunicação "cliente-cliente", no qual cada computador tem equivalentes capacidades e responsabilidades.

Em um primeiro momento a definição pode parecer um tanto quanto obscura, mas há uma grande diferença entre o modelo cliente-servidor e o P2P. No primeiro há uma clara distinção entre a parte que provê e a que usa o serviço. Já no segundo, não existe uma definição fixa entre clientes e servidores. Um outro fato que difere os dois modelos é que no P2P não há um ponto central em suas redes, por isso são geralmente simples. Isso quer dizer que ele não possui um ponto central de falhas.

Esse modelo possibilita que as pessoas possam fazer conexões diretas para interação em tempo real. Essa interação pode ser uma simples conversa via mensagens instantâneas, ou uma teleconferência, ou mesmo um compartilhamento de arquivos.

Contudo, o P2P apresenta alguns problemas. Ele não oferece o mesmo desempenho do cliente-servidor. Em geral a localização do recurso necessário é custosa, pois exige a busca em diversos nós, ao passo que no cliente-servidor, basta procurar no servidor.

4. Modelo Proposto

Com o objetivo de minimizar os problemas apresentados pelo tradicional modelo cliente-servidor na transmissão de mídia pela Internet, apresenta-se um novo modelo de comunicação, o WaveNet.

De forma semelhante ao CoopNet (Cooperative Networking), o modelo criado utiliza alguns de seus clientes para, através de conexões peer-to-peer, servir conteúdo para aqueles que requisitarem o recebimento da mídia quando o servidor estiver sobrecarregado. Chamamos esses clientes que auxiliam o servidor de clientes cooperativos.

O novo modelo apresenta um sistema de agrupamento dos clientes de acordo com o trecho da mídia que estão compartilhando. Esse trecho da mídia é o que chamamos de onda ou wave.

Essa divisão em ondas possibilita que a rede assuma uma configuração semelhante na transmissão de vídeos sob-demanda e na transmissão ao vivo, diferindo basicamente no número de ondas simultâneas. Nas sessões seguintes discutimos isso mais detalhadamente.

O modelo WaveNet consiste basicamente de duas partes: cliente e servidor. O cliente é a entidade que solicita e recebe a mídia. O servidor é quem distribui a mídia. Porém, este não é um servidor comum, ele possui um elemento adicional, o tracker, que cataloga cada peer da rede e retorna listas com o endereço deles.

A interação entre esses três elementos ocorre como mostrado no diagrama de seqüência da figura 1. O cliente que requisitar a mídia quando o servidor estiver sobrecarregado recebe uma lista de endereços de clientes que já estão recebendo conteúdo. Recebida a lista, o cliente escolhe um dos clientes na lista e negocia com ele a transmissão.

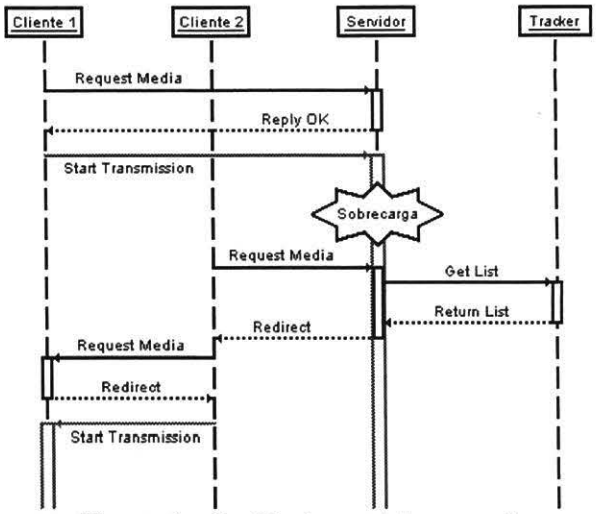

Figura 1 - Seqüência geral de operação

Há duas formas de oferecer conteúdo para os clientes através da rede. Uma dessas formas é usar a transmissão ao vivo, a outra é usar a transmissão sob-demanda. O modelo aqui descrito assume um comportamento semelhante nos dois casos, ao contrário do que acontece nos sistemas de distribuição de conteúdo tradicionais. Nas sessões seguintes veremos isso com mais detalhes.

4.1 Transmissão ao vivo

A transmissão de mídia ao vivo é caracterizada pela distribuição de conteúdo sendo obtido de algum dispositivo de captura, como a cobertura de um evento, ou pré-gravado e distribuído apenas em um único instante, como acontece na televisão.

Nesse tipo de transmissão, o conteúdo que um usuário recebe e vê é muito próximo do que os outros usuários recebem e vêem. Explicando de uma outra

forma, todos os clientes devem estar reproduzindo o mesmo trecho da mídia quase ao mesmo tempo.

Devido a essa sincronia de conteúdo a ser exibido, a transmissão de mídia ao vivo compõe um cenário favorável para a aplicação de nosso modelo. Quando ocorre uma sobrecarga no servidor, os clientes passam a compartilhar o conteúdo que já possuem em seus buffers com aqueles que solicitarem a mídia.

A figura 2 mostra a configuração de uma rede WaveNet quando isso ocorre. Na figura, o M[1] ao lado das setas indica a transmissão do instante 1 e o M[1]⁻ representa um instante anterior a 1.

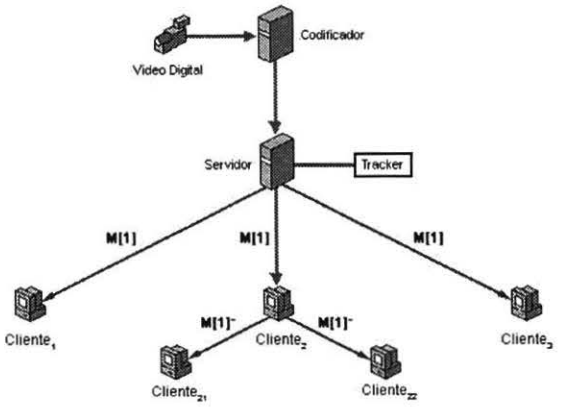

Figura 2 - WaveNet na transmissão ao vivo

4.2 Transmissão Sob-Demanda

A transmissão de mídia sob-demanda refere-se àquela que inicia quando o conteúdo é solicitado. São os próprios usuários que decidem quando a transmissão deve iniciar.

Diferentemente do que acontece na transmissão ao vivo, na sob-demanda raramente todos os usuários estão reproduzindo o mesmo trecho da mídia. Para o streaming progressivo isso é problema, mas para o streaming de tempo real tal fato reduz a potencialidade de um cliente servir conteúdo para os demais. Como o conteúdo é descartado após algum tempo, o trecho que um cliente possui pode não ser o que os demais estão precisando. Por exemplo, se o buffer de um cliente guardar até 10 segundos de transmissão, o seu conteúdo não será o que mais precisam, aqueles que conectaram 10 segundos mais tarde.

Contudo, em situações onde há uma sobrecarga devido ao volume de requisições, podemos encontrar vários usuários que iniciaram a transmissão ao mesmo tempo. Em WaveNet esses usuários sincronizados são separados em grupos, de acordo com a onda em que estão. Dessa forma, usuários na mesma onda podem compartilhar conteúdo entre si. Repare que no caso da transmissão ao vivo o que temos é um sistema de apenas uma onda.

No que diz respeito ao funcionamento e ao gerenciamento dos buffers, tudo ocorre da mesma forma que na transmissão de mídia ao vivo. Um outro ponto que deve ser ressaltado é que ao contrário de CoopNet, o nosso modelo não faz cache na transmissão sob-demanda, pois isso exige um considerável espaço em disco.

4.3 Gerenciamento dos Buffers

No streaming de tempo real, geralmente, o conteúdo é descartado a medida em que é exibido. Por esse motivo, o modelo WaveNet utiliza um mecanismo de gerenciamento dos buffers a fim de evitar que o conteúdo não seja descartado quando os demais ainda precisam dele.

O buffer usado em WaveNet é o circular, isso significa que ao ser totalmente preenchido, ele passa a sobrepor os dados mais antigos, porém sem sobrescrever informações ainda não lidas pelo reprodutor de mídia. Caso o usuário precise do espaço ocupado por dados que algum cliente esteja solicitando para armazenar conteúdo, a conexão com o solicitante é fechada, o que força este a buscar outro cliente que possua o trecho solicitado, só então os dados são sobrescritos.

4.4 Montagem da Lista

Como no Napster, os clientes devem reportar o seu perfil (banda) para o servidor junto ao seu pedido de conexão. Quando o servidor solicita ao tracker uma lista de endereços, ele seleciona dentre os peers que entraram na rede mais recentemente, aqueles que possuem um perfil semelhante ao do solicitante. São considerados de perfil semelhante aqueles que possuem conexões com largura de banda próxima e pertencem ao mesmo grupo (tab. 1).

A escolha dessa estratégia se deve ao fato de que para um cliente utilizando ADSL não há vantagem em tentar obter conteúdo de um usuário de um modem de 56 KB. O mesmo vale para o caso oposto, o usuário do modem não tem muito a ganhar conectando-se ao ADSL, pois ele está limitado à sua própria a banda e mesmo os dados tendo a possibilidade de serem enviados com rapidez, eles seriam recebidos lentamente. Além disso, caso os peers sejam todos redirecionados para os de banda larga, quando outro com o mesmo tipo de conexão tentar entrar na rede, os peers que poderiam o atender melhor teriam pouca banda disponível, já que estariam sobrecarregados.

Contudo, peers com maior largura de banda podem suportar vários clientes de conexões mais lentas, equilibrando melhor o sistema. Por isso, também são incluídos na lista, peers com maior largura de banda, para caso não haja peers de perfil semelhante que possam atender ao solicitante. E para que clientes de banda larga possam ser atendidos satisfatoriamente, mesmo com outros clientes conectados ao mesmo peer, todos os peers são atendidos a uma velocidade proporcional ao seu perfil.

Tabela 1 - Classificação dos clientes de acordo com a banda e grupo

Perfil	Grupo
14.4 Kbps modem	Banda Estreita
28.8 Kbps modem	
56 Kbps modem	
64 Kbps ISDN	
112 Kbps ISDN	Banda Média
256 Kbps DSL/cable modem	Banda Larga
512 Kbps DSL/cable modem	
Acima de 512 Kbps	

4.5 Seleção do Peer

Após receber a lista, o cliente deve tomar uma importante decisão, que é a escolha de qual peer obter conteúdo. Obviamente, ele deve escolher aquele que está em melhores condições para distribuir a mídia, contudo, descobrir qual é, não é tarefa trivial. Alguns trabalhos como [15] e [16] propõem que os melhores peers são aqueles mais próximos, mas para tráfego pesado como mídia, a largura de banda é uma medida melhor. Em [17] os autores propõem o uso de árvores de decisão para classificar e um algoritmo de aprendizado baseado em informações coletadas de downloads anteriores para selecionar os peers. Porém isso exige uma grande coleta de dados. Do mesmo problema sofrem alguns trabalhos de estimativa de capacidade [18] e banda disponível [19, 20, 21, 22], que poderiam servir para classificar os peers.

De forma a evitar todos esses problemas, adotamos uma estratégia mais simples. Para descobrir qual peer dispõe de mais banda, o cliente solicita para até 5 peers da lista que estimem a banda que poderão disponibilizar e enviem o resultado. A banda disponível é estimada dividindo a velocidade nominal que é possível transmitir pelo número de clientes já conectados ao peer mais um. Se em 10 segundos nenhum dos peers responder ou se as respostas forem muito baixas, o cliente escolhe outros 5 peers na lista e pede a mesma informação para eles. Os algoritmos de todo esse processo são mostrados nas figuras 3 e 4.

```
do forever
  m_q ← receive(*)
  if (m_q == REQUEST_MEDIA)
    if (getUsedBandwidth() < SERVER_BANDWIDTH)
      T = T U {(A_q,P_q)}
      streamMedia(q)
    else
      L ← getMostRecentlyConnected(P_q);
      send(REDIRECT, L) to q

  else if (m_q == CONFIRM_REDIRECTION)
    T = T U {(A_q,P_q)}
```

Figura 3 - Algoritmo do servidor. Onde m_q é a mensagem recebida de q, A_q é o endereço de q, P_q é o perfil de q, L é a lista e T é a lista de peers no tracker

```
do forever
  send(REQUEST_MEDIA) to s
  m_S ← receive(s)
  if (m_S == REDIRECT)
    n ← min(5, |L_S|)
    C = <pick n nodes form L_S>
    while (C ≠ VAZIO ^ NOT found)
      i = 0
      while (i < n)
        send(ESTIMATE_SPEED) to Ci
        b_i ← receive(C_i)
        if (b_max < b_i) b_max ← b_i

      if (b_max is enough to stream) found ← true
      else C = <pick n nodes form L_S>

  else if (m_S == RECEIVE_MEDIA)
    receiveMedia()
```

(a) Thread ativa

```
do forever
  m_q ← receive(*)
  if (m_q == ESTIMATE_SPEED)
    do for every r E CLIENTS
      l ← l + <download speed of r>
    send(l_P/2 - l) to q
  else if (m_q == REQUEST_MEDIA)
    CLIENTS ← CLIENTS U {q}
    send(CONFIRM_REDIRECTION) to s
    streamMedia(q)
```

(b) Thread passiva

Figura 4 – Algoritmos do cliente. s é o servidor, m_s é a mensagem recebida de s, Ls é a lista recebida de s, b_i é a banda disponível em i, l é a carga total do peer, l_P é a carga nominal do perfil P e CLIENTS é o conjunto de clientes conectados ao peer.

4.6 Gerenciamento da Árvore de Distribuição

Em WaveNet, conforme os clientes vão conectando à rede, uma árvore de distribuição vai sendo formada, tendo o servidor como raiz. Nessa árvore cada nó é um

cliente que retransmite a stream recebida para cada um de seus nós filhos. A quantidade de filhos que o nó pode ter, ou seja, seu grau de saída, é condicionado à banda que o nó possui.

O gerenciamento da árvore resume-se a administrar eficientemente as chegadas e saídas de nós, de forma que ao entrar na rede cada cliente seja redirecionado para o nó mais apropriado e que as saídas sejam mais transparentes quanto possível para os demais nós.

Quando um cliente deseja entrar no sistema, ele primeiro avisa o servidor. Essa informação é então repassada para o tracker que monta uma lista com os peers que ingressaram na rede mais recentemente, e que não façam parte de sua sub-árvore. Essa lista é então enviada para o cliente, que dentre os que estão na lista, escolhe aquele em melhores condições.

Quanto à saída de nós da árvore, um tratamento mais cuidadoso é necessário para evitar que os nós órfãos inundem o servidor com mensagens de solicitação de um novo nó pai. A saída pode ocorrer de duas formas: intencionalmente ou não. No primeiro caso, o nó que deseja sair informa sua intenção antes de deixar a rede. Já no segundo caso, o nó sai por algum defeito, e por isso, fica impossibilitado de avisar.

Na saída intencional, assim que o servidor recebe a notificação de que um nó vai sair, ele imediatamente seleciona um novo nó pai (ou uma lista de possíveis pais) e envia essa informação para o nó que está saindo, e este retransmite a mensagem para seus filhos.

Já na situação em que um dos nós falhar, o problema é tratado junto à perda de pacotes. A falha do nó corresponde a uma condição especial onde há uma perda de 100% dos pacotes. Sendo assim, quando a taxa de perda atingir um ponto inaceitável (ou estiver muito próximo disso), o nó contata seu pai para ver se este está tendo o mesmo problema. Se a resposta for positiva, a fonte do problema está acima dele e o melhor é deixar que os nós que estiverem acima resolvam o problema. Agora, se a resposta for negativa ou se ele não responder, o próprio nó envia uma mensagem para o servidor solicitando um novo pai.

4.7 Balanceamento Dinâmico de Carga

Para contornar eventuais problemas decorrentes de variações nas condições de recebimento e transmissão dos clientes na rede, mecanismos de monitoramento da velocidade de recepção dados devem ser providos. Isso permite que os próprios clientes possam buscar sempre maximizar as suas taxas de download, monitorando-as e solicitando ao tracker clientes em melhores condições de lhe servir conteúdo em tempo de execução.

Para o caso específico da transmissão sob-demanda essa troca de peer servidor ainda pode envolver a mudança de onda. Caso na mesma onda não haja clientes em condições de lhe servir conteúdo, pode-se deslocar o cliente para uma onda próxima. Se seus níveis de qualidade de serviço permitirem, ainda pode-se introduzir um atraso para o cliente esperar outra onda.

5. Simulação

Em nossas simulações, consideramos uma rede consistindo de uma grande coleção de nós com identificadores únicos e que comunicam entre si através de troca de mensagens. Tal rede é altamente dinâmica e novos nós podem entrar e sair a qualquer hora, tanto voluntariamente, quanto por falha. Uma vez que saídas voluntárias podem ser facilmente tratadas, vamos nos concentrar na discussão de como lidar com falhas.

Assumimos que os nós são conectados através de uma rede como a Internet, onde cada nó pode se comunicar com qualquer outro nó. A relação entre esses nós compõe a topologia da rede simulada.

Os nós são heterogêneos, ou seja, diferem em seu poder computacional, capacidade de armazenamento e de transmissão. Essas características interferem na escolha do nó que deve servir conteúdo para os demais. Cada nó conhece esses seus parâmetros.

Para executar nossos experimentos, usamos uma versão modificada do PeerSim [26], um simulador de redes peer-to-peer que oferece simulação de ambientes de grande escalabilidade e dinamismo.

5.1 Metodologia

Foram executados 20 experimentos independentes. Salvo quando dito o contrário, nesses experimentos os parâmetros eram fixos. Para o tamanho da rede foi escolhido 500 nós, sendo que cada um deles foi configurado aleatoriamente como tendo conexão de 110 Kbps, 256 Kbps, 512 Kbps ou 1Mbps.

Os resultados são expressos em ciclos, sendo que o δ, que representa a duração de cada um deles, define a complexidade de convergência. A escolha de um d resulta em uma rápida convergência, porém com um alto custo de comunicação por unidade de tempo. Caso com um pequeno número de ciclos seja possível obter uma configuração otimizada, valores altos para o d são permitidos.

Como configuração inicial, nós configuramos um servidor, sem clientes conectados, com capacidade para atender seus clientes a uma taxa de 100 Mbps e então começamos a adicionar os peers. Assumimos que os

peers chegam a uma taxa A, fixa, e que cada requisição leva um tempo t para ser processada.

5.2 Resultados

A primeira característica a ser testada é o número de clientes podem ser atendidos com o nosso sistema de redirecionamento. Em média, 210 peers podem ser atendidos pelo servidor, e outros 290 podem ser servidos por clientes. Os dados desse teste podem ser vistos no gráfico da figura 5, onde a parte de baixo de cada coluna indica o número de clientes atendidos pelo servidor e a parte de cima indica o número de clientes atendidos por outros clientes.

O tamanho do ciclo foi configurado como o tempo entre duas chegadas consecutivas mais o tempo gasto para processar a requisição, ou seja, $\delta = (1/A) + t$. Assim, com esse mesmo teste podemos constatar o tempo que o servidor consegue operar normalmente até ser sobrecarregado e os peers que forem chegando começarem a ser redirecionados. Temos que, em média, o servidor foi sobrecarregado após 210 * δ.

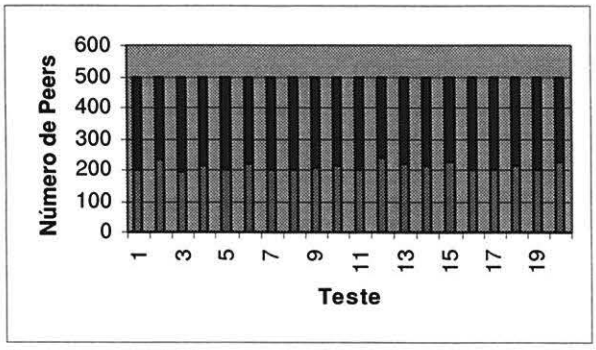

Figura 5 – Número de Peers atendidos pelo Servidor e pelo Cliente

Esse mesmo teste foi repetido para um número maior de clientes. Escolhendo 1000 entradas consecutivas, tivemos uma média de 683,6 peers atendidos por clientes e quase 213 peers conectados diretamente ao servidor. Os demais tiveram serviço negado e não puderam ser acomodados pelo sistema, mesmo após 2000 ciclos de execução.

6. Estudo de Caso

Por proporcionar um ambiente mais interessante para a aplicação do modelo, o estudo ficou concentrado na transmissão de conteúdo ao vivo. A aplicação desenvolvida é composta de um módulo cliente que solicita o recebimento de mídia contínua para um servidor com capacidade para redirecionamento de requisições para outros clientes.

6.1 Servidor

O servidor, desenvolvido em linguagem Java, é alimentado com arquivos MPEG2-TS [23], criados com o software VLC [24]. Esse tipo de arquivo é um formato de multiplexação de pacotes de sinais de áudio e vídeo, preparado para ser transportado em redes, carregando informações úteis em ambientes onde há ruído ou perda de pacotes. A transmissão da mídia é iniciada quando o primeiro cliente conecta-se a rede e encerrada no momento em que a rotina de envio atingir o fim do arquivo. Nesse momento, um sinal de fim de stream é enviado.

6.2 Cliente

Assim como o servidor, o cliente também foi desenvolvido em plataforma Java, sendo que a reprodução das streams ficou a cargo da Java Media Framework API (JMF) [25]. Como as classes presentes nessa API não são capazes de manipular streams corretamente, foram implementados um novo PullDataSource e um novo PullSourceStream com buffer circular de 512KB para manipular a mídia da forma desejada.

Seguindo o modelo proposto, quando o cliente deseja receber a mídia, ele solicita ao servidor e caso o mesmo não possa atender, uma lista de possíveis fontes é recebida e o peer é escolhido de acordo com as estratégias descritas anteriormente.

Tão logo a conexão seja estabelecida, um processo de preenchimento do buffer (buffering) tem início. O cliente segue copiando os dados até que o tempo para preencher o restante do buffer seja menor que o tempo para reproduzir o conteúdo já obtido. Por exemplo, caso o vídeo tenha uma codificação de 128 Kbps e o cliente possua uma conexão de 112 Kbps, conhecendo o tamanho do buffer, com um simples cálculo é possível saber que a exibição do conteúdo não deve ser interrompida se o tempo de espera for de pelo menos 19,6 s. Esse valor é recalculado a cada segundo para contornar flutuações na banda. Consideramos que caso o cliente não possua conexão rápida, interrupções na transmissão são toleráveis. Caso o cliente tenha mais banda que a taxa de codificação, a exibição começa imediatamente após receber os primeiros bytes.

Em cada conexão, há um tempo máximo de espera por dados. Se o cliente permanecer 10 segundos sem receber qualquer dado de sua fonte, ele deve procurar um outro peer na lista que possa atendê-lo. A lista também deve ser atualizada regularmente. Seja N o número de endereços recebidos na lista anterior, caso

já tenha se passado mais de 6 * N segundos, uma nova lista atualizada é solicitada ao servidor.

7. Conclusões e Trabalhos Futuros

Neste artigo apresentamos o WaveNet como uma solução para a redirecionamento de carga na comunicação de streams.

Pelos resultados dos testes simulados é possível concluir que o modelo proposto faz com que um número maior de clientes seja atendido, mas há limites. Como os clientes não são originalmente criados com o intuito de servir conteúdo, a capacidade deles é limitada e dependendo da configuração da rede que se formar pode não haver muitos clientes com largura de banda em certos momentos.

A implementação de um sistema de transmissão ao vivo de streams de vídeo MPEG que utiliza o modelo proposto foi concluída e alguns experimentos já puderam ser feitos.

Devido às limitações impostas pelo ambiente de trabalho, no que diz respeito ao número de computadores disponíveis para serem utilizados como clientes, não foi possível exaurir o sistema da mesma forma que na simulação. Contudo, isso é uma limitação bem conhecida dos testes experimentais, como discutido nesse trabalho. Com os resultados obtidos nos testes é possível notar algumas vantagens do modelo proposto em relação ao tradicional cliente-servidor.

Dentre os trabalhos futuros estão experimentação e simulação do modelo proposto em um ambiente de grande escala na transmissão sob-demanda.

8. Referências

[1] Content Delivery Network, Article, Wikipedia, 2005. http://en.wikipedia.org/wiki/Content_Delivery_Network.

[2] CDN (Content Delivery Network), Pc-news.com, 2005. http://www.pc-news.com/detalle.asp?sid=&id=44&Ida=750.

[3] H. Deshpande, M. Bawa e H. Garcia-Molina. "Streaming Live Media over a Peer-to-Peer Network", Technical Report, Stanford University, Stanford, CA, EUA, Agosto/2001. http://dbpubs.stanford.edu:8090/pub/2001-31

[4] M. A. Vasconcelos, V. Almeida e S. Campos, "Análise da Distribuição de Streaming Media em Arquiteturas do Tipo Peer-to-Peer", VI Semana de Pós-Graduação em Ciência da Computação, Setembro/2002.

[5] D. A. Tran, K. A. Hua e T. T Do, "A Peer-to-Peer Architecture for Media Streaming", IEEE Journal of Selected Areas in Communications, V. 22, Nº1, Jan/2004, p. 121-133.

[6] L. B. Pinho, "Implementação e Avaliação de um Sistema de Vídeo Sob-demanda Baseado em Cache de Video Cooperativa", Universidade Federal do Rio de Janeiro, COPPE, 2002, 71 p., Dissertação de Mestrado.

[7] T. T. Do, K. A. Hua, M. Tantaoui, "P2VoD: Providing Fault Tolerant Video-on-Demand Streaming in Peer-to-Peer Environment". IEEE International Conference on Communications, Paris, Junho/2004.

[8] V. N. Padmanabhan e K. Sripanidkulchai, "The Case for Cooperative Networking", Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02). MIT Faculty Club, Cambridge, MA, USA, Fevereiro/2002.

[9] V. N. Padmanabhan, H. J. Wang, P. A. Chou, e K. Sripanidkulchai, "Distributing Streaming Media Content Using Cooperative Networking", ACM/IEEE NOSSDAV, Miami, FL, EUA, Maio/2002.

[10] X. Jiang, Y. Dong e D. Xu, "GnuStream: A MPEG Video Streaming System Over Peer-to-Peer Network", CS590N P2P Networks and Services, Dezembro/2002.

[11] J. Ritter. "Why Gnutella can't scale. No, really.", Fev/2001, http://www.darkridge.com/jpr5/doc/gnutella.html.

[12] Napster.com, 2005, http://www.napster.com/.

[13] Gnutella.com, 2005, http://www.gnutella.com/.

[14] B. Cohen, "Incentives build robustness in BitTorrent", Proceedings of the 1st Workshop on the Economics of Peer-to-Peer Systems, Berkeley, CA, Junho/2003.

[15] B. Krishnamurthy and J. Wang. "On Network-Aware Clustering of Web Clients", ACM SIGCOMM, Agosto/2001.

[16] Z. Xiang, Q. Zhang, W. Zhu, Z. Zhang e Y. Zhang, "Peer-to-Peer Based Multimedia Distribution Service", IEEE Transactions on Multimedia, Vol. 6, No. 2, Abril/2004.

[17] D. S. Bernstein, Z. Feng, B. N. Levine, S. Zilberstein, "Adaptive peer selection". Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03), Berkeley, CA, EUA, Fevereiro/2003.

[18] K. Lai, M. Baker, "Measuring link bandwidths using a deterministic model of packet delay". Proceedings of ACM SIGCOMM 2000. Agosto/2000.

[19] J. Strauss, D. Katabi, and F. Kaashoek. A Measurement Study of Available Bandwidth Estimation Tools. Proceedings of ACM Internet Measurement Conference, Miami, Florida, USA, Outubro/2003.

[20] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, B. Hendricks, R. Baraniuk, "Multifractal Cross-traffic Estimation". ITC Conference on IP Traffic, Modeling and Management. 2000.

[21] M. Jain, C. Dovrolis, "End-to-end Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput". Proceedings of ACM SIGCOMM. 2002.

[22] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, L. Cottrell, "pathChirp: Efficient Available Bandwidth Estimation for Network Paths". Proceedings of PAM Workshop. 2003.

[23] ISO. "ISO/IEC 13818-1 - Information Technology – Generic Coding of Moving Pictures and Associated Audio Information – Part 1: Systems (MPEG-2 Systems)", 1996.

[24] VideoLan, 2005, http://www.videolan.org.

[25] Java Media Framework API (JMF), Sun Microsystems, 2005, http://java.sun.com/products/java-media/jmf/index.jsp.

[26]Peersim Simulator, 2005, http://peersim.sourceforge.net/.