Um Serviço de Distribuição de Vídeo Sob Demanda Baseado em uma Rede de Servidores D-VoD

Resumo. *Este artigo apresenta a arquitetura de um serviço de distribuição de vídeo e a implementação de servidores que dão suporte a arquitetura proposta. O serviço foi concebido para distribuir vídeo em rede de vídeo digital composta por servidores (equipamentos e software) interligados por uma rede IP. O serviço foi testado no backbone da RNP, mostrando que é viável distribuir vídeo de alta qualidade em ambientes com restrição de banda passante.*

1. Introdução

Este trabalho apresenta a arquitetura de um serviço de distribuição de vídeo baseado em uma rede de servidores D-VoD implementados e testados pela equipe do GTVD-RNP. O Grupo de Trabalho de Vídeo Digital (GTVD) é uma iniciativa da RNP que visa induzir o desenvolvimento de uma nova geração de aplicações de vídeo digital para explorar o potencial de redes de alta velocidade no país. Como aplicação piloto foi proposta uma infra-estrutura baseada na RNP para implantar um serviço de vídeo sob demanda com suporte a replicação de servidores e otimização da utilização dos recursos de rede.

O presente artigo está estruturado da seguinte forma. Na seção 2 será apresentada a arquitetura proposta para o serviço. A seção 3 lista alguns trabalhos relacionados. As seções 4, 5 e 6 descrevem respectivamente a arquitetura, aspectos de implementação e resultados de testes realizados com o servidor D-VoD. Por fim, a seção 7 traz as conclusões e perspectivas futuras do trabalho.

2. Arquitetura do Serviço de Vídeo Sob Demanda

A arquitetura proposta para o serviço de distribuição de vídeo baseia-se em uma estrutura de múltiplos níveis de distribuição: servidor-proxy, proxy-proxy e proxy-cliente. Vale ressaltar que podem existir diversos níveis de distribuição proxy-proxy. A Figura 1 ilustra a segunda geração da arquitetura de distribuição de vídeo.

Os servidores de vídeo compõem o repositório permanente de vídeo digital. A idéia é que as instituições colaboradoras (provedoras de conteúdo) responsáveis pelas aplicações usuárias do serviço, mantenham seus próprios servidores, que armazenarão apenas os vídeos disponibilizados pela respectiva instituição no serviço. Opcionalmente, para incrementar os requisitos de tolerância a falhas, uma instituição poderá utilizar diversos servidores de vídeo, onde cada vídeo poderá ser armazenado em um subconjunto qualquer destes servidores. Ou seja, a arquitetura é baseada em um esquema de replicação parcial, onde os servidores não precisam estar sincronizados em todo o seu conteúdo, mas apenas nas réplicas de vídeo que coexistem simultaneamente em alguns deles.

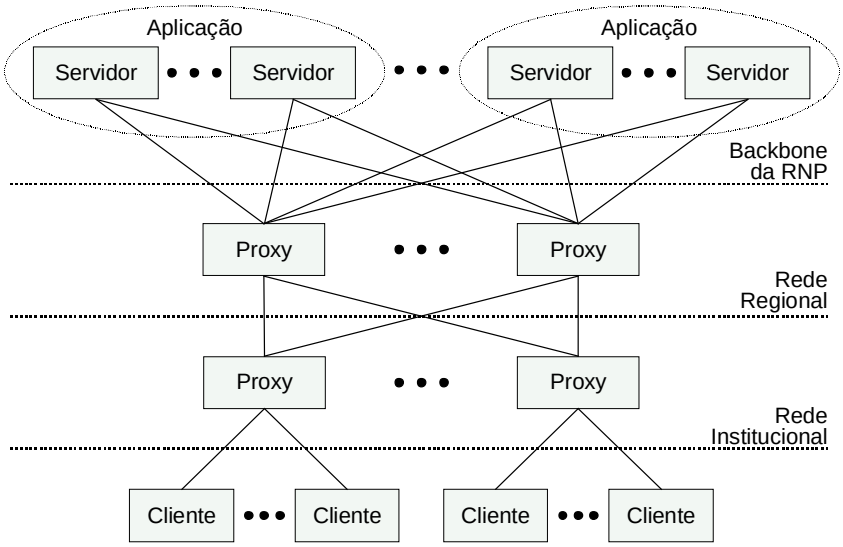

Figura 1 – Arquitetura de Distribuição do GTVD

Um proxy também implementa uma cache para aqueles vídeos de maior audiência em sua área de atendimento, definida pela proximidade dos usuários. A arquitetura permite a conexão em cascata de proxies, configurando assim uma hierarquia de múltiplos níveis de cache. A principal vantagem desta nova abordagem é a distribuição otimizada do tráfego no backbone, redes regionais, redes institucionais e até mesmo nas redes locais.

Embora os servidores e proxies que compõem a segunda geração da arquitetura de distribuição possuam papéis distintos e bem definidos, os mesmos possuem em comum o fato de recuperarem o vídeo a partir de uma determinada fonte e distribuírem o mesmo para um determinado destino. No caso do servidor de vídeo, a fonte e destino são o sistema de arquivos e algum proxy, respectivamente. No caso do proxy, a fonte pode ser um servidor de vídeo ou outro proxy, enquanto o destino pode ser outro proxy ou o cliente. Com base nesta observação, percebeu-se que as funções desempenhadas pelos servidores e proxies diferem apenas nos tipos de fontes e destinos, enquanto o controle de fluxo entre a fonte e o destino é basicamente o mesmo.

Para satisfazer aos requisitos da arquitetura proposta, foi realizada uma implementação modular composta de diversos componentes para tratar a diversidade de fontes e destinos de vídeos, e um único componente de controle. Em tempo de configuração, ou até mesmo em tempo de execução, é possível acoplar novos tipos de fontes e destinos, viabilizando assim uma implementação única para servidores e proxies de vídeo. A figura 2 ilustra a arquitetura de software com a proposta de integração.

No caso de um servidor de vídeo, o componente de controle deverá utilizar como fonte o componente baseado em arquivo. Uma vez que um determinado servidor pode distribuir diversos vídeos simultaneamente, cada um destes pode ser enviado a um proxy utilizando qualquer componente de destino (HTTP, UDP, RTSP/RTP, etc).

No caso do proxy, o componente de controle deverá ser configurado para utilizar como fonte o componente baseado em cache, que por sua vez está recebendo blocos de vídeo de um servidor primário ou outro proxy. Assim, qualquer componente de destino (HTTP, UDP, RTSP/RTP, etc) pode distribuir blocos de vídeo para um componente cache. Considerando que um proxy pode distribuir diversos vídeos para outros proxies e

clientes de forma simultânea, cada um destes vídeos poderá ser enviando utilizando qualquer componente de destino.

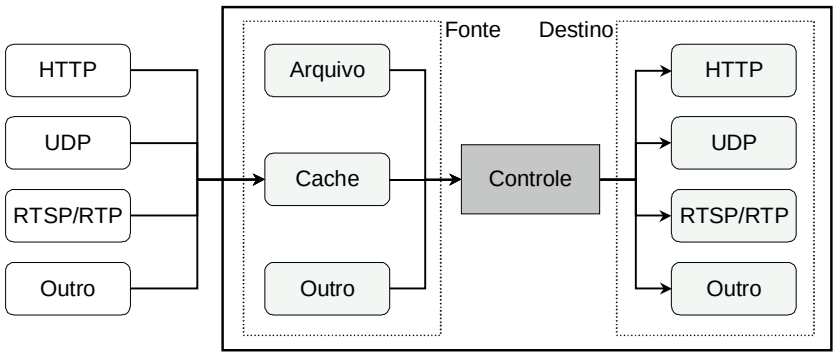

Figura 2 – Arquitetura Integrada do Servidor e Proxy

Vale ressaltar que esta arquitetura modular permite uma única entidade atuar simultaneamente como servidor de vídeo e proxy, bastando para tal utilizar os componentes baseados em arquivo e cache. Neste caso, para um determinado subconjunto de vídeos digitais, a entidade será vista como um servidor de vídeo que armazena de forma permanente réplicas dos mesmos. Por outro lado, para os demais vídeos distribuídos pelo serviço, a entidade apenas manterá aqueles de maior audiência em uma cache local.

A Figura 3 ilustra a seqüência de ações executada quando um cliente acessa o serviço. Inicialmente o cliente acessa via Internet uma aplicação de busca, que consulta um repositório de metadados dos vídeos disponibilizados pela instituição responsável. Os metadados são todas as informações sobre os vídeos, adicionado dos endereços dos servidores fonte. Esta característica é importante, pois torna o serviço proposto pelo GTVD independente de um servidor de vídeo específico e de uma aplicação de busca específica. Toda aplicação de busca deverá retornar os seguintes elementos: o identificador do vídeo desejado, os endereços dos servidores fonte do vídeo, e o endereço do serviço de gerenciamento.

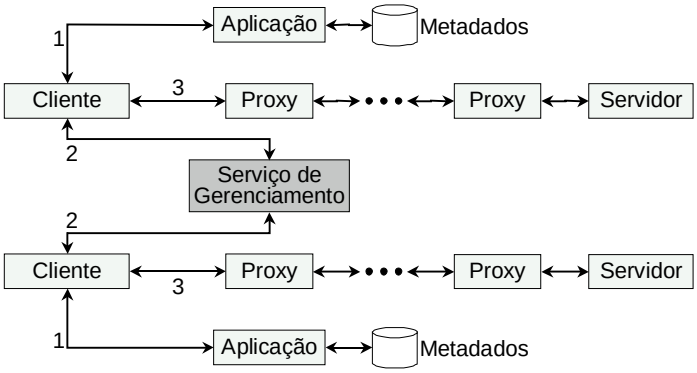

Figura 3 – Seqüência de Acesso no GTVD Fase II

Uma vez recuperado o identificador do vídeo, o cliente então envia uma requisição ao serviço de gerenciamento, informando o identificador do vídeo desejado e os endereços dos servidores fonte daquele vídeo. O serviço de gerenciamento é responsável por

selecionar um dos servidores disponíveis, juntamente com a cadeia de proxies, que serão utilizados para atender a solicitação do cliente. Como resultado, o serviço de gerenciamento retorna a melhor rota de distribuição (servidor e proxies) a ser utilizada para distribuir o vídeo até o cliente.

Em seguida, o cliente envia uma requisição do vídeo desejado ao primeiro proxy da rota de distribuição. Por sua vez, o proxy requisitado solicita o vídeo ao próximo proxy da rota de distribuição, até que a requisição chega ao servidor fonte do vídeo. Neste momento, a distribuição é iniciada e o cliente começa a receber o fluxo de vídeo. Na verdade, quando parte do vídeo desejado encontra-se armazenado na cache de algum proxy intermediário, o cliente pode começar a receber o fluxo antes que a requisição atinja o servidor fonte. Caso o vídeo esteja totalmente armazenado na cache de algum proxy intermediário, a requisição se propaga apenas até este proxy.

Com o objetivo de otimizar o tráfego no backbone da RNP, redes regionais e redes institucionais, a arquitetura de distribuição do GTVD suporta múltiplos níveis, permitindo o fluxo de vídeo ser distribuído entre uma cadeia de proxies (Figura 3). Neste caso, o serviço de gerenciamento deve possuir a noção de localização dos diversos servidores fonte e proxies, de modo que uma rota de distribuição otimizada possa ser estabelecida. A solução implementada para a definição da rota de proxies explora o conceito de localização dos blocos de endereços alocados na RNP. Neste caso, com base na distribuição dos blocos de endereços na RNP (bloco x localização) é possível representar a localização destes blocos, servidores fonte e proxies em um grafo de conectividade. Baseado em algoritmos de determinação do melhor caminho entre dois pontos, é viável calcular a melhor rota de distribuição entre um determinado cliente e os servidores fonte de um vídeo específico.

Para criar este grafo de conectividade, a equipe do GTVD especificou um esquema XML capaz de representar nós da rede, conexões entre estes nós, associação de blocos aos nós, e distribuição de servidores e proxies nestes diversos nós. A partir da especificação da rede em XML, o serviço de gerenciamento gera um grafo e aplica algoritmos de determinação de melhor caminho para calcular a melhor rota de distribuição entre um cliente e um servidor fonte de um determinado vídeo desejado pelo usuário.

3.Trabalhos Relacionados

Existem atualmente uma grande variedade de sistemas disponíveis capazes de transmitir vídeos sob demanda pela internet. A seguir serão apresentados exemplos desses sistemas, assim como suas principais características.

O *Windows Media Services[2]* da *Microsoft* torna possível a distribuição de vídeos sob demanda codificados no seu formato privado (ASF) os seguintes formatos: BMP, WAV, WMA, WMV, AVI, ASF e MPEG-1, podendo transmiti-los utilizando os seguintes protocolos: UDP, TCP, HTTP / TCP e IP Multicast. O *Windows Media Services* possui recursos como proteção e confidencialidade dos dados, proteção dos direitos autorais e interatividade com o cliente. Porém é uma solução comercial fechada, o que implica que alguns dos aplicativos e protocolos envolvidos no processo de distribuição devem seguir a mesma solução privada. Outras desvantagens desse componente incluem sua

arquitetura centralizada e estática, o que impossibilita uma flexibilidade na distribuição de vídeos, além de seu custo bastante elevado.

Helix Universal Server [3] é o servidor de vídeo sob demanda da Real Networks. Ele transmite através do protocolo RTSP dados codificados nos formatos Real Media, Windows Media, QuickTime e MPEG-4 usando os protocolos de transporte UDP, TCP, IP Multicast e RTP, alem de HTTP / TCP. O *Helix Universal Server* pode ser configurado em uma arquitetura distribuída (através do uso de outro pacote, o *Helix Universal Gateway*) permitindo a configuração de em uma rede de servidores. Como o custo da solução é função do número de servidores e do número de clientes suportado por servidor, para aplicações com muita demanda o custo das licenças atinge valores elevados. A solução não inclui suporte para distribuição automatizada dos clientes entre os servidores.

Vídeo LAN Server[3] é um dos melhores softwares livres de vídeo sob demanda disponíveis. Ele é capaz de transmitir vídeos capturados de DVD, Satélite, TV Digital, ou codificadores MPEG-1, MPEG-2 e MPEG-4. Transmite vídeos em IP multicast, TCP e UDP em IPv6 e IPv4, também em HTTP, mas com a ajuda do servidor Apache. O grupo Vídeo LAN também produziu um exibidor de vídeos para seu servidor, o *Vídeo LAN Client*. Sua arquitetura é elegante, o exibidor *VLC* permite que o cliente também sirva vídeos para outros clientes. Sua licença GNU o torna uma solução de baixo custo. No entanto sua implementação não prevê recursos de configuração dinâmica ou otimização na distribuição dos vídeos.

ALMADEM-VoD [6] é um bom exemplo de servidor de vídeo sob demanda na área acadêmica brasileira. Ele faz parte do projeto ALMADEM, e é desenvolvido pela UFMG, com a parceria de diversas outras universidades brasileiras e estrangeiras. Sua principal característica é usar apenas componentes de baixo custo, que faz com que seja um servidor com poucos requisitos de hardware. Com a utilização de QoS ele permite a utilização ótima dos recursos de rede [12]. Sua arquitetura é centralizada e não prevê o uso distribuído de recursos da rede, nem configuração dinâmica. Sua modelagem é inflexível e usa seu próprio protocolo de controle de fluxo de dados, o que requer a utilização de um cliente especial desenvolvido pelo mesmo grupo para captura e exibição dos dados do servidor.

4. Arquitetura do Servidor D-VoD

A arquitetura do D-VoD, como ilustrado na Figura 4, consiste basicamente em um pipe de 3 camadas. A primeira camada, "*Sources*", é responsável por adquirir dados usando algum tipo de protocolo de rede ou lendo diretamente de um dispositivo local de entrada de dados.

Do outro lado do pipe encontram-se os *Modules*, responsáveis pela interface do servidor com o cliente, isto é, ler dados das fontes e enviá-los a um cliente usando um protocolo implementado por ambos.

Entre o Source e os Modules se encontra a camada de controle do D-VoD, que é responsável por todas as ações de controle, como a associação de fontes aos módulos e o gerenciamento dessas fontes. Dentro dessa camada temos um módulo especial, o *Manager*, que permite o controle dinâmico automático pelo gerente do serviço VOD, ou

manual através da interface gráfica externa que se conecta a ele, o XD-VoD. Dentro dessa camada encontra-se também o sistema de LOG, que é responsável pelo registro de todas as ações do servidor e pelo recolhimento dos dados para o calculo das estatísticas do servidor.

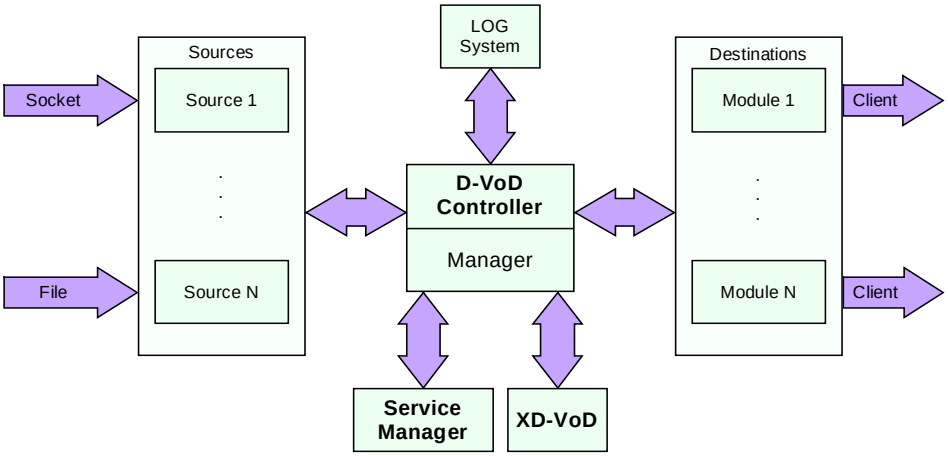

Figura 4. Arquitetura do D-VoD.

Esta arquitetura é suficientemente flexível para atender os requisitos do serviço VOD. Ela abstrai a fonte de dados adotada para cada cliente e possibilita que servidor possua quantas fontes diferentes quantas forem necessárias. Essa característica permite ao D-VoD poder ser configurado como *Source Server*, ou um *Proxy Server*, dinamicamente, dependendo de cada cliente.

4.1. Modelagem da Arquitetura do D-VoD

Ao se analisar a especificação da arquitetura do D-VoD, imediatamente vê-se a necessidade de uma modelagem orientada a objeto. A Figura 55 mostra a modelagem dessa arquitetura, apresentada através da notação UML (*Unified Modeling Language*).

Como na especificação da arquitetura (Figura 5), o modelo divide-se em 3 camadas. A camada de controle tem como principal componente a classe "*D-VoD Controller*", que é responsável pelo gerenciamento dos módulos, das fontes e de suas configurações. Seus principais atributos são uma lista de fontes "*TableOfSources*" e uma lista de módulos "*TableOfModules*", esses atributos são administrados pela classe *Administrator*.

Dentro dessa camada também se encontra o módulo *Manager*. Este módulo especial tem como principal função ser um "controle remoto" da classe *Administrator*. Ele pode se conectar ao gerente do serviço VOD, ou a sua interface gráfica (XD-VoD), e receber comandos para criar ou remover fontes ou tanto habilitar e desabilitar os módulos como mudar suas configurações. Também é função do módulo *Manager* divulgar as estatísticas colhidas pela classe *LOG* através de um documento XML. Então o módulo *Manager* é, também, um servidor HTTP para receber tais comandos ou divulgar suas estatísticas.

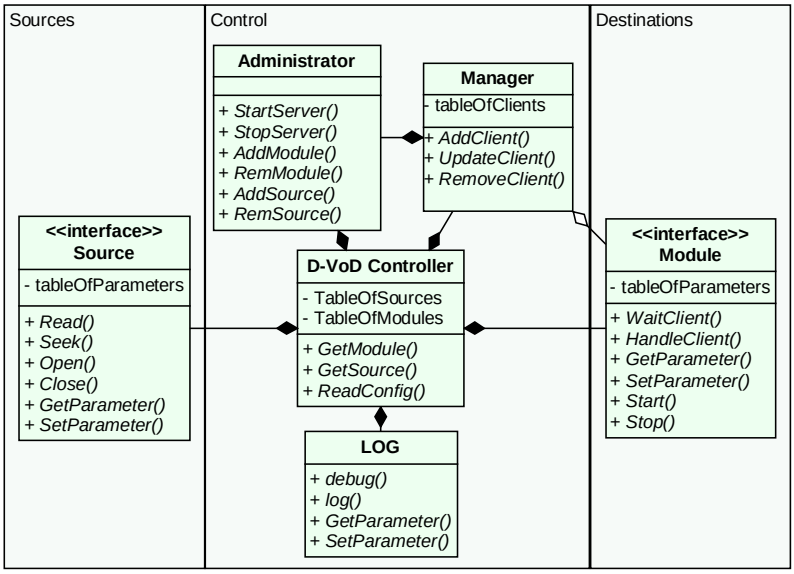

Figura 5. Modelagem da arquitetura do D-VoD.

Nas outras camadas, temos as classes de fachada *Source* e *Module*. Cada fonte de dados do D-VoD irá herdar a classe *Source* e implementar seus métodos virtuais *Read(), Open(), Seek()* e *Close()*, caso necessário. Note que certos tipos de fontes não necessitam implementar todos os métodos, por exemplo, uma fonte de dados que lê de um dispositivo de vídeo em tempo real não irá precisar do método *Seek()*.

Cada tipo de destino diferente (i.e. protocolo de transmissão diferente) precisará herdar a classe *Module* e implementar todos os seus métodos virtuais. Cada módulo é uma coleção de classes que implementam um ou mais protocolos de transferência de dados.

5. Implementação do D-VoD

A arquitetura do D-VoD foi implementada usando-se C++ e pode ser compilada para vários sistemas operacionais, porém atualmente ele está sendo compilado e distribuído apenas para o sistema operacional Linux. O D-VoD faz uso de algumas ferramentas livres, como a LibXML [9] e um coletor de lixo para C++, o *Boehm-Demers-Weiser Conservative Garbage Collector* [11].

A implementação segue o modelo proposto e consiste primeiramente da camada de controle. A classe *Controller* é implementada para, ao ser iniciada, fazer uma análise em um arquivo XML para então obter os parâmetros e montar suas listas internas de módulos e fontes através do método *ReadConfig()*. Em seguida solicita que o objeto do *Administrator* instanciado inicie os seus módulos pelo método *StartServer()*.

Para iniciar os módulos, o objeto *Administrator* lê a tabela de módulos *TableOfModules* do *Controller* e cria uma *thread* para cada entrada dessa tabela, que fica bloqueada executando o método virtual *WaitClient()* implementado por cada tipo de módulo. Esse método basicamente irá esperar um cliente e iniciar uma thread para tratar sua requisição pelo método *HandleClient()*, e este por sua vez pode solicitar dados a qualquer fonte de dados presente na tabela de fontes pela chamada do método

GetSource() do *Controller* e pelos métodos virtuais *Open()*, *Read()* e *Close()* implementados especificamente por cada tipo de fonte.

5.1. Módulos

Os módulos no D-VoD são a interface com os clientes e implementam o protocolo de transmissão de dados usado para envio dos vídeos. Sua principal função é ler dados de uma fonte qualquer, que é abstraída pelo *Controller*, para então os enviar usando o protocolo por ele implementado.

A escolha dos protocolos implementados no D-VoD baseou-se no requisito compatibilidade com os exibidores de vídeos disponíveis no mercado. Foi implementado o protocolo http (*Hyper-Text Transfer Protocol*), que é o protocolo de transferência de dados mais utilizado na *web*. Foi também implementado um protocolo mais específico para a transmissão de dados em tempo real, ou *Streaming* de dados, o RTSP ou "*Real Time Streaming Protocol*". A modelagem UML na qual se baseou a implementação desses protocolos é mostrada na Figura6.

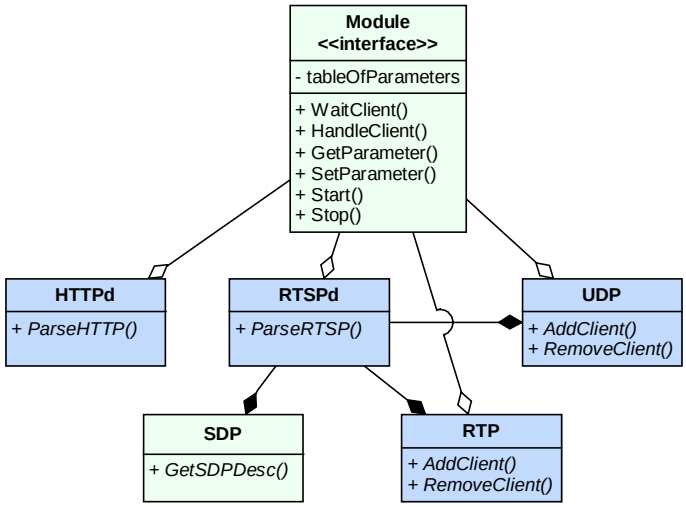

Figura 6. Implementação dos módulos.

Para transmissão do fluxo de dados utilizando-se o protocolo HTTP, o método *WaitClient()* do módulo *HTTPd* foi implementado de forma a aguardar em um *socket* [10] TCP um *handshake* HTTP ser realizado com algum cliente. Durante este *handshake*, o método *ParseHTTP()* é chamado para se avaliar a requisição do cliente e decidir qual fonte de dados será usada para o cliente, assim o módulo informa ao cliente que o conteúdo não deve ser mantido em *cache*, através dos campos de cabeçalho HTTP: "Pragma: no-cache" e "Cache-Control: no-cache". O formato do fluxo é então avaliado e informado através do cabeçalho "Content-Type", e o seu tamanho em bytes é informado no cabeçalho "Content-Length". Depois de adicionado a tabela de clientes, o cliente começa a receber então o fluxo de dados proveniente da fonte selecionada.

O módulo RTSPd implementa o protocolo RTSP. Este protocolo não dá suporte a transferência dos dados propriamente ditos, mas trata da negociação dos parâmetros de controle do fluxo de dados com o cliente e do controle do fluxo de dados que são

transferidos através de outros módulos usando os protocolos de transporte UDP ou RTP.

O RTSP também faz uso do protocolo SDP ou *Session Description Protocol* que é usado para informar ao cliente metadados do acervo de vídeo oferecido pela fonte de dados. Assim, os clientes podem receber informações técnicas acerca dos dados, ou obter uma grade de programação de vídeos para serem escolhidos.

Os módulos UDP e RTP são implementados de forma a possibilitar a inserção e remoção manual de clientes através de outros módulos, como o RTSPd ou o *Manager*. Quando há a inserção de um cliente da tabela de clientes, um *socket* UDP é criado e os dados são enviados através dele ou enviados na forma de pacotes RTP no módulo RTP.

5.2. Fontes de dados

Nessa seção são apresentadas as fontes de dados que foram ou estão sendo implementadas no D-VoD para o serviço de VOD do GTVD. A Figura7 ilustram tais fontes, assim como elas se encaixam no modelo da arquitetura proposta na Figura 4.

As fontes presentes na Figura7 são necessárias para tornar o modelo do D-VoD compatível com os requisitos arquitetura do serviço VOD. A fonte *File* lê vídeos que estão no disco local, enquanto a fonte *Cache* dá ao modelo a sua característica distribuída permitindo que sejam lidos dados de fluxos HTTP ou UDP provenientes dos módulos HTTPd e UDP de outro D-VOD ou de qualquer outro servidor que suporte tais protocolos.

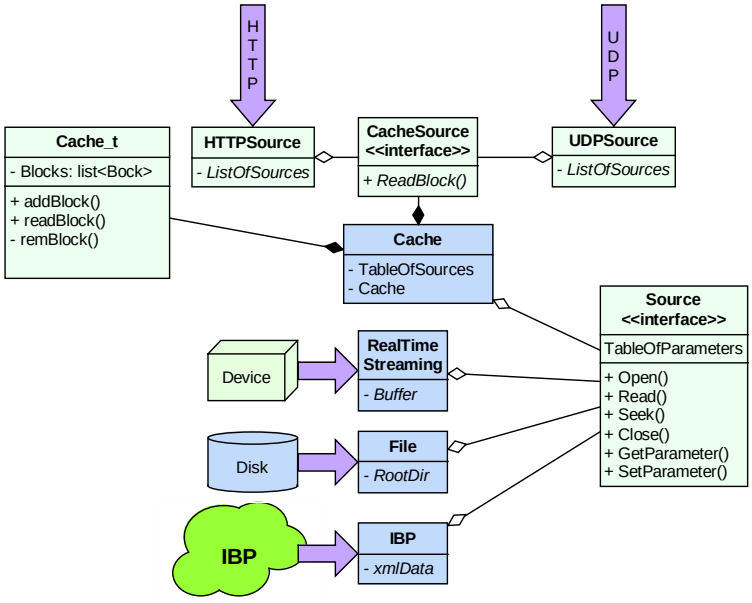

Figura 7. Implementação das fontes de dados.

A fonte de dados *File* foi implementada para ler dados do sistema de arquivos do sistema operacional. Pelo fato dessa fonte ter acesso direto aos dados localmente, todo servidor D-VoD que está configurado para usar este tipo de fonte será um servidor *source* do serviço VOD. Sua implementação é simples e consiste apenas na tradução dos parâmetros dos métodos virtuais da classe de fachada *Source* (*Open()*, *Read()*, *Seek()* e

Close()), para os das chamadas de sistema de função correspondente para ler arquivos do sistema de arquivo que estão localizados no diretório raiz identificado pelo atributo *RootDir*.

A fonte de dados *Real Time Streaming* por enquanto foi apenas modelada e está sendo implementada. Ela tem como principal característica acessar os dados diretamente de um buffer que estará sendo alimentado a uma taxa constante com dados provenientes de um dispositivo de captura de vídeo em tempo real. Esta fonte tem acesso direto aos dados na sua memória local, portanto um servidor executando essa fonte também será um servidor *source* do serviço VOD.

A fonte de dados IBP também está apenas modelada. Quando implementada, ela tornará possível a integração do serviço de VOD do GTVD com o protocolo de distribuição de armazenamento de dados *Internet Backplane Protocol* [7,8]. Esta fonte está sendo implementanda em uma parceria com o Logistical Computing and Internetworking Laboratory (LoCI) da Universidade do Tennessee. O IBP é suportado por um conjunto de servidores, conhecido como o Logistical Backbone (L-Bone), que são usados por centenas de pesquisadores e usuários para armazenamento de dados digitais (em qualquer formato), especialmente arquivos grandes (acima de 1GB). Atualmente, o L-Bone contém cerca de 253 servidores IBP, instalados em aproximadamente 20 paises. Este sistema de arquivos virtual possui uma capacidade aproximada de 22TB.

5.2.1. Fonte *Cache*

Para suprir a necessidade de distribuição dos vídeos na rede, tornou-se necessário o projeto e implementação de servidores atuando como *proxy*, cuja principal característica é a presença de uma *cache* de armazenamento temporário. Nesses servidores os vídeos são guardados apenas temporariamente nessa *cache* em disco e, quando o vídeo solicitado pelo cliente a um servidor *proxy* não existe na sua *cache* local, a própria *cache* é responsável por buscar este vídeo de servidores *source* ou mesmo de outro servidores *proxy*.

Como esses servidores *proxy* geralmente são máquinas menos potentes que os servidores *source*, o espaço em disco para armazenamento dos vídeos é limitado. É imperativo, portanto, que se utilize uma política de gestão de *cache* para armazenar novos vídeos.

A política adotada no D-VoD para a *cache* faz uso do particionamento do vídeo em blocos, de tal forma que, sempre que for necessário liberar espaço, o bloco removido será o último bloco presente na *cache* do vídeo que foi acessado a mais tempo. Sempre que um cliente solicita um vídeo ao servidor *proxy*, a data de acesso deste vídeo é atualizada. Se o vídeo não estiver presente na *cache* e esta estiver cheia, isto é, o limite de espaço em disco para a *cache* foi alcançado, um bloco deve ser removido para dar lugar ao novo bloco que será trazido de um servidor *source* para a *cache*. Esta, por sua vez, guarda a informação da última data de acesso de todos os vídeos de tal forma que a busca pelo vídeo que foi acessado há mais tempo é linear. Uma vez determinado qual vídeo deve ter um bloco removido, é realizada uma nova busca para se determinar qual o último bloco deste vídeo que está presente na *cache*. Note que, quando dizemos o último

bloco de um vídeo estamos nos referenciando ao bloco mais distante do início do vídeo, de tal forma que se tivermos os blocos 1 a 10 de um vídeo, o bloco 10 será o último.

Essa política foi adotada no D-VoD devido visando aumentar a permanência do início dos vídeos *cache*. Desta forma, sempre que um cliente solicitar um vídeo que esteja presente na *cache*, é certo que o início da transmissão será imediata, mesmo que nem todo o vídeo esteja presente na *cache*. Outras políticas foram estudadas e implementadas, mas esta foi a que melhor se adaptou às necessidades do sistema proposto.

Para a busca de blocos de vídeos em outros servidores a *cache* utiliza um outro módulo. Atualmente a busca é feita via HTTP ou UDP. Este módulo possui uma lista de fontes de vídeo – servidores *source* ou mesmo outros servidores *proxy*. Quando a *cache* solicita um bloco a este módulo pela primeira vez são disparadas duas *threads* para busca deste bloco nos dois primeiros servidores presentes na lista de fontes. A *thread* que realizar o trabalho mais rápido será selecionada como fonte principal deste vídeo. Se a lista de fontes possuir mais de dois servidores, a próxima solicitação de um bloco também irá disparar duas *threads*. A primeira *thread* tendo como fonte o servidor principal atualmente selecionado, e a segunda tendo como fonte um servidor que ainda não tenha sido testado. Este processo é repetido até que todas as fontes tenham sido testadas. Esta é apenas uma forma simples de realizar um balanceamento de carga entre as fontes deste servidor, e tem se mostrado bastante eficiente nos testes realizados.

Outras características implementadas no D-VoD são o suporte a busca antecipada e um mecanismo de inércia nas requisições. Na busca antecipada, quando o cliente solicita um vídeo que não está completamente na *cache*, a própria *cache* inicia um processo de busca dos blocos restantes para que não ocorra interrupção da transmissão no momento que seja enviado ao cliente o último bloco presente na *cache*. O mecanismo de inércia consiste no seguinte. Quando um cliente solicita um vídeo e assiste a apenas uma parte dele, a *cache* continua a buscar mais alguns blocos do vídeo. Pois, com os testes realizados com o D-VoD percebemos que a maioria dos abandonos do serviço é provocado ocorre em blocos que não estão na cache e precisam ser buscado em fontes, cuja taxa de transferência muitas vezes é inferior a taxa de codificação dos vídeos. Em muitos casos o cliente tenta posteriormente assistir o mesmo vídeo. Com o mecanismo de inércia, na segunda tentativa o usuário conseguirá assistir um trecho maior do vídeo.

6. Resultados Obtidos

O Serviço VOD do GTVD foi testado na RNP. Neste cenário de testes foi implementado um motor de busca em PHP que procura em uma base de dados LDAP (www.natalnet.br/~gtvd). No teste foram utilizados 7 servidores Proxy e dois *servidores fonte* interconectados como ilustrado na Figura 8. A Tabela 1 mostra a velocidade dos enlaces que conectando os servidores.

Foram realizados testes com servidores instalados em Brasília e em Natal, aplicações de busca instaladas no Rio de Janeiro e em Natal e servidores proxy instalados em SP, RJ, PE, DF, PB e SC.

Os testes consistiram em acessar o serviço através da aplicação de busca implementada pela equipe do GT de Diretórios, escolher e assistir vídeos nos exibidores preferidos

pelos usuários que testaram o serviço. Para aumentar o volume dos testes a equipe do GTVD implementou adicionalmente um robô de testes em duas versões uma Java, que foi usada para testar o serviço de máquinas de usuários, e outra usando scripts Linux, usada para testar o serviço a partir das maquinas onde foram instalados servidores.

Figura 8 - Rede de Vídeo Digital sobre a RNP

Enlace	Velocidade
CE ↔ RJ	6 MBps
CE ↔ SP	4 MBps
RN ↔ RJ	4 MBps
RN ↔ SP	4 MBps
PB ↔ RJ	2 MBps
PB ↔ SP	2 MBps
PE ↔ RJ	8 MBps
PE ↔ SP	5 MBps
DF ↔ RJ	22 MBps
DF ↔ SP	20 MBps
SC ↔ RJ	24 MBps
SC ↔ SP	12 MBps
RS ↔ RJ	25 MBps
RS ↔ SP	15 MBps

Tabela 1: Velocidade dos enlaces conectando os servidores

Como esperado, nos primeiros acessos, onde o vídeo precisava ser transferido do servidor primário em natal para o servidor secundário próximo ao cliente, a banda passante sustentada pelo backbone da RNP se mostrou insuficiente. Porém, em tentativas posteriores de acesso, com os vídeos já no cache dos servidores secundários, os usuários conseguiram assistir os vídeos satisfatoriamente.

Os problemas reportados no acesso ao serviço, na maioria das vezes se deveram a problemas na configuração dos exibidores nas maquinas e na instalação de decodificadores compatíveis com os formatos dos vídeos disponíveis.

Identificamos problemas de parada no funcionamento dos servidores. Para contornar estes problemas foram implementados e instalados procedimentos para aumentar a confiabilidade do serviço: uma aplicação de monitoramento que gera mensagens reportando falhas nos servidores e outra que, ao detectar um servidor em falha o reinicia.

Os resultados dos testes realizados foram armazenados em logs nos servidores primários, secundários, nas aplicações de gerencia e monitoramento do serviço, módulo de tolerância a falhas e nos robôs de teste. Segue um resumo dos dados coletados e respectiva análise.

Os dados mostrados na Tabela 2 referem-se a medidas realizadas nas conexões entre um servidor instalado em Brasília e proxies instalados em diferentes localidades. Foram medidos o retardo para transferência de blocos de 500 KBytes e a vazão nas conexões entre servidores e proxies.

Avaliando o resultado percebe-se que a vazão sustentada pelo servidor primário chegou a 102Mbps em Brasília, com o servidor e o proxy sendo executados na mesma máquina. Resultados intermediários foram obtidos pelos proxies no Rio e em São Paulo (entre 1 e 2 Mbps de vazão), depois vieram os servidores de Santa Catarina e Rio Grande do Sul com vazão na faixa dos 700 Kbps. Por fim, o servidor da Paraíba, que está instalado na UFPB que é ligada ao PoP de Campina Grande por um enlace de 2 Mbps completamente saturado, como era esperado, apresentou os piores resultados de vazão e retardo. Dos dados colhidos conclui-se que as taxas sustentadas pelo backbone da RNP são suportadas pela versão implementada do servidor.

Tabela 2: Medidas realizadas nas transmissões entre servidores e proxies D-VoD

Proxy	Amostras	Retardo			Vazão		
		min	med	max	min	med	max
Brasília	5.202	0.001s	0.292s	0.861s	6Mbps	55Mbps	102Mbps
Paraíba	4.120	1,79s	20,44s	551,9s	60bps	21Kpbs	57Kbps
Santa Catarina	13.859	0,213s	2,136s	1047,7s	100bps	104Kbps	618Kbps
Rio de Janeiro	38.695	0,188s	1,948s	12,189s	680bps	156Kbps	1Mbps
São Paulo	367.338	0,056s	0,085s	3,942s	1Kbps	1,5Mbps	1,8Mbps

Um resumo dos dados levantados nos testes da comunicação entre os SS e os clientes foi o seguinte: maior vazão para proxy e cliente na mesma máquina: 180 Mbps (SP); maior vazão para proxy e cliente na mesma rede: 59 Mbps (RN); maior vazão para proxy e cliente na RNP: 1.8 Mbps (SP) e maior vazão para proxy e cliente em backbones distintos: 56 Kbps (SC).

Os dados mostraram que para proxy e cliente em uma mesma rede de campus as taxas permitem o fornecimento de vídeos com taxa acima de MPEG-2 HDTV. Com cliente e proxy ambos no backbone da RNP é possível servir vídeo MPEG-1. Para clientes fora do backbone a vazão máxima sustentada só permite a transmissão de vídeo de baixa qualidade. Para contornar este problema estamos implementando uma versão Java do servidor secundário que poderá ser instalada nas máquinas dos clientes. A idéia é permitir que os clientes agendem o acesso a vídeos de alta qualidade que serão transferidos para sua máquina sempre que ele estiver acessando a Internet. Quando o vídeo estiver disponível localmente o serviço irá enviar uma mensagem para o cliente informando que o vídeo está a sua disposição.

7.Conclusões e Perspectivas Futuras

A Rede de Vídeo digital montada está operacional e os testes realizados demonstram que o serviço é capaz de distribuir vídeos de alta qualidade mesmo considerando restrições nos enlaces interligando os servidores D-VoD.

Como perspectivas futuras cabe mencionar a necessidade de continuar trabalhando na evolução da implementação dos componentes do serviço melhorando sua confiabilidade, performance e integração com exibidores. É importante também destacar a necessidade de motivar produtores de conteúdo a disponibilizar seu acervo através do serviço. Outro

aspecto que precisa ser investigado e que, na realidade, já está sendo é a questão de segurança. Estamos trabalhando na especificação e implementação de suporte a autenticação de usuários, controle de acesso ao serviço e confidencialidade dos dados.

8. Referências

[1] GTVD - Grupo de Trabalho de Vídeo Digital <http://www.natalnet.br/~gtvd> e RNP – Rede Nacional de Ensino e Pesquisa. <http://www.rnp.br>.

[2] Microsoft, Inc., "*Windows Media Technologies*". <http://www.microsoft.com/windowsmedia/>

[3] RealNetworks. "*Real Networks Media Delivery Technologies*". <http://www.realnetworks.com/products/media_delivery.html>.

[4] KeyLabs. "Helix Universal Server from RealNetworks Comparative Load Test Results" <http://www.keylabs.com/results/realnetworks/helixcomparativeload.pdf>

[5] École Centrale Paris. "*VideoLan*". < http://www.videolan.org/ >

[6] UFMG – "Projetos *Video on Demand*" < http://www.vod.dcc.ufmg.br/ >

[7] Rich Wolski, Martin Swany, J. S. Plank, T. Moore, Graham Fagg, M. Beck, A. Bassi. "The Internet BackPlane Protocol: A Study in Resource Sharing". Second IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 2002). Maio 2002.

[8] J. S. Plank, T. Moore, M. Beck. "Scalable Sharing of Wide Area Storage Resources". University of Tennessee Department of Computer Science. Janeiro 2002.

[9] "LibXML - The XML C parser and toolkit of Gnome" <http://xmlsoft.org/>

[10] Stevens, W. R.. "*UNIX Network Programing - Networking APIs: Sockets and XTI* ". Prentice-Hall, Inc., 1998. Segunda Edição.

[11] "Boehm-Demers-Weiser Conservative Garbage Collector - A garbage collector for C and C++" <http://www.hpl.hp.com/personal/Hans_Boehm/gc/>

[12] Berthier, Ribeiro N; Campos, Sergio. "*Video on Demand Servers*". Computer Science Department. Federal University of Minas Gerais.