Compartilhamento nas nuvens para redes locais baseado em P2P

Jerônimo Feijó Noble da Rosa[1]**,Msc. André Moraes**[1]**, Msc. Eduardo M. Monks**[1]

[1]Redes de computadores – Faculdade de tecnologia SENAC Pelotas (FATCPEL)
Caixa Postal 15.064 – 91.501-970 –Pelotas – RS – Brazil

{kastanho,chameoandre,emmonks}@gmail.com

Abstract. *This article describes the development of an application for sharing files over a network using P2P sites. This application aims to facilitate the user's life assuming all the tasks to convert, distribute and retrieve the data allowing the user to engage only with the act of sending, receiving and managing your files. All this using the space that is often wasted on workstations or desktops. Another function is to also perform the decentralization of information from the servers thus increasing the fault tolerance.*

Resumo. *Esse artigo descreve o desenvolvimento de uma Aplicação para compartilhamento de arquivos em rede locais utilizando P2P. Essa aplicação tem como objetivo facilitar a vida do usuário assumindo toda as tarefas de converter,distribuir e recuperar os dados deixando que o usuário se envolva somente com ato de enviar, receber e gerenciar os seus arquivos. Tudo isso utilizando o espaço que muitas vezes é desperdiçado nos workstations ou desktops. Outra função é também realizar a descentralização das informações dos servidores aumentando assim a tolerância a falhas.*

1. Introdução

Hoje me dia com o aumento exponencial das unidades de armazenamento e com seu custo despencando em mesma taxa, as estações de trabalho geralmente estão com sua capacidade sub-utilizada, a partir disso foi criada a ideia de usar esse espaço desperdiçado para armazenar os dados que em outro caso estariam em um servidor de arquivos.

Com a utilização desta aplicação é possível distribuir dados nas estações de trabalho participantes sem que o usuário se envolva ou mesmo saiba onde esses arquivos realmente estão. O Usuário apenas interage com uma interface simples que permite enviar, organizar,baixar e apagar seus arquivos. Ao utilizar uma estrutura descentralizada é possível reduzir ou eliminar os problemas de sobrecarga e disponibilidade, pois quanto maior a rede, maior são as fontes de arquivos e menor a probabilidade que um determinado host seja sobrecarregado ou mesmo sofra uma falha qualquer fazendo com que a informação se torne indisponível. Para que possível a criação dessa estrutura descentralizada utiliza-se o princípio de P2P que hoje em dia é muito utilizado na internet para compartilhamento de arquivos.[Kurose and Ross 2009]

Para a tarefa de localização de hosts na rede é utilizado o multicast.[Kurose and Ross 2009] O seu funcionamento é parecido ao do *broadcast* a diferença é que o último envia uma mensagem para um endereço especial e todos os demais hosts recebem e tratam a mensagem como se fosse para eles mesmos, enquanto

no *multicast* é possível criar grupos de hosts através de um endereço multicast. Ao escolher um endereço IP, esse é compartilhado pelos hosts membros e somente eles tratam as mensagens enviadas para o endereço escolhido.

A linguagem de programação utilizada foi o *Python 2.7* uma linguagem de programação multi-plataforma e orientada a objetos que dá possibilidade de migração para quase qualquer sistema operacional, incluindo dispositivos móveis. Além das vasta quantidade de bibliotecas e documentação disponíveis. [Rhodes and Goerzen 2010]

2. Conceitos básicos

2.1. Peer to Peer

O funcionamento do *Peer to peer* é interessante, pois dá a possibilidade de cada host entrar,sair e voltar ao compartilhamento a qualquer momento causando pequeno ou nenhum efeito na disponibilidade do conteúdo como um todo. Ao tentar baixar um arquivo o host conecta-se a uma rede P2P, nessa existem outros membros tentando baixar o mesmo arquivo. Esses hosts se conectam entre si e começam a baixar os pedaços que necessitam e enviar os que eles já tem. No P2P não existe uma estrutura cliente e servidor clara, pois cada membro dessa determinada rede pode ser ao mesmo tempo servidor de um e cliente de outro, basta apenas que um host tenha um pedaço que outro precise. Essa habilidade de criar várias conexões com diferentes fontes para obter um recurso faz com que a rede se torne escalável ou seja quanto maior o tamanho, maior será a velocidade de download contando que todos os hosts estejam compartilhando seus recursos.

Entre as várias aplicações baseadas em compartilhamento P2P, esse projeto foi inspirado no protocolo *Bittorrent*, desenvolvido principalmente para compartilhamento de arquivos na internet. [Kurose and Ross 2009]

O *Bittorrent* utiliza um servidor que guarda as informação do conteúdo a ser baixado e quais hosts estão ativos no momento. Esse servidor é chamado de *tracker* ou rastreador, e para localizar-lo é necessário baixar um arquivo *.torrent* que fornece informações para conexão e para o conteúdo a ser baixado.

O uso do rastreador é um ponto fraco do protocolo *Bittorrent*, pois se ele estiver inoperante os hosts não serão capazes de encontrarem, claro que apesar do risco também é uma excelente forma de localização através da internet.

3. Funcionalidades

Essa seção descreve as funções que os usuários tem acesso permitindo que os mesmos manipulem, organize e excluam seus arquivos. Existem dois níveis de usuários os primeiros são os usuários comuns que tem funções básicas para utilização da aplicação, essas atividades estão descritas abaixo:

- Download;
- Upload;
- Criação, manipulação e exclusão de diretórios;
- Troca de senha.

Todas as funções descritas fornecem todo acesso necessário que um usuário comum precisa para a função principal da ferramenta que é o envio e e recebimento de arquivos.

Existem também os usuários que pertencem a grupos de administradores esses grupo tem a função de gerenciar a aplicação, eles podem podem executar comandos de controle como:

- Criar, excluir e modificação grupos e usuários;
- Informações de rede;
- Troca de senha dos usuários.

É interessante lembrar que os administradores também podem executar funções básicas. Para esse projeto foi criada uma interface em modo texto e alguns dos comandos utilizados são bem parecido aos que os sistemas operacionais atuais utilizam. A figura 1 mostra

Figura 1. prompt de comando

o prompt de comando primeiramente é apresentado o usuário ativo que no caso é o "admin"logo após o diretório atual que é o raiz e então o cursor fica esperando que o usuário digite algum comando. Isto tudo é mostrado após se efetuar a autenticação com sucesso e que as informações de usuário sejam guardadas em uma sessão. outras informações são armazenados na sessão como o código do diretório atual e anterior e as informações dos grupos que o usuário é membro. É considerado diretório raiz quanto o código do diretório é igual a zero. Os dois níveis de acesso são definidos pelo grupo e não pelo usuário. O usuário pode pertencer a diversos grupos ao mesmo tempo e no caso de qualquer um dos grupos ter um nível administrador o usuário automaticamente se torna administrador. Ao tentar executar um comando é necessário que o usuário tenha permissões adequados, caso contrário será retornado uma mensagem de erro. Ao criar um arquivo ou diretório novos serão criadas referências a todos os grupos que o usuário é membro além de criar uma referência para si mesmo. Não existem permissões de leitura,escrita e execução então somente os proprietários do arquivo ou pastas podem visualizar e baixar esse conteúdo.

Uma vantagem desta aplicação é o reaproveitamento de arquivos já enviados. Isso é feito para reduzir o desperdício de espaço em disco, a ideia é bem simples ao tentar enviar um arquivo para a nuvem uma *hash* é gerada para que seja possível comparar com os demais arquivos já armazenados. Se for encontrado um outro arquivo com a mesma *hash* então será criado um tipo de atalho e as informações de nomeação, diretório na nuvem,extensão e até permissões serão os dados fornecidos ao executar o comando de envido de arquivos. Enquanto as informações do nome do primeiro pedaço, quantidade de pedaços e o tamanho do total do arquivo utilizam as informações do arquivo já registrado. A figura 2 mostra um exemplo simplificado da criação de um atalho.

Para cópia de arquivos é feito algo parecido com a criação de atalho, pois o usuário deve informar o arquivo de origem e destino então basta copiar as informações do registro de origem para o registro de destino. Para mover um arquivo de uma pasta para outra só é necessário alterar o código do diretório para o do novo caminho. A utilização de atalhos traz vantagens tanto para o uso na rede, quanto para o espaço em disco e processamento, pois antes de enviar o arquivo deve ser dividido consumindo recursos locais, depois esse arquivo é enviado para rede que exigem mais recurso e ao chegar ao destino os arquivos devem ser armazenados em disco. Por serem distribuídos em diversos pedaços entre os

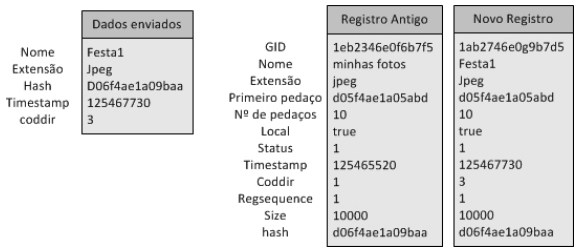

Figura 2. Exemplo de criação de arquivos com hash iguais na nuvem.

hosts na rede o risco de perda de um determinado arquivo é pequeno justificando assim o uso de atalhos nas cópias e criação de arquivos.

Ao remover todas as cópias de um arquivo os dados também são removidos do disco para isso é feito uma checagem de existência de um arquivo com mesma hash caso não seja encontrada então é executada uma solicitação de remoção de arquivo.

4. Base de dados

As principais tabelas são usuários,grupos,pastas,arquivos e pedaços. Na figura 3 as duas primeiras tabelas(usuário,grupos) são utilizados exclusivamente para controle de acesso e administração da aplicação e a distinção do conteúdo de um determinado grupo e/ou usuário. A tabela de usuário fornece informações para acesso à aplicação, a tabela de grupo permite a criação de níveis de grupos que torna possível a administração da ferramenta. A tabela de arquivos é utilizada para armazenar informações de identificação de um ar-

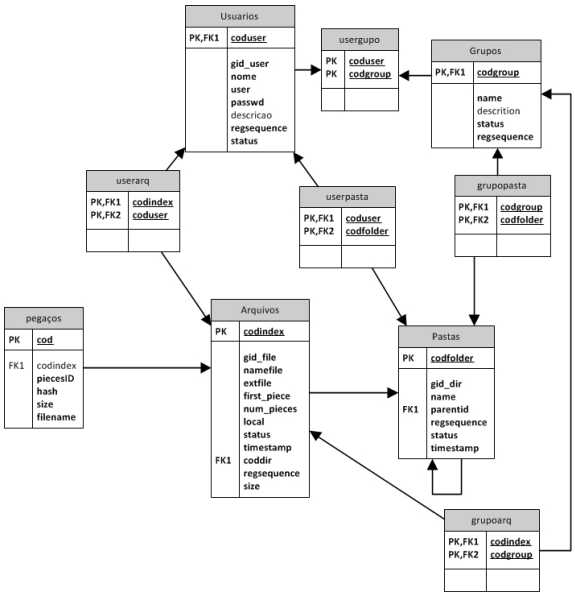

Figura 3. Estrutura da base dados

quivo, mas não mantem as informações da localização em disco. Os únicos detalhes são o nome do primeiro pedaço, quantidade de pedaços e se existe algum pedaço armazenado localmente, isso é feito para reduzir a utilização de rede ao máximo se fosse necessário guardar as informação de cada pedaço isso tornaria a base de dados cheia de informações desnecessárias e exigiria muito da rede para manter as tabelas atualizadas. Na tabela

de pedaços estão guardadas as informações sobre os pedaços dos arquivos armazenados localmente. Nesta tabela existe referencia ao caminho e nome de arquivo no sistema de arquivos local. Existe também uma tabela de diretórios que é utilizada para organização dos arquivos, essa referência é feita através do campo *coddir* na tabela de arquivos essa ligação seria representada na aplicação da seguinte forma */diretorio1/arquivo1*. Um diretório pode ser atribuído a outro de forma recursiva(*/diretorio1/diretorio2/diretorio3*) através da utilização do campo *parentid* onde é inserido o diretório pai.

Como pode ser visto na figura 3 os usuários podem pertencer a vários grupos e cada grupo pode ter vários usuários. Os arquivos e pastas também podem pertencer tanto ao grupo onde um determinado usuários é membro ou então podem pertencer ao próprio usuário, mas é importante lembrar que apenas usuários que sejam donos de um arquivo podem visualiza, baixar ou excluir o mesmo.

5. Distribuição de pedaços

Para fazer a distribuição dos arquivos foi levado em conta o princípio que seria necessário pelo menos 50% dos hosts ativos para ser possível recuperar um determinado arquivo, mas claro que quanto maior a rede maior a diferença de hosts entre o requisito mínimo que é 50% e o requisito máximo que é 70%, ou seja uma rede com quatro hosts a diferença é de um. Em uma rede de com cinquenta hosts a diferença é de dez, isso possibilita que aja uma variação na distribuição em redes maiores desde que a porcentagem, mas importante lembrar que a porcentagem não pode ser menor que 55%. Abaixo é apresentado o algorítimo que faz o calculo da média:

```
1. calc=float(nodes)*50/100
2. calc2=float(nodes)*70/100
3. media=70-(calc2-calc)*0.5
4. if media<55:
5.   media=55
6. if media>100:
7.    media=100
8. return int(nodes*media/100)
```

No código apresentado anteriormente primeiramente são realizados dois cálculos iniciais um para saber qual é o numero de hosts necessários para ter 50% da rede e o segundo para ver o necessário para ter 70% da rede. Na terceira linha é feita a média diminuindo o valor calculado na linha 1 e 2 multiplicado por 0.5 para que a variação não seja muito grande e depois é diminuído de 70. Da linha 4 até a linha 7 é feita uma verificação para saber se não foi ultrapassado o valor minimo ou máximo. Na linha 8 é calculado o valor total de hosts que devem receber um determinado pedaço.

Um teste utilizando uma rede com quatro hosts como o da figura 4 necessita pelo menos três hosts receberem o arquivo, ela também mostra um teste com a distribuição dos pedaços a esquerda mostra a combinação dos hosts e mostra os cenários que foram testados com sucesso, e a direita mostra a distribuição de cada host por pedaço nos três cenários testados.
Para realizar a distribuição mostrada na figura 4, primeiramente realiza-se o cálculo para saber quantos hosts receberão a determinado cópia de um pedaço .

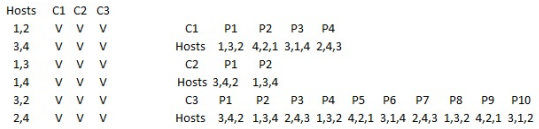

Figura 4. Teste de distribuição de hosts por pedaços

Logo após descobrir a quantidade de host que vão receber uma cópia começa a distribuição propriamente dita, primeiramente é escolhido um número aleatório entre o host 1 e o numero total de host, depois que foi descoberto esse primeiro número é pulado uma casa e pega o próximo host como mostra a figura 5.

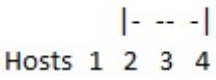

Figura 5. Escolha de hosts utilizando salto

Se o host seguinte for maior que o numero do total de hosts então volta para a o início como mostra a figura 6. É importante lembrar que não é possível repetir o numero do host ou seja um determinado host não pode receber o mesmo pedaços mais de uma vez, então caso caia no mesmo numero de host que já foi sorteado anteriormente e já está na lista esse valor é somado com mais um e verificado novamente até que não aja um resultado positivo. Após todos os hosts que vão receber os determinado pedaços

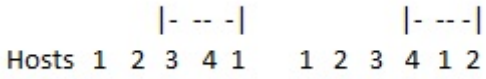

Figura 6. Escolha de hosts

foram definidos o próximos hosts que serão atribuídos serão os que sobraram como mostra a figura 7 isso é feito para que os pedaços fiquem completamente distribuído entre os membros da rede.

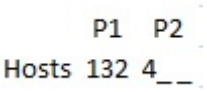

Figura 7. Organização de Hosts no utilizados

Depois que os hosts que não foram inserido na lista do pedaço anterior forem inserido no próximo pedaço, então a regra padrão segue como antes, por exemplo 4,6,2 até que todos os pedaços sejam distribuídos.

6. Comunicação

A comunicação na rede é feita por duas portas que começam a ser escutadas assim que a aplicação inicia . Uma delas é uma porta UDP utilizando multicast e outra TCP através de unicast . Essas portas podem ser alteradas no arquivo de configuração, mas para que os hosts se encontrem a porta Multicast deve ser igual para todos eles. Isto acontece porque os hosts trocam mensagens de descobrimento e mensagens de atualização de índice. A porta TCP pode usar um número de porta diferente em cada host, pois esse informação é enviada junto como as mensagens de descobrimento de host. A transferência de arquivo utiliza uma ou várias portas UDP dependendo quantos blocos estão sendo transmitidos ou recebidos no momento, a porta TCP serve como controle durante essas transferências.

6.1. Descobrindo Vizinhos

Antes que tudo mais seja possível é necessário que sejam localizados os demais hosts na rede, por isso a porta de multicast deve ser igual em todos os eles. O descobrimento começa na inicialização do software quando uma mensagem de descobrimento é enviada para o endereço de multicast, essa mensagem contém informações como o nome do host, o endereço IP e porta, o espaço em disco, o uptime do host, e o IP do índice raiz e no destino é inserido uma validade para o registro antes desse ser inserido na lista, após o host remoto receber a mensagem de descobrimento esse host remoto responde ao endereço de unicast TCP da origem as suas próprias informações exemplo nome, endereço, porta e etc. Isso é feito para reduzir o numero de mensagens multicast e fazer que somente quem necessita receber a informação receba a mensagem. Depois de um tempo pré definido é enviada uma mensagem de atualização para que os demais hosts saibam que a origem ainda está ativa, um host remoto ao receber essa mensagem atualiza o tempo de validade do registro com seu próprio valor, esse é definido no arquivo de configuração. Quando o tempo de validade de um registro expira em um host remoto uma mensagem de solicitação por muticast é enviada e a reposta deve voltar da mesma forma, para que os demais hosts possam saber que o solicitado está ativo e o mesmo não receba uma enxurrada de mensagens e tenha que responder cada uma, caso do tempo de validade dos registros seja igual para todos os hosts. Se mensagem de solicitação não for respondida o registro deve ser removido da lista do hosts remotos assim que a sua requisição expire.
Abaixo é apresentado um exemplo de mensagem de descobrimento:

Id msg.:Nome do host:Uptime:endereço IP:Porta:IP do índice raiz:Validade do registro
1:Host1:999999999:192.168.0.1:8000:1024:192.168.0.50:9000000

O campo "id msg"é usado para identificar o tipo de mensagem que estão sendo carregadas. o campo "nome do host"é utilizado para facilitar a visualização do usuário. o campo "uptime"informa a quanto tempo o host remoto está ativo, esse campo é utilizado para permitir a eleição do host raiz. Os campos "endereço IP"e "Porta"só são utilizados para identificar o host remoto e a porta TCP para comunicação unicast. o campo "IP do índice raiz"é usado para informar qual o endereço do índice raiz do host remoto. E por ultimo é o campo de "validade do registro"nada mais é que o timestamp que é inserido quando a mensagem chega, de tempo em tempo, todos os registro são verificados para saber se algum desses não estão expirados. A função de descobrimento é inspirada na comunicação entre roteadores com a utilização do protocolo OSPF.

6.2. índice

Esta é uma das partes mais importantes da ferramenta, pois permite realizar a distribuição das informações dos arquivos, usuários, grupos, e diretórios. Sem isso nem é possível acessar a aplicação porque o login e senhas de usuários devem ser distribuídos entre os hosts. Além disso a utilização de bases distribuídas por diversos hosts permite que se tenha uma rápida recuperação de falhas e maior confiabilidade no armazenamento dos dados porque é gerado um tipo de backup involuntário do índice em cada membro dessa rede.

6.2.1. Eleição do índice raiz

O software ao iniciar procura buscar suas configurações de elegibilidade, se no seu arquivo de configuração constar que ele pode concorrer, então esse host candidato aguarda que a lista de vizinhos seja montada para que ele possa verificar o *uptime* dos demais membros da rede. Caso seja constatado que ele tem o direito de assumir o posto de host raiz, será então enviada uma mensagem para o endereço de *multicast* informando que ele está assumindo a função. Os demais membros aceitam caso concordem com os dados fornecidos. a figura 8 exemplifica a eleição de forma simplificada.

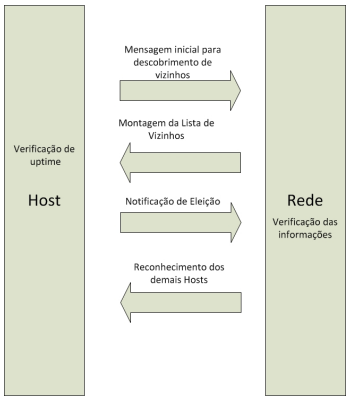

Figura 8. Eleição do índice raiz

Nas mensagens de descobrimento existe vários campo entre eles tem o campo *uptime* nesse campo são inseridos os tempos de atividade do host ou então as configurações de elegibilidade. Esse *uptime* é o tempo de execução da aplicação e não do sistema operacional. No caso de um nível ser configurado o valor de *uptime* vai ser igual a esse nível e o tempo de atividade do host não será utilizado. abaixo são descritos os níveis possíveis.
0 - Não elegível;
1 - Prioritário;
2-9 - Nível de prioridade mais baixo.
São vários níveis de prioridade por exemplo o nível zero não permite que o host se torne um índice raiz, caso o nível seja um o host terá preferencia sobre qualquer host com nível mais baixo ou então com qualquer *uptime*. A prioridade é definida em ordem sequencial de baixo para cima. Em caso de dois ou mais hosts estejam em nível igual ou com o mesmo uptime serão levados em conta as configurações de hardware de cada um deles e

se mesmo assim não houver uma decisão então vai ser eleito o host com o menor endereço IP, como mostra o exemplo abaixo:

192.168.0.5
192.168.0.7
192.168.0.3

6.2.2. Mantendo índice

Ao inserir,remover ou alterar um registro nas tabelas usuários,grupos,pastas,arquivos e as tabelas que fazem a ligação entre elas é gerada uma mensagem para o índice raiz que notifica a alteração do registro.

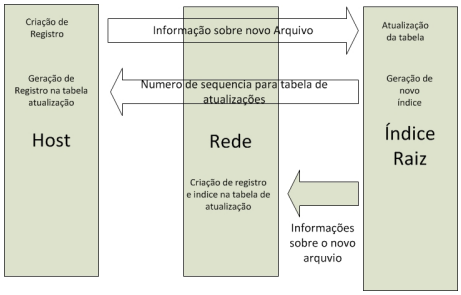

Figura 9. Mantendo índice

O índice raiz recebe as informações e gera um registro em sua base de dados, em seguida é gerado um registro na tabela de atualização, nessa tabela é inserido um número de sequência que posteriormente será mandado através da conexão já estabelecida com o host de origem para que esse possa finalizar a criação do registro. Então o índice raiz envia uma mensagem via multicast para todos os demais membros da rede informando da existência de um novo registro e nesse mesma mensagem vai anexado as informações para ser inseridas na tabela de atualizações. A figura 9 exemplifica o processo de troca de mensagens. Para uso em rede cada registro utiliza um identificador único gerado pelo algoritmo SHA1. A estrutura 'base distribuída' é mostrado na figura 10. É importante lembra que localmente é utilizado um número inteiro sequencial para servir como chave primária e esse valor pode ser diferente nos diversos membros da rede.

6.2.3. Sincronização de índice

Ao iniciar a aplicação é necessário realizar atualizações nas tabelas, pois enquanto o host estava desligado alterações podem ter sido realizadas e esse host estará com suas tabelas desatualizadas, então após fazer a inicialização do aplicativo começará a atualização. primeiramente será verificado na tabela de atualizações qual o numero de sequencia mais alto, depois disso uma mensagem será enviada via *unicast* TCP para o índice raiz solicitando os números maiores do que foi informado. é informado uma mensgem parecida com a apresentada abaixo:

Sequencia:GID:GID2:Num. sequencia do registro:operação

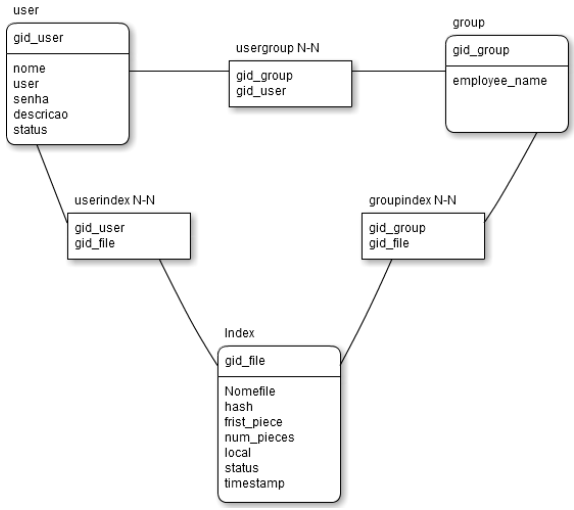

Figura 10. Estrutura da base dados na rede

O número de sequencia é atribuido pelo servidor para a tabela de atualizações. O GID é a identificação única de cada registro que será utilizado na rede. O campo GID2 é utilizado somente quando é modificada uma ligação entre duas tabelas como por exemplo o a inserir um novo membro a um grupo. O campo de sequência do registro é utilizado para informar qual o número de sequencia do próprio registro isso é feito para que impeça o host de solicitar um registro que já está atual. Por exemplo na figura 11 o texto marcado representa um mesmo registro sendo atualizado diversas vezes, na sequencia 4 ele é criado, na sequencia 6 e 7 são feitas atualizações. A cada modificação que é feita o campo "reg. sequencia"da tabela atualizações e da própria tabela do registro são incrementados, quando o host recebe a informação contidas na figura 11 ele solicita os dados do primeiro registro para o índice raiz. Esse por sua vez responde com o registro desejado ao fazer a alteração na tabela o novo registro já vem com o número de sequência da última atualização, então quando por exemplo o host chegar na sequencia 6 vai verificar que o valor do campo reg. sequência no registro é maior que o do mesmo campo na tabela de atualização e não o solicitará novamente, pois esse já estará atualizado reduzindo assim o tráfego desnecessário na rede.

Sequencia	GID	GID2	Reg sequencia	OP
7	6aea660e924eabf03b692f3f9a4b06d5b1c47ed4		3	2
6	6aea660e924eabf03b692f3f9a4b06d5b1c47ed4		2	2
5	aaa6aea660e924eabf03b692f3f9a4b06d5b1c47	aca3aea960e904ecbfc3b632f3f9a4b06d5b1c47	2	3
4	6aea660e924eabf03b692f3f9a4b06d5b1c47ed4		1	1

Figura 11. Exemplo da tabela lista de atualizações

As operações aceitas são:
1- Criação;
2- Alteração;
3- Exclusão.
A estrutura do campo operações é feita dessa forma para que seja possível guardar a remoção de um registro.

7. Transferência

Essa seção do documento descreve a transmissão de arquivos entre hosts. começando pelo envio de arquivo, depois passa pelo recebimento e por último aborda o envio como ele é realmente realizado.

7.1. Envio de arquivos

Ao receber uma solicitação da camada de usuário para envio de arquivos a função de envio vai primeiramente pegar as informações do arquivo e armazena-los na base dados essas informações são o nome do arquivo,extensão, a hash, timestamp e etc. Após as informações serem gravadas o arquivo é convertido para o formato BASE64 e dividido em pedaços. Esses pedaços contém as informações do nome do primeiro pedaço e o total de pedaços, eles também são inseridos na base de dados. Depois que o registro está completo as informações são enviadas para o índice raiz para que ele possa fazer a atualização do índice nos demais membros da rede e para que ele também retorne o número de sequência para que seja inserido na tabela de atualização. Após realizar essa tarefa é recuperado o número de hosts ativos com espaço para receber cada pedaço e é iniciada a distribuição. Ao receber a lista de qual host vai recebe o determinado pedaço é iniciada a conexão com os hosts contidos na lista. essa conexão acontece de forma simultânea. Esses hosts são informados que irão receber um arquivo, se aceitaram então a transferência é inciada. quando todos os pedaços são enviado com sucesso o usuário é informado que o arquivo foi inserido na nuvem.

7.2. Recebimento de arquivos

Ao receber uma requisição da camada de usuário para que seja recuperado um determinado arquivo que está na nuvem. É iniciada uma busca local para verificação da existência de pedaços desse arquivos caso exista pedaços locais as informações de serão inseridos na lista de recuperação, depois de executar a pesquisa local é requisitado via endereço *multicast* fontes para download. Os hosts remotos montam uma mensagem de resposta com os pedaços que cada um pode fornecer e envia para do endereço *unicast* TCP do host solicitante. Ao receber as informações o host solicitante segue montando a lista com detalhes de cada um dos pedaços a serem baixados e divide esses pedaços em blocos para tornar possível o download concorrente de cada um deles. Em relação aos arquivos locais eles somente são copiados para a mesma pasta temporária onde os outros estão sendo baixados. Após todos os pedaços serem recebidos no destino o arquivo é montado e convertido de Base64 para o seu formato anterior e a hash é comparada com a que está armazenada na tabela de índices e depois esse arquivo é movido para o destino escolhido pelo o usuário. importante lembrar que no caso de não ser possível encontrar as fontes necessárias para baixar todos os pedaços será enviada uma mensagem para o usuário informando do fato.

7.3. Transferência de arquivos

Após montar a lista com todos os blocos é iniciada a conexão com cada um dos hosts remotos e a partir dai são solicitados os blocos para download. Após receber todos os blocos de um determinado pedaço as hashs são comparadas e o pedaço é montado e então a conexão é fechada. Ao iniciar uma conexão o remetente envia

para o destinatório as informações iniciais do bloco e a ordem que ele deve ser montado. Por o UDP não ser confiável cada datagrama é numerado e uma hash do conteúdo acrescentada no final. Após isso 1386 bits de dados são inseridos e o datagrama é enviado.

identificação do datagrama:hash dos dados:dados

No destinatário ao receber o datagrama os dados são ordenados se necessário ou então são solicitados novamente caso os mesmos não estejam íntegros, na origem são mantidos em uma lista alguns datagramas já enviados o objetivo disto é que no caso de serem solicitados novamente pelo destinatário o datagrama é re-enviado, caso já não esteja mais na lista então o numero de sequência deve ser inserido em outra lista. Para que o arquivo seja percorrido após o envio de todos os demais datagramas. Isso deve ser feito em ordem numérica do menor para o maior. por que fazer isso? A resposta é simples porque o pedaço deve ser lido novamente então é mais fácil juntar todos datagramas que não foram entregues e enviar novamente de uma vez só fazendo a leitura de forma sequencial do pedaço.

8. Projetos Futuros

Para os trabalhos futuros serão implementadas uma interface gráfica para facilitar a interação com o usuário, também é importante a proteção dos dados tanto durante a transmissão quanto no armazenamento local. Também é importante a criação de cotas de usuários e grupos como já existe em muitos sistemas utilizados hoje em dia. Essa função dá poder ao administrador limitar a quantidade de dados que determinado usuário pode enviar para a nuvem. Outra funcionalidade que deve ter uma atenção especial é a modificação da estrutura das permissões de arquivos e diretório para que seja acrescentadas as permissão de leitura, escrita e execução.

Para permitir uma maior disponibilidade dos arquivo será importante o desenvolvimento de alguma forma de redistribuição de pedaços para quando um novo host for ligado ou reconectado na rede que este possa receber um ou mais pedaços de um determinado arquivo que no momento estiver com baixa disponibilidade tornando possível que sempre aja fontes ativas para ele.

9. Testes

A aplicação não foi completada, alguns teste foram feitos como a transmissão de mensagens via multicast. Diversos hosts ligados em uma rede ficaram escutando em um endereço de multicast e um outro começou a enviar mensagens curtas, os hosts que estavam escutando começaram e receber e mostrar as mensagem na tela. Também foram feitos testes de divisão e remontagem de arquivos existiam vários arquivos de tamanhos diferentes e esses foram convertidos para o formato BASE64 e depois divididos, remontados e testados para verificar o funcionamento do algoritmo.

Foram realizados testes no algoritmo de distribuição que permite realizar a tarefa de criar uma lista de qual hosts devem receber um determinado pedaços. essa lista divide o número de pedaços e apresenta na tela a sua distribuição. Também foi criado um método para realização de teste para ver se seria possível a recuperação dos mesmos em diferentes situações.

Existe um ambiente de interação com o usuário parcialmente funcional onde muitas das tarefas foram implementadas, mas claro que essa interface deve ser melhorada para que o usuário possa interagir de forma mais intuitiva.

10. Conclusão

O projeto que foi proposto era grande de mais para o tempo de execução e não pode ser finalizado, mas muitas das suas funcionalidades e classes estão implementadas, claro que ainda não é uma versão operacional, pois ainda necessita a criação de algumas funções importantes.

Durante a pesquisa e desenvolvimento foi constatado que é a construção dessa ferramenta é viável,e pode se tornar totalmente funcional e por isso o projeto vai ter uma continuação em um próximo trabalho, onde as funções hoje implementadas serão completadas com as tarefas ainda pendentes e também serão complementadas com novas funcionalidades não abordadas neste documento.

No decorrer do desenvolvimento da aplicação foi realizado uma breve pesquisa para encontrar outras aplicações com funcionalidades semelhantes a essa, mas a pesquisa não encontrou nenhuma que fosse especificamente para redes locais e que fizesse a distribuição de pedaços entre os hosts ou que tornasse transparente para o usuário o ato de envio e armazenamento de arquivos.

A linguagem de programação utilizada se mostrou bastante confiável e completa. Dando suporte a todas várias necessidades de biblioteca que foram necessárias durante a criação do conteúdo desde o nível de sockets até na parte de manipulação de arquivos. Essa linguagem também permite o desenvolvimento em uma fração do tempo se comparados com algumas outras linguagens. além de possibilitar o uso de orientação a objetos que tornou mais claro o entendimento do código além de permitir o reaproveitamento do mesmo.

Referências

Kurose, J. F. and Ross, K. W. (2009). *Redes de computadores e a internet: uma abordagem top-down*. Pearson Education, 5th edition.

Rhodes, B. and Goerzen, J. (2010). *Foundations of Python Network programming*. Apress, 2th edition.