Universidade de Brasília
Departamento de Ciência da Computação

29 de outubro de 2015
(2015/2)

Teleinformática e Redes 1 – Turma A – Trabalho 1
Professor: Marcos F. Caetano <mfcaetano@unb.br>
Monitor: Guilherme David Branco <gdbranco@gmail.com>
Resumo: Implementação do protocolo DHT.

1 Introdução

Um banco de dados numa rede P2P é usualmente feito a partir do protocolo DHT, *Distribuited Hash Table*, DHT é feito a partir de uma estrutura de pares, como *(chave,valor)*, onde neste exemplo chave será o nome de um arquivo, "filme.mkv"e valor será o *IP:Porta* onde está armazenado. Numa versão distribuída cada *peer* guarda um conjunto destas estruturas de pares, e qualquer um destes *peers* podem perguntar ao DHT a partir de uma chave qualquer e obter a informação de quem possui a chave, assim como pode também inserir/remover novos pares de *chave,valor* no banco.

O objetivo deste trabalho é a implementação de um DHT, de acordo com as funcionalidades descritas neste documento. Maiores informações podem ser obtidas no livro do Kurose, conforme bibliografia informada em sala. Dúvidas devem ser encaminhas ao fórum da disciplina.

2 Descrição e implementação

O protocolo Distribuited Hash Table – DHT refere-se a uma estrutura de dados bastante utilizada em redes P2P (peer-to-peer). Com o objetivo de obter e separar informações de uma forma descentralizada, escalonável, balanceada e com excelente performance, por meio da distribuição de arquivos entre seus usários.

Neste trabalho, a estrutura DHT a ser implementada deve conter algumas funcionalidades obrigatórias, tais como: rendezvous, estrutura de dados, ordenamento, busca e armazenamento, inserção e remoção de nós, reatamento, atalhos (finger-tables) e também inserção, atualização e remoção de dados. Maiores detalhes serão apresentadas nas próximas seções.

2.1 Rendezvous

No contexto de redes p2p, servidores *rendezvous* possuem endereços IPs fixos e previamente conhecidos, que servem como ponto inicial de encontro na rede. Ou seja, nós que desejam entrar na rede utilizam os *rendezvous* como ponto de partida para encontrarem os demais, uma vez que nós em uma rede p2p utilizam endereços IPs dinâmicos.

Desta forma deve existir um servidor previamente conhecido na rede, a fim de entregar aos nós entrantes um ID para colocá-los no DHT. A comunicação entre os nós e o servidor *rendezvous* será feita utilizando *socket* com protocolo UDP. Ou seja, a comunicação não será orientada a conexão.

Para que seja possível iniciar um DHT, a comunicação se estabelece primeiramente com o envio de uma mensagem *hello*, do novo nó para o servidor, que responde com a oferta de ID (único) para o acesso ao DHT. O mecanismo assemelha-se ao funcionamento do protocolo DHCP (do inglês, *Dynamic Host Configuration Protocol*). Ou seja, ao receber a mensagem com ID, o nó retorna um *Ack* (do inglês, *Acknowledgement*) ao servidor, confirmando sua oferta. A partir deste momento, o servidor *rendezvous* possui a associação ID para IP:*Porta* do cliente registrado.

Os IDs válidos a serem utilizados encontram-se especificados por um arquivo de configuração a ser fornecido como parâmetro de entrada para o servidor *rendezvous*. Deverá ser aceito duas formas válidas de formação dos IDs.

O primeiro com IDs na faixa válida entre $[0, K]$, onde K representa o número máximo de nós na rede. O segundo, com IDs de 2 a K, sendo todo elemento uma potência de 2. Os IDs devem ser atribuídos de forma aleatória, dentro da faixa, aos nós ingressantes.

É importante ressaltar que a aplicação a ser implementada precisa observar e tratar os casos em que o protocolo não se comporta conforme o descrito. Por exemplo:

- O cliente envia a mensagem *hello* mas não recebe a mensagem resposta do servidor;
- O servidor envia a mensagem com o ID pré-alocado mas não recebe a confirmação de *Ack* do cliente;
- Mensagens chegam duplicadas no servidor;
- Mensagens chegam fora da ordem ao servidor (UDP);

O servidor precisa ter a capacidade de distinguir cada nó durante a fase de registro do protocolo, uma vez que as mensagens *hello*, provenientes de diferentes nós, podem chegar a qualquer momento. A medida em que o servidor registra os clientes, ele deve montar localmente a estrutura DHT (Figura 1). Esta estrutura deverá ser utilizada somente para o acompanhamento dinâmico da topologia da rede. Ou seja, o servidor deverá apresentar, em seu terminal, a topologia do DHT ao logo da execução do protocolo. Em nenhum momento, a topologia deverá ser utilizada pelos nós da rede para facilitar a localização de qualquer tipo informação no DHT, pois cada nó possui somente uma visão local da topologia.

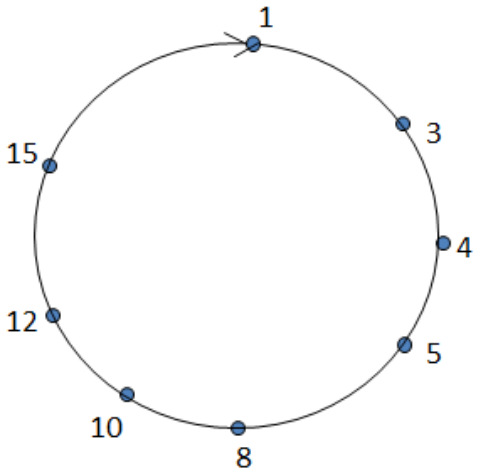
Por fim, o servidor também deve informar ao nó ingressante a existência prévia ou não do DHT. Caso não haja, o servidor marcará o nó como sendo o primeiro da rede (*root node*). Esta informação será informada ao nó e este tomará conhecimento de que ele é o *root node* da rede. Entretanto, caso já exista um DHT formado, o servidor informará ao nó o IP:$Porta$ do *root node* existente. Desta forma, o nó ingressante entrará em contato com o *root node* para iniciar o processo de registro no DHT.

2.2 "Estrutura de Dados"

Do ponto de vista da abstração, o DHT se faz representar como uma lista circular duplamente encadeada. Contudo, do ponto de vista da implementação, os nós não possuem uma visão global da topologia da estrutura. Cada nó só conhece o IP:$Porta$ de seus vizinhos.

Para o projeto a ser desenvolvido, o DHT será pensado como uma lista duplamente encadeada, a fim de se formar um círculo entre os nós. A formação desta lista é distribuída, ou seja, cada nó possui somente uma visão parcial da estrutura. Na Figura 1, por exemplo, o nó 1 conhece somente o IP:$Porta$ referente ao nó 3 (seu sucessor) e IP:$Porta$ referente ao nó 15 (seu antecessor). Este conceito vale para os demais nós da rede. A comunicação entre os nós é feita utilizando *socket* e o protocolo UDP.

Figura 1: DHT Circular.



2.3 Ordenamento

Deve ser definida uma ordem para que se possa determinar o "próximo"nó da lista. Será utilizado o método proposto no DHT Chord, que consiste em:

- Ordenar os nós de maneira circular (lista duplamente encadeada) a partir do ID. O sucessor de cada nó deverá ser o de menor distância no sentido horário, sendo que para a maioria deles o sucessor consiste no ID mais próximo, contanto que maior que o seu próprio ID. A exceção será o nó de maior ID da lista, para o qual o seu sucessor deverá o de menor ID, na formação do círculo. A métrica de distância pode ser calculada com o seguinte algoritmo:

Algorithm 1 Algoritmo para encontrar a distância entre dois nós

```
int distancia(a,b):
if (a == b) then
    return 0
else if (a < b) then
    return b-a
else
    return (2**k)+(b-a)
end if
```

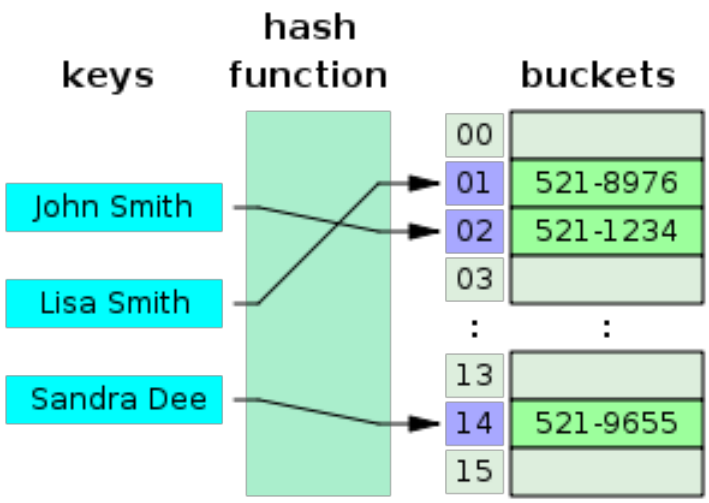
2.4 Busca e Armazenamento

Cada nó compreende uma tabela *hash* comum, também conhecida como *Tabela de Dispersão* ou *Tabela de Espalhamento*. Ela consiste em uma estrutura de dados especial, que associa chaves de pesquisa a valores. A Figura 2, apresenta o esquema de uma função *hash*. De acordo com a figura, a chave *john Smith* ao ser aplicada a função *hash* será mapeada para o valor 02, que está associado ao *bucket* 521-8976. Ou seja, a partir de uma *chave* é possível retornar um *valor*.

No protocolo DHT, os *buckets* representam os nós da rede. Ou seja, cada nó é responsável por um conjunto de valores. O mecanismo de busca em uma rede DHT consiste em se aplicar a função *hash* a uma *chave* e baseado valor obtido, utilizar esta informação para percorrer o DHT (sentido horário) em busca do ID mais próximo, de tal forma que $IDNode > IdKeyHash$.

Pegamos a *chave*: *john Smith* como exemplo. Aplicando na função *hash* da Figura 2, o valor obtido é **02**. Iniciando a procura no DHT representado pela Figura 1 e supondo que o nó que está realizando a consulta é o nó 12, o nó responsável por armazenar o valor 521-8976 (caso o valor exista), é o nó 3. Sendo assim, o nó 12 dispara uma mensagem de consulta para seu sucessor (sentido horário) e esta mensagem seguirá o seguinte caminho: $15 \rightarrow 1 \rightarrow 3$. Caso o valor 521-8976 exista, este será

Figura 2: Tabela hash.



encaminhado pelo nó 3 ao nó origem 12. Para isso, na mensagem de descoberta, informações como IP:$Porta$ devem constar na consulta.

2.5 Inserção e Remoção de nós

A inserção e a remoção de nós, no protocolo DHT, devem ser convencionadas, a fim de que todo nó entenda as mensagens trocadas no momento em que um nó entre ou saia da rede. O formato destas mensagem fazem parte do protocolo de aplicação DHT que será definido e implementado pelos grupos. O protocolo de inserção de nós deve ter o seguinte comportamento:

- Nó ingressante no DHT entra em contato com o servidor *rendezvous* para obter o IP:$Porta$ referente ao *root node*;
  - caso ele seja eleito o *root node* o protocolo é encerrado;
- procura-se o sucessor do nó ingressante na rede a partir da busca explicada nos itens anteriores;
  - a partir do IP:$Porta$ do *root node*, o nó ingressante iniciará o processo de descobrir qual nó é o seu sucessor (baseado no seu ID). Este processo consiste em consultar um nó e verificar se ele é o seu sucessor. Caso não seja, o nó ingressante solicita quem é o sucessor do nó avaliado (IP:$Porta$);
- este novo nó deve ser inserido entre o sucessor achado e seu predecessor;
  - De uma forma distribuída e seguindo o protocolo DHT definido pelo grupo, o nó irá informar ao sucessor achado e seu predecessor que agora ele deve ser "encaixado"entre os dois. Por exemplo, na Figura 1, o nó 4 precisou contactar o nó 3 e informar que agora ele é o seu sucessor, da mesma forma que contactou o nó 5 e informou que agora ele era o seu predecessor.
- ele torna-se responsável por parte das chaves gerenciadas pelo nó predecessor;
- para garantir o funcionamento das buscas, a cópia das chaves deve ser feita antes de se ajustar as ligações de antecessor e sucessor do novo nó.

No projeto a ser desenvolvido será usado exatamente metade das chaves gerenciadas. Portanto, faz-se necessária a troca de mensagens entre os nós até que as tabelas hash de ambos estejam

balanceadas. Ou seja, a remoção dos itens a serem transferidos para o novo nó na tabela já existente. É recomendado que se utilize uma variável que guarde não somente o IP:$Porta$ dos sucessores mas também o ID, a fim de se descobrir rapidamente qual o IP:$Porta$ do sucessor mais apropriado para o nó ingressante.

O protocolo para remoção compreende uma ação mais simples:

- transferência, para seu predecessor, das chaves gerenciadas pelo nó a ser retirado;
- ajuste dos "ponteiros", a fim de que o predecessor aponte para o sucessor do nó retirado e vice-versa.
  - este processo de remoção é feito mediante a troca de mensagens entre os nós envolvidos;

Exemplificando:

Um nó deseja se registrar na estrutura do DHT. Para isso, ele executa os seguintes passos:

1. Entra em contato com o servidor *rendezvous*. Recebe um ID válido e também o IP:$Porta$ do *root node* da rede, caso ele exista. Caso ele seja o primeiro nó a se registrar no DHT, ele é informado de que ele é o *root node* da rede;
2. No caso do DHT já estar formado:
   - Uma mensagem deve ser enviada ao IP:$Porta$ do *root node*. Este responde com o seu ID;
   - Seguindo a regra de ordenamento (Seção 2.3), o nó passará a entrar em contato com os demais nós (IP:$Porta$) do DHT. Sempre questionando o IP:$Porta$ do sucessor ou antecessor do nó (de acordo com a regra de ordenamento), até que seu lugar no DHT seja encontrado.

2.6 Reatar

No caso de falha em algum dos nós do DHT, há a necessidade do reestabelecimento das conexões perdidas. Na Figura 1, por exemplo, caso o nó 3 falhe, é necessário reestabelecer a ligação do DHT. Ou seja, o sucessor do nó 1 deve passar a ser o nó 4 e o antecessor do nó 4 precisar ser o nó 1.

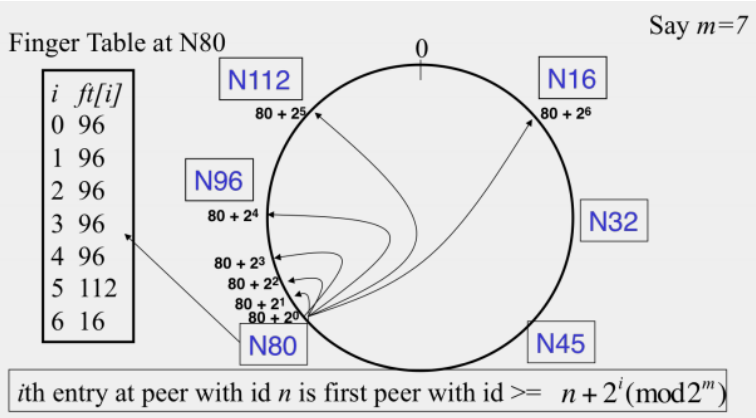
Para a implementação desta funcionalidade, dois problemas precisam ser resolvidos. Primeiro, cada nó deve realizar o monitoramento periódico dos seus vizinhos. A cada X segundos, os nós devem verificar se os seus vizinhos continuam respondendo. Ou seja, se ao enviar uma mensagem, o vizinho (sucessor ou antecessor) irá responder a esta mensagem (*keep alive*). O segundo problema é como recuperar o DHT a partir do momento em que é identificado a ausência de um vizinho. Este problema pode ser resolvido fazendo com que cada nó, além de conhecer o seu sucessor e predecessor, passe a conhecer o sucessor do seu sucessor e o predecessor do seu predecessor. Maiores informações sobre este mecanismo podem ser obtidas no livro do Kurose (2.6 *Peer-to-Peer Applications*).

2.7 Finger-Tables (atalhos)

Finger-tables refere-se à consulta para determinar a entrada cujo campo de início consiste no predecessor mais próximo da chave. Para melhorar o desempenho do DHT circular com predecessor e sucessor, se faz necessário utilizar "tabelas de alcance", pois elas permitem o acesso a diversos nós a partir de apenas um, não se limitando meramente ao predecessor e ao sucessor, diminuindo, assim, a quantidade de saltos numa busca. No projeto a ser desenvolvido, duas maneiras de criar as tabelas de alcance devem ser consideradas. A primeira, encontra-se explicitada no livro "Redes de Computadores e a Internet – uma abordagem top-down" de James Kurose, no qual o autor evidencia a necessidade de um segundo nível, explicado no item 2.6. Já a segunda abordagem, um pouco mais

sofisticada, abrange a ideia de que pode existir M nós na rede. Onde M é definido globalmente e representa a quantidade de entradas na tabela de alcance. Estas entradas são dadas a partir da fórmula: IDNode +2i(mod2m), onde i é o index da entrada na tabela. Lembre-se que ao utilizar as tabelas de alcance, sempre que um nó for inserido ou removido, deve-se balancear as tabelas de todos os nós que possuem atalho para o novo nó ou para o nó removido, a fim de se manter a eficiência do DHT.

Figura 3: DHT Circular.



3 Relatório

Um relatório final do projeto deve ser apresentado. Este relatório deve conter:

- Apresentação teórica sobre DHT, incluindo os conceitos envolvidos neste trabalho. Exemplos deverão ser apresentados;
- Documento apresentando a arquitetura do sistema desenvolvido;
- Explicação da arquitetura produzida e da relação entre dos principais componentes;
- Doxygen de todo o código produzido;
- *Screenshots* e explicação do funcionamento das funcionalidades implementadas;

4 Avaliação

A avaliação consiste em 2 etapas:

- Código e funcionamento do projeto.
  - O código e o relatório deverão ser enviados para o email: *gdbranco@gmail.com* até o dia **29/11 as 23:55h**;
  - A apresentação do trabalho deverá ser agendada pelo email: *gdbranco@gmail.com*. As apresentações acontecerão no período de **30/11 a 4/12** e **7/12 a 11/12**. As apresentações serão feitas fora do horário de aula;
  - Não serão aceitos trabalho enviados fora do prazo;
  - Trabalhos não apresentados também não serão considerados.
- Nota do Relatório.

5 Observações

As seguintes observações deverão ser consideradas pelos alunos que forem fazer o trabalho:

- O trabalho deve rodar **OBRIGATORIAMENTE** na plataforma GNU/Linux;
- O programa pode ser feito em qualquer linguagem;
- Pode-se utilizar funções hash de algum biblioteca, não é obrigatório a implementação destes.
- Não é permitido utilizar bibliotecas externas, exceto pela parte de *hashing* caso seja necessário.
- No relatório do trabalho deve conter as instruções para compilação do código e a relação de todas as bibliotecas utilizadas;
- Códigos copiados serão considerados "cola"e todos os alunos envolvidos ganharão nota zero;
- Dúvidas sobre o trabalho deverão ser tiradas **utilizando a lista de email da disciplina**. Desta forma, dúvidas comuns e esclarecimentos poderão ser respondidos uma única vez para toda a turma.