INSTITUTO SUPERIOR TÉCNICO

Sistema Entre Pares e Redes Sobrepostas

P2PBay - Peer to Peer auctions

Grupo 4

Rui Pereira	João Domingos	Karan Balu
nº 70600	nº 73847	nº 73937

1 INTRODUÇÃO

Este relatório tem como objectivo definir o planeamento do projecto P2PBay, começando por descrever a razão pela escolha do protocolo de rede *peer-to-peer*, continuando com uma explicação do que acontece quando um nó entra/sai da rede e concluindo com uma descrição da abordagem para cada função tanto para o cenário *user* como para o cenário de *manager*.

2 KADEMLIA VS CHORD VS PASTRY

Todos estes algoritmos analisados nas aulas têm como objectivo armazenar dados de forma descentralizada e escalável, sendo as pesquisas feitas <*key,value*>. Após termos analisado as três implementações sugeridas no enunciado (*Open Chord*, *TomP2P* e *FreePastry*) decidimos desenvolver o projecto com o protocolo *kademlia*. Escolheu-se este protocolo uma vez que este consegue realizar queries a qualquer peer dentro de um intervalo sendo possível escolher a melhor rota com base em latência ou realizar *queries* paralelas e assincronas. Mantem a métrica de routing XOR durante todo o processo de routing enquanto que Pastry necessita de mais uma estrutura algorítmica para descobrir o *peer* entre os *peers* que partilham um mesmo prefixo mas o próximo b-bit digito é diferente, aumentando assim o custo de *bootstrapping* e *maintenance*.

Sendo que o que se pretende implementar é um sistema de leilões onde a comunicação entre os peers deve ser o mais eficiente possível para manter a informação actualizada, decidiu-se usar a *framework TomP2P*.

3 Node Join Process

Inicialmente apenas existe um *Master Peer* que no nosso caso será o sigma.ist.utl.pt, este *well known peer* é considerado como um *entry point* da rede. Quando um novo nó se quiser juntar à rede, liga-se em primeiro lugar ao *Master Peer* onde recebe informações de configuração como os nós presentes na rede. Este usa esta informação para se ligar ao nós restantes.

4 Node leave Process

Quando um nó sai da rede, seja por logout ou seja porque houve uma falha, terá sempre todo o conteúdo que possui replicado nos nós mais próximos. Desta forma quando o utilizador voltar a efectuar o login numa outra máquina, conseguirá ter acesso a registos de bids, registos de compras efectuadas e items que colocou à venda.

5 P2PBay - Usage scenario

5.1 Login

Para efectuar o login no sistema do P2PBay, um utilizador providenciará um *username* e uma *password* sendo que nosso sistema vai considerar uma lista de *usernames* pré-definidos e caso o *username* que o utilizador digitou corresponder a um dos elementos na lista, este será reencaminhado para um menu onde terá accesso às funções descritas nas próximas secções.

5.2 Offer item to sale

Para permitir que um utilizador venda um item, será necessário criar uma class *itemSimple* cujo o construtor recebe um título, uma descrição, uma lista de objectos do tipo Bid que contem como argumentos um bid value, um peerID e um boolean para indicar se o item foi vendido ou não. De seguida associa-se uma chave a este objecto que será um hash do título do item.

5.3 Accept bid

Quando uma *bid* for aceite pelo vendedor o *item* deixa de ficar disponível para os outros poderem licitar. Numa classe *itemSimple*, já previamente mencionada, actualiza-se o valor do *boolean* chamado *'sold'* para *true* . Desta forma é possível saber quais os *items* que já foram vendidos, impedindo assim que outros peers façam bid.

Finalmente é criada uma nova instância de item cujo o construtor contem o título, conteúdo em bytes e o valor da compra, sendo este objecto enviado para o peer que efectuo a compra.

5.4 Search item

Para procurar um item um utilizador poderá usar apenas uma palavra como chave de procura, ou então usar operações booleanas como *and*, *or* e *not*. Por exemplo, procurar

pelo item que contenha as duas palavras ‘case *and* iphone‘ irá devolver todos os items que contenham a palavra ‘case‘ e a palavra ‘iphone‘. Pensamos que a solução mais correcta a usar, será o uso de *indexes*, em que indexamos os items por partes do seu nome. Por cada parte usamos como valor todos os utilizadores que contenham essa parte, no nome de um item que tenham posto à venda.

Por exemplo os utilizadores X,Y e Z puseram cada um, um item a venda:

- Utilizador X pôs o item ’Iphone Case’ à venda
- Utilizador Y pôs o item ’Iphone 5S’ à venda
- Utilizador Z pôs o item ’Samsung Case’ à venda

A tabela de *indexes* será a seguinte:

Key	Value
Samsung	Z
Iphone	X, Y
Case	Z, X
5S	Y

5.5 BID ON ITEM

Um utilizador que queira efectuar uma *bid* deverá previamente procurar o *item* ao qual deseja fazer uma *bid* obtendo desta forma, também o valor da última licitação. A única restrição em relação às licitações é o utilizador ter sempre de licitar com um valor superior à *bid* actual do *item*. Não há qualquer limitação em relação ao valor da licitação desde que cumpra a regra anterior descrita. O peer cria uma nova instância de bid onde coloca como argumentos o valor da bid e o seu *peer Address*, faz *append* deste objecto à lista e volta a disponibilizar o objecto *itemSimple* na rede. Caso um peer não consiga fazer o upload do objecto itemSimple porque dois peers tentaram fazer o bid, o peer volta a realizar o upload passado um tempo curto aleatório.

5.6 VIEW DETAILS ON ITEM

Para obter toda a informação acerca de um determinado *item*, mais uma vez é introduzida a classe *itemSimple*. Este irá conter conter certas estruturas que irão conter todos os dados acerca do *item*. Estas serão enunciadas de seguida.

- *description*: Breve descrição acerca do item.
- *name*: Nome do *item*.
- *allBidders*: Uma lista de todos os utilizadores que fizeram uma *bid* no *item*.
- *dealer*: O nome do utilizador que pôs o *item* à venda.

5.7 View purchase/bidding history

De modo a fornecer ao utilizador um histórico de todas as suas compras e bids, irá estar disponível numa classe chamada *user* uma lista que vai conter todos os *items* comprados e uma outra lista com todos os bids efectuados. O utilizador poderá consultar estas listas após ter efectuado *login* na aplicação.

6 P2PBay - Management scenario

6.1 Number of nodes running P2PBay

O número de nós que iremos usar é relevante devido ao facto de um maior número de nós proporcionar uma maior aproximação à realidade, e assim verificar com maior fiabilidade a *performance* do nosso algoritmo. Também quanto maior for o número de nós, maior é a capacidade de ter utilizadores activos, tendo em conta que em cada nó, apenas pode estar um utilizador activo. Para descobrir o número de nós activos, iremos implementar o protocolo *Gossip* no nosso projecto.

6.2 Number of users registered

O número de utilizadores será guardado localmente em cada nó, ou seja, quando um nó entra na rede, já conhece *a priori* todos os utilizadores registados.

6.3 Number of items on sale

O número de items à venda afecta de modo que quanto maior for o seu número, maior é o custo de manutenção dos *indexes* a usar na procura de items. Sempre quando é criado um item, o peer incrementa uma variável denominada numItems e disponibiliza na rede.