TP3 — Sistema *peer-to-peer*

Adler Melgaço e Ronald Davi

1 – Introdução

O trabalho consiste na implementação da funcionalidade básica de sistema chave-valor do tipo *peer-to-peer*, ou P2P. Através desse método, é possível encontrar o valor associado a uma chave específica, fazendo com que os programas que mantêm tal funcionalidade possam agir tanto como cliente quanto servidor. Tais sistemas são utilizados quando a prioridade é a descentralização de recursos e a sua heterogeneidade, visto em aplicações como BitTorrent e DNS.

2 – Arquitetura

Nesse trabalho o protocolo utilizado foi o UDP, pois o fato de ser um protocolo sem conexão garante uma entrega mais rápida quando se trata de pacotes que são compostos em sua maioria por dados, como é o caso em questão.

3 – Servent

O *servent* é o programa responsável pelo controle da troca de mensagens e o armazenamento da base de dados chave/valor, os quais estão em um arquivo de texto simples e foram implementados utilizando-se a estrutura de dados do tipo dicionário. Cada *servent* conhece o endereço de seus parceiros, embora não haja uma conexão permanente entre eles, justamente por se tratar de um protocolo UDP.

Quando se é necessário propagar uma mensagem pela rede, o *servent* precisa primeiramente verificar se a mensagem nunca foi recebida por ele, com o objetivo de evitar duplicatas, utilizando uma lista de mensagens já recebidas pelo *servent* para esse fim. Caso ele já tenha recebido a mensagem, nada é feito. Porém, caso a mensagem seja nova, o *servent* verifica se o TTL da mensagem, após decrementá-lo em uma unidade, é maior que 0. Caso isso se

cumpra, ele armazena a mensagem como recebida e retransmite a mensagem para todos os *servents* vizinhos, além de, em alguns casos, enviar uma resposta ao cliente que enviou a requisição original.

Sua implementação é feita utilizando-se duas classes: *ServentInfo* e *Message*. A primeira é encarregada de representar cada um dos *servents* e possui métodos responsáveis pela estruturação da mensagem, como a construção da lista que guarda as mensagens recebidas e a inicialização do IP. Já a segunda representa a mensagem, dispondo métodos que têm a função de repassar a mensagem para os outros *servents,* decrementar o TTL e assim por diante.

3 – Client

O programa *client* funciona como uma interface de rede para o usuário, permitindo que uma consulta por uma chave possa ser feita, além de ser possível saber como é a topologia de rede.

A implementação é realizada a partir de duas classes também: *ClientInfo* e *Message*. *ClientInfo* se encarrega de criar a estrutura do cliente, inicializando seu IP e seu PORT, além de setar o número de sequência para zero. A classe *Message* possui métodos que fazem a função de mandar as mensagens do tipo *keyReq* e *topoReq*, possuindo verificações de temporização para uma resposta à mensagem enviada chegar e imprimindo-a, após o tratamento e verificação dela.

4 – Discussão

Detalhes de implementação

O desenvolvimento do trabalho seguiu as orientações dadas pela especificação, já que ela estava mais detalhada, não havendo nenhuma decisão de implementação específica da nossa parte.

Testes

O programa foi testado com várias entradas diferentes e argumentos na linha de comando, com o objetivo de verificar as seguintes características no trabalho:

- Ausência da chave requisitada em alguns *servents*
- Não repetição de mensagens, permitindo ciclos na lista de hosts
- Funcionamento do TTL, através de *servents* inalcançáveis
- Consideração de apenas o segundo valor relacionado a uma chave, através de chaves duplicadas
- Encaminhamento correto do UDP, com mais de dois *clients* sendo executados e realizando requisições simultâneas

Tanto os arquivos *.txt* para esses testes e o *Makefile* se encontram na pasta *test*.

5 – Conclusão

O trabalho foi desenvolvido de maneira tranquila e nos serviu como um bom aprendizado a respeito da maneira como o protocolo UDP funciona e nos permitiu alcançar um grau de conhecimento maior com relação a aplicações tão usadas na Internet.

Gostaríamos de elogiar dessa vez a especificação do trabalho, já que ela estava detalhada e não dava margem para muitas ambiguidades, fazendo com que a dificuldade estivesse relacionada apenas à implementação propriamente dita, e não a uma questão de interpretação de texto.

6 – Referências bibliográficas

Funcionamento do UDP:

[1] http://www.binarytides.com/programming-udp-sockets-in-python/

[2] https://pymotw.com/2/socket/udp.html