Trabalho Prático 3 de Redes

Um sistema peer-to-peer de armazenamento chave-valor

Douglas Rodrigues de Almeida
douglasralmeida@live.com

1. Introdução

O objetivo deste trabalho é implementar um sistema peer-to-peer que permite um cliente pesquisar em uma rede entre pares que armazenam dados indexados através de chaves.

O cliente inclui um interpretador de linhas de comando básico onde o usuário pode pesquisar por uma chave ou pela topologia da rede que ele consegue alcançar.

O sistema foi desenvolvido na linguagem Python utilizando a versão 3.6 e utiliza o protocolo TCP para transmissão dos dados.

2. Arquitetura

O sistema trabalha em uma rede conhecida como P2P – peer-to-peer, uma arquitetura de sistemas distribuídos onde tarefas são divididas entre pares. Cada par é um nó interconectado na rede que compartilha informações entre si sem a necessidade de um servidor centralizado.

Figura 1 - Uma rede P2P é uma arquitetura de sistemas distribuídos

Cada host da rede entre pares é chamado de *servent* (server/client) pois atua como servidor e cliente ao mesmo tempo.

3. Servent

O servent é responsável por receber consultas de um cliente e respondê-las adequadamente. Ele possui um banco de dados de chaves e valores e uma lista de vizinhos (outros servents) conhecidos que ele se conecta para retransmitir as consultas recebidas.

As consultas são retransmitidas pela rede por uma técnica conhecida como alagamento confiável. Cada servent retransmite apenas uma única vez e, ao serem geradas, recebem um tempo de vida (TTL) limitado a 4 retransmissões. Essa técnica impede que uma consulta fique sendo retransmitida na rede indefinidamente. Para tal, cada consulta é identificada por um identificador único, pelo IP do cliente que a gerou e pela porta que o cliente recebe respostas. Quando um servente recebe uma consulta, ele

armazenada esses dados de identificação em um conjunto de dados (set) para consulta futura e impedir mais de uma retransmissão.

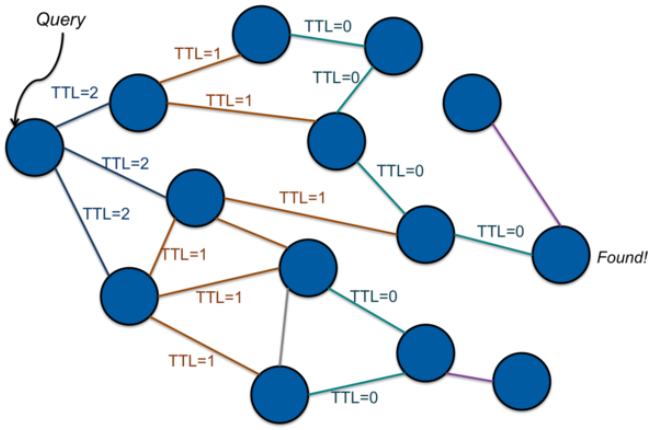

Figura 2 - O alagamento confiável impede o servent transmita a mesma consulta duas vezes e limita a quantidade de retransmissões.

Quando um servent possui a chave pesquisada ele envia o conteúdo armazenado diretamente para o cliente que realizou a consulta.

O servent precisa lidar com múltiplas conexões. Entretanto a função accept() do soquete que recebe conexões bloqueia o programa sempre enquanto uma conexão não é estabelecida. Para impedir este problema, o módulo select do Python é utilizado pois permite monitorar vários soquetes de conexão ao mesmo tempo.

A manipulação das múltiplas conexões é feita com o código abaixo:

```
while (True):
   lista = select.select(lista_soquetes, [], [])
   if not lista:
      continue
   for soquete in lista:
      if soquete is soquete_que_aceita_conexoes:
         registra nova conexao
         salva novo soquete na lista_soquetes
      else:
         recebe dados
```

O servente possui uma lista de soquetes que recebem dados de outros serventes e clientes, além do soquete que aguarda por novas conexões da rede. A função select() monitora esta lista e retorna uma sublista de soquetes que possuem dados advindos da rede para serem lidos. Se for uma nova conexão, um novo soquete é gerado e adicionado na lista. Do contrário os dados são recebidos e processados pelo programa.

4. Cliente

O cliente funciona com uma interface de linha de comando que irá receber consultas do usuário e a transmitirá para um servent onde está conectado. O cliente aceita três comandos:

- ? chave: O sinal de interrogação seguido de uma palavra fará com que o cliente consulte esta palavra na rede P2P.
- T: A letra t maiúscula fará com que o cliente consulte pela topologia da rede alcançável.
- Q: A letra q maiúscula ou o sinal EOF encerram o programa.

Ele se conecta a um único servente que será responsável por retransmitir as consultas pela rede. Após uma consulta, ele espera por 4 segundos por uma resposta e a exibe na tela. Como vários servents

podem responder, este contador é reiniciado a cada reposta que chega pela rede. Mensagens inesperadas ou com o sequencial diferente da consulta inicial são descartadas, mas uma mensagem de erro é exibida na tela. Se, após os quatro segundos, nenhuma mensagem for recebida, será exibida uma mensagem de aviso e o programa voltará a aguardar por novas consultas do usuário.

5. Discussão

Para facilitar o entendimento do código, quase todo o sistema foi implementado com classes. Durante a análise do código será notado que foram criadas classes para tudo. A lista de clientes, de vizinhos servents, o banco de dados, o gerenciador de soquetes, o gerador e o processador de mensagens, a linha de comando e tratador de dados transmitidos possuem classes próprias.

A maior dificuldade encontrada durante a implementação do sistema foi de depuração. É muito difícil depurar os erros nos servents pois eles funcionam de modo distribuído. Houve dificuldade de saber se uma informação foi transmitida, retransmitida ou recebida por um host e qual mensagem estava sendo transmitida. No início do trabalho os dados das mensagens e o texto estavam sendo transmitidos separadamente o que gerou indefinições nas consultas. Isso foi difícil de corrigir pois não havia nada que indicava que o texto não estava sendo transmitido. Apenas com tentativa e erro percebi que a melhor maneira era simplesmente concatenar ambos antes do início da transmissão.

Os dados para teste do programa estão na pasta *data*. Existem dois scripts na pasta raiz, *topologia1* e *topologia2*, que testam o sistema usando os dados de teste. O primeiro script testa se o servent está bloqueando a retransmissão repetida de consultas conforme previsto pelo alagamento confiável. Nele, são criado quatro serventes na forma de losango onde o último servent sempre receberá duas consultas repetidas. Pelos testes, apenas uma será transmitida e, se for o caso, respondida. No segundo script os servents são implementados sequencialmente para testar o tempo de vida das consultas. Existem seis servents nesta topologia, mas o último da lista nunca receberá consultas pois com o tempo de vida de 3 TTL, uma consulta alcançará, no máximo, cinco servents.