Redes de Computadores – DCC023
Trabalho Prático III – Servidor de Mensagens

Igor Dias Cangussu – 2013029980
Moises Mendes de Assis – 2014015524

1. Introdução

Neste trabalho implementou-se um simples servidor de mensagens, com funcionamento similar ao da plataforma Twitter. O programa foi estruturado no formato cliente-servidor, em que vários clientes podem se conectar ao servidor para enviar e receber mensagens. As mensagens dos clientes podem conter *tags* que indicam que outros clientes com a mesma *tag* também devem recebê-las. Para isso, foi necessário implementar a comunicação cliente-servidor, guardar uma tabela com cada cliente e suas *tags* cadastradas e permitir que clientes cadastrem e descadastrem *tags*.

Abaixo está apresentado um esquema geral do funcionamento do fluxo do programa:

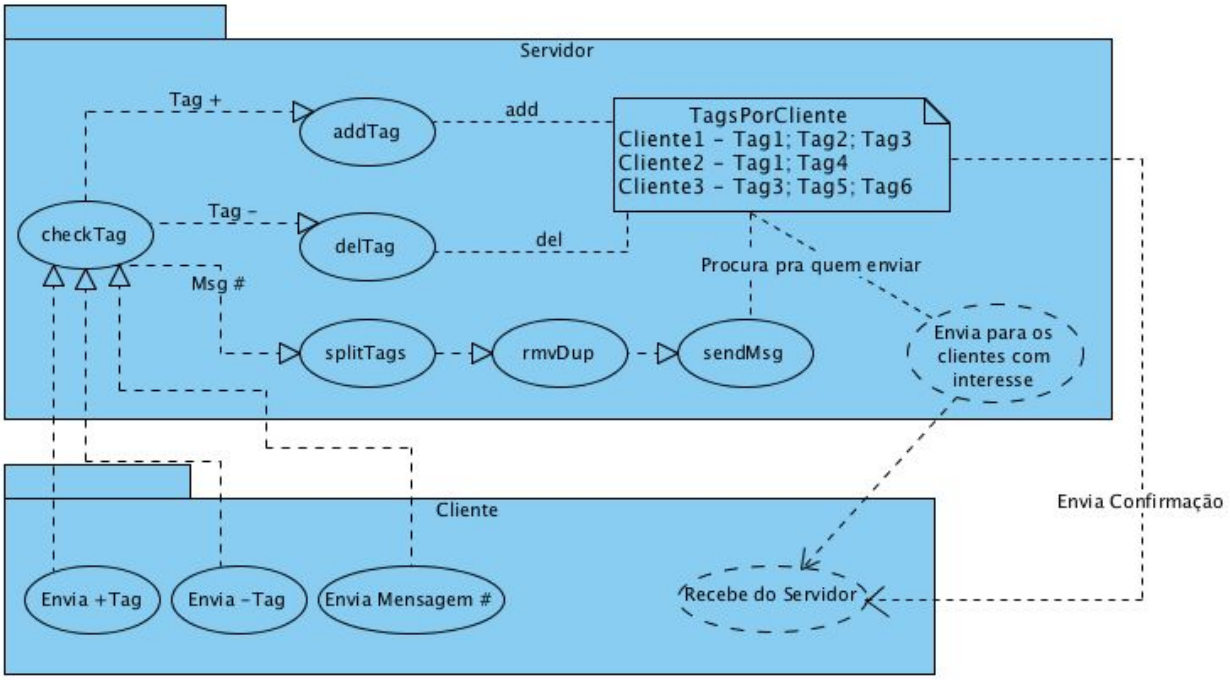

Fig. 1. Fluxograma geral do funcionamento do servidor de mensagens

2. Comunicação cliente-servidor

Os clientes e o servidor trocam mensagens por meio do protocolo UDP, que é baseado na ideia de "melhor esforço", i.e., ele não possui otimizações para garantir a entrega dos pacotes, como no TCP. O cliente se conecta ao servidor ao informar pela linha de comando a porta que ele utilizará para se comunicar e o IP e a porta em que o servidor utilizará. Já o servidor é iniciado passando apenas a porta que ele utilizará para se comunicar pela linha de comando.

O cliente utiliza a função `select` do Python para ser capaz de ler mensagens da entrada padrão (teclado) e receber dados do *socket* do servidor ao mesmo tempo. Isso evita que a execução do cliente trave em uma

dessas atividades e torna o funcionamento do Servidor de Mensagens implementado similar a um serviço real de mensagens.

O servidor estabelece uma conexão com o cliente e entra em seu *loop* principal, em que basicamente ele aguarda receber mensagens e as processa quando recebe. Essas mensagens podem ser de inserção ou remoção de *tags* ou podem ser mensagens que devem ser transmitidas aos outros clientes de acordo com as *tags* contidas nelas. Mensagens de inserção de *tags* têm um caractere **(+)** seguido da *tag* a ser cadastrada, enquanto mensagens de remoção têm um **(-)**, então elas podem ser distinguidas de mensagens de texto normais. Estas mensagens são enviadas pelo cliente ao servidor e podem conter citações de *tags*, o que é feito através de um caractere **(#)** seguido da *tag* de interesse. As mensagens com citação de *tag* são enviadas para todos os clientes que cadastraram interesse nessa *tag*, menos para o cliente que enviou a mensagem. Os passos para tratar cada caso são detalhados a seguir.

3. Inserção e remoção de *tags*

A estrutura utilizada para armazenar as informações das *tags* de cada cliente foi um dicionário (também referenciado ao longo do texto como tabela) chamado **`TagsPorCliente`**. As chaves desse dicionário são tuplas contendo IP e porta de cada cliente. E para cada chave é armazenada uma lista com as *tags* para as quais aquele cliente se cadastrou.

A função de adicionar *tags* é chamada **`addTag`**. Caso o cliente ainda não esteja presente na tabela, que ocorre quando um cliente cadastra sua primeira *tag*, ele é inserido na tabela e sua *tag* é cadastrada. Caso o cliente já esteja na tabela, verifica-se se essa *tag* já não foi inserida para aquele cliente, e só então ela é adicionada à lista de *tags* daquele cliente. É enviada então uma confirmação ao cliente, informando que a *tag* foi adicionada e qual é essa *tag*. O código dessa função é apresentado abaixo.

A função de remoção *tags* é chamada **`delTag`**. Ela primeiro verifica se a *tag* que se deseja remover foi cadastrada para o cliente em questão e caso negativo, ela retorna uma mensagem avisando que aquela *tag* não existe para aquele cliente. Caso a *tag* exista, ela é deletada da tabela *TagsPorCliente*. Nesse caso, envia-se uma mensagem ao cliente informando que a *tag* requisitada foi removida e é impressa uma mensagem na tela do servidor indicando a operação de remoção da *tag*.

4. Tratamento das mensagens normais (com e sem citações)

Quando o servidor recebe uma mensagem, ela é repassada à função **`checkTag`**. Essa função verifica se é uma mensagem de inserção ou remoção de *tags*, ao procurar pelos símbolo de mais ou de menos, ou se é uma mensagem normal. Toda mensagem normal é impressa na tela do servidor para indicar que ele a recebeu.

Como o tratamento dos casos de inserção e remoção de *tags* já foi detalhado acima, resta o caso de envio de uma mensagem normal. Essa mensagem pode ou não conter citações, que são *tags* na mensagem que vêm antecedidas de um caractere tralha (#). Assim que uma mensagem normal é identificada, ela é repassada à função **`splitTags`**, que tem o objetivo de identificar citações em uma mensagem. Ela encontra todas as *tags* presentes em uma mensagem através de todos os caracteres tralha presentes na mensagem. Essas *tags* identificadas, juntamente com a mensagem e o cliente que a enviou, são passadas à função que envia as mensagens a outros clientes, que é a função **`sendMsg`**.

Foram implementadas duas otimizações no envio de mensagens:

1. **O cliente que envia a mensagem não a recebe de volta.** Isso foi feito pensando em um serviço de mensagens como o Twitter, em que um usuário não recebe sua própria mensagens, mas os outros usuários é que a recebem.
2. ***Tags* repetidas em uma mensagem não fazem com que ela seja enviada diversas vezes, mas apenas uma.** Nesse caso, como a *tag* é igual, só é necessário enviar a mensagem aos outros clientes para cada *tag* diferente na mensagem.

Isso é feito através da implementação da função **`sendMsg`**. Quando essa função recebe as *tags* presentes em na mensagem, ela chama a função **`rmvDup`**, que remove as *tags* repetidas dentre as *tags* identificadas. Depois disso, ela percorre a tabela *TagsPorCliente* e verifica quais clientes cadastraram cada uma das *tags* identificadas. Caso um cliente tenha cadastrado uma *tag* presente na mensagem e esse cliente não foi o cliente que enviou a mensagem, a mensagem é enviada para ele. E para indicar o processo na tela do servidor, é impressa uma mensagem que informa que a mensagem foi enviada e também imprime a mensagem.

Escolheu-se imprimir diversas informações na tela para que o programador possa acompanhar o procedimento realizado pelo código e compreender melhor seu funcionamento. Também dada a ausência de testes automáticos, foram utilizados padrões próprios definidos pelo grupo para imprimir na tela essas informações.

5. Testes

Para demonstrar o funcionamento do programa, foram realizados os seguintes testes com dois clientes:

```
Cliente 1: +agua
Cliente 2: +agua
Cliente 1: +suco
Cliente 2: +refri
Cliente 2: +chuva
Cliente 1: Mensagem Teste #chuva (Cliente 2 recebe)
Cliente 1: Mensagem Teste #agua (Cliente 2 recebe; Cliente 1 não recebe por ser o emissor)
Cliente 2: Mensagem Teste2 #agua (Cliente 1 recebe agora)
Cliente 1: -suco
Cliente 2: Mensagem Teste3 #suco (nenhum cliente recebe)
```

Os resultados com os prints da tela do servidor e dos dois clientes estão apresentados na página a seguir na seguinte sequência: servidor, cliente 1, cliente 2.

Vale ressaltar que foram feitos testes com mensagens variadas, dentre elas as indicadas abaixo.

- **+#** com **##**
- **+**œ∑´®†¥ (fora da especificação)
- **#**
- **++**
- **#+** .

Todos funcionaram corretamente e como esperado. As mensagens que eram de cadastro de *tag* foram corretamente identificadas, mesmo com caracteres e símbolos diversos. Também foram identificadas corretamente *tags* com os próprios caracteres de cadastro (mais, por exemplo).

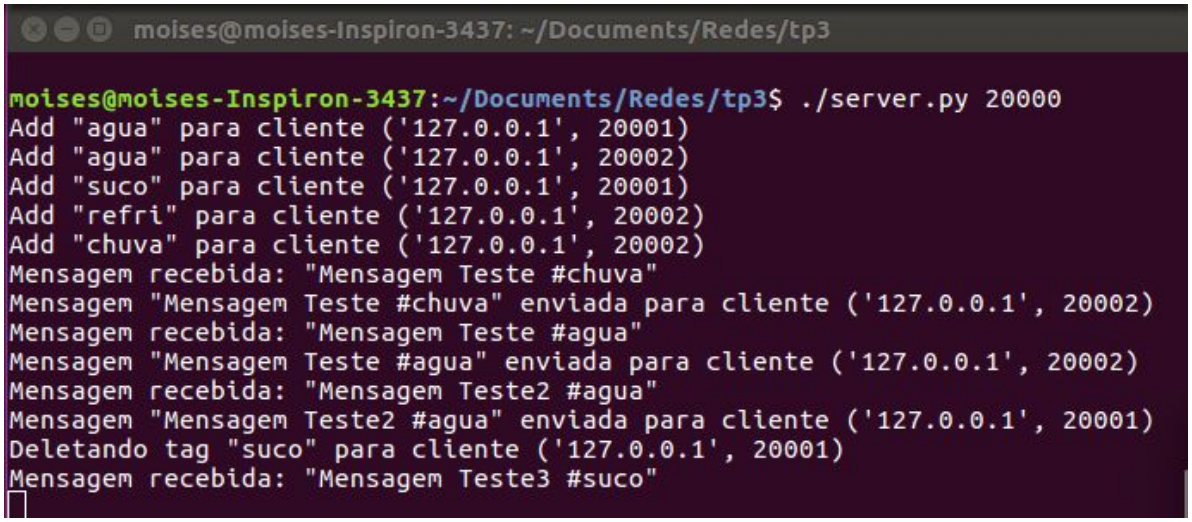

Fig. 2. Servidor com *prints* das mensagens de funcionamento

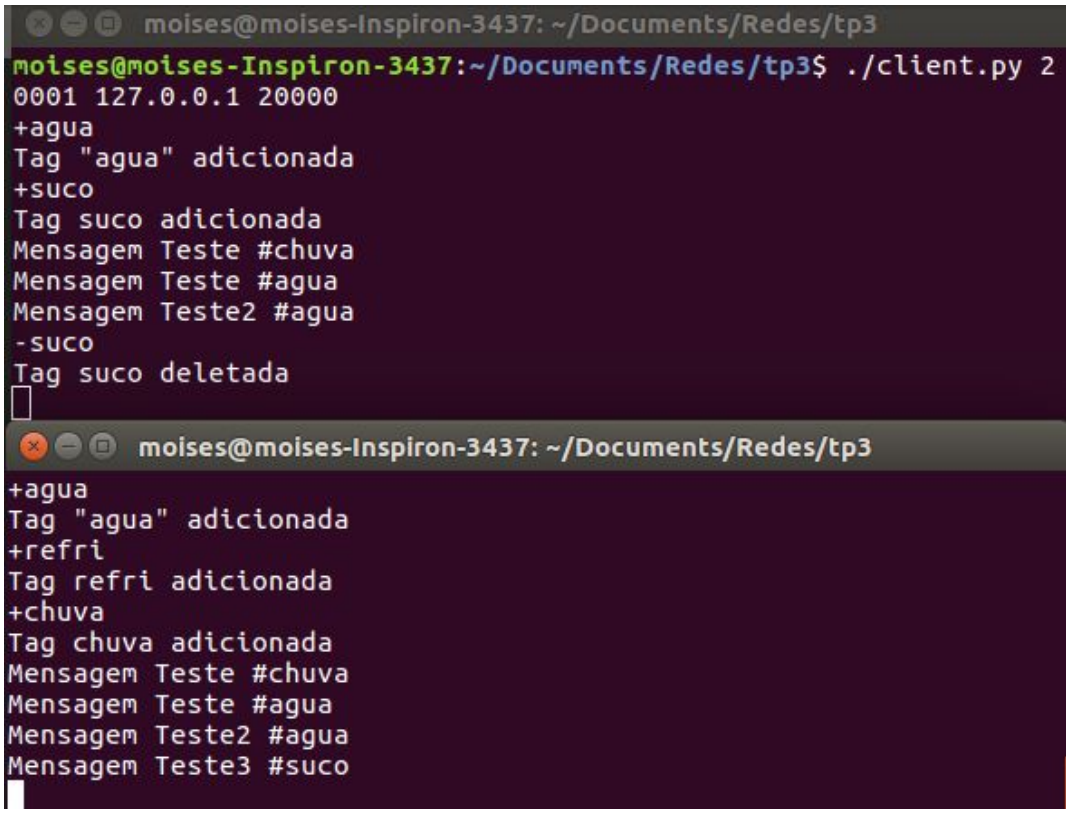

Fig. 3. Cliente 1 e 2, respectivamente, e suas mensagens enviadas e recebidas

6. Dificuldades e desafios

A principal e única dificuldade que o grupo ressalta foi a compreensão de como utilizar a função **`select`** para controlar a leitura do teclado e o recebimento de dados da rede. Outros pontos duvidosos surgiram devido à falta de definições claras na especificação, mas todas as considerações feitas foram devidamente documentadas.