**Universidade Federal de Minas Gerais**
**EE - Engenharia de Sistemas**
**DCC023 - Redes de Computadores**

# Trabalho Prático III - DCCRIP: Servidor de Mensagens

Humberto Monteiro Fialho - 2013430811
Rafael Carneiro de Castro - 2013030210

9 de julho de 2018

## 1 Introdução

Neste trabalho será implementada uma troca de mensagens entre duas partes, servidor e cliente. O servidor receberá o conteúdo das mensagens e nela estarão inclusas algumas *tags* que serão utilizadas para realizar o envio dessas mensagens aos outros clientes interessados no respectivo assunto. Para realizar essa tarefa serão aplicados conceitos de procotolo UDP e do uso da função *select()*, para a leitura de entradas por *socket* e terminal nos clientes. Cada cliente consegue se registrar para uma tag enviando para o servidor a mensagem *+tag*, substituindo a palavra *tag* pela tag desejada. Qualquer mensagem enviada por outros clientes que contenham esta tag, precedida de #, serão enviadas para este cliente registrado. Para desfazer o registro de uma tag, basta o cliente enviar a mensagem *-tag*. A linguagem utilizada para o desenvolvimento do trabalho é o *Python*.

## 2 Desenvolvimento

Quando o servidor é inicializado, ele recebe sua porta de operação. O *socket* UDP é inicializado com esta porta. Para que as menasgens recebidas tenham a identificação do remetente, a função *recvfrom* do socket é utilizada (no lugar de *recv*), para se obter o endereço de quem enviou tal mensagem.

O primeiro procedimento feito pelo servidor na chegada de uma mensagem é checar se a menasgem que chegou é a de registrou ou remoção de registro de alguma tag para algum cliente. Estas mensagens são precedidas dos sinais + ou -, conforme explicado na *Introdução*. Para se extrair das mensagens as palavras que começam com + ou -, foi utilizada a biblioteca *re* do *Python*, com a função *findall*, usada para se operar com *regex* em *strings*. Os padrões $(?:\backslash+)(\backslash w+)$ e $(?:-)(\backslash w+)$ foram utilizados para obter da *string* da mensagem, através de *regex*, todas as palavras que começam com + ou -, respectivamente. Caso a mensagem seja de registro de tag para um cliente, o endereço do cliente é guardado em um dicionário. As chaves do dicionário são as tags, e os valores dos dicionários são *sets*, que correspondem a *arrays* sem elementos repetidos, para que mensagens duplicadas não sejam enviadas para os clientes. O *set* contem os endereços dos clientes registrados para aquela tag. Caso a mensagem seja de remoção de tag, basta remover o endereço daquele cliente do *set* correspondente à tag.

Quando o servidor recebe uma mensagem que não é de adição ou remoção de registros de tags, todas as tags contidas no texto da mensagem (palavras que começam com #) são extraídas, mais uma vez com o método *re.findall*, mas agora com o *regex* $(?:\#)(\backslash w+)$. Com as tags em mãos, basta obter do dicionário de registros todos os clientes que estão cadastrados para estas tags, e enviar a mensagem para todos eles. Caso um cliente esteja cadastrado para mais de uma tag da mensagem, ele recebe a mensagem apenas uma vez. Vale salientar que, caso o cliente que enviou a mensagem esteja registrado para alguma das tags contidas na mensagem, ele também vai receber a própria mensagem.

A aplicação cliente é iniciada recebendo sua porta de operação, o IP e porta do servidor de mensagens. Como é uma aplicação que escuta entradas tanto do *scoket* (mensagens vindas do servidor) quanto do terminal (mensagens escritas pelo usuário), a função *select* é utilizada para selecionar as entradas ordenadamente, conforme especificado. Caso a entrada seja uma mensagem vinda do servidor, esta mensagem é escrita no terminal. Caso a entrada seja um comando vindo do terminal, uma mensagem é enviada para o servidor, contendo o que o usuário escreveu no terminal. A função *sendto* do *socket* é utilizada para enviar a mensagem para o endereço correto do servidor.

# 3 Conclusão

Através deste trabalho foi possível colocar em prática os conceitos dados na parte final da disciplina, trabalhando com aplicações que utilizam protocolos de transporte para o envio de dados. É possível utilizar o cliente (ou servidor) aqui desenvolvido com o servidor (ou cliente) do professor ou qualquer colega, implementados na mesma especificação, e *vice-versa*. Conclui-se que os objetivos do trabalho foram alcançados, e que os conceitos vistos em sala de aula foram devidamente praticados.