

Transmissão de Dados - Trabalho Final

Bruno Andreghetti Dantas
Universidade de Brasília

brunoandreghetti@gmail.com

1. Introdução

Este documento consiste no relatório de desenvolvimento do projeto final da disciplina de Transmissão de Dados no 2º semestre do ano de 2018, ministrada pelo professor Marcos F. Caetano e pela monitora Mariana Makiuchi.

O trabalho consiste no desenvolvimento de um sistema de chat online a partir da implementação de um protocolo na camada de aplicação. Tanto o servidor quanto o cliente devem ser desenvolvidos pelo aluno utilizando a linguagem de sua preferência.

Para o desenvolvimento desse trabalho, foi escolhida a linguagem Go, desenvolvida pela empresa Google com foco em programação concorrente e produtividade.[1] Além disso foi utilizada a plataforma Docker para virtualização do servidor e dos clientes a fim de demonstrar o funcionamento do sistema.[2]

2. Desenvolvimento

O desenvolvimento foi guiado pelo documento de especificações provido no ambiente Aprender. As funcionalidades desenvolvidas estão pontuadas nas subseções a seguir, assim como a máquina de estados referente ao servidor e ao cliente.

2.1. Protocolo

Para estabelecimento da comunicação cliente-servidor, foi desenvolvido um protocolo para a troca de mensagens. O protocolo consiste na construção, serialização, envio e decodificação das mensagens.

A estrutura das mensagens foi definida da seguinte forma:

```
type ChatMsg struct {  
    Command int  
    Status  int  
    Payload []byte  
}
```

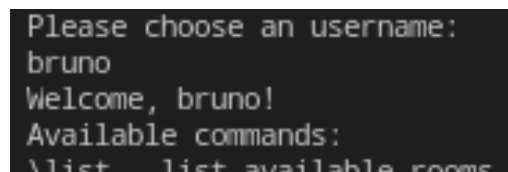
Onde os campos `Command` e `Status` são inteiros definidos no arquivo do pacote `msgs`, `msgs.go`, e o campo `Payload` é um vetor de bytes cujo significado muda de

acordo com o contexto definido pelos campos `Command` e `Status`.

A estrutura é então serializada em formato JSON [3] e enviada via TCP. Na aplicação de destino, o JSON é decodificado e a estrutura é tratada de acordo com seu conteúdo.

2.2. Cadastro de Usuários

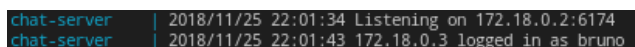
O cadastro do usuário se dá no momento da inicialização da aplicação cliente. A aplicação pede para o usuário seu nome e em seguida envia requisição de cadastro para o servidor.



```
Please choose an username:  
bruno  
Welcome, bruno!  
Available commands:  
\list - list available rooms
```

Figura 1: Exemplo de cadastro de usuário do ponto de vista do cliente.

Do ponto de vista do servidor, é verificada a validade do nome de usuário e retornado erro correspondente caso não seja possível realizar o cadastro. Caso tudo dê certo, o usuário é cadastrado e uma mensagem de log é gerada para indicar o cadastro do usuário.



```
chat-server | 2018/11/25 22:01:34 Listening on 172.18.0.2:6174  
chat-server | 2018/11/25 22:01:43 172.18.0.3 logged in as bruno
```

Figura 2: Log registrado no servidor pelo cadastro de usuário.

2.3. Entrar e sair de uma sala existente

A entrada em uma sala é realizada através do comando `join <room-name>`, onde o usuário seleciona a sala onde ele quer entrar.

Do ponto de vista do servidor, é feita a associação entre o usuário e a sala, permitindo que o usuário obtenha as mensagens daquela sala.

Na aplicação do cliente, é aberta uma rotina concorrente para requisitar as novas mensagens do servidor a cada

meio segundo enquanto o usuário estiver na sala, podendo o usuário também enviar mensagens para esta sala simplesmente digitando-as no terminal.

Para sair de uma sala, é utilizado o comando `\leave`.

2.4. Buscar informações sobre as salas

Para obter informações sobre as salas, o usuário deve utilizar o comando `\list [room-name]`. Caso o usuário deseje saber quais as salas existentes e quantos usuários estão em cada uma delas, basta digitar o comando sem argumentos. Caso queira saber a lista de usuários presentes em uma sala específica, deve selecionar a sala pelo nome através do argumento opcional `[room-name]`.

2.5. Criação e deleção de salas

A criação e deleção de salas é feita através dos comandos `\create <room-name>` e `\delete <room-name>`. O usuário é avisado do resultado da operação por parte do servidor, que retorna uma mensagem de "ok" no campo Status caso a sala tenha sido criada ou deletada.

É importante destacar que as salas apenas podem ser deletadas caso estejam vazias, retornando um erro para o usuário caso não estejam.

2.6. Logs

É mantido nos registros do Docker um histórico das requisições executadas, tal como visto na Figura 3. Esses registros ficam no container do servidor e são facilmente acessíveis através do comando `docker logs chat-server`.

```
chat-server 2018/11/26 00:03:12 Listening on 172.18.0.2:6174
chat-server 2018/11/26 00:03:21 172.18.0.3 logged in as alice
chat-server 2018/11/26 00:03:33 (172.18.0.3) alice created a new room named newroom
chat-server 2018/11/26 00:03:47 172.18.0.4 logged in as bruno
chat-server 2018/11/26 00:03:55 (172.18.0.3) alice joined newroom
chat-server 2018/11/26 00:04:05 (172.18.0.4) bruno requested a list of all rooms
chat-server 2018/11/26 00:04:08 (172.18.0.4) bruno requested a list of the users in newroom
chat-server 2018/11/26 00:04:13 (172.18.0.4) bruno joined newroom
chat-server 2018/11/26 00:04:13 (172.18.0.3) alice got newroom room's messages from 1 up to message number 2
chat-server 2018/11/26 00:04:17 (172.18.0.4) bruno posted hey!
chat-server 2018/11/26 00:04:18 (172.18.0.3) alice got newroom room's messages from 2 up to message number 3
chat-server 2018/11/26 00:04:18 (172.18.0.4) bruno got newroom room's messages from 2 up to message number 3
chat-server 2018/11/26 00:04:25 (172.18.0.3) alice left newroom room
chat-server 2018/11/26 00:04:25 (172.18.0.4) bruno got newroom room's messages from 3 up to message number 4
chat-server 2018/11/26 00:04:30 (172.18.0.3) alice deleted empty room named general
chat-server 2018/11/26 00:04:34 (172.18.0.3) alice logged out
chat-server 2018/11/26 00:04:39 (172.18.0.4) bruno logged out
```

Figura 3: Exemplo de log de várias requisições.

3. Conclusão

O chat, apesar de funcional, tem espaço para diversas melhorias. Entre elas:

- Simplificar e aprimorar o gerenciamento dos usuários conectados
- Suporte a salas privadas
- Suporte a envio de arquivos

- Melhorar a interface com o usuário no terminal ou interface gráfica
- Tornar o protocolo mais eficiente, aumentando o percentual de dados úteis do campo Payload uma vez que o formato JSON torna os dados "inchados".
- Distinguir usuários com permissão de administração das salas, podendo utilizar o comando `\kick`, por exemplo.

Apesar disso, o trabalho foi bem sucedido na implementação de um serviço de chat através do qual usuários podem se comunicar por texto através da internet.

Referências

- [1] Site oficial da linguagem Go. golang.org.
- [2] Site oficial da plataforma Docker. docker.com.
- [3] Site oficial do padrão JSON. www.json.org.