Universidade Federal de Alagoas
Instituto de Computação
Redes de Computadores I: Trabalho de Implementação

Professor: Leandro Melo de Sales
Alunas: Bruna Leal Torres Silva e Eirene de Oliveira Fireman

1.Funcionalidades

Nossa aplicação consiste em uma simulação simples de jogo entre o cliente e o servidor. Nesse contexto, o objetivo do usuário é vencer um monstro através de comandos simples, escolhidos a cada rodada, como:

A - ataque;
AE - ataque especial;
C - cura;
D - desistir;
? - em caso de dúvida sobre os comandos;

Tanto o monstro quanto o usuário iniciam com 100 de vida e o jogo termina quando alguém morre(vida = 0) ou quando há desistência por parte do cliente.
A lógica principal do jogo ocorre no servidor e, como essa lógica depende dos comandos indicados pelo usuário, que é nosso cliente, é necessária a troca de mensagens entre eles. Assim, o servidor faz a requisição ao cliente e este envia os comandos através de mensagens.

2.Possíveis melhorias

Com o uso, é normal que tratamentos de erros fiquem mais evidentes. Trazendo isto para nosso contexto, acreditamos que podemos testar mais o nosso código a ponto de prever prováveis bugs ou falhas em nossa lógica. Também gostaríamos de ter implementado uma interface gráfica, mesmo que simples, a fim de dar mais dinamicidade ao jogo. Além disso, seria interessante prover a disputa entre dois usuários diferentes, ou seja, dois clientes representando dois usuários adversários jogando no mesmo servidor.

3.Dificuldades

A principal dificuldade que tivemos foi em fazer o envio e recebimento dos dados entre o servidor e o cliente, pois acontecia deles ficarem armazenados em variáveis que não eram a correspondente, ou seja, não estava recebendo corretamente a referência o que acabou gerando um grande problema na hora de usá-las. Para resolver, compilamos os dados em um único pacote e enviamos (antes estávamos enviando os dados aos poucos). Do outro lado, implementamos um “decodificador” que ficou responsável por separar os dados e armazenar em suas respectivas variáveis sempre na mesma ordem para manter o controle. Lado a isto, nossa

primeira versão de código consistia em um servidor que fazia três requisições para o cliente durante uma iteração no loop. Percebendo que isto estava sendo custoso, decidimos reduzir esse número de requisições a uma.