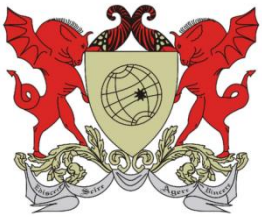

Universidade Federal de Viçosa – Campus Florestal

Curso Bacharelado em Ciência da Computação

Redes de computadores - CCF 452F

RELATÓRIO DO TRABALHO PRÁTICO 02

REDES DE COMPUTADORES

SOCKETS DE BERKELEY

Cíntia Santos de Resende Silva Costa 1810

Samuel Jhonata Soares Tavares 2282

Florestal-MG

2017

Introdução

O presente trabalho tem como objetivo definir um protocolo da camada de aplicação usando *Sockets de Berkeley*, para colocar em prática as teorias estudadas em sala de aula.

Para isto, deve ser feito um jogo, *Pedra-Papel-Tesoura*, onde usaremos os protocolos **IP** na camada de redes e **TCP** na camada de transporte.

Nesta aplicação, os dois jogadores são os clientes e o servidor faz o controle das regras entre eles.

O jogo consiste em:

- Pedra ganha de tesoura
- Tesoura ganha de papel
- Papel ganha de pedra
- Símbolos iguais denotam empate.

Cada jogada de um cliente deve ser enviada ao servidor e esse deve repassar a informação para o outro cliente, certificando que nenhuma jogada foi ilegal e que os jogadores só saberão a jogada um do outro quando os dois já tiverem feito suas respectivas jogadas. O jogo terá fim quando um dos jogadores conseguir somar 3 pontos e o servidor deve mostrar para cada um quem foi o vencedor.

A seguir é mostrado um fluxo de como se dará o jogo:

1. Servidor é iniciado e espera a conexão dos clientes (jogadores);
2. Os clientes são iniciados e pedem permissão ao servidor para jogar;
3. Servidor aceita dois e somente dois jogadores e a partida começa pelo jogador que foi aceito primeiro;
4. Os jogadores efetuam suas jogadas e o servidor anuncia o vencedor da rodada, repetindo este passo até que um dos jogadores vença;
5. Ao chegar ao fim do jogo, ou seja, quando um dos jogadores completar 3 pontos, o jogo termina, os jogadores são avisados sobre quem foi o vencedor e a conexão é desfeita, voltando para o passo 1.

Desenvolvimento

Para o desenvolvimento deste trabalho, foi utilizada a linguagem Java, a biblioteca *Sockets* e a *API Java Swing*. Também foi criada uma biblioteca *ClienteServidor*, com as classes descritas abaixo:

- Classe *Servidor*:

Para o servidor foi utilizado a classe *ServerSocket*, da Biblioteca *Sockets*. Foram implementados os métodos:

public Servidor(): Construtor da classe.
publicbooleanisServidorEstabelecido(): Método que verifica se a conexão do servidor está estabelecida no momento.
publicintgetPorta(): Método que retorna o número da porta em que o servidor está conectado.
publicintqtdClientes(): Método que verifica quantos jogadores estão conectados ao servidor.
publicbooleanestabeleceServidor(): Método responsável por estabelecer o socket de conexão do servidor.
publicbooleanaceitaConexao(): Método que espera a conexão dos clientes ao servidor.
privatebooleanestabeleceComunicacao(): Método responsável por estabelecer a comunicação do servidor com os clientes, conectando as entradas e saídas de comunicação.
privatebooleanescolheCliente(): Método que escolhe qual cliente deve fazer comunicação com o servidor no momento.
privatebooleanenviarMensagem(): Método responsável por enviar mensagens para o cliente escolhido no método anterior.
privateStringrecebeMensagem(): Método responsável por receber mensagem enviada pelo cliente escolhido.
publicStringfazComunicacao(): Método responsável por fazer a comunicação do servidor com o cliente escolhido, onde primeiro recebe a mensagem do cliente e depois envia outra mensagem para ele.
publicbooleanencerraConexao(): Método responsável por encerrar a conexão cliente/servidor no fim de cada jogo ou quando algo inesperado ocorrer durante o jogo.
publicbooleanencerraServidor(): Método responsável por encerrar o servidor.

- Classe *Cliente*:

Para o cliente foi utilizado a classe *Socket*, da Biblioteca *Sockets*. Foram implementados os métodos:

public Cliente(): Construtor da classe, que recebe a porta e o host em que o cliente deverá se conectar.
publicintgetPorta(): Método que retorna a porta em que o cliente está conectado (ou deseja se conectar).
publicStringgetHost(): Método que retorna o host em que o cliente está conectado (ou deseja se conectar).
publicbooleanpedeConexao(): Método que solicita iniciar uma conexão com o servidor(usado pelo cliente quando este quiser entrar no jogo).
privatebooleanestabeleceComunicacao(): Método responsável por estabelecer a comunicação do cliente com o servidor, conectando as entradas e saídas de comunicação.

privatebooleanenviarMensagem(): Método responsável por enviar mensagens para o servidor.
private String recebeMensagem(): Método responsável por receber mensagem enviada pelo servidor.
public String fazComunicacao(): Método responsável por fazer a comunicação do cliente com o servidor, onde primeiro envia mensagem ao servidor e depois recebe outra mensagem do mesmo.
publicvoidencerraCliente(): Método responsável por encerrar o cliente.

Foram criados outros dois projetos (*ProgramaCliente* e *ProgramaServidor*), onde ambos utilizam a biblioteca *ClienteServidor* descrita acima e a *API Java Swing*, para a elaboração da interface gráfica.

- Projeto *ProgramaServidor*:

Apresenta as classes *Jogo* (onde são implementadas as regras e o funcionamento do jogo) e *ServidorJogo*(com a interface do servidor, onde ele e o jogo são criados).

- Projeto *ProgramaCliente*:

Apresenta as classes *InterfacePrincipal* (onde o jogador escolhe seu nome, host e porta aos quais ele deverá conectar ao servidor, para iniciar o jogo) e *InterfaceJogo* (com a interface do jogo, que é iniciada somente depois de os dois jogadores terem se conectado com o servidor onde ele e o jogo são criados).

Na *InterfaceJogo* é onde o jogo acontece para o cliente, onde o mesmo recebe as mensagens do servidor e envia as suas jogadas, sendo mostrado o placar do jogo e as opções de jogada, bem como, em tela dividida, a sua jogada e a jogada do seu oponente (somente ao fim da rodada).

Conclusão

Durante a execução deste trabalho, foi possível ver, de forma prática, como se dá a conexão cliente/servidor e como o servidor gerencia as interações com seus clientes, bem como um maior aprendizado sobre outras teorias estudadas no decorrer da disciplina.

Ao se fazer este trabalho, foi preciso uma maior abstração, uma vez que este trabalho não trata de problemas relacionados com sistemas distribuídos, com o intuito apenas de dar aos alunos uma visão de como funciona as interações entre máquinas conectadas em uma rede de computadores.