UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

Trabalho de Redes: Urna Eletrônica Utilizando Sockets

Nome	**Número USP**	**email**
Felipe Kazuyoshi Takara	8921026	felipe.takara@usp.br
Leonardo Urbano	8532280	leonardo.urbani@usp.br
Jefferson Ihida	8922368	jefferson.ihida@usp.br

Conteúdo

1 – Introdução

Este trabalho tem como objetivo a aplicação prática da matéria referente a sockets. Nele é feita a implementação de uma urna eletrônica, sendo construído tanto o cliente como o servidor, ambos possuindo duas versões de implementação: JAVA e C. O servidor será executado no cluster do cosmos e será feito uma conexão entre ele e o cliente.

Uma urna eletrônica (coletor eletrônico de votos) é uma máquina utilizada nas eleições do Brasil que registra os votos realizados pelos eleitores, é um facilitador durante as eleições, pois simplifica a maneira de votar e de computar votos e agiliza todos o processo de decisão dos candidatos eleitos. A projeto implementado é uma versão simplificada, que busca a compreensão da matéria de redes sendo aplicada a um problema real e palpável.

2 – Fundamentação teórica

Os principais conceitos utilizados foram comunicação via socket e transferência de dados pelo protocolo TCP.

2.1 – Sockets

É uma interface, uma porta local ao host que é utilizado pela aplicação e controlado pelo Sistema Operacional, e através dele o processo de aplicação envia e recebe mensagens para outro processo de aplicação.

O socket é utilizado, em suma, para enviar e receber dados, sendo que possui um endereçamento único na máquina ao qual pertence, ou seja, o diente possui o endereço (o “localhost”) que deve ser informado junto com a porta que se conectará, como por exemplo, a porta “2525”, e após isso a comunicação desta aplicação acontecerá através desta porta combinada com o endereço, da forma “localhost”, ”2525”.

2.2 – Transferencia de Dados TCP

TCP é um protocolo da camada de transporte sobre o qual se assentam a maioria das aplicações cibernéticas. O TCP é um protocolo full duplex (possível transferir dados simultaneamente em ambas direções) e orientado à conexão, significando que há o estabelecimento de conexão ponto a ponto (entre origem e destino), sendo que para o estabelecimento e finalização da conexão é utilizado o mecanismo "Handshake". Além disso, o TCP utiliza-se de várias técnicas para proporcionar uma entrega confiável dos pacotes de dados, além de realizar controle de fluxo e de congestionamento.

3 – Detalhamento da Solução

3.1 – Linguagem JAVA

Foram feitos dois projetos, uma para o cliente e outro para o servidor.

O projeto "Cliente" foi implementado com três classes: classe "Candidato", classe "Cliente" e classe "EntradaTeclado". A classe "Candidato" possui a estrutura base do candidato da eleição, seus atributos são os mesmo que foram especificados no enunciado do trabalho. A classe "EntradaTeclado" é uma classe auxiliar para ler entradas do teclado. Por fim a classe "Cliente" é a principal classe do projeto, possuindo as opções da uma descritas a seguir.

Opção "999": Carrega à lista de servidor

Opção "888": Enviar lista para servidor caso tenha sido feita ao menos uma votação

Opção "1": Permite à votação caso já tenha sido carregada à lista

Opção "2": Permite votos brancos

Opção "3": Permite votos nulos

Ao inicializar o cliente uma nova conexão é estabelecida e espera-se o código "999" por parte por cliente e após carregar a lista, a conexão é encerrada. Só é permitido votar caso tenha sido carregada a lista de candidatos de servidor e só é permitido enviar a lista para o servidor caso tenha feito uma votação.

Após a opção "888" uma nova conexão é aberta, o cliente envia os dados que foram gerados durante a execução para o servidor e encerra à conexão.

O projeto servidor também possui a classe “LerEntrada” (análoga a classe “EntradaTeclado”) , a classe “Candidato” (igual a classe de mesmo nome de projeto cliente, porém esta possui o vetor de candidatos declarado estaticamente), a classe “Metodos_Servidor”, que possui as implementações dos “opcode” “999” e “888”, a classe “Servidor” que extends Threads e chama os métodos relacionado aos “opcodes”, e a classe “main()”, que lê a porta de conexão e possui um loop infinito que abre uma nova thread após o término da anterior.

Os votos nulos e brancos, assim como os candidatos e seus respectivos votos, são guardados em variáveis estáticas que são atualizadas após receber os dados dos clientes, e após a finalização do cliente a atualização dos votos é impressa na tela.

3.2 – Linguagem C

Também foram feitos 2 projetos, um cliente e um servidor. Os mesmos padrões de “opcode” foram utilizados para versão C:

Opção “999”: Carrega à lista de servidor

Opção “888”: Enviar lista para servidor caso tenha sido feita ao menos uma votação

Opção “1”: Permite à votação caso já tenha sido carregada à lista

Opção “2”: Permite votos brancos

Opção “3”: Permite votos nulos

Porém devido a dificuldade de implementação houve algumas mudanças com relação ao que foi descrito no enunciado do trabalho. Todas as operações do cliente são realizadas em uma única conexão, e, após o envio do opcode “888” a conexão é encerrada. Assim como na versão JAVA o servidor está rodando em um loop infinito, e tenta iniciar uma nova conexão após o encerramento da conexão anterior com o cliente

Para a implementação de socket em C foram utilizadas as bibliotecas e “sys/types.h” , “sys/socket.h”, “netinet/in.h” , “arpa/inet.h” para à manipulação de sockets em C. Foi feito uma estrutura “candidatos” similar ao feito em código JAVA, com os atributos exigidos na descrição de trabalho, além disso também é armazenado, para ter registro da votação total, a quantidade de votos brancos e nulos, podendo assim saber para onde a totalidade dos votos foi distribuída.

O código servidor está dividido em “candidato.c”, “candidato.h”, “main.c”, “servidor.c”, “servidor.h”, onde a “main()” simplesmente inicia algumas variáveis e chama à função servidor, que é o servidor.c, que possui toda a implementação da conexão e o candidato.c, que possui funções para criar candidatos. O código cliente está dividido em “candidato.c”, “candidato.h” e “main.c”, onde a implementação de conexão está na “main()”.

Devido a dificuldade de implementação o grupo optou por não utilizar threads em C, já que as tentativas de uso se mostraram falhas e frustrantes. Apesar do grupo estar ciente da necessidade de utilização de threads em conjunto com socket, até o final de prazo não foi possível realizar sua implementação de forma efetiva e sem erros.

4 – Dificuldades

Devido a não familiaridade com a programação em sockets, muitos problemas surgiram durante a implementação. Erros referentes a comunicação via sockets foram frequentes, mas foram contornados.

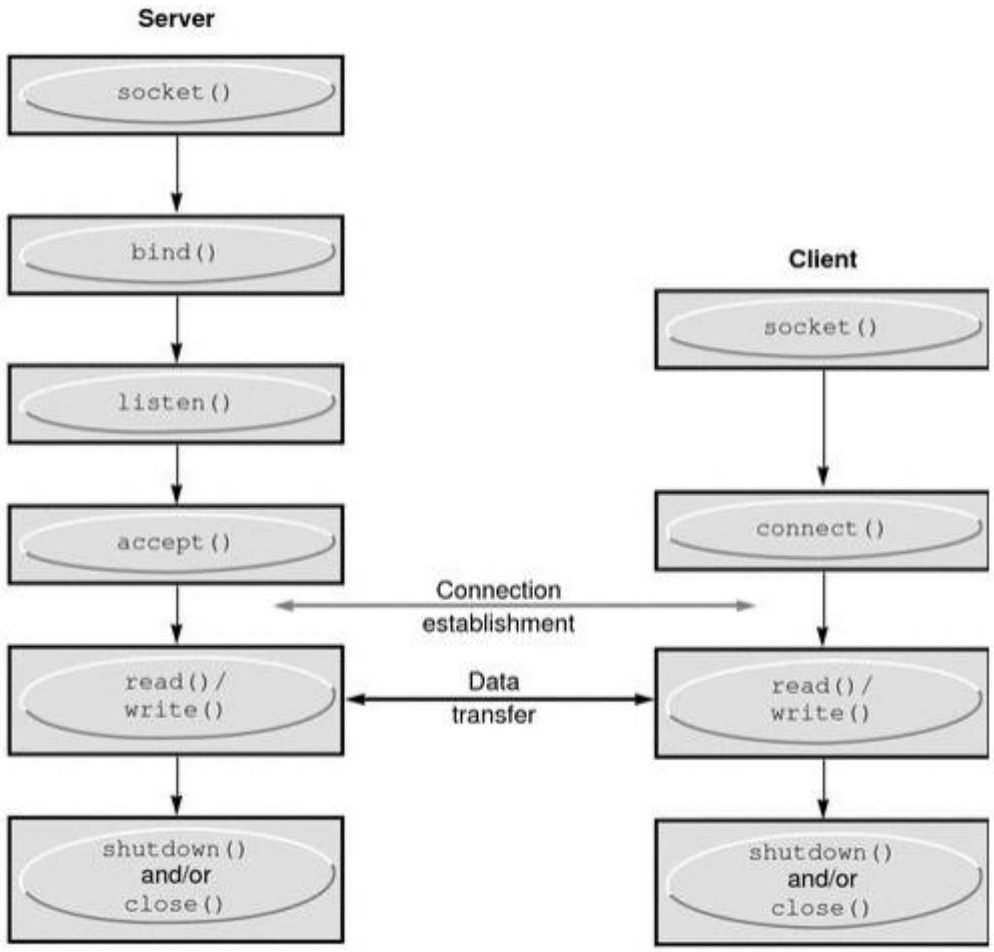
Pode-se citar como a principal dificuldade, em relação a implementação JAVA, a utilização de variáveis de escrita e leitura em socket que recebiam diretamente “socket.getInputStream()”. Isso deve-se às suas limitações, como a necessidade de transmitir e ler os dados em “bytes”, não permitindo passar diretamente números maiores que 9, e em vários casos perdendo informações do buffer durante o envio de dados o que fazia o programa não avançar e ficar eternamente esperando o recebimento de dados via socket. A maior partes dos problemas, se não todos, foram resolvidos após utilizar uma variável “DataOutputStream”, que permitia o envio e leitura direto de números inteiros e corrigiu o erro do programa não receber os dados enviados e facilitou a transferência de dados entre servidor e cliente. Mesmo assim ainda foram utilizados ambos os tipos de variáveis.

Quanto a implementação em C a dificuldade foi a própria linguagem, por ser uma linguagem rústica acaba proporcionando vários empecilhos para a implementação de socket, além da necessidade de utilizar bibliotecas não compatíveis entre sistemas linux e windows. O processo de envio e recepção de mensagens foi um problema, a utilização de variáveis das bibliotecas de sockets não era de fácil assimilação, com nomes e DEFINES estranhos e de difícil compreensão.

5 – Metodologia, Ambientes e Ferramentas Utilizadas

Para a implementação java foi utilizado a IDE Netbeans em ambiente windows, que se mostrou um grande facilitador durante o desenrolar do projeto.Para a implementação C foi utilizado o editor Visual Code e o compilador gcc em ambiente linux.

6 – Modelagem de sistema



https://docs.oracle.com/cd/E19120-01/open.solaris/817-4415/images/7099.gif

7 – Conclusões

A utilização de sockets para a implementação de um sistema Cliente - Servidor é um modo prático e eficaz para a escrita e leitura de dados entre ambas aplicações. Em linguagens mais recentes a existência de uma biblioteca robusta facilita à implementação de sockets, com funções de fácil assimilação e inúmeras possibilidades para envio e leitura de dados. Porém para um funcionamento real do sistema é necessário utilizar o conceito de sockets em conjunto com Threads de modo a permitir o estabelecimento de múltiplas conexões.

8 – Execução do código

8.1 – Linguagem Java

O código foi compilado em NetBeans e para executar basta dar dois cliques no .jar.

8.2 – Linguagem C

Para compilar os código basta dar o comando “Make” no terminal tanto do servidor quanto do cliente. Para rodar o cliente basta digitar “./cliente” e o servidor basta digitar “./servidor”.

9 – Referências Bibliográficas

9.2 – Sites

9.2.1 - Voto branco x voto nulo: saiba a diferença

http://www.tre-es.jus.br/imprensa/noticias-tre-es/2014/Outubro/voto-branco-x-voto-nulo-saib a-a-diferenca.html

9.2.2 - Programming Interfaces Guide

https://docs.oracle.com/cd/E19120-01/open.solaris/817-4415/6mjum5sou/index.html

9.2.3 - Bibliotecas e funções

http://www.cplusplus.com/

9.3 – Outros

Slides da matéria e exemplos de códigos disponibilizados pelo monitor.