Universidade Federal do Ceará – Campus de Quixadá
Disciplina: QXD0043 - Sistemas Distribuídos
Cursos: SI, RC, ES, CC e EC
Professor: Rafael Braga

Trabalho 1 – Comunicação entre processos (Capítulo 4)

Sockets e Streams

1. Um registro de uma pessoa é constituído de um nome, seu CPF e sua idade. Crie uma classe Pessoa que represente esta informação. Crie uma subclasse de OutputStream chamada PessoasOutputStream que envia os dados de um conjunto (array) de Pessoas, seguindo a seguinte regra:

a) Primeiro envie o número de pessoas que terão dados enviados pelo stream;
b) Depois para cada pessoa, deve ser enviada o número de bytes utilizados para gravar o nome da pessoa, o nome da pessoa, seu CPF e sua idade.
c) O construtor da subclasse deve receber como parâmetros: (i) um array de Pessoas que representam os dados a serem transmitidos e um OutputStream de destino, para onde as informações do conjunto de pessoas devem ser enviadas.
d) Teste sua implementação utilizando como destino a saída padrão (System.out).
e) Teste sua implementação utilizando como destino um arquivo (FileOutputStream).
f) Teste sua implementação utilizando como destino um servidor remoto (TCP).

2. Crie uma subclasse para InputStream chamada PessoasInputStream que lê os dados gerados pelo stream do exercício anterior.

a) O construtor da subclasse deve receber como parâmetro um InputStream de origem, de onde as sequências de bytes serão lidas.
b) Teste sua implementação utilizando como origem a entrada padrão (System.in)
c) Teste sua implementação utilizando como origem um arquivo (FileInputStream).
d) Teste sua implementação utilizando como destino um servidor remoto (TCP).

Serialização Java

3. Implemente um serviço remoto através da comunicação cliente-servidor. A comunicação entre cliente e servidor deve ser implementada via *sockets* (TCP ou UDP) que trocam fluxos de bytes. Cada estudante deve definir um serviço remoto, sugestões.

As estruturas de dados devem ser 'serializadas' para serem enviadas em mensagens, ou seja, deve ser feito o empacotamento e desempacotamento das mensagens no lado cliente e no lado servidor, ou seja:

- O cliente deve empacotar a mensagem de **request** antes de enviar para o servidor;
- O cliente deve desempacotar a mensagem de **reply** enviada pelo servidor**;**
- O servidor deve desempacotar a mensagem de **requisição** do cliente;
- O servidor deve empacotar a mensagem de **reply**e enviar para o cliente;

Representação externa de dados

4. Implemente uma aplicação distribuída que suporte um sistema de votações. O envio de votos têm um prazo máximo, finalizado esse tempo o servidor não aceita mais votações e calcula o total de votos, respectivas percentagens e candidato ganhador. O eleitor começa a votação através de um login. Em resposta, o servidor envia-lhe uma lista de candidatos que estão em votação, o que permitirá ao eleitor votar em um determinado candidato. Além dos eleitores, existem também os administradores do sistema que têm a capacidade de introduzir e remover candidatos para votação e que poderão enviar notas informativas para os eleitores.

O login, envio da lista de candidatos e de votos deverá ser implementada usando comunicação *unicast* com API de *sockets* TCP em Java, enquanto que toda a comunicação *multicast* será feita utilizando *sockets* UDP. O multicast será utilizado exclusivamente para as notas informativas enviada pelos administradores do sistema. O servidor deverá ser *multi-threaded*.

Para a representação externa de dados nas chamadas remotas (métodos, argumentos e resultados), sugere-se que seja implementada através de *protocol buffers*. Contudo, versões em **XML** ou **JSON** também são aceitas.

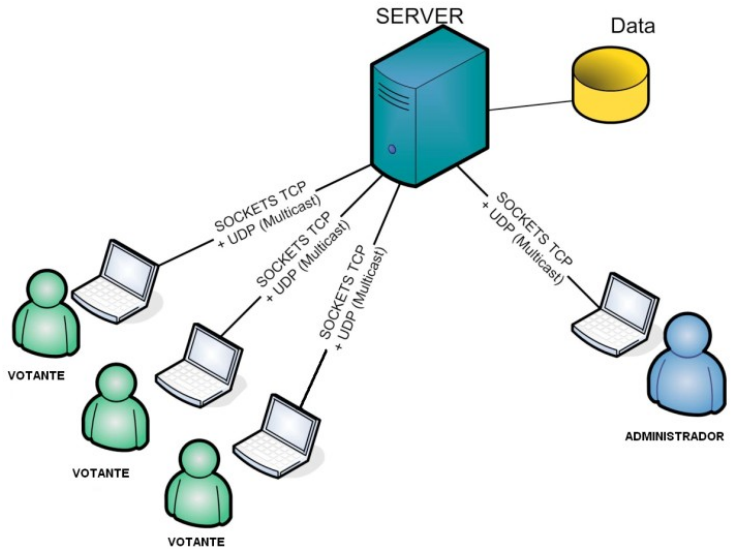

Figura 1: Arquitetura da aplicação, utilizando Java Sockets

Bom trabalho!