Tarefa 03 - Computação Distribuída

Luiza Ávila

1)

i- É idempotente.

ii - É idempotente.

iii- Não é idempotente.

Como a operação idempotente é aquela que pode ser realizada diversas vezes, sem efeito colateral, a falta de uma dependência de estado é necessária para a sua realização.

2) CORBA IDL:

```
interface Election {
        void vote(in string nomeCandidato, in long numeroVotante);
        void result(out string nomeCandidato, out long QtdVotos);
};
```

JAVA RMI:

```
class Resultado {
        String nomeCandidato;
        int QtdVotos;
}
import java.rmi.*;
public interface Election extends Remote{
        void vote(String nomeCandidato, int numeroVotante) throws RemoteException;
        Resultado result () throws RemoteException;
};
```

Os argumentos de entrada tem estruturas parecidas. Se um método retornar mais de um resultado, JAVA RMI é mais inconveniente, pois todos os argumentos de saída devem estar juntos.

3) A semântica do talvez é inapropriada para o serviço Election. O usuário pode votar somente uma vez, e o servidor mantém guardado o número de cada eleitor. A semântica apropriada seria no máximo uma vez, pois reenvia e verifica se existe uma duplicata, garantindo ao eleitor seu voto, e ao sistema que só houve um voto de cada um.

4) No primeiro caso, o desenvolvedor assume que se o cliente observa falhas por omissão, não pode ser dito se é devido a perda de mensagens tipo request ou de reply, o servidor ter caído, ou o servidor estar tomando mais tempo. Portanto, quando o request é retransmitido, o cliente pode receber resposta do primeiro request. No segundo caso, a observação de falhas por omissão, não pode ser porque o servidor demorou muito, então após um tempo T, se o cliente enviar novamente o request, a resposta não será do request anterior. No primeiro caso, o desenvolvedor deve lidar com respostas tardias, no segundo, não.