FORMATIVA 13 - SOCKETS UDP

|  |
| --- |
| EXERCICIO 1: |
| TESTE 1  Executar o programa servidor01.py no IDE (porta 9999)  Executar o cliente01.py a partir de duas consoles do Windows  Executar os testes necessários para responder as seguintes perguntas: |
| QUESTÃO 1:  a) O cliente indica erro se o servidor não tiver sido iniciado primeiro (Sim/Não)?  **R: Não. O cliente pode enviar mensagens, mas não vão para lugar nenhum.**  b) O cliente é notificado se enviar uma mensagem para um servidor que não existe (Sim/Não)?  **R: Não. As mensagens enviadas são descartadas porque o servidor não foi iniciado.**  c) O cliente é notificado se o servidor for encerrado (Sim/Não)?  **R: Não. O cliente consegue ainda enviar mesagens sem ser notificado que o servidor foi encerrado.**  d) O servidor consegue ouvir mensagens de múltiplos clientes (Sim/Não)?  **R: Sim, por ser UDP o servidor não trava os clientes.** |

|  |
| --- |
| EXERCICIO 2: |
| TESTE 2  Executar o programa servidor02.py a partir de duas consoles do Windows informando a mesma porta 9999, mas em endereços IP diferentes (127.0.0.2 e 127.0.0.3)  Executar o cliente01.py no IDE.  Executar os testes necessários para responder as seguintes perguntas: |
| QUESTÃO 2:  a) O cliente pode enviar mensagens para servidores diferentes no mesmo socket (Sim/Não)?  **R: Sim, porque o comportamento do UDP permite.**  b) O cliente consegue enviar mensagens para o endereço de broadcast (Sim/Não)?  **R: Sim.**  Dica: Usar o endereço 127.255.255.255  c) Explique a dificuldade de criar um mecanismo confiável para mensagens de Broadcast.  **R: O cliente não consegue saber de qual servidor ele recebeu o ACK. Como nessa questão são dois servidores, para cada mensagem que ele enviar em Broadcast, ele vai receber duas ACK e não vai conseguir identificar.** |

|  |
| --- |
| EXERCICIO 3: |
| TESTE 3.1  *Abrir uma janela CMD e executar servidor03.py, passando o endereço 127.0.0.2 e a porta 9999*  *Abrir uma janela CMD e executar cliente02.py, enviando as seguintes mensagens (nessa ordem) sem teclar nada no servidor*  *teste1*  *teste2*  *teste3*  *teste 4*  *Teclar algo no servidor para receber as mensagens* |
| TESTE 3.2  *Enviar usando cliente02.py as seguintes mensagens (nessa ordem) sem teclar nada no servidor*  *teste1*  *teste muito grande 2*  *teste3*  *teste muito grande 4*  *Teclar algo no servidor para receber as mensagens* |
| QUESTÃO 3:  a) Caso existam várias mensagens no buffer o servidor consegue receber mais de uma mensagem com um único comando recvfrom (sim/Não)?  **R: Sim, ele armazena as mensagens uma a uma.**  b) Caso a mensagem recebida no servidor seja maior do que o buffer, é possível receber a mensagem por partes (sim/Não)?  **R: Não, como definimos que o buffer aceita 10 bytes, quando passa desse número ocorre um erro.** |

|  |
| --- |
| EXERCICIO 4: |
| TESTE 4  Abrir duas janelas CMD  Executar servidor04.py no endereço 127.0.0.2 porta 9999  Executar cliente03.py  Tente enviar as mensagens  teste1  teste2  teste3  teste4 |
| QUESTÃO 4:  a) É possível enviar uma sequência de mensagens no cliente sem ter recebido o ACK (Sim/Não)?  **R: Não, é necessário desbloquear a mensagem no servidor para que, o cliente poder enviar uma próxima mensagem (implementação conexão confiável) .**  b) A mensagem ACK que o cliente recebe é sempre da última mensagem enviada (Sim/Não)?  **R: Sim.** |

|  |
| --- |
| EXERCICIO 5: |
| TESTE 5  Abrir duas janelas CMD  Executar servidor04.py no endereço 127.0.0.2 porta 9999  Executar cliente04.py e enviar a mensagem teste 1  Teclar algo no servidor o mais rápido possível  Observar se a confirmação (ACK) foi recebida no cliente  Enviar a mensagem teste2 no cliente  Esperar mais de 10 segundos antes de teclar algo no servidor  Observar se a confirmação (ACK) foi recebida no cliente  **- Mostra para o cliente que o servidor não respondeu.** |
| QUESTÃO 5:  a) Para que serve a chamada setblocking(0)?  **R: O servidor fica mais livre, mais dinâmico, porém o cliente não vai saber qual mensagem o ACK está se referindo.**  b) Com a opção setblocking(0) o cliente não sabe com certeza a qual mensagem enviada o ACK que ele recebeu se refere. Isso ocorre porque a mensagem de recepção não bloqueia. Como o cliente poderia saber qual mensagem o ACK está confirmando o recebimento?  **R: Colocar uma thread separada para cada cliente, assim ele vai saber qual mensagem o servidor sabe.** |