	UNIVERSIDADE FEDERAL DA PARAÍBA **CENTRO DE INFORMÁTICA** **DISCIPLINA: Redes de Computadores I**

PROJETO DE REDES

O programa especificado abaixo deverá ser implementado utilizando-se a linguagem C++ ou a linguagem Python, no Windows ou no Linux. O trabalho será em trios e deverá ser enviado pelo SIGAA até as **23:59h do dia 08/05/2024**.

Implemente dois clientes (um utilizando socket UDP e outro utilizando socket RAW) de uma aplicação do tipo cliente/servidor que encaminha requisições para o servidor que está executando através dos protocolos UDP/IP no endereço IP **[server_ipv4]** e porta **50000**. Cada cliente deve solicitar ao usuário a escolha de um dos tipos de requisição abaixo:

1. Data e hora atual;
2. Uma mensagem motivacional para o fim do semestre;
3. A quantidade de respostas emitidas pelo servidor até o momento.
4. Sair.

Uma vez que o usuário tenha feito a sua escolha, o cliente deve encaminhar uma requisição devidamente formatada para o servidor, de acordo com o formato de mensagem especificado abaixo. O servidor por sua vez emitirá uma resposta de volta para o cliente utilizando o mesmo formato de mensagem. Em seguida o cliente deverá exibir a resposta recebida pelo servidor de uma forma adequada para a legibilidade pelo usuário final. Por fim, o programa cliente deverá aguardar novas requisições do cliente até que o usuário selecione a opção “Sair”.

FORMATO DAS MENSAGENS DE REQUISIÇÃO/RESPOSTA

4 bits	4 bits	16 bits	8 bits	8 bits	8 bits	
req/res	**tipo**	**identificador**	**tamanho da resposta**	**byte 1 da resposta**	**byte 2 da resposta...**	…

- **req/res:** indicação para mensagem do tipo **requisição** (bits 0000) ou resposta (bits 0001);
- **tipo:** indicação do tipo de requisição ou resposta. Bits 0000 para solicitação de data, bits 0001 para solicitação de frase motivacional para o fim do semestre e bits 0010 para quantidade de respostas emitidas pelo servidor. O servidor ainda pode emitir uma resposta com o tipo 0011 para indicar que recebeu uma requisição inválida do cliente;
- **identificador:** número não negativo de 2 bytes determinado pelo cliente. O cliente deve sortear um número entre 1 e 65535 toda vez que for enviar uma nova requisição para o servidor. O identificador 0 é reservado para o servidor informar o recebimento de uma requisição inválida;
- **tamanho da resposta:** campo utilizado apenas em respostas geradas pelo servidor. Indica o tamanho da resposta propriamente ditas, em número de bytes (1 a 255). O tamanho 0 é reservado para quando o servidor envia uma resposta indicando o recebimento de uma requisição inválida;

- **bytes da resposta propriamente dita:** uma sequência de bytes contendo a resposta solicitada pelo usuário. Caso o servidor esteja informando o recebimento de uma requisição inválida, nenhum byte é encaminhado neste campo.

Abaixo ilustraremos alguns exemplos de requisições e de respostas com suas respectivas representações em hexadecimal.

Requisição: Tipo 0 (data e hora), identificador 14161

Byte 0	Byte 1	Byte 2
0x00	0x37	0x51

Resposta: Tipo 0 (data e hora), identificador 14161, tamanho 26, resposta propriamente dita “Fri Apr 26 02:28:39 2024\n”

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0x10	0x37	0x51	0x1A	‘F’	‘r’	‘i’	‘ ‘	‘A’	‘p’	‘r’	‘ ‘	‘2’	‘6’	‘ ‘

15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
‘0’	‘2’	‘:’	‘2’	‘8’	‘:’	‘3’	‘9’	‘ ‘	‘2’	‘0’	‘2’	‘4’	‘\n’	‘\0’

Requisição: Tipo 1 (frase motivacional para o fim do semestre), identificador 30225

Byte 0	Byte 1	Byte 2
0x01	0x76	0x11

Resposta: Tipo 1 (frase motivacional), identificador 30225, tamanho 12, resposta propriamente dita “Seja forte!\0”

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x11	0x76	0x11	0x0C	‘S’	‘e’	‘j’	‘a’	‘ ‘	‘f’	‘o’	‘r’	‘t’	‘e’	‘!’	‘\0’

Requisição: Tipo 2 (número de respostas enviadas), identificador 589

Byte 0	Byte 1	Byte 2
0x02	0x02	0x4D

Resposta: Tipo 2 (identificador 589, propriamente dita 42 (número inteiro sem sinal de 4 bytes)

0	1	2	3	4	5	6	7

0x12	0x02	0x4D	0x04	0x00	0x00	0x00	0x2A

Uma vez que o cliente utilizando socket UDP tenha sido finalizado, a equipe deverá fazer uma nova versão do programa cliente, porém utilizando socket RAW (SOCK_RAW) e protocolo IPPROTO_UDP. Nesse tipo de socket, o próprio programador deve estruturar o cabeçalho do protocolo UDP (camada de transporte). Para isso, cada equipe deverá criar uma função para anexar ao início de uma mensagem um cabeçalho UDP (formando um segmento).

Observações importantes:

- Os grupos não devem se preocupar em validar as entradas de dados. Assumam que o usuário sempre digitará valores válidos como entradas de dados;
- Cada grupo deverá tomar medidas para garantir que o código-fonte possua boa legibilidade (comentários são cruciais nesse sentido) e que o programa seja minimamente eficiente, ou seja, o programa não deverá realizar ações claramente desnecessárias para a solução do problema;
- O código implementado deve ser original, não sendo permitidas cópias de códigos inteiros ou trechos de códigos de outras fontes (incluindo inteligências artificiais generativas, como ChatGPT).

O cliente utilizando socket UDP irá contribuir com 6 pontos na nota, enquanto que o cliente com socket RAW e protocolo IPPROTO_UDP valerá mais 4 pontos, totalizando uma nota máxima de 10 pontos.

--- Boa sorte! ---

CONSIDERAÇÕES SOBRE CABEÇALHOS UDP E IP

Abaixo você encontrará algumas informações complementares para ajudar você a estruturar corretamente o cabeçalho UDP.

- **Cabeçalho UDP**

A estrutura de um segmento UDP simples (cabeçalho + *payload*), já considerando o protocolo de aplicação descrito nesta especificação, encontra-se exemplificada abaixo.

Cabeçalho UDP	
Porta de origem	**0xE713** (porta 59155)
Porta de destino	**0xC350** (porta 50000)
Comprimento do segmento	**0x000B** (comprimento 11, ou seja, 8 bytes do cabeçalho + 3 bytes do payload)
Checksum	**0xE0B3** (valor final do checksum. No momento em que o checksum está sendo calculado, o valor provisório desse campo deve ser **0x0000**)

Payload	
Payload	**0x025CE1** (requisição da quantidade de respostas enviadas pelo servidor, com identificador da requisição 23777)

Para o cálculo do checksum UDP, a RFC 768 afirma que precisamos calcular o checksum de 16 bits considerando o complemento-de-um da soma porções de 2 bytes de um **pseudo cabeçalho IP**, do **cabeçalho UDP** e do **payload**. Caso o último número a ser somado tenha apenas 1 byte, deve-se adicionar mais um byte 0 à direita (operação chamada de *padding*). O objetivo da inclusão do pseudo cabeçalho IP, de acordo com a RFC, é fornecer ao protocolo proteção contra datagramas roteados de forma equivocada. O pseudo cabeçalho IP possui a estrutura abaixo.

Pseudo Cabeçalho IP	
IP de origem	**0xC0A8 0169** (IP 192.168.1.105)
IP de destino	**0x0FE4 BF6D** (IP [server_ipv4])
Byte 0 + Número de protocolo de transporte	**0x0011** (byte com valor decimal 0 seguido de byte com valor decimal 17, que é o número do protocolo UDP de acordo com a *Internet Assigned Numbers Authority* - IANA)
Comprimento do segmento UDP	**0x000B**

Dessa forma, o somatório necessário para o cálculo do checksum é o abaixo:

0xC0A8 + 0x0169 + 0x0FE4 + 0xBF6D + 0x0011 + 0x000B + 0xE713 + 0xC350 + 0x000B + 0x0000 + 0x025C + 0xE100

- Em vermelho: conjuntos de 2 bytes do pseudo cabeçalho IP
- Em verde: conjuntos de 2 bytes do cabeçalho UDP
- Em azul: conjuntos de 2 bytes do payload UDP (o byte 00 foi adicionado ao último byte do payload para que todos os conjuntos da soma possuam 2 bytes)

O resultado do somatório acima é 41F48. Para fazer o *wraparound* de uma vez só de modo a tornar o resultado da soma um número de 2 bytes, você pode considerar esse resultado como o número de 32 bits 0x00041F48 e fazer a soma dos 16 bits mais significativos com os 16 bits menos significativos, ou seja, 0x0004 + 0x1F48, o que produzirá o resultado **1F4C**. O checksum será o complemento de 1 de 1F4C, que é **E0B3**, conforme podemos observar na representação binária abaixo:

1 F 4 C
0001 1111 0100 1100
(complemento de 1 abaixo)
1110 0000 1011 0011
E 0 B 3

RECEBENDO RESPOSTAS COM SOCK_RAW E IPPROTO_UDP

É importante ressaltar que ao utilizar SOCK_RAW com o protocolo IPPROTO_UDP, a resposta recebida consistirá de um datagrama completo, incluindo cabeçalho IP, cabeçalho UDP e payload! As equipes deverão pular os 20 bytes do cabeçalho IP e os 8 bytes do cabeçalho UDP para chegarem ao conteúdo do payload enviado pelo servidor.