MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PIAUÍ - UFPI
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS - PICOS
CURSO BACHARELADO EM SISTEMAS DE INFORMAÇÃO
DISCIPLINA: REDES DE COMPUTADORES II
PROFESSOR(A): RAYNER GOMES SOUSA
ALUNO: VANDIRLEYA BARBOSA DA COSTA

1º Avaliação

1. VISÃO GERAL

Este projeto compreende dois elementos centrais: um SuperServidor e um Cliente, desenvolvidos nos arquivos "SuperServidor.py" e "Cliente.py", respectivamente. O SuperServidor tem a capacidade de administrar vários servidores TCP e UDP, cada um identificado por um ID exclusivo. O Cliente foi concebido para interagir com o SuperServidor e requisitar serviços específicos.

- SuperServidor

Estrutura "SuperServidor": Aceita um "host" e uma "porta" como parâmetros. Esses são armazenados como atributos da instância. Além disso, dois dicionários vazios "servidores_tcp" e "servidores_udp" são inicializados para armazenar os servidores TCP e UDP criados, respectivamente. Um dicionário "threads_servidor" também é inicializado para armazenar as threads dos servidores.
Método iniciar: Este método é usado para iniciar o SuperServidor. Ele cria um novo socket, vincula o socket ao host e à porta especificados, e começa a ouvir conexões. Em seguida, ele entra em um loop onde aceita novas conexões de clientes e inicia uma nova thread para lidar com cada cliente.
Método lidar_com_cliente: Este método é usado para lidar com a comunicação com um cliente específico. Ele lê a solicitação do cliente e executa a ação apropriada com base na solicitação. As ações possíveis incluem desligar o SuperServidor, criar um novo servidor TCP ou UDP, listar todos os servidores, e encerrar um servidor específico.
Métodos criar_servidor_tcp e criar_servidor_udp: Estes métodos são usados para criar um novo servidor TCP ou UDP, respectivamente. Eles geram um ID único, encontram uma porta disponível, criam uma nova instância do servidor e iniciam o servidor em uma nova thread.
Métodos iniciar_servidor_tcp e iniciar_servidor_udp: Estes métodos são usados para iniciar um servidor TCP ou UDP, respectivamente. Eles criam um novo socket, vinculam o socket a uma porta específica, e começam a ouvir conexões. Em seguida, eles entram em um loop onde aceitam novas conexões de clientes e iniciam uma nova thread para lidar com cada cliente.
Métodos lidar_com_cliente_tcp e lidar_com_cliente_udp: Estes métodos são usados para

lidar com a comunicação com um cliente específico de um servidor TCP ou UDP, respectivamente. Eles leem a solicitação do cliente, tentam avaliar a solicitação como uma expressão matemática (no caso do servidor TCP) ou como um comando de envio/recebimento de arquivo (no caso do servidor UDP), e enviam a resposta de volta ao cliente.
Método parar: Este método é usado para parar o SuperServidor. Ele faz isso fechando o socket e juntando todas as threads dos servidores.
Método encontrar_porta_disponivel: Este método é usado para encontrar uma porta disponível no sistema operacional.
Método listar_servidores: Este método é usado para listar todos os servidores TCP e UDP atualmente gerenciados pelo SuperServidor.
Método encerrar_servidor: Este método é usado para encerrar um servidor específico.

- Cliente

Estrutura "Cliente": aceita um "host", uma "porta" e um "protocolo" como parâmetros. Esses são armazenados como atributos da instância. Além disso, um atributo "soquete" é inicializado como "None".
Método conectar: Este método é usado para conectar o cliente ao servidor. Ele cria um novo socket, que é um ponto de extremidade de uma comunicação bidirecional entre o servidor e o cliente. O tipo de socket criado depende do protocolo especificado (`tcp` ou `udp`). Em seguida, o socket é conectado ao host e à porta especificados.
Método enviar_requisicao: Este método é usado para enviar uma requisição ao servidor e receber uma resposta. A requisição é enviada como uma string codificada em UTF-8. A resposta é recebida como bytes, que são decodificados de volta para uma string se o parâmetro "decodificar" for "True".
Método enviar_dados: Este método é usado para enviar dados brutos (bytes) ao servidor. Isso é útil para enviar arquivos, por exemplo.
Método fechar_conexao: Este método é usado para fechar a conexão com o servidor.
Método receber_arquivo: Este método é usado para receber um arquivo do servidor. Ele lê os dados do socket em blocos de 1024 bytes de cada vez e os grava em um arquivo até que receba um indicador de fim de arquivo (`b"FIM DO ARQUIVO"`).

2. CABEÇALHO TCP

Para monitorar o cabeçalho pelo TCP, primeiro ligo o superserver que está no protocolo TCP, e estabelece conexão com Cliente.

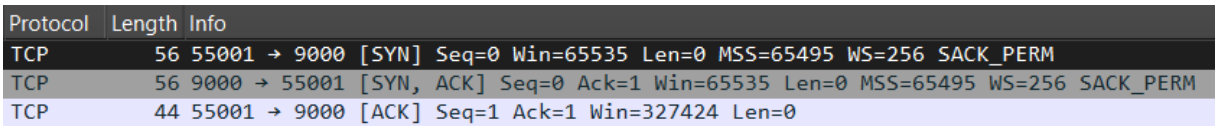
```
PS C:\Users\vandi\OneDrive\Documentos\RedesII> & C:/Users/vandi/AppData/Local/Programs/Python/Python312/python.exe c:/Users
/vandi/OneDrive/Documentos/RedesII/PROVA/SUPERSERVIDOR/SuperServidor.py
Iniciando Super Servidor TCP em localhost:9000...
```

Servidor escutando, logo depois de estabelecer conexão com cliente, no wireshark aparecer o processo do estabelecimento de conexão com handshake de 3 vias.
Segue as informações do **estabelecimento** de conexão no wireshark.
TCP 56 55001 → 9000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM:
Nesta etapa inicial, conhecida como o handshake de três vias, o cliente (porta 55001) inicia

a comunicação enviando um pacote SYN para o servidor (porta 9000), solicitando a abertura de uma conexão TCP. Os parâmetros incluem um tamanho de janela (Win) de 65535, um tamanho máximo de segmento (MSS) de 65495, uma escala da janela (WS) de 256 e a permissão SACK ativada.

TCP 56 9000 → 55001 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM: Em resposta ao pacote SYN enviado pelo cliente, o servidor (porta 9000) responde com um pacote SYN-ACK, indicando que reconheceu a solicitação do cliente e também está solicitando uma conexão. O servidor mantém os mesmos parâmetros que o cliente, incluindo um tamanho de janela (Win) de 65535, um MSS de 65495, uma escala de janela (WS) de 256 e a permissão SACK ativada.

TCP 44 55001 → 9000 [ACK] Seq=1 Ack=1 Win=327424 Len=0: Por fim, o cliente (porta 55001) envia um pacote ACK de volta para o servidor (porta 9000) para confirmar o recebimento do pacote SYN-ACK, finalizando assim o handshake de três vias. Com isso, a conexão TCP é estabelecida com sucesso. Notavelmente, o tamanho da janela do cliente é ajustado para 327424.



Segue as informações do **encerramento de conexão** no wireshark.

TCP 44 9000 → 55001 [FIN, ACK] Seq=6 Ack=19 Win=2161152 Len=0: Este é o primeiro passo no encerramento de uma conexão TCP. O servidor (porta 9000) envia um pacote FIN para o cliente (porta 55001) indicando que terminou de enviar dados. O servidor também envia um ACK para o cliente, reconhecendo os dados recebidos do cliente. O tamanho da janela do servidor é 2161152.

TCP 44 55001 → 9000 [ACK] Seq=19 Ack=7 Win=327424 Len=0: Em resposta ao pacote FIN do servidor, o cliente (porta 55001) envia um pacote ACK de volta para o servidor (porta 9000) para reconhecer o pacote FIN do servidor. O tamanho da janela do cliente é 327424.

TCP 44 55001 → 9000 [FIN, ACK] Seq=19 Ack=7 Win=327424 Len=0: Agora, o cliente (porta 55001) envia um pacote FIN para o servidor (porta 9000) indicando que também terminou

de enviar dados. O cliente também envia um ACK para o servidor, reconhecendo os dados recebidos do servidor. O tamanho da janela do cliente é 327424.

TCP 44 9000 → 55001 [ACK] Seq=7 Ack=20 Win=2161152 Len=0: Finalmente, o servidor (porta 9000) envia um pacote ACK de volta para o cliente (porta 55001) para reconhecer o pacote FIN do cliente. Isso completa o encerramento da conexão TCP. O tamanho da janela do servidor é 2161152.

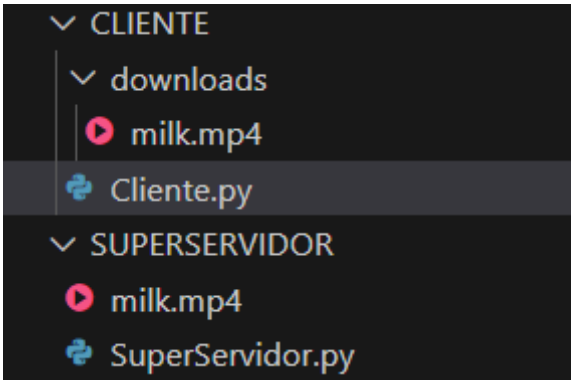
```
TCP        44 9000 → 55001 [FIN, ACK] Seq=6 Ack=19 Win=2161152 Len=0
TCP        44 55001 → 9000 [ACK] Seq=19 Ack=7 Win=327424 Len=0
TCP        44 55001 → 9000 [FIN, ACK] Seq=19 Ack=7 Win=327424 Len=0
TCP        44 9000 → 55001 [ACK] Seq=7 Ack=20 Win=2161152 Len=0
```

3. ANÁLISE DE TRÁFEGO

Para realizar a análise de tráfego com wireshark, inicializamos o servidor de acordo com solicitação do cliente, e o cliente seleciona a opção de “download”, para requisitar a transmissão de um arquivo que está na pasta do servidor para a pasta de downloads com o cliente.

Servidor UDP inicializado.

Client requisita a opção de servidor file, depois conecta com o servidor, e solicita o download do arquivo milk.mp4. Servidor envia o arquivo. Cliente recebe o arquivo, e assim ficam as pastas organizadas:



E assim, o monitoramento com wireshark ficou o seguinte:

Campo	Valor	Unidade

Porta A	53542	N/A
Porta B	55242	N/A
Pacotes	1077	pacotes
Bytes	1134649	bytes
ID do Fluxo	0	N/A
Total de Pacotes	1077	pacotes
Percentual Filtrado	100	%
Pacotes A → B	1	pacotes
Bytes A → B	56	bytes
Pacotes B → A	1076	pacotes
Bytes B → A	1134593	bytes
Início Relativo	0	segundos
Duração	0.012159	segundos
Bits/s A → B	36845	bits/s
Bits/s B → A	746504153	bits/s

- **Porta A** e **Porta B**: São as portas de comunicação usadas pelos pontos de extremidade A e B, respectivamente.
- **Pacotes**: É o número total de pacotes transmitidos na sessão.
- **Bytes**: É o número total de bytes transmitidos na sessão.
- **ID do Fluxo**: É um identificador único para a sessão de fluxo.
- **Total de Pacotes**: É o número total de pacotes transmitidos na sessão.
- **Percentual Filtrado**: É a porcentagem de pacotes filtrados na sessão.
- **Pacotes A → B** e **Bytes A → B**: São o número de pacotes e bytes, respectivamente, enviados do ponto A para o ponto B.
- **Pacotes B → A** e **Bytes B → A**: São o número de pacotes e bytes, respectivamente, enviados do ponto B para o ponto A.

- **Início Relativo**: É o tempo de início da sessão em relação ao início da captura de pacotes.
- **Duração**: É a duração total da sessão.
- **Bits/s A → B** e **Bits/s B → A**: São as taxas de bits por segundo do ponto A para o ponto B e do ponto B para o ponto A, respectivamente.