Guilherme Ribeiro Rodrigues (0024299)
Flávio Eduardo Keglevich de Buzin (00229724)
Lucas Matheus Dias Brum (00261614)
Bruno Longo Farina (00263412)

TRABALHO PRÁTICO PARTE 1: THREADS, SINCRONIZAÇÃO E COMUNICAÇÃO

- (A) Como foi implementado cada subserviço;
    - O serviço de **monitoração** é feito através da thread `requestStatus,` que é criada após a conexão de um Cliente ao sistema. Esta o envia mensagens de recorrentes a fim de verificar se estão acordados. Na ausência de resposta a três pacotes seguidos com timeout de 2seg entre eles, muda o status do Cliente.
    - O serviço de **descoberta,** através do método `recvfrom` em `server_udp.c` usa dados advindos de uma máquina (IP, Mac Add, Hostname) para popular uma estrutura HOST [vide (C)] e manipular a sua presença e status na tabela criada. As conexões são estabelecidas a partir de sockets criados entre Servidor e Cliente.
    - O serviço de **interface**, as impressões da da tabela em tela para o usuário sinalizando alterações ocorrerão quando do adição ou remoção de um e são executadas pelo método `printTable.`

- (B) Em quais áreas do código foi necessário garantir sincronização no acesso a dados;
    - Foi necessário usar uma mutex para sincronizar o acesso dentro da maioria das funções que acessam diretamente a tabela de participantes (hosts) dentro do arquivo table.c. Assim a tabela só é acessada pelas funções, e dentro das funções, o acesso à estrutura interna da tabela só é feito usando a mutex.
    - Também foi necessário o uso de um semáforo para estabelecer um acesso restrito à inicialização da thread requestStatus de modo que a criação de uma nova thread desta função só é liberada no momento em que os argumentos passados por parâmetro são armazenados em uma variável local, impedindo que a estrutura passada por parâmetro seja sobrescrita por uma nova thread antes de ser armazenada localmente.

- (C) Descrição das principais estruturas e funções que a equipe implementou;
    - Funções principais do arquivo table.c:
        - Estrutura HOST: Representa uma entrada na tabela de participantes do processo *manager*. Armazena seu hostname, endereço MAC, endereço IP, status (acordada ou dormindo) e a sua posição na tabela.
        - função create_host: Cria um HOST baseado nos seus parâmetros.
        - função **init_table**: Inicializa a tabela de participantes.
        - função **insertHost**: Insere um HOST na tabela de participantes.

    - função **removeHost**: Remove um HOST na tabela de participantes dado a posição da sua entrada.
    - função printHost: Imprime um HOST na saída padrão.
    - função **printTable**: Imprime a tabela toda na saída padrão.
    - função **wakeUpHost**: Muda o estado de uma entrada para acordada.
    - função **sleepHost**: Muda o estado de uma entrada para dormindo.
    - função **findHostByName**: Obtém um HOST baseado no hostname dele.

      Nota: as funções em negrito sincronizam o acesso à tabela interna usando uma mutex.
- Principais funções no sleep_server.c:
    - Somente a função main que é ponto de entrada da aplicação e define se irá ser executada a função de cliente ou servidor com base nos parâmetros inseridos.
- Principais funções client_udp.c:
    - Estrutura pcInfo: Armazena o hostname, endereço IP, endereço MAC da máquina atual e a posição da estrutura HOST na tabela de participantes.
    - checkHostName: Verifica se o hostname é valido.
    - getipNumber: Obtém o número IP da máquina local.
    - getMac: Obtém o número MAC da máquina local.
    - getHost: Obtém o hostname da máquina local.
    - getIPandName: Copia os dados da máquina local para uma estrutura pcInfo.
    - sendStatus: Função da thread que recebe a requisição de status do servidor e manda que está acordada. Essa thread termina quando o usuário digita EXIT.
    - clientUDP: Espera pela mensagem em broadcast do servidor informando seu IP, envia uma mensagem contendo as suas informações relevantes para o servidor no IP indicado e espera pela mensagem informando a sua posição na tabela, caso ela seja -1, o cliente não conseguiu se conectar devido ao fato de o servidor estar lotado, caso contrário, a função inicializa a thread sendStatus e fica em loop até que o usuário digite “EXIT”.
- Principais funções no server_udp.c:
    - caller: Função de thread chamada no inicio da execução do serverUDP que cria um socket e envia repetidamente em broadcast o IP da máquina executando o código (IP do servidor).
    - requestStatus: Função de thread chamada toda vez que um cliente novo consegue estabelecer a conexão com o servidor, a função fica repetidamente enviando mensagens de status e recebendo mensagens enquanto o cliente estiver ativo, caso, mesmo enviando repetidas mensagens, ele pare de receber respostas por um certo tempo, ele vai assumir que o cliente está dormindo e irá alterar o status do cliente para asleep, caso ele volte a receber mensagens de resposta, ele retornará o status do cliente para awaken. Caso a mensagem recebida for de término de conexão, ele remove o cliente da tabela e encerra os sockets.

        - wakeUpThread: Código principal da thread que lê os comandos do usuário. Ela é responsável por ler os comandos WAKEUP *hostname* que o usuário digitar na aplicação, e alterar a flag que envia o pacote mágico Wake On Lan na thread requestStatus.
        - serverUDP: Função principal, inicializa a thread caller, realiza um loop esperando receber informações de clientes tentando se conectar, se houver espaço na tabela, retorna para o cliente a sua posição na tabela e inicializa a thread requestStatus passando como parâmetro as informações deste cliente para então, retornar à sua espera por novas conexões.

- (D) Explicar o uso das diferentes primitivas de comunicação
    - A principal primitiva de comunicação utilizada foram sockets criados em diferentes portas da aplicação.
        - O socket onde foi realizado o broadcast informando o IP do servidor para os clientes utilizou a porta 5000.
        - O socket em que o servidor espera repetidamente por solicitações de conexão das máquinas de clientes utilizou a porta 40000.
        - Os sockets utilizados para fazer a checagem de status de cada cliente utilizaram as portas a partir do índice 10000, a porta de comunicação era incrementada conforme a posição do cliente na tabela armazenada no servidor.

- Observações finais:

Infelizmente a gente não conseguiu testar diretamente a funcionalidade de envio de pacotes Wake On Lan para acordar a máquina cliente no trabalho. Como as máquinas dos laboratórios estudantis do INF não possuem *hostname*, não conseguimos testar diretamente, e também não conseguimos testar nas nossas máquinas/redes pessoais.

Nota: seria importante implementar uma exclusão mútua para a flag wakeUpFlag, porém isso não foi implementado.