UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
DEPARTAMENTO DE INFORMÁTICA APLICADA

INF01151 – SISTEMAS OPERACIONAIS II N – Turma A
PROF. WEVERTON CORDEIRO
SEMESTRE 2024/1
TRABALHO PRÁTICO PARTE 1: THREADS, SINCRONIZAÇÃO E COMUNICAÇÃO

ESPECIFICAÇÃO DO TRABALHO

A proposta de trabalho prático é implementar um serviço de "gerenciamento de sono" (*sleep management*) de estações de trabalho que pertencem a um mesmo segmento de rede física de uma grande organização. O objetivo do serviço é promover a economia de energia ao incentivar que os colaboradores coloquem suas estações de trabalho para dormir após o termino do expediente na organização. Neste caso, o serviço deve garantir que as estações poderão ser acordadas caso o colaborador(a) desejar acessar, remotamente, um serviço na sua estação de trabalho (por exemplo, um serviço de compartilhamento de arquivos, para acessar arquivos pessoais).

Sabe-se que uma das formas de acordar estações de trabalho de forma remota é via Wake-On-LAN (WoL)[1]. WoL é um padrão de rede que permite ligar ou acordar estações de trabalho a partir do envio de um "*pacote mágico*"[2] (*magic packet*) para a placa de rede da estação que se deseja acordar. Para isso, a placa de rede deve suportar WoL e a placa mãe da estação do trabalho deve estar configurada para aceitar WoL.

No Linux, um programa que pode ser usado para acordar estações de trabalho usando o padrão Wake-On-LAN é o `wakeonlan`. Para usá-lo, faz-se necessário indicar o endereço MAC da estação de trabalho que se deseja acordar:

```
$ wakeonlan 00:01:02:03:04:05
```

Uma limitação do WoL é que o pacote magico não é roteável pela Internet[3]. Para acordar uma estação de trabalho em um determinado segmento físico de rede faz-se necessário, portanto, um gateway WoL no mesmo segmento e que possa receber requisições diretamente da Internet (usando, por ex., TCP ou UDP para esse fim). A solução baseada em gateway WoL depende, no entanto, de uma estação dedicada por segmento físico de rede, o que pode não ser viável

[1] Há outros mecanismos similares, como Wake-on-Wireless LAN (WoWLAN), no caso de computadores conectados via rede Wi-Fi.

[2] O *pacote mágico* é um *frame*, frequentemente enviado via *broadcast*, que contém em qualquer lugar no seu *payload* 6 bytes 255 (0xFF 0xFF 0xFF 0xFF 0xFF 0xFF em hexadecimal), seguido por dezesseis repetições do endereço MAC de 48 bits da placa de rede da estação de trabalho (total de 102 bytes). O pacote mágico é verificado apenas para a *string* descrita anteriormente, e não é analisado por uma pilha de protocolos completa. Portanto, ele pode ser enviado como *payload* de qualquer protocolo de rede e de camada de transporte, sendo normalmente enviado como um datagrama UDP para a porta 0 (número da porta reservada), 7 (Echo Protocol) ou 9 (Discard Protocol), ou diretamente pela Ethernet como EtherType 0x0842 (Fonte: Wikipedia).

[3] Essa limitação pode ser superada com o uso de IP Direct Broadcast, embora esse recurso seja indiferente no contexto da proposta de trabalho prático.

tecnicamente. Considerando essa limitação, o objetivo da proposta de trabalho é desenvolver um serviço de gerência de sono (*sleep management service*) que opere de forma descentralizada em um segmento físico de rede. A proposta de trabalho é inspirada no GreenUp[4], solução para o monitoramento de consumo energético e gerenciamento de sono de estações de trabalho desenvolvido pela Microsoft Research [1].

A proposta deverá ser desenvolvida em duas etapas. A primeira etapa compreenderá funcionalidades que dependerão de tópicos como *threads*, processos, comunicação e sincronização para serem implementadas. A aplicação deverá executar em ambientes Unix (Linux), mesmo que tenha sido desenvolvida em outras plataformas. O projeto deverá ser implementado em C/C++, usando a API User Datagram Protocol (UDP) para descoberta e Transmission Control Protocol (TCP) para comunicação.

DESCRIÇÃO DO SERVIÇO E FUNCIONALIDADES BÁSICAS

O projeto compreenderá UM ÚNICO PROGRAMA, que executará em múltiplas estações, e que deverá oferecer as funcionalidades previstas nos subserviços abaixo. Sugere-se à equipe organizar cada subserviço do sistema em módulos para implementá-las.

- **Subserviço de Descoberta (*Discovery Subservice*)**, para identificar quais estações de trabalho no mesmo segmento físico de rede passaram a executar o programa (denominadas *participantes*) e quais passaram a não fazer mais parte do serviço. Sugere-se usar *sockets* UDP configurados com a opção *broadcast* para implementar as rotinas de descoberta. Na primeira parte do trabalho prático, este subserviço (i) operará de forma passiva (isto é, recebendo e respondendo pacotes em *broadcast* do tipo *sleep service discovery*) somente em uma estação de trabalho, denominada líder (*manager*) e (ii) operará de forma ativa (isto é, enviando pacotes do tipo *sleep service discovery*) nas demais estações participantes;
- **Subserviço de Monitoração (*Monitoring Subservice*)**, para identificar quais participantes passaram para o modo sono (*asleep*) e quais estão acordadas (*awaken*). Na primeira parte do trabalho prático, este subserviço (i) operará de forma ativa (isto é, enviando pacotes do tipo *sleep status request*) somente na estação *manager* e (ii) operará de forma passiva (isto é, recebendo e respondendo pacotes do tipo *sleep status request*) nas demais estações participantes;
- **Subserviço de Gerenciamento (*Management Subservice*)**, para manter uma lista das estações participantes e o status de cada participante (*asleep* ou *awaken*). Na primeira parte do trabalho prático, este subserviço manterá a lista somente na estação *manager*;
- **Subserviço de Interface (*Interface Subservice*)**, para exibir a relação atualizada de participantes na tela e permitir a interação com usuários (para, por ex., enviar um comando para acordar uma estação). Este subserviço deverá ficar ativo em todas as estações participantes, incluindo a *manager*.

A definição da estação de trabalho *manager* será feita via argumento de linha de comando passado como parâmetro para o programa durante a inicialização. Isso significa que a mesma implementação do projeto deverá permitir que uma estação atue como *manager* ou como participante. Na segunda parte do trabalho pratico, a estação *manager* será definida usando um algoritmo de eleição de líder.

[4] https://www.microsoft.com/en-us/research/project/greenup/

Cada estação de trabalho participante deverá ser identificada pelo nome da máquina (*hostname*), por exemplo disponível no arquivo `/etc/hostname`. Caso um *hostname* não esteja disponível, a participante deverá ser identificada pelo endereço IP da interface de rede.

O serviço deve garantir, ainda:

- **Consistência nas estruturas de armazenamento:** As estruturas de armazenamento de dados no serviço devem ser mantidas em um estado consistente. Por exemplo, as informações sobre participantes e o status de cada participante devem ser mantidas de forma consistente.
- **Persistência de dados no servidor**: As estruturas de armazenamento de dados no serviço devem ser mantidas somente em memória. Na segunda parte do trabalho, mecanismos de replicação serão usados para garantir a persistência consistente das estruturas em múltiplas estações.

IMPLEMENTAÇÃO DO SISTEMA

A proposta de trabalho prático está dividida em duas etapas, sendo que a segunda etapa irá contemplar funcionalidades adicionais. Portanto, considere uma implementação modular e com possibilidade de extensão, e o encapsulamento das funções de comunicação do cliente e do servidor em módulos isolados.

Para o funcionamento do serviço, o servidor deverá manter, em uma estrutura de dados, a lista de participantes. Cada participante é caracterizado pelo *hostname*, o endereço IP, o endereço MAC e o *status* do participante (*asleep* ou *awaken*). Ao descobrir um novo participante (via subserviço de descoberta), uma nova entrada deverá ser incluída na lista. Da mesma forma, quando um participante deixar de fazer parte do serviço, a sua respectiva entrada deverá ser removida. A Tabela 1 ilustra as informações mantidas nessa estrutura.

Tabela 1. Exemplo de estrutura de dados mantida pelo servidor.

Hostname	Endereço MAC	Endereço IP	Status
host1	01:01:01:01:01:01	1.1.1.1	awaken
host2	02:02:02:02:02:02	1.1.1.2	awaken
host3	03:03:03:03:03:03	1.1.1.3	ASLEEP

O processo *manager* deverá implementar um esquema baseado no modelo leitor/escritor para gerenciar a leitura e escrita de dados na tabela. Isso significa que o acesso concorrente à tabela deverá ser controlado por primitivas de exclusão mútua. O subserviço de descoberta (primeiro escritor) ficará responsável por incluir/remover entradas da tabela (sempre que uma estação ingressar/sair do serviço, respectivamente). O subserviço de monitoração (segundo escritor), por sua vez, deverá atualizar o *status* de cada participante. Já o subserviço de interface (leitor) deverá aguardar por atualizações na tabela, e escrevê-las na tela para o(a) usuário(a). Observe que a escrita na tabela será uma operação não bloqueante, enquanto a leitura será uma operação bloqueante (o leitor ficará bloqueado até que uma nova atualização esteja disponível para leitura). Observe também que, enquanto a tabela estiver sendo lida, nenhum escritor poderá modificá-la.

INTERFACE DO USUÁRIO

O serviço deverá ser iniciado via linha de comando de forma simples:

```
$ ./sleep_server
```

Sem parâmetros, ele deverá ser iniciado na estação no modo participante. Com o parâmetro *manager*, o serviço deverá ser iniciado na estação como *manager*.

Após iniciar o serviço, a estação *manager* deverá manter constantemente atualizada na tela a lista com todas as estações participantes no serviço. Ou seja, sempre que mudar o status de qualquer estação participante, essa informação deverá ser atualizada na tela. As estacoes participantes deverão exibir na tela os dados (nome, endereço MAC e endereço IP) da estação *manager*. Note que é possível que a estação *manager* seja iniciada depois das participantes.

Nas estações participantes, o único comando que poderá ser informado pelo(a) usuário(a) é:

```
EXIT
```

Nesse caso, a estação em questão deverá enviar para a *manager* uma mensagem do tipo *sleep service exit*, notificando que ela está deixando o serviço e, portanto, deve ser removida da tabela. Após isso, o processo deverá ser encerrado nessa estação.

Na estação *manager*, o único comando que poderá ser informado pelo(a) usuário é:

```
WAKEUP hostname
```

Onde *hostname* é o nome da estação conforme exibido na tabela. Esse comando deverá acordar uma máquina (via pacote WoL). Note que uma máquina também poderá acordar espontaneamente, por ação do(a) usuário(a).

A interface do cliente deve ter uma *thread* para escrever as mensagens na tela, e outra thread para ler os comandos digitados pelo(a) usuário(a). Ao apertar CTRL+C (interrupção) ou CTRL+D (fim de arquivo), o processo cliente deverá encerrar, sinalizando ao *manager* que o(a) usuário(a) está saindo do serviço (similar a EXIT).

FORMATO DE ESTRUTURAS

A equipe tem liberdade para definir o tamanho e formato das mensagens que serão usadas para troca de dados entre os processos executando em cada estação. Sugere-se a especificação de uma estrutura para definir as mensagens trocadas. Abaixo é apresentada uma sugestão de como implementar a estrutura para a troca de comandos e mensagens entre os processos.

```
typedef struct __packet{
  uint16_t type;       //Tipo do pacote (p.ex. DATA | CMD)
  uint16_t seqn;       //Número de sequência
  uint16_t length;     //Comprimento do payload
  uint16_t timestamp;  // Timestamp do dado
  const char* _payload;  //Dados da mensagem
} packet;
```

DESCRIÇÃO DO RELATÓRIO

A equipe deverá produzir um relatório fornecendo os seguintes dados:

- Explicação e respectivas justificativas a respeito de:
  - (A) Como foi implementado cada subserviço;
  - (B) Em quais áreas do código foi necessário garantir sincronização no acesso a dados;
  - (C) Descrição das principais estruturas e funções que a equipe implementou;
  - (D) Explicar o uso das diferentes primitivas de comunicação;
- Descrição dos problemas que a equipe encontrou durante a implementação e como estes foram resolvidos (ou não).

A **nota será atribuída baseando-se nos seguintes critérios**: (1) qualidade do relatório produzido conforme os itens acima, (2) correta implementação das funcionalidades requisitadas, (3) qualidade do programa em si (incluindo uma interface limpa e amigável, documentação do código, funcionalidades adicionais implementadas, etc.) e (4) qualidade da apresentação (o que inclui domínio sobre o trabalho realizado por cada integrante da equipe).

MÉTODOS DE AVALIAÇÃO

O trabalho deve ser feito em grupos de **QUATRO INTEGRANTES**. As pessoas participantes da equipe devem estar claramente identificadas no relatório e na apresentação. A avaliação do trabalho será pela análise da implementação, do relatório produzido e da apresentação. A ausência de um(a) integrante da equipe no dia da apresentação implicará em conceito zero para a pessoa ausente (salvo por motivos excepcionais como por ex. de saúde, que deverão ser registrados via junta médica da UFRGS).

A apresentação dos trabalhos e demonstração prática em laboratório dos sistemas implementados será realizada presencialmente, conforme o cronograma da disciplina.

Faz parte do pacote de entrega os códigos-fonte da implementação, um tutorial de como compilar e executar os códigos e o relatório em um arquivo ZIP. A implementação deve estar funcional para demonstração durante a apresentação pela equipe, em laboratório. A compilação deverá ser feita via scripts automatizados (por ex., Makefile), de modo a facilitar o processo de avaliação do projeto submetido.

DÚVIDAS, QUESTIONAMENTOS E SUGESTÕES

Dúvidas, questionamentos e sugestões podem ser enviados por e-mail ou Moodle.

REFERÊNCIAS

[1] Sen, Siddhartha, Jacob R. Lorch, Richard Hughes, Carlos Garcia Jurado Suarez, Brian Zill, Weverton Cordeiro, and Jitendra Padhye. "Don't Lose Sleep Over Availability: The GreenUp Decentralized Wakeup Service." In 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), pp. 211-224. 2012.