Universidade do Vale do Itajaí
Disciplina: Redes de Computadores I
Professor: Felipe Viel
Avaliação – Camada de Transporte

Instruções

1. Esta avaliação deve ser feita individualmente ou em até 5 pessoas.

2. Data de entrega: 20/10/2021 até 18:55. Serão aceitos trabalhos entregues em atraso, com desconto de 2,0 ponto por dia de atraso, incluindo final de semana, independente da nota da avaliação.

3. Esta avaliação tem por objetivo consolidar o aprendizado sobre algoritmos usados para a construção da camada de transporte.

4. O uso de bibliotecas de terceiros deverá ser explicado com uma justificativa válida que ficará ao critério do professor aceitar ou não.

5. Poderá ser feito nas linguagens C/C++, Rust, Python e Java. Abstrações por meio de bibliotecas poderão ser feitas mediante autorização do professor.

6. O sistema deve ser entregue funcionando corretamente.

7. Cópias de outros alunos ou da internet, mesmo que parcial, quando detectada implicará em nota zero para todos os envolvidos.

8. Deve ser apresentado um relatório eletrônico em formato PDF (em outro formato é descontado 1,5 ponto) que contenha:
    a. Identificação do autor e do trabalho.
    b. Enunciado do projeto.
    c. Explicação e contexto da aplicação para compreensão do problema tratado pela solução.
    d. Resultados obtidos com as simulações.
    e. Códigos importantes da implementação.
    f. Resultados obtidos com a implementação (tabelas, gráficos e etc).
    g. Análise e discussão sobre os resultados finais.

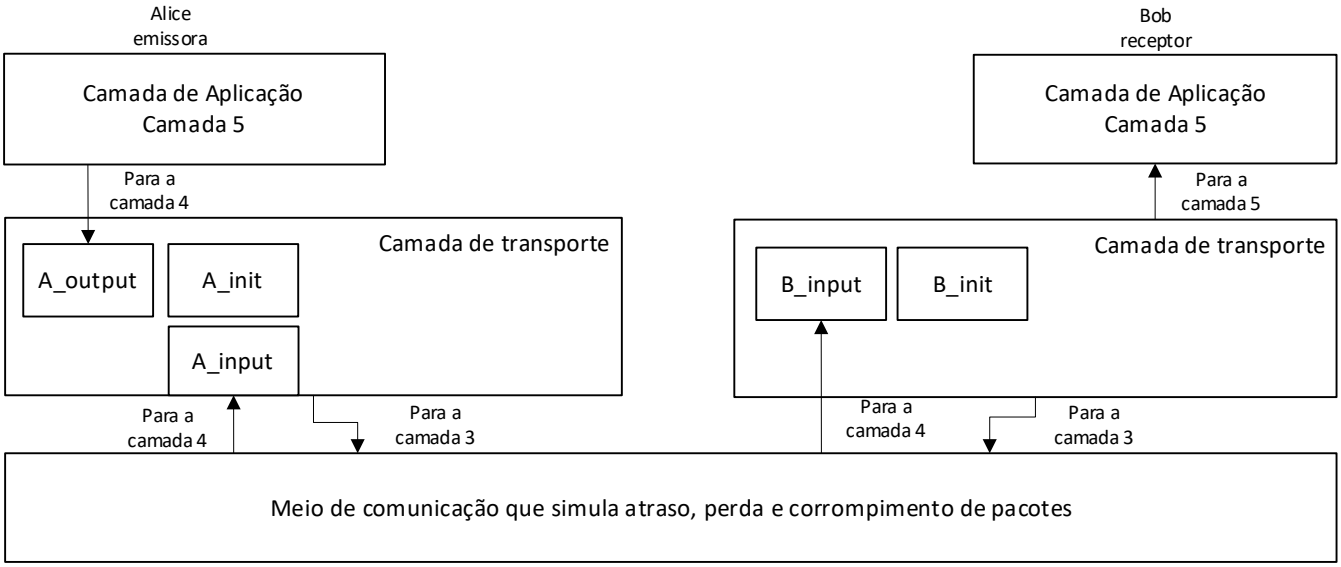
9. Deve ser disponibilizado os códigos da implementação juntamente com o relatório. O código deverá ser feito entregue via repositório no Github. O repositório deve ser fechado e deve ser aberto somente no dia da entrega (o(s) aluno(s) podem optar por adicionar o professor no Github – VielF). Também deve ser apresentado o trabalho, ou por vídeo ou em aula para o professor, apresentando todo o código e compilando no memento da apresentação. Aos que optarem pelo vídeo, podem gravá-lo e disponibilizar no Youtube como não listado e incluindo o link na estrega/relatório. Na apresentação do trabalho não é necessário utilizar slides, apenas apresentar o código e sua execução (compilar na apresentação). Aos que optarem por apresentar em aula, avisar o professor previamente.

Descrição do projeto a ser desenvolvido

Projeto 1

Neste projeto, você irá implementar a versão do algoritmo que define o protocolo Reliable Data Transfer (RTP) na versão 2.1 e versão 2.2 (como descrita no livro). Para isso, você deve simular algumas questões (abstraindo a complexidade) que são as camadas acima e abaixo da pilha de operações de referência em uma rede de computadores. No caso, a camada 5 (acima) representa a

camada de aplicação, já a camada 3 (abaixo) representa a camada de rede. Isso permite fazer uma alusão a pilha TCP/IP. A Figura abaixo ilustra a situação:



Os dados gerados pela camada de aplicação deverão seguir uma estrutura de mensagem com valores de 8 bits com no mínimo 20 valores. Como exemplo, pode ser usado a estrutura abaixo.

- Exemplo: char dados[20];

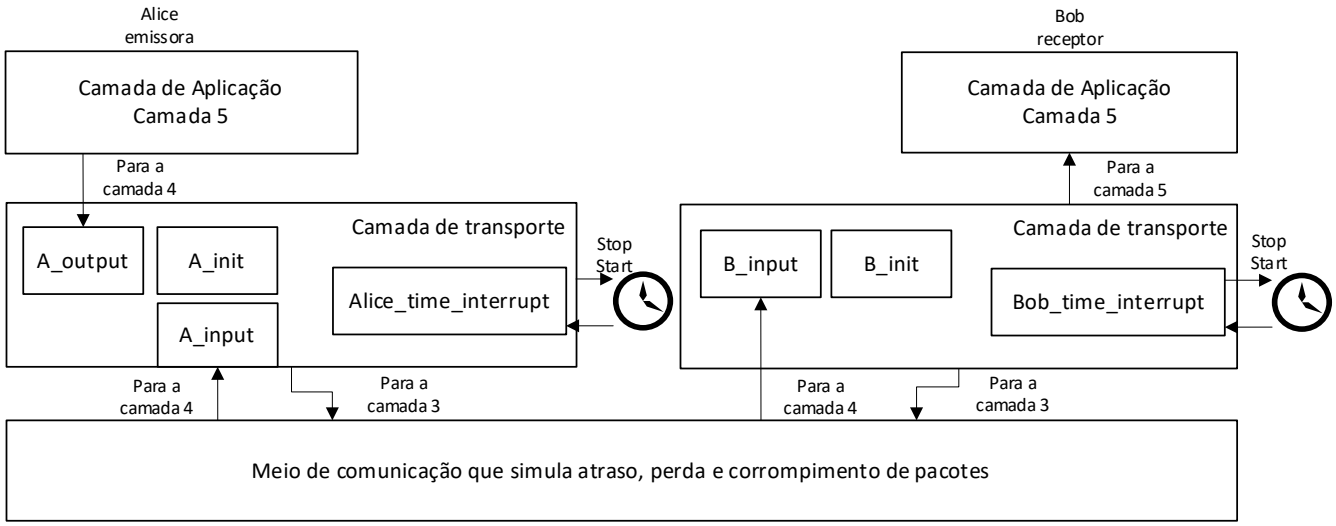
A estrutura do pacote formado na camada de transporte implementada por você(s) deverá seguir uma estrutura com no mínimo os campos abaixo.

- Seqnum – sequência do pacote enviado;
- Acknum – número do Ack gerado;
- Checksum – valor da soma de verificação calculado;
- carga útil

Lembre-se que a camada de transporte deve receber e enviar dados para a camada de rede e de aplicação, sendo assim, deve-se simular a comunicação. Além disso, você(s) deve(m) implementar o cálculo do checksum manualmente, não sendo aceito o uso de bibliotecas. Para tornar uma simulação mais realística, você deve gerar uma implementação de função que simula a inserção de atraso de propagação, perda do pacote e corrompimento dos dados contidos em todo o pacote quando solicitado (por exemplo, aleatoriamente).

Projeto 2

Com a implementação da versão 2.1 e 2.2 realizadas no Projeto 1, você(s) deve(m) agora implementar a versão 3.0 do RTP, a qual adiciona o temporizador. A Figura a seguir ilustra essa situação:



Toda a implementação feita no Projeto 1 pode ser reaproveitada. Mas, para esse projeto, você deverá implementar uma função que simule a contagem de tempo e deve seguir as máquinas de estado (FSM) dos algoritmos presentes no material usado na disciplina. A função que simula a passagem de tempo pode ser implementada usado a biblioteca timer.h (ou equivalente).

Considerações sobre ambos os projetos

As funções ilustradas devem seguir a seguinte lógica de implementação:

- Alice_output - onde a mensagem é uma estrutura do tipo mensagem, contendo dados a serem enviados para o Bob. Essa função será chamada sempre que a camada superior do lado de envio (Alice) tiver uma mensagem para enviar. É função do seu protocolo garantir que os dados em tal mensagem sejam entregues em ordem e corretamente à camada superior do lado receptor.
- Alice_input - onde pacote é uma estrutura contendo os campos citados acima. Esta rotina será chamada sempre que um pacote enviado por Bob (repassado para a camada 3 e depois para a camada de transporte da Alice) chega para Alice.
- Alice_time_interrupt - está função será chamada quando o temporizador da Alice expirar (gerando assim uma interrupção do temporizador). Você usará essa rotina para controlar a retransmissão de pacotes. Como complementar, você também deverá criar as rotinas start_timer e stop_timer para controla o temporizador.
- Alice_init - está rotina será chamada uma vez, antes que qualquer uma das outras rotinas do lado da Alice sejam chamadas. Utilize para fazer qualquer inicialização necessária.
- Bob_input - onde pacote é uma estrutura contendo os campos citados acima. Esta rotina será chamada sempre que um pacote enviado por Alice (repassado para a camada 3 e depois para a camada de transporte do Bob) chega para Bob.
- Bob_time_interrupt - está função será chamada quando o temporizador Bob expirar (gerando assim uma interrupção do temporizador). Você usará essa rotina para controlar a retransmissão de pacotes. Como complementar, você também deverá criar as rotinas

start_timer e stop_timer para controla o temporizador. Essa função existe para padronizar as camadas, mas pode ser usada para adicionar um plus no projeto (sempre bem visto).

- Bob_init - está rotina será chamada uma vez, antes que qualquer uma das outras rotinas do Bob sejam chamadas. Ele pode ser usado para fazer qualquer inicialização necessária.

Interfaces de software

Você deverá desenvolver funções de interface entre as camadas, fazendo uma alusão a sockets que poderiam ser usados em uma implementação real. Abaixo a sugestões:

to_network_layer - onde o pacote é uma estrutura com os campos citados acima. Está rotina fará com que o pacote seja enviado para a rede, com destino à outro host.

to_app_layer – usado para repassar a mensagem que segue a estrutura citada anteriormente. Com a transferência de dados unidirecional.

Simulação da comunicação

Para ambos os projetos, você(s) deve(m) implementar uma função (dentro da simulação que simulará o meio de comunicação) que implicará se a transmissão do pacote será normal, com atraso, com perda do pacote ou com corrompimento dos dados. Você pode seguir qualquer distribuição probabilística que quiser para indicar as possibilidades citadas. Simule uma comunicação que estabeleça o envio de pelo menos 20 mensagens entre os dois hosts do projeto. Quanto ao tempo para o temporizador usado nos hosts, você pode utilizar um tempo mais adequado, pode utilizar tempos de ping reais (adaptados para a simulação) para utilizar como timeout.

O ideal é reproduzir ambos os projetos com os mesmos cenários de envio para mediar a capacidade de mitigar problemas, fazendo uma comparação entre as versões 2.1, 2.2 e 3.0 do RTP.

Pontuação Extra (1,0 no trabalho ou 0,5 na prova):

Como forma de estimular uma comunicação mais real, deverá ser implementado uma comunicação bidirecional (modelo full duplex ou half duplex), onde tanto Alice quanto Bob enviam mensagens um para o outro.

Dicas:

Você pode utilizar programação com threads para facilitar funções de simulação.