Redes de Computadores
Projeto de Programação – Comunicação Confiável

Revisão 1: 11/01/2024.

1. Definição do Sistema

Neste exercício deverá implementar um protocolo com paralelismo de transporte confiável, executado acima do UDP, assumindo que o TCP não existe. Para isso, **será obrigatório utilizar os conceitos de "Go-Back-N" ou de "Repetição seletiva" (você escolhe um deles)** e a comunicação será intermediada pelo componente Canal desenvolvido no projeto anterior.

O protocolo será utilizado por um programa que realiza a transmissão de mensagens entre um *sender* (quem envia) e um *receiver* (quem recebe). A mensagem a ser transmitida deverá ter um formato (cabeçalho) que atenda as funcionalidades.

2. Recomendação Inicial

Assistir o vídeo do link https://www.youtube.com/watch?v=nysfXweTI7o e implementar os exemplos mostrados em Java. Só assistir o vídeo não lhe será de utilidade quando tiver que implementar funcionalidades mais complexas.

3. Funcionalidades

1. Implementar o formato (cabeçalho) da mensagem que será transmitida. Crie a classe *SegmentoConfiavel* com os atributos que considerar necessários para o correto funcionamento.
2. Implementar o tratamento de pacotes perdidos (uso de temporizador e timeout).
3. Implementar o tratamento de pacotes fora de ordem.
4. Implementar o tratamento de pacotes duplicados.
5. Implementar o tratamento de pacotes lentos (uso de temporizador e timeout).
6. Implementar o buffer de pacotes (recebidos, ack, enviados, etc) no *sender* e no *receiver*.
7. Mostrar na console do *sender* e na console do *receiver* as informações descritas na Seção 4.
8. Inicialização da captura/leitura de IP e porta do *sender* e o *receiver*. Deixe como default o IP 127.0.0.1 (só apertando a tecla *enter* já considerará esse IP)

Para o caso dos números de sequência e de ack, deve assumir que cada um corresponde a um pacote (cadeia de bytes completa) e não a um byte específico da cadeia (como no TCP) ou terá nota zero.

4. Mensagens (prints) apresentadas na console

Na console deverão ser apresentadas “exatamente”, nem mais nem menos, as seguintes informações usando o print (e.g, System.out.println no java).

4.1. No caso do *sender*:

- Capturar do teclado a mensagem e a quantidade de vezes a ser enviada pelo componente Canal. Para o exemplo, vamos supor que a mensagem foi “oi” com identificador X a ser enviada 1000 vezes.

- Quando receber o reconhecimento do *receiver*, mostre na console:

  Mensagem id X recebida pelo *receiver*.

- Quando o temporizador der timeout e não receber o reconhecimento:

  Dependendo da implementação, mostre o identificador do pacote específico a ser reenviado (Repetição Seletiva) ou os identificadores de todos os pacotes a serem reenviados (Go-Back-N).

4.2. No caso do *receiver*:

- Quando receber uma mensagem fora de ordem, mostre:
  Mensagem id X recebida fora de ordem, ainda não recebidos os identificadores [lista de identificadores não recebidos]

- Quando receber uma mensagem na ordem, mostre:
  Mensagem id X recebida na ordem, entregando para a camada de aplicação.

Cabe destacar que as informações do componente Canal e sua consolidação no final do teste da próxima seção também deverão ser mostradas.

5. Teste a ser realizado

5.1. Teste Local

O professor compilará o código e abrirá 2 consoles. Uma delas será o *receiver* e as outra será o *sender*. Exemplo das consoles pode ser observado no link: https://www.youtube.com/watch?v=FOwKxw9VYqI

No teste, envie 1000 mensagens sequenciais e mostre no final a consolidação dada pelo componente Canal.

Cabe destacar que:

- Inicialmente, o professor executará o *receiver*. A seguir, executará o *sender*.
- Você não precisará de 2 computadores para realizar a atividade. Basta usar 2 consoles (que correspondem a 2 processos) em 1 computador.

5.2. Teste Remoto

Mostre duas consoles, uma delas será o *receiver* e as outra será o *sender*.. O *receiver* e o *sender* deverão estar separados obrigatoriamente por uma WAN.

Os testes a serem realizados serão:

a) Mostre tanto o IP quanto a localização geográfica da máquina.
b) Realize os mesmos testes da seção 5.1.

6. Código fonte

- Deverá criar somente as classes Sender, Receiver e SegmentoConfiavel. A última deverá ser utilizada **obrigatoriamente** para o envio e recebimento de informações (nas requisições e respostas). Caso envie novas classes, serão descontados 3 pontos da nota final.
    - A única exceção é a criação das classes para Threads, mas elas deverão ser criadas dentro de uma das classes acima (e.g, classes aninhadas).

- O código fonte deverá apresentar claramente (usando comentários) os trechos de código que realizam as funcionalidades mencionadas na Seção 3.

- Comente para que serve cada atributo da classe SegmentoConfiavel

- Independente da linguagem, o uso de bibliotecas que realizem parte das funcionalidades pedidas não será aceito. Caso tenha dúvidas de alguma específica, pergunte ao professor.

7. Entrega

A entrega é individual e consistirá em: (a) um relatório com nome SeuRA.pdf; (b) o código fonte do programa com as pastas. A entrega será como definida no plano de ensino, não sendo aceita por outras formas.

- Caso tenha utilizado uma biblioteca permitida pelo professor, envie-a também.

O relatório deverá ter obrigatoriamente as seguintes seções:

a) Nome e RA do participante
b) Formato da classe SegmentoConfiavel
c) Link do vídeo do funcionamento (screencast). O vídeo do screencast deverá conter no máximo 5 minutos, mostrando a compilação, o funcionamento do código no teste local e

no teste remoto. **Não envie o vídeo do screencast**, envie só o link do vídeo, o qual pode disponibilizar no Youtube ou em outro lugar semelhante (como vimeo). Lembre-se de dar permissão para visualizá-lo.

d) No vídeo do screencast do ponto anterior, deve realizar e mostrar os testes (local e remoto) da Seção 5.
e) Breve explicação ("alto nível") do tratamento de mensagens lentas e perdidas.
f) Breve explicação ("alto nível") do tratamento de mensagens fora de ordem e duplicadas.
g) Breve explicação ("alto nível") do funcionamento e consumo do buffer.
h) Cada estrutura de dados criada deve ser explicada (para que serve, o que armazena, etc).
i) Explicação do uso das threads, mencionado as linhas do código fonte que fazem referência.
j) Explique quais foram as diferenças nas informações apresentadas nos tipos de testes local e remoto. Caso não tenha percebido diferenças, modifique os parâmetros do arquivo CONFIG do Canal. Cabe destacar que os mesmos parâmetros deverão ser usados em ambos tipos de testes.
k) Links dos lugares de onde baseou seu código (caso aplicável). Prefiro que insira os lugares a encontrar na Internet algo similar.

8. Observações importantes sobre a avaliação

A seguir mencionam-se alguns assuntos que descontarão a nota.

- Código fonte não compila ou usa bibliotecas externas sem aprovação do professor (nota zero).
- Usou ack por cada byte e não pelo pacote inteiro (nota zero).
- Não criou e usou a classe SegmentoConfiavel para a comunicação (menos 3 pontos).
- Não enviou o relatório (menos 2 pontos).
- Não enviou o link do vídeo, o link não está disponível ou o vídeo não mostra o funcionamento, separado nas consoles (menos 2 pontos).
- Enviou outros arquivos do código fonte além dos citados na Seção "Código fonte", por exemplo os .class ou outras classes (menos 3 pontos).
- Código fonte sem comentários que referenciem as funcionalidades da Seção 3 (menos 2 pontos)

9. Links recomendados (para Java)

- Para baixar a JDK 8
  https://www.oracle.com/br/java/technologies/javase/javase-jdk8-downloads.html

- Vídeo explicativo sobre programação UDP, TCP e Threads:
  https://www.youtube.com/watch?v=nysfXweTI7o

- Informações sobre programação com UDP podem ser encontradas em:
  https://www.baeldung.com/udp-in-java

https://www.geeksforgeeks.org/working-udp-datagramsockets-java/
https://docs.oracle.com/javase/tutorial/networking/datagrams/index.html

- Informações sobre Threads, que permitem que o servidor ou peer receba e envie informações de forma simultânea:
  https://www.baeldung.com/a-guide-to-java-sockets (Seção 6: TCP com muitos clientes)
  https://www.tutorialspoint.com/java/java_multithreading.htm

10. Ética

Cola, fraude, ou plágio implicará na nota zero a todos os envolvidos em todas as avaliações e exercícios programáticos da disciplina.