Sistemas Distribuídos (noturno) – Q2/2019
Exercício Programático 2

1. Definição

Neste EP você deverá criar um sistema que permita a um programa cliente pesquisar e baixar um arquivo de vídeo (por TCP ou UDP) armazenado em *peers*. O vídeo terá formato MP4 e deverá ter mais de 100 MBytes.

O programa cliente deverá ser capaz de efetuar, no mínimo, as seguintes operações:

1. Realizar uma consulta. O cliente poderá, escrevendo na console, realizar uma consulta por o nome exato do arquivo de um vídeo (e.g., A.mp4). Para isso, o cliente escolherá, de forma aleatória, um dos *peers* e lhe enviará a consulta.

2. Esperar pela resposta. O cliente esperará um determinado período de tempo por uma resposta. Caso não a receba nesse período, o cliente mostrará na console que não foi possível encontrar o arquivo A.mp4.

3. Baixar o arquivo. Caso o cliente receba uma resposta de algum *peer*, baixará o arquivo A.mp4 automaticamente desse *peer* para uma pasta local do cliente. Não poderá utilizar bibliotecas externas que façam a transferência do arquivo (e.g. apache http, apache commons, etc).

4. Visualizar o arquivo. Após ter baixado o arquivo, a pessoa (professor) poderá visualizar o vídeo utilizando um reprodutor externo ao sistema (e.g., Windows Media Player, VLC, etc).

A especificação do ***peer*** X terá os seguintes itens:

5. Obter metadados. Cada certo tempo, o *peer* X obtém os arquivos presentes em uma pasta local e armazena (em memória) os seus metadados. Esse conjunto de metadados será denominado de **estado do *peer*** X.

6. Receber a consulta e enviar a resposta. O *peer* X que receber uma consulta por um arquivo observará se o mesmo está armazenado no seu estado. Se esse for o caso, enviará a resposta ao cliente de que X tem o arquivo.

7. Encaminhar a consulta para outro *peer*. Caso o *peer* X não tenha o arquivo, encaminhará a consulta para um dos W *peers* armazenados por X. A escolha por W será de forma aleatória. O encaminhamento deve seguir o mecanismo de ***flooding*** **com TTL** - *Time to Live* – e o valor do TTL deverá ser enviado como parâmetro ao iniciar a execução do *peer*.

8. Verificar se a consulta é repetida. O *peer* que receber a consulta de outro *peer* realizará as mesmas atividades 6 ou 7 somente se a consulta **não foi recebida anteriormente** (mensagem duplicada).

9. Enviar o arquivo a um cliente. O *peer* enviará o arquivo ao cliente que pediu para baixá-lo no item 3.

* Assuma que o cliente conhece todos os endereços e portas dos *peers*.
* Assuma que o *peer* conhece todos os endereços e porta dos outros *peers*.

A consulta a ser encaminhada via *Flooding* deverá ter um formato apropriado para que seja possível ao peer que a receberá atender todos os itens.

A execução do programa deverá poder ser realizada com um número arbitrário de peers. Na avaliação, será exigida a sua execução com no mínimo 10 (dez) *peers* e 2 clientes.

2. Entrega

A entrega poderá ser realizada por um **grupo de até duas pessoas** e consistirá em um relatório e o código fonte do programa a ser entregue pelo TIDIA, na aba atividades.

O relatório deverá ter obrigatoriamente as seguintes seções:

a) Nome e RA dos participantes
b) Formato da mensagem encaminhada entre *peers* via *flooding*.
c) Explicação em "alto nível" do *flooding* e TTL implementado.
d) Explicação em "alto nível" do tratamento de mensagens duplicadas.
e) Links dos lugares de onde baseou seu código (caso aplicável).

O código fonte (e na avaliação) deverá apresentar claramente na console as execuções dos sete itens definidos na Seção 1. Por exemplo:

- Console cliente 1: pesquisando por arquivo A.mp4 no *peer* X.

Console *peer* X: recebendo pesquisa A.mp4, tenho o arquivo A.mp4 no meu estado.
Console cliente 1: baixando arquivo A.mp4 do *peer* X.
Console cliente 1: arquivo A.mp4 baixado

- Console cliente 2: pesquisando por arquivo B.mp4 no *peer* W.

Console *peer* W: recebendo pesquisa B.mp4, NÃO tenho o arquivo, encaminho para Z.
Console *peer Z*: recebendo pesquisa B.mp4, NÃO tenho o arquivo, encaminho para X.
Console cliente 2: tempo esgotado para consulta por arquivo B.mp4.

- Console cliente 1: pesquisando por arquivo C.mp4 no *peer* W.

Console *peer* W: recebendo pesquisa C.mp4, NÃO tenho o arquivo, encaminho para Z.
Console *peer Z*: recebendo pesquisa C.mp4, NÃO tenho o arquivo, encaminho para X.
Console *peer X*: recebendo pesquisa C.mp4, NÃO tenho o arquivo, encaminho para Y.

Console *peer Y*: recebendo pesquisa C.mp4, NÃO tenho o arquivo, encaminho para Z.
Console *peer Z*: recebendo pesquisa C. mp4, MSG DUPLICADA NÃO ENCAMINHO.
Console cliente 1: tempo esgotado para consulta por arquivo C.mp4

- Console cliente 1: pesquisando por arquivo D.mp4 no peer Z.

Console *peer Z*: recebendo pesquisa D.mp4, NÃO tenho o arquivo, encaminho para X.
Console *peer X*: recebendo pesquisa D.mp4, NÃO tenho o arquivo, encaminho para Y.
Console *peer Y*: recebendo pesquisa D.mp4, TTL=ZERO NÃO ENCAMINHO.
Console cliente 1: tempo esgotado para consulta por arquivo D.

3. Datas

A entrega tanto do relatório quanto do código fonte deverá ser realizada até o dia 8 de agosto às 19.00 (antes da aula prática) **somente via TIDIA**. Envios por email ou outra forma não serão aceitos.

4. Avaliação

A avaliação será realizada na aula prática do dia 8 de agosto. De forma aleatória o professor escolherá os participantes a serem avaliados e lhes fará perguntas. Caso seu grupo não seja selecionado, a avaliação será realizada na próxima aula prática, com o código entregue no dia 8 de agosto.

No dia da avaliação, deixe pronto o sistema para funcionar e o código fonte aberto para responder as perguntas.

Observações:

- Caso o participante não esteja presente na hora da chamada terá nota zero.
- Caso o código não compile ou não esteja pronto para mostrar, terá nota zero, independentemente de ter entregue o relatório.

5. Atrasos

Para os atrasos, a nota máxima (após a avaliação na aula prática) será baseada na tabela abaixo.

Dias de atraso	Nota máxima
0	10
1	7
2	6
3	5

+3	0

7. Bônus

O participante terá um bônus de 2 pontos se os 10 *peers* e os 2 clientes forem executados em 12 múltiplas instâncias Ubuntu (do EC2 da Amazon) diferentes. Caberá ao participante mostrar o funcionamento das instâncias na hora da apresentação.

Para ter acesso aos benefícios da Amazon (para alunos da UFABC) realize o cadastro com seu email da UFABC no site: https://aws.amazon.com/pt/education/awseducate/

8. Links recomendados

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 o *flooding* podem ser encontradas em:

https://www.youtube.com/watch?v=P6hn1kSECng
Seção 2.3 do livro Sistemas Distribuídos (3ª ed. inglês) do Tanenbaum. pp 85.

Informações sobre como transferir arquivos via TCP podem ser encontradas em:

https://www.devmedia.com.br/file-transfer-between-2-computers-with-java/24516

9. Ética

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