Especificação do Trabalho III – ELC1018 Sistemas Distribuídos
Programação de um middleware para Comunicação *multicast* com Mecanismo de Estabilização para descarte de mensagens do buffer baseado em vetor de vetor de relógios lógicos

O trabalho consiste em **implementar um *middleware* para envio de mensagens *multicast* e realizar o controle de estabilização de mensagens para descarte das mensagens do buffer**. Consulte o material teórico e os slides para compreender o **algoritmo para tratar para tratar estabilização de mensagens**. O trabalho deve ser feito em dupla.

O *middleware* deve ser um único *packet* Java nomeado `StableMulticast` que será importado por algum programa Java (usuário) e que oferecerá através de sua API a facilidade de enviar uma mensagem *multicast* para um conjunto de destinatários (grupo) com descarte de mensagens estáveis do buffer. Vide detalhes da API abaixo. Funcionalmente, o *middleware* deve implementar a comunicação *multicast* entre suas instâncias (programas que usam o *packet* e que executam em diferentes nodos/computadores) através do envio de mensagens *unicast* não confiável (sockets UDP) a todos os membros do grupo destinatário. O middleware deve usar IP *multicast* para realizar a descoberta das instâncias do usuário que executam o middleware (participantes da computação) e a comunicação entre instâncias do middleware deve usar sockets UDP. Neste trabalho você não deve usar RMI.

A interface do *middleware* (API do packet `StableMulticast`) deve oferecer um método `msend(msg, this)`, para os usuários enviarem mensagens *multicast* com ordenamento causal, e um método `deliver(msg)`, para o usuário receber mensagens. Note que o parâmetro `this` do método `msend` é quem passa a referência do objeto remoto para que a resposta seja recebida por *call-back* no método `deliver`. O serviço de descoberta deve permanecer sempre ativo, a fim de permitir atualização dinâmica dos membros do grupo. Como cada instância do *middleware* deve conter as informações sobre quem participa do grupo (resultado da etapa de descoberta), e a atualização é realizada dinamicamente, o encaminhamento das mensagens *multicast* será realizado ao grupo corrente.

Importação do pacote `StableMulticast`:
```
import StableMulticast.*;
```

Interface que deve ser implementada por todo usuário do pacote `StableMulticast`:
```
Public interface IStableMulticast {
    public void deliver(String msg);
}
```

API oferecida pelo pacote `StableMulticast`:
```
public StableMulticast(String ip, Integer port, IStableMulticast client)
public void msend(String msg, IStableMulticast cliente)
```

Para possibilitar a correção do trabalho, faça o envio de cada mensagem *unicast* ser controlado via teclado, ou seja, deve haver uma pergunta antes de cada envio *unicast* (controle) questionando se é para enviar a todos ou não. O caso "não" deve permitir o envio um a um no *unicast*, aguardando posteriormente para que o professor/aluno decida sobre o envio da mensagem. Por exemplo, com três processos, deve-se poder escolher pelo menos um para envio posterior (atrasado). O conteúdo do buffer e dos relógios lógicos também precisam ser permanentemente demonstrados na tela/terminal. Não é necessário implementar uma GUI na aplicação do usuário.

O algoritmo de estabilização a ser implementado pode ser o proposto no artigo *Fundamentals of Distributed Computing: A Practical Tour of Vector Clock Systems*, disponível em Slides no Moodle e na Web. Porém outros algoritmos com vetores de relógios lógicos também podem ser implementados (indique isto na documentação, se ocorrer).

Na avaliação o seu middleware será testado com um cliente do professor! Respeite as especificações!

O algoritmo para estabilização das mensagens está descrito a seguir e o seu comportamento resumido está ilustrado na figura 1.

```
procedure mcsend(msg)                                  % realizado por Pi
    msg.VC ← MCi[i][*]                                 % constrói o timestamp de msg
    msg.sender ← i
    for all P do send(msg) to Pj enddo                 % multicast
    MCi[i][i] ← MCi[i][i]+1                            % Pi fez mais um multicast

when Pi receives msg from Pj                           % msg traz msg.VC por piggyback
    deposit(msg)                                       % coloca msg no buffer
    MCi[j][*] ← msg.VC                                 % atualiza visão do Pi com visão de Pj
    if i ≠ j then MCi[i][j] ← MCi[i][j]+1              % mais 1 msg de Pj entregue
    deliver msg to the upper layer                     % evento de entrega

when (existe msg no buffer_i AND msg.VC[msg.sender] ≤ min_{1≤x≤n}(MCi[x][msg.sender])
    discart(msg)                                       % elimina msg do buffer local
```

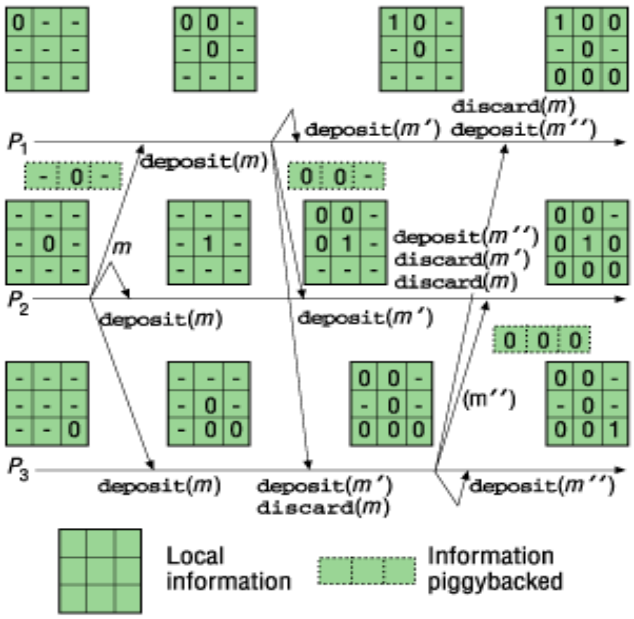

Figura 1 – Funcionamento do algoritmo para estabilização de mensagens.

A entrega do trabalho deve ser no Moodle e a apresentação em sala de aula.