Trabalho de Sistemas Distribuídos - Implementação Casa de Leilão
Parte 1

Nomes: Rafaela Martins VIeira — Matrícula:2852
Rodrigo César SIlveira — Matricula:2316
Saulo Camargo — Matricula:2783

IFMG - Campus Formiga
06 de Junho de 2018

Resumo das funções e Implementação:

O trabalho possui seis classes, sendos estas: **Item**, **Sala**, **SistemaInterface**, **SistemaNucleo**, **SalaInterface**, **ChapeuDeLeiloeiro**.

- A Classe **Item**, basicamente possui sets e gets do ID do item, nome do item, dono do item, data de uso e flag de verificação(Emuso).
- A Classe **Sala**, possui get e sets do nome da sala, um objeto item, lance atual,uma flag de leilão finalizado e usuário que deu o último lance.
- A Classe **SistemaInterface**: Está classe estende o ReceivedAdapter do JGroups e implementa o RequestHandle. Responsável por controlar toda a interface do usuário com o sistema e também o envio e despacho de mensagens multicast para todos os usuários. Faz papel de View e Controler do programa.
- A Classe **SistemaNucleo**: Possui uma lista de Itens e uma lista de Salas, é a parte de A A Classe Modelo do código, onde as informações são salvas dentro de variáveis e listas.
- A Classe **SalaInterface**: É uma View e Controler igual ao SistemaInterface porém está controla e interfaceia as operações e exibições das salas de leilões que estão sendo executadas. O primeiro usuário é o coordenador da sala.
- A Classe **ChapeuDeLeiloeiro**, é uma interface de thread que de 10 em 10 segundos atualiza um contador e escreve uma frase de “Dou-lhe n” sendo n a fase do lance. Após 30 segundos sem nenhum novo lance ele finaliza o leilão voltando pro chat global.

Decisões de Arquitetura do Sistema [XML]

Utilizou-se **UDP** como camada de Transporte. É mais leve, não possui algoritmo de HandShake que o TCP utiliza, assim se adicionarmos um protocolo de recebimento como o NACKAK ele garantirá a entrega e ficará mais leve que o TCP por possuir menos conteúdo nos blocos de mensagem. **Ping** para explorar vizinha. Utilizou-se o Ping para garantir consistência dos membros dentro do grupo. Um **NACKAK** para garantir o recebimento das mensagens por todos os membros do grupo. Como se utilizou o UDP e este não garante entrega o NACKAK fica responsável por trazer essa confirmação de recebimento. **Central Lock** para não permitir que dois usuários alterem o mesmo item simultaneamente. Se dois usuário ao mesmo tempo criarem um item o ID ficará repetido para ambos os itens. Os IDs por sua vez devem ser únicos para cada item, desta forma foi-se necessário travar esta ação para garantir

que o ID seja modificado atomicamente. **StateTransfer**, utilizado no getState para atualizar os novos membros que chegarem no grupo com as versões mais recentes.

FRAG2 para fragmentar as mensagens. Determina o tamanho do pacote. **Verify Suspect** também foi utilizado para suspeitar se o usuário saiu ou está demorando enviar as respostas. O Ping possuía verificações precipitadas sobre membros que saíram do grupo, o **VerifySuspect** foi usado para verificar se o membro está demorando a responder ou se ele realmente saiu.