Paradigmas de Sistemas Distribuídos
Sistemas Distribuídos em Grande Escala
"Ordenação causal num sistema reativo"

Trabalho Prático - I

2022/2023

Informações gerais

- Cada grupo deve ser constituído por até 4 elementos, de ambas ou de uma das UCs.
- Deve ser entregue o código fonte e um relatório de até 6 páginas (A4, 11pt) no formato PDF.
- O trabalho deve ser entregue até às 23:59 do dia 21 de abril de 2023 no *eLearning*.
- A apresentação do trabalho ocorrerá em data a anunciar.
- O trabalho será classificado separadamente para cada uma das UCs, de acordo com os objetivos de cada uma delas.

Resumo

A ordenação causal de mensagens facilita várias tarefas em sistemas distribuídos e é uma parte importante de muitos algoritmos. É por isso útil implementar essa funcionalidade como um componente de software que possa ser reutilizado em diferentes contextos. Tendo em conta que vai ser utilizada em servidores que devem ser capazes de acomodar um grande número de ligações, pretende-se que o componente de software a desenvolver respeite as interfaces e regras de composição do paradigma ReactiveX.

Funcionalidade básica (até 16 valores)

Pretende-se um operador reativo para fazer respeitar a ordem causal de mensagens durante o seu processamento, por exemplo, para ser usado assim:

```
var in = ...                                  // Mensagens desordenadas
                                              //  da classe CausalMessage<T>
in.lift(new CausalOperator<T>(n))             // Ordenação
 .map(payload -> process(payload))            // Processamento de mensagens
                                              //  ordenadas da classe T
 .subscribe();
```

Pretende-se assim implementar uma única classe, `CausalOperator`, que processa uma sequência de objetos da classe `CausalMessage` que incluem:

- etiqueta necessária para a ordenação;

- um conteúdo arbitrário.

Estes representam mensagens difundidas entre um grupo de processos segundo *causal broadcast*, e que estão a chegar ao componente por uma ordem arbitrária, possivelmente com duplicados, mas sem perda de mensagens. O operador deve produzir uma sequência de conteúdos de acordo com a ordem causal das etiquetas. Isto significa que:

1. não deve entregar mensagens desrespeitando a ordem causal;
2. não deve omitir nem duplicar mensagens na sequência entregue.

Podem usar o esqueleto para a classe `CausalOperator`, a classe `CausalMessage` e os testes unitários em anexo como ponto de partida.

Funcionalidade avançadas (mais do que 16 valores)

Na implementação avançada deste operador devem ainda:

1. respeitar *back pressure* (i.e., implementar `Flowable` e não apenas `Observable`), admitindo que pode ser usada num contexto *multi-thread*;
2. escolher uma representação e implementação eficiente da ordem causal, minimizando o estado necessário e o número de operações ao ordenar mensagens.

Objetivos

PSD O trabalho deve demonstrar para programação por eventos de acordo com o paradigma ReactiveX (i.e., em Java, usando RxJava) e ter em conta o comportamento dinâmico.

SDGE O trabalho deve demonstrar um algoritmo de ordenação causal e ter em conta considerações de eficiência.