	UNIVERSIDADE FEDERAL DA PARAÍBA CENTRO DE INFORMÁTICA	
	Disciplina	Linguagem de Programação II
	Semestre	2016.2
	Professor	Bruno Jefferson de Sousa Pessoa

PROJETO DA DISCIPLINA

1. Introdução

O projeto consiste na implementação de um *proxy* de vídeo utilizando os conceitos e técnicas de programação concorrente apresentados durante o curso.

2. Definição do problema

Um *proxy* de vídeo é um software que recebe pacotes de dados através de uma fonte de vídeo (*streamer*) e os distribui para um conjunto de clientes (*players*) que exibirão o vídeo transmitido.

O proxy a ser desenvolvido deverá ser composto por um conjunto de threads divididas em dois grupos principais, representados pelo retângulo principal da Figura 1. O primeiro conterá a thread produtora, cujo objetivo é capturar os pacotes enviados pelo *streamer* e armazená-los em um *buffer* limitado. O segundo grupo diz respeito às threads consumidoras. Elas terão a finalidade de ler os pacotes do buffer, retransmitindo-os para seus respectivos clientes. A implementação desses componentes deve seguir o modelo de sincronização produtor/consumidor, consistindo no **primeiro** mecanismo de sincronização abordado no projeto.

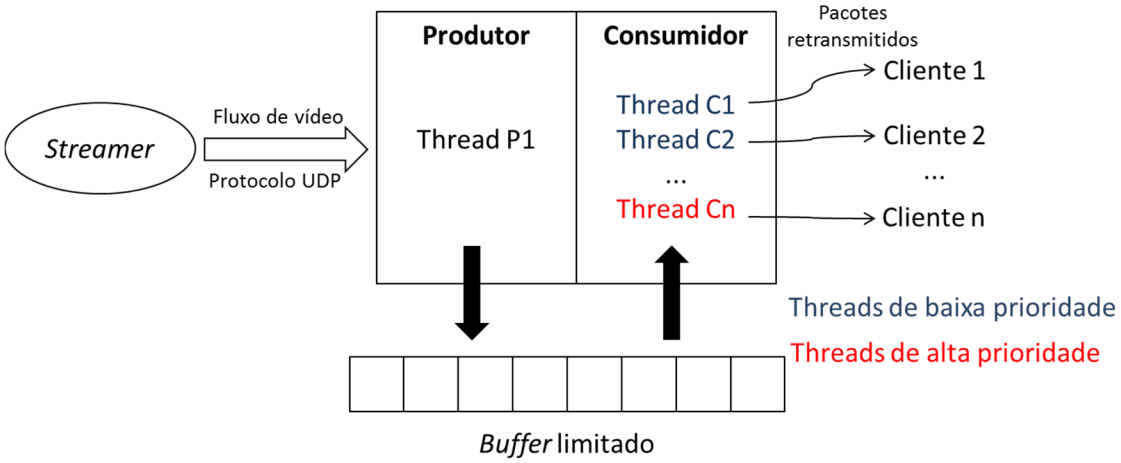

Figura 1. Representação ilustrativa dos componentes de software do projeto e os seus relacionamentos.

As threads consumidoras são classificadas em dois tipos: as threads de prioridade baixa e de prioridade alta. As threads do primeiro tipo não possuem nenhum requisito especial em relação aos recursos da rede, portanto podem executar em paralelo. O resultado esperado é a exibição simultânea do vídeo transmitido por vários clientes distintos. No caso das threads de alta prioridade, há uma demanda por toda à largura de banda da rede (como se elas fossem destinadas a tocar vídeos 4K, por exemplo), fazendo com que apenas uma thread desse tipo seja executada por vez. Esse modelo de sincronização corresponde ao problema dos Leitores e Escritores, no qual os leitores podem ser vistos como as threads de baixa prioridade e os escritores como as de alta prioridade. Tal sincronização representa o **segundo** mecanismo de sincronização abordado no projeto.

3. Instruções gerais

- O presente projeto deve ser desenvolvido em grupo de no máximo 3 alunos.
- Sua entrega está marcada para o dia **09/06/2017.**
- Deverá ser entregue um arquivo zipado, contendo o código fonte, no formato descrito a seguir:
    - LP2-Projeto-grupo.zip
    - Ex.: LP2-Projeto-jose-maria.zip
- O projeto será apresentado pelo grupo no dia de sua entrega.
- Os seguintes itens serão avaliados: execução, interface, código-fonte, utilização da técnica de Passagem de Bastão, solução do problema Produtor/Consumidor, solução do problema dos Leitores/Escritores e apresentação. A última terá uma avaliação individual para cada aluno.

4. Instruções para implementação

- O projeto deve ser desenvolvido em Java ou em C++.
- A transmissão de pacotes de vídeo deve ser realizada através do protocolo UDP.
- Semáforos devem ser utilizados para sincronização de acesso às seções críticas.
- A implementação da sincronização Leitores/Escritores deve utilizar a técnica de passagem de bastão.
- No mínimo 5 (oito) threads consumidoras deverão estar executando simultaneamente durante a apresentação do projeto.
- A interface do usuário deverá possuir, no mínimo, os seguintes campos:
    - Produtor (Servidor): IP e porta;
    - Consumidor (Cliente): IP, porta, tempo de execução ou botões play/pause e o tipo do cliente (baixa ou alta prioridade).
- Abaixo há uma sugestão de interface com campos requeridos:

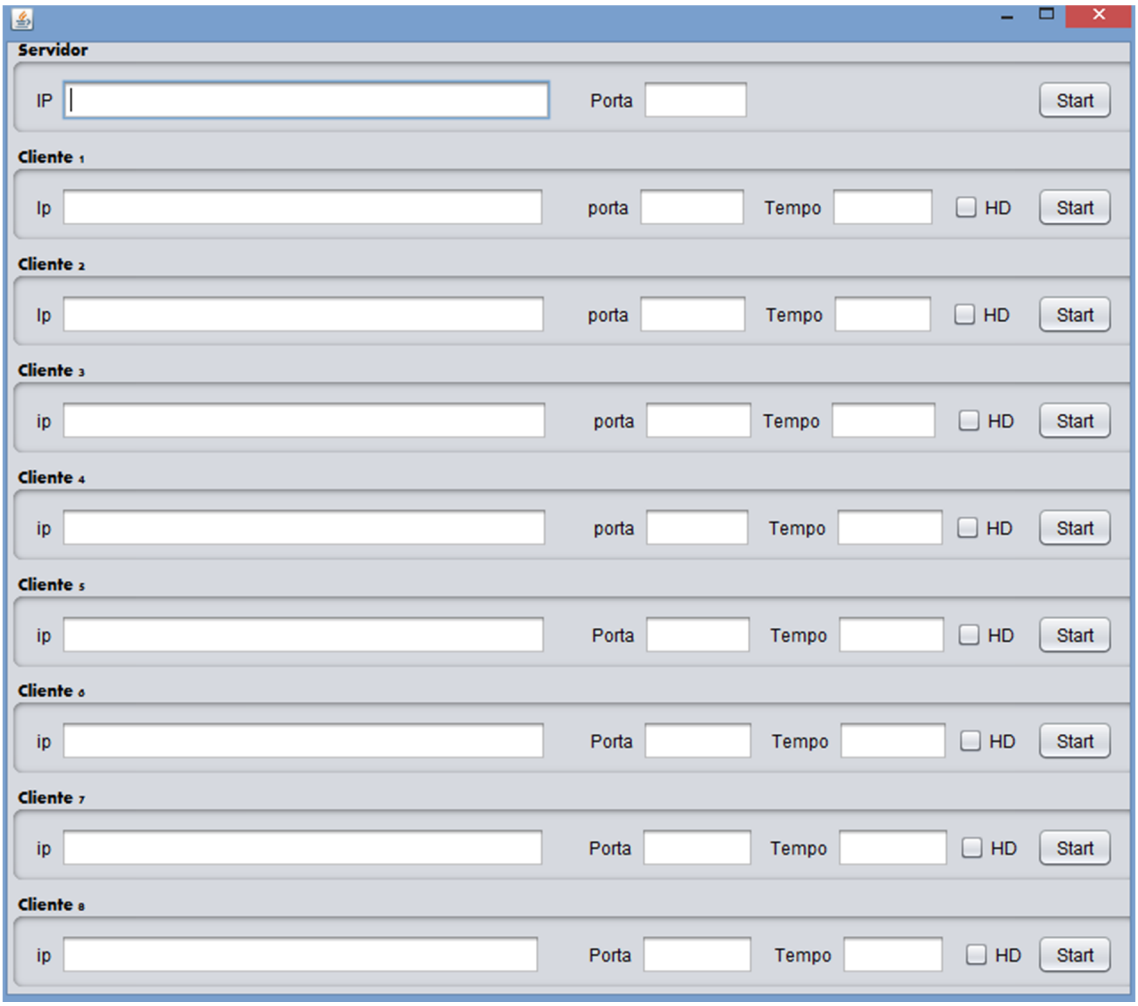

Figura 2. Exemplo de interface com os campos requeridos.

- Como sugestão de solução, segue abaixo um esboço dos principais componentes proxy:

```
process Produtor {
   while (true) {
      Recebe pacotes da rede.
      Protocolo de entrada (Buffer Limitado).
      Escreve no buffer.
      Protocolo de saída (Buffer Limitado).
   }
}
```

```
process Consumidor [i=1 to n] {
   while (true) {
      Protocolo de entrada (Leitores e Escritores).
      while (true) {
        Protocolo de entrada (Buffer Limitado).
        Lê dados do buffer.
        Protocolo de saída (Buffer Limitado).
        Envia pacote para o cliente.
        if (paused) break;
      }
      Protocolo de saída (Leitores e Escritores).
   }
}
```