	UNIVERSIDADE FEDERAL DA PARAÍBA CENTRO DE INFORMÁTICA	
	Disciplina	Linguagem de Programação II
	Semestre	2017.2
	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 com *buffer* limitado.

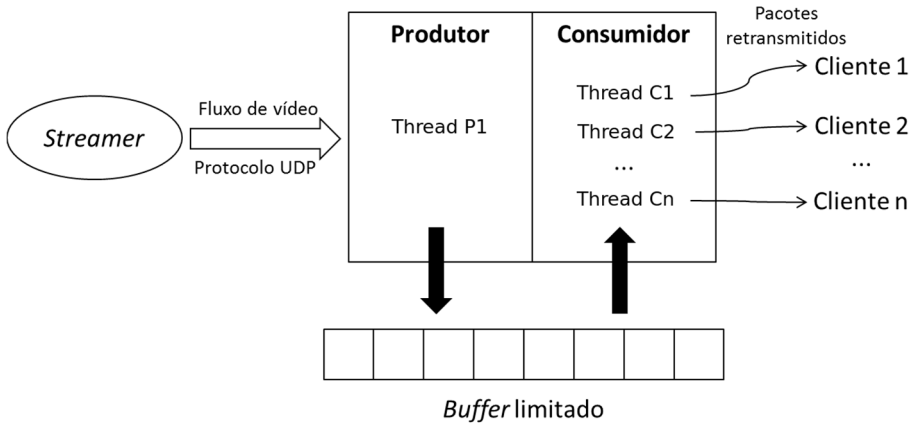
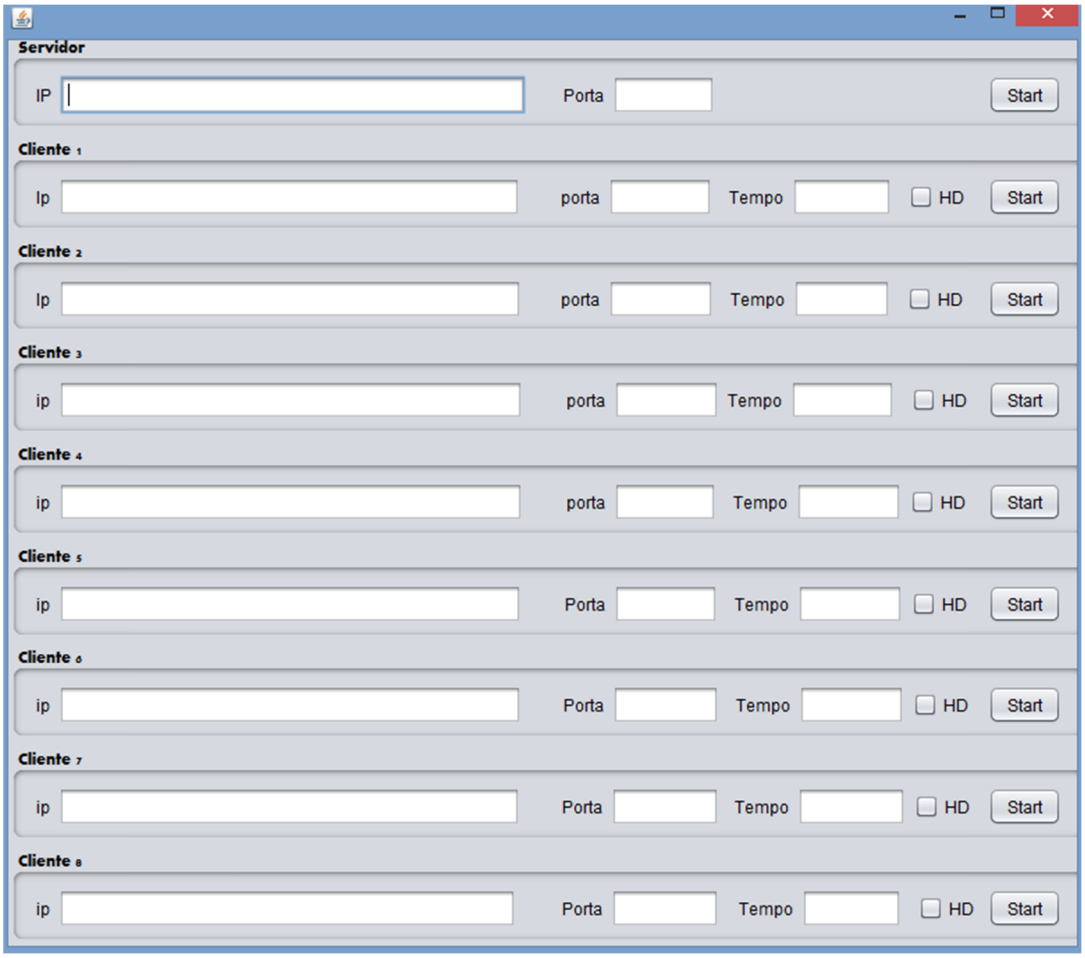


Figura 1. Representação dos componentes de software do projeto e seus relacionamentos.

A thread produtora irá esperar até que exista ao menos um *slot* disponível no *buffer* limitado para o depósito de um pacote. Para um *slot* ficar disponível, é necessário que **todas as threads consumidoras** tenham lido o pacote nele armazenado. Por suas vezes, as threads consumidoras irão aguardar até que exista ao menos um pacote novo depositado pela thread

produtora, isto é, um pacote ainda não lido por elas anteriormente. As threads consumidoras irão atuar de forma totalmente **independente entre si**, ou seja, não haverá sincronização entre elas. O número total de threads deve ser variável e sua instanciação será realizada a partir da solicitação do usuário. A Figura 2 ilustra um exemplo de interface gráfica com os campos que deverão ser preenchidos pelo usuário.

Figura 2. Exemplo de interface com os campos requeridos.



3. Instruções gerais

- O presente projeto deve ser desenvolvido em grupo de no máximo 3 alunos.
- O projeto deverá ser apresentado no dia 19/06/2018 **e** entregue até às 23:59 dia anterior.
- 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
- Os seguintes itens serão avaliados: execução, interface, código-fonte, solução do problema Produtor/Consumidor 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.
- O projeto deverá conter no mínimo duas classes: **Produtor** e **Consumidor**.
- As variáveis compartilhadas devem ser declaradas como membros de classes.
- A interface do usuário deverá possuir, no mínimo, os seguintes campos:
    - Produtor (Servidor): IP, porta e um botão para iniciar a thread.
    - Consumidor (Cliente): IP, porta e um botão para iniciar a thread.
- Como sugestão de solução, segue abaixo um pseudocódigo dos principais componentes do proxy:

Solução 1

```
sem empty = n, /* n é o número de slots no buffer */
    full[m] = ([m] n), /* m é o número de consumidores */
    mutex = 1;

int rear = 0, front[m] = ([m] 0);

bool s_read[m][n]; /* Indica se um slot já foi lido por uma thread
consumidora*/

process Produtor {
   while (true) {
      P(empty);
      Recebe pacotes da rede;
      buffer[rear] = pacote;
      rear = (rear + 1) % n;
      for (int i =1; i <= m; i++)
         V(full[i]);
   }
}


process Consumidor[j=1 to m] {
   int aux;

   while (true) {
      P(full[j]);
      aux = front[j];
      pacote = buffer[aux];
      Envia pacote para o cliente;
      front[j] = (front[j] + 1) % n;

      P(mutex);
      s_read[j][aux] = true;

      for (int i=1; i <= m; i++)
         if (s_read[i][aux] == false) {
            V(mutex);
            Vá para o início do while;
         }

      for (int i=1; i <= m; i++)
          s_read[i][aux] = false;

      V(empty); // Libera o produtor
      V(mutex);
   }
}
```

Solução 2

```
sem empty[m] = ([m], n) /* n é o número de slots no buffer */
    full[m] = ([m] 0); /* m é o número de consumidores */

int rear = 0, front[m] = ([m] 0);

process Produtor {
   while (true) {
      for (int i = 1; i <= m; i++)
         P(empty[i]);
      Recebe pacotes da rede;
      buffer[rear] = pacote;
      rear = (rear + 1) % n;
      for (int i = 1; i <= m; i++)
         V(full[i]);
   }
}


process Consumidor[i=1 to m] {
   while (true) {
      P(full[i]);
      pacote = buffer[aux];
      Envia pacote para o cliente;
      front[i] = (front[i] + 1) % n;
      V(empty[i]);
   }
}
```