Definição do Trabalho Final

A definição do trabalho consiste em implementar uma aplicação que simula o funcionamento de uma rede local em anel. A aplicação deverá implementar a transmissão de mensagens entre as máquinas que compõem o anel, utilizando o protocolo UDP como transporte. Deve ser implementada uma fila para armazenar as mensagens que serão enviados por cada máquina da rede, e somente um item da fila (mensagem) pode ser transmitido por vez.

Em uma rede em anel, há o uso de ***tokens***, que são pacotes especiais que circulam na rede e permitem a transmissão das mensagens por cada máquina da rede. Desta forma, a aplicação deverá implementar este *token* que ficará circulando na rede.

O programa deverá possuir dois tipos de pacotes: o ***token*** e os **dados**. Ao iniciar o programa, o usuário deverá informar o endereço IP da máquina que está a sua direita, um apelido e o tempo do *token* e dos dados.

As máquinas da rede serão identificadas por apelidos e deve-se especificar o tempo que elas permanecerão com os pacotes (para fins de depuração), em segundos. Tais informações devem ser inseridas na aplicação através de um arquivo de configuração. Além disso, apenas uma determinada máquina deve ser a responsável por gerar o *token* a primeira vez (esta máquina deve ter o valor *true* no arquivo de configuração, enquanto as outras terão o valor *false*).

O arquivo de configuração deverá seguir o seguinte formato:

```
<ip_destino_do_token>:porta
<apelido_da_máquina_atual>
<tempo_token>
<token>
```

Exemplo:

```
10.32.143.12:6000
Bob
1
true
```

A máquina que gera o *token* deve enviá-lo para a máquina que está a sua direita no anel (IP configurado no arquivo de configuração). Caso a máquina que recebeu o *token* não tenha dados para transmitir (fila de mensagens vazia), o *token* será enviado para a próxima máquina do anel (máquina a direita - IP configurado no arquivo de configuração). Caso contrário, a primeira mensagem é retirada da fila e é enviada para a máquina a sua direita, ou seja, os dados também devem seguir a ordem do anel.

Os dados enviados deverão retornar à máquina origem e somente depois disso o *token* poderá ser enviado para a próxima máquina.

Quando a máquina origem enviar um pacote de dados, um campo no cabeçalho do pacote deverá ser marcado como "`naoexiste`". Esse pacote poderá retornar para a máquina origem com uma das seguintes configurações:

- "**`naoexiste`**": significa que a máquina destino não se encontra na rede ou está desligada. Neste caso, uma **mensagem na tela** deve informar o ocorrido, a mensagem deve ser retirada da fila e o *token* deve ser transmitido para a próxima máquina do anel;
- "**`NACK`**": significa que a máquina destino identificou um erro no pacote e o mesmo deverá ser retransmitido pela origem. Neste caso, uma **mensagem na tela** deve informar o ocorrido, o *token* deve ser transmitido para a próxima máquina do anel, e a mensagem não deve ser retirada da fila, sendo retransmitida na próxima passagem do *token*;
- "**`ACK`**": significa que o pacote foi recebido corretamente pela máquina destino. Neste caso, uma **mensagem na tela** deve informar o ocorrido, a mensagem deve ser retirada da fila e o *token* deve ser transmitido para a próxima máquina do anel.

Antes de enviar uma mensagem, o cálculo de controle de erro deverá ser inserido na mensagem pela máquina origem. Deve-se utilizar o **CRC32** como técnica de controle de erro. A aplicação deve implementar um **módulo de inserção de falhas** que force as máquinas a inserirem erros aleatoriamente nas mensagens. Este módulo deve trabalhar com alguma probabilidade para inserir erro nas mensagens.

Ao receber uma mensagem, a máquina destino deve recalcular o controle de erro. Caso a mensagem tenha sido recebida com erro, ela deve ser marcada com "**`NACK`**", caso contrário ela deve ser marcada com "**`ACK`**".

Além disso, deverá ser implementada uma **fila de mensagens** em cada máquina. Esta fila poderá estar vazia ou não. A fila poderá conter **até 10 mensagens**. Para cada mensagem adicionada, deve ser armazenado também o apelido da máquina destino.

Os serviços de envio de dados oferecidos pela aplicação devem contemplar duas formas de transmissão:

- ***Unicast***: envia o pacote para um único destino;
- ***Broadcast***: envia o pacote para todas as máquinas da rede usando o apelido **TODOS**. Neste caso, o módulo de inserção de falhas deve manter o pacote em "**`naoexiste`**".

Descrição dos Pacotes

A implementação da aplicação deve seguir fielmente o formato dos pacotes descritos a seguir, pois durante a apresentação do trabalho, **aplicações de grupos diferentes deverão se comunicarem**. As corretas interações entre as diferentes implementações fazem parte da avaliação do trabalho.

Token

O *token* será formado por uma sequência numérica em formato string e terá o valor 9000, como mostra o exemplo a seguir:

```
9000
```

Pacote de dados

Um pacote de dados é formado por outra sequência numérica em formato string e terá o valor 7777. Neste caso, o valor 7777 será seguido de um '**:**' e pelos campos:

<controle de erro>;<apelido de origem>;<apelido do destino>;<CRC>;<mensagem>.

Exemplo:

```
7777:naoexiste;Bob;Mary;19385749;Oi pessoal!
```

No destino
Ao receber um pacote de dados, a estação identifica se o mesmo é endereçado a ela, verificando o apelido do destino. Caso não seja, este pacote deve ser enviado para seu vizinho da direita. Caso o pacote seja para ela, a aplicação deve recalcular o CRC, imprimir o apelido da origem e a mensagem, e deve também enviar o pacote de volta, alterando o campo `naoexiste` para `ACK` ou `NACK`.

Na origem
Caso o pacote de dados seja recebido por quem o originou (o apelido de origem será igual ao seu apelido), será necessário verificar o controle de erro, pois a mensagem deu toda a volta no anel. Ao receber o pacote com o campo em `naoexiste` ou `ACK`, um *token* deve ser enviado para seu vizinho da direita. Caso o pacote venha com `NACK`, o mesmo deve ser retransmitido apenas uma vez na rede, trocando o `NACK` por `naoexiste`, colocando a mensagem original sem erro e enviando a mensagem para a máquina a sua direita na próxima passagem do *token*.

Controle do Token
A máquina que gera o *token* a primeira vez também deve controlá-lo. Essa máquina irá verificar se o *token* está passando por ela dentro de um determinado tempo. Dois problemas podem ser detectados por essa estação:

1. o *token* não passa dentro de um tempo estipulado (*timeout*): um novo *token* deverá ser gerado, pois o mesmo foi perdido por uma estação do anel. Para tanto, deverá haver uma opção de retirada do *token* do anel por uma das máquinas; OU
2. um *token* passa por ela em um tempo menor que o tempo mínimo: neste caso há mais de um *token* circulando na rede e, portanto, o *token* deverá ser retirado da rede. Nesse caso, deverá haver uma opção de geração de *token*, para que seja possível gerar *tokens* por qualquer máquina quando a rede estiver em funcionamento.

Visualização e depuração

A demonstração deverá acontecer, no mínimo, em 3 máquinas.

Deve ser possível:

- Especificar, a qualquer momento, uma mensagem a ser enviada por uma máquina;
- Visualizar onde o *token* e o pacote de dados se encontram durante a execução do programa;
- Avisar quando houver retransmissões;
- Saber o que está acontecendo no anel (o que cada máquina está fazendo);
- Saber se houve *token* perdido ou se há mais de um *token* circulando na rede.

Regras Gerais

Grupos: Até 3 componentes.

Data de entrega e apresentação: 05/06
Obs.: Todos os participantes devem estar presentes

Entrega final:

- Relatório descrevendo a estrutura da solução dada, envolvendo estruturas de dados, threads, classes, mecanismos de sincronização utilizados, CRC, exemplos de execução, etc.
- Código fonte comentado.

IMPORTANTE: Não serão aceitos trabalhos entregues fora do prazo. Trabalhos que não compilam ou que não executam não serão avaliados. Todos os trabalhos serão analisados e comparados. Caso seja identificada cópia de trabalhos, todos os trabalhos envolvidos receberão nota ZERO.