Pontifícia Universidade Católica do Rio Grande do Sul
Laboratório de Redes / Redes de Computadores
Prof. Cristina Nunes
2022/1

Relatório - Trabalho final

Arthur Henz, Augusto Bottega e Pablo Alles

1. Introdução

Neste relatório, estaremos discorrendo sobre o trabalho final das disciplinas de Laboratório de Redes e Redes de Computadores no semestre 2022/1. Nele, iremos explorar a solução gerada para a criação e funcionamento de uma rede em anel, com a inclusão de um Token como uma forma de controle de envio dos pacotes a serem circulados no sistema. A partir de um arquivo de configuração as máquinas do anel são configuradas com um nome de origem, um IP de outra máquina, a qual ela enviará as mensagens, um tempo de sleep para envio de mensagens e se ela iniciará com o Token ou não e ao iniciar sua execução o usuário pode inserir mensagens a serem enviadas máquina destino na rede informando o nome da máquina que foi inserido no arquivo de configuração de cada uma e a mensagem que quer enviar. Ao receber uma mensagem, as máquinas verificam se a mensagem é para elas, caso seja a máquina seja o destino a mensagem é apresentada no terminal de execução do programa, caso seja a origem um aviso de se houve confirmação de entrega da mensagem, ou se houve algum erro na mensagem (nesse caso haverá apenas uma possibilidade de retransmissão) ou se a máquina destino não foi encontrada a mensagem é retirada da fila. Paralelamente ao envio e recebimento de mensagens temos o controle do Token sendo realizado pela máquina que originalmente começou com ele a fim de controlá-lo no sentido de caso receba antes do tempo ideal ou que nunca receba (*timeout*). Por fim, o programa também permite ao usuário digitar mensagens que queira enviar para uma máquina específica ao fornecer o apelido que foi configurado aquela máquina destino da mensagem e o conteúdo da mensagem.

2. Estrutura da solução

Neste parágrafo principal, estaremos explicando as estruturas e elementos criados para que nosso sistema pudesse criar um programa que se comunicasse com outros computadores da mesma rede. A seguir, são mostradas as seções principais do trabalho explicando o funcionamento do programa.

2.1 Estruturas de dados

Usamos um módulo nativo do python que implementa filas com um produtor e um consumidor, ou seja uma fila que é sincronizada entre *threads* diferentes. Sendo a inserção de mensagens aquele que insere mensagens à fila e o que as retira a função de recebimento de mensagens, uma vez que uma mensagem só é retirada da fila ao ser recebida uma confirmação de sua chegada ao destino (ou a confirmação de que a máquina não existe) retornando ACK à origem, caso encontre um erro o aviso será NAK. Dessa forma, toda vez que o socket da origem recebe uma confirmação (ACK), ou que a máquina não foi encontrada (maquinanaoexiste) ou que a mensagem já foi retransmitida mas chegou novamente com erro (NAK e Retransmits>0) deve ser retirada a mensagem da fila e o Token passado adiante.

Nesse sentido, implementamos dois sockets UDP, ou seja com mecanismo de datagrama, um para receber e outro para enviar os pacotes. O que envia extrai o IP e a porta da próxima máquina do arquivo de configuração, enquanto o de recebimento configura o próprio IP e a porta do arquivo (por isso, caso se queira realizar a execução em apenas uma máquina, apenas alterando a porta, deve se alterar essa porta no recebimento para que não seja a mesma do envio). Dessa forma, é possível enviar pacotes entre estações para que a mensagem possa trafegar na rede. Por fim, uma última observação é que o socket de recebimento recebe pacotes com 1024 bytes na sua entrada.

2.2 Threads

Neste trabalho, foi necessária a criação de três *threads*, para que a execução de tarefas específicas e definidas acontecesse paralelamente. Primeiramente, criamos funções que mantivessem as *threads* executando de forma ininterrupta (*while True*). Dessa forma, foi feita uma *thread* que executa os *sockets* de recebimento e envio de mensagens, em que no recebimento é realizado o tratamento da mensagem, caso eu seja a origem é apresentado na tela qual foi o resultado .

Em segundo lugar, temos outra *thread* com a função de receber os parâmetros de apelido da máquina destino e a mensagem escrita pelo usuário, pondo-as na fila. Esse é um jeito possível do usuário poder adicionar mensagens no final da fila ao mesmo tempo que o programa executa outras funções utilizando outras *threads*.

Por fim, há também uma *thread* responsável pela administração do token, caso ocorra um *timeout*. No caso, ele ocorre quando o tempo limite na máquina que gerou o token inicial atinge o valor pré-determinado no arquivo de configuração, fazendo assim com que outro token seja disparado na rede por essa mesma máquina. Além disso, esse *timer* também faz controle do tempo mínimo o qual essa mesma máquina que gerencia o token deve receber o próximo, assim sendo, se algum deles for recebido antes desse tempo, o token recebido é descartado.

2.3 Funções e variáveis globais

A seguir são apresentadas as variáveis globais que compõem o programa bem como para que cada uma serve:

- Token: inicializado como False, porém de acordo com o arquivo de configuração inserido caso esteja lá que a máquina inicia com o Token ele é atualizado para True. Logo, é uma variável que realiza o controle de se a máquina tem o Token ou não.
- Control: caso no arquivo de configuração seja configurado que a máquina em questão é a máquina que começa com o Token a mesma

realiza o controle do Token e assim essa variável diz para a máquina que o controle deve ser feito.

- Retransmits: inicializado como 0, contabiliza o número de retransmissões que foram feitas, nesse caso toda vez que há uma retransmissão essa variável deve ser incrementada e ao receber uma confirmação com ACK, maquinanaoexiste, ou um NAK que já foi transmitido uma vez é zerada novamente a variável.
- tokenTime: variável que utiliza o módulo do Python time e com isso realiza o controle de tempo, ou seja é uma variável que é utilizada no controle do tempo mínimo de recebimento e *timeout* do Token.
- sleepTime: variável que é configurada com base no arquivo de configuração que diz respeito a quanto tempo de sleep é dado para cada mensagem antes de ser enviada.
- qtdMachines: variável que contabiliza quantas máquinas além da sua existem na rede sendo usado para o cálculo de tempo mínimo de recebimento do Token com base nesse número.
- minimumTime: variável que determina o tempo mínimo que a máquina que controla o Token pode receber outro Token, sendo que o cálculo desse tempo é o produto do sleepTime com a qtdMachines. Dessa forma, caso um Token seja recebido antes do tempo ele é descartado com base nessa variável.
- timeout: definido como 20 segundos por ter sido considerado um valor aceitável para nossos testes, assim caso forem inseridas muitas máquinas ou o tempo de sleep for muito grande esse valor deve ser aumentado. Sendo que ele diz respeito ao tempo máximo que a máquina que gerencia o Token pode ficar sem receber nenhum Token, caso esse tempo estoure um novo Token é gerado na rede.
- nakChance: porcentagem a ser definida de chance de a máquina inserir algum erro na mensagem a fim de gerar erros e testar a funcionalidade de NAK.
- tokChance: porcentagem a ser definida de chance de a máquina inserir um Token extra na rede ou de um Token não ser enviado em todo envio de Token gerando erros de Token a fim de testar funcionalidades do seu controle.

A seguir são apresentadas as principais funções que compõem o programa bem como o que cada uma faz:

- readFile (name)

Função necessária para extrair os dados do arquivo de configuração em que o parâmetro “name” refere-se ao nome do arquivo, neste caso, nomeado de “arq.txt”, que contém o IP da máquina destino de conexão e sua porta, o apelido da máquina que executa este método, o tempo de sleep com que a máquina se segura antes de enviar qualquer coisa e se a máquina em que o arquivo de configuração se

encontra iniciará com o token ou não. Esta função só é chamada no começo da execução do programa.

- getMessageConsole ()

Função responsável pelos tratamentos do console recebendo as entradas do usuário relacionadas ao destino e o conteúdo de uma mensagem que é inserida na fila para ser enviada pelo programa ao ser posta no *input* do terminal sendo que primeiro é inserido o destino e após a mensagem em dois *inputs* diferentes. Esta função é chamada em uma *thread* para ser executada paralelamente a outras funções.

- sendMsg ()

Função que realiza o envio de mensagens da fila, caso não tenham mensagens a serem enviadas apenas passa o token adiante. Além disso, ao pegar uma mensagem da fila tem uma probabilidade de inserir um erro antes de realizar o envio sem que a máquina de origem saiba e só seja descoberto ao ser recebido pela máquina destino. Esta função só é executada quando a máquina possui o Token.

- timing ()

Função responsável pelo *timer* que realiza controle do *timeout* e do tempo mínimo que o token deve chegar à máquina em que ele originou. Caso o token não chegue no tempo estipulado deve gerar um novo token e caso o token chegue antes de um tempo mínimo esse token deve ser descartado por conta da verificação à esse *timer*. Esta função é chamada em uma *thread* para ser executada paralelamente a outras funções.

- receiveMsg ()

Função responsável por receber as mensagens e assim tratar o cabeçalho dos pacotes no sentido de que se a mensagem é para uma determinada máquina e ela chegou nela deve imprimir na tela e enviar a confirmação (ACK) ou pedido de retransmissão (NAK), ou se for o caso de ser para a origem apresentar que a mensagem enviada anteriormente foi confirmada ou que deve ser retransmitida. Além disso, como é aqui que é feita a análise do cabeçalho do pacote também é feita a verificação de CRC. Caso não seja o caso da máquina ser origem nem destino da mensagem, ela apenas passa a mensagem adiante no anel. Quando se trata da origem, a função por meio da máquina pode enviar um Token após receber uma confirmação, visto que terminou o ciclo da mensagem que estava em rede. Aqui também é conferido o tempo mínimo do Token, visto que é por meio do recebimento que temos a informação do recebimento dele. Esta função é chamada em uma *thread* para ser executada paralelamente a outras funções.

2.4 Mecanismos de sincronização utilizados

Para realizar a sincronização entre *threads* utilizamos a *Queue* do Python que é um módulo de fila que é multi-produtor e multi-consumidor, ou seja uma *thread* pode consumir seus elementos enquanto outra insere. Dessa forma, impedimos com que houvesse problemas de sincronização da fila entre as *threads* que captura a entrada do usuário para inserir na fila e a que envia e recebe mensagens. Além disso, implementamos um timer que funciona em uma *thread* separada em sincronia com as demais *threads*, pois esta implementa a funcionalidade de realizar o controle de tempo mínimo de recebimento e de *timeout* do Token.

2.5 CRC

O CRC funciona como um mecanismo de detecção de erro, onde conseguimos usar o módulo 'binascii' para aplicar o algoritmo 'crc32' na mensagem recebida e assim executar a verificação de erros. Para o programa inserir um erro durante a execução ele deverá passar pela porcentagem dele ocorrer que é definida em uma variável global "nakChance" dessa forma antes de a mensagem ser enviada a máquina que enviou irá modificar o NAK que já havia sido calculado anteriormente quando a mensagem foi inserida na fila. Dessa forma, ao chegar no destino a máquina irá verificar que o CRC da mensagem não é o mesmo que está no cabeçalho do pacote e assim retornar NAK para a origem dessa forma possibilitando visualizar as retransmissões e o controle de caso a mensagem venha com erros.

2.6 Exemplos de execução

- Exemplo de execução enviando mensagem e recebendo ACK:

Máquina 1

```
1: 192.168.15.16:6000
2: Bob
3: 20
4: true
Tempo minimo para receber o token: 5
Nenhuma mensagem na fila, vou enviar o Token adiante
Digite o apelido da maquina destino:
RobVou receber um pacote

Digite a mensagem:
1234
Digite o apelido da maquina destino:
Recebi o Token!
Vou enviar uma mensagem
Vou receber um pacote
Recebi uma mensagem!
ACK: (2222;ACK:Bob:Rob:2615402659:1234)
Vou receber um pacote
Recebi o Token!
```

Máquina 2

```
1: 192.168.15.162:6000
2: Rob
3: 20
4: false
Tempo minimo para receber o token: 5
Digite o apelido da maquina destino:
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
Recebi uma mensagem!
1234 Cliente:('192.168.15.162', 56580)
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
```

- Exemplo de mensagem para TODOS e de timeout do token por conta da máquina não ter enviado ele:

Máquina 1

```
1: 192.168.15.16:6000
2: Bob
3: 20
4: true
Tempo minimo para receber o token: 5
Nenhuma mensagem na fila, vou enviar o Token adiante
Digite o apelido da maquina destino:
Vou receber um pacoteTODOS

Digite a mensagem:
1234
Digite o apelido da maquina destino:
Recebi o Token!
Vou enviar uma mensagem
Vou receber um pacote
Recebi uma mensagem!
Removi uma mensagem que foi enviada para TODOS da fila.
Nenhuma mensagem na fila, vou enviar o Token adiante
Nao enviei o token!!!Timeout do token, reenviando novo token

Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
```

Máquina 2

```
1: 192.168.15.162:6000
2: Rob
3: 20
4: false
Tempo minimo para receber o token: 5
Digite o apelido da maquina destino:
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
Recebi uma mensagem!
1234 Cliente:('192.168.15.162', 64572)
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
```

- Token recebido antes do tempo por duplicação de token:

Máquina 1

```
1: 192.168.15.16:6000
2: Bob
3: 20
4: true
Tempo minimo para receber o token: 5
Nenhuma mensagem na fila, vou enviar o Token adiante
Digite o apelido da maquina destino:
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
Recebi o Token!
Removendo o token por ter sido recebido antes do tempo mínimo.
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
Recebi o Token!
Removendo o token por ter sido recebido antes do tempo mínimo.
Vou receber um pacote
Recebi o Token!
```

Máquina 2

```
1: 192.168.15.162:6000
2: Rob
3: 20
4: false
Tempo minimo para receber o token: 5
Digite o apelido da maquina destino:
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Novo token gerado aleatoriamente!!!
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Novo token gerado aleatoriamente!!!
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
```

- Exemplo com ACK, maquinanaoexiste, NAK, remoção e *timeout* do Token por ter sido recebido antes do tempo mínimo ()

Máquina 1

```
1: 192.168.15.16:6000
2: Bob
3: 5
4: true
Tempo minimo para receber o token: 5
Nenhuma mensagem na fila, vou enviar o Token adiante
Nao enviei o token!!!
Digite o apelido da maquina destino:
Vou receber um pacote
Rob
Digite a mensagem:
```

```
123
Digite o apelido da maquina destino:
Rob
Digite a mensagem:
1234
Digite o apelido da maquina destino:
rob
Digite a mensagem:
1235
Digite o apelido da maquina destino:
Timeout do token, reenviando novo token
Recebi o Token!
Vou enviar uma mensagem
Vou receber um pacote
Recebi uma mensagem!
ACK: (2222;ACK:Bob:Rob:2286445522:123)
Vou receber um pacote
Recebi o Token!
Removendo o token por ter sido recebido antes do tempo mínimo.
Vou receber um pacote
Recebi o Token!
Vou enviar uma mensagem
Vou receber um pacote
Recebi uma mensagem!
NAK: (2222;NAK:Bob:Rob:2615402660:1234)
Vou receber um pacote
Recebi o Token!
Vou enviar uma mensagem
Será uma retransmissao!!!
Vou receber um pacote
Recebi uma mensagem!
NAK and already Retransmitted: (2222;NAK:Bob:Rob:2615402660:1234)
Vou enviar uma mensagem
Vou receber um pacote
Recebi uma mensagem!
maquinanaoexiste: (2222;maquinanaoexiste:Bob:rob:3974418485:1235)
Vou receber um pacote
```

Máquina 2

```
1: 192.168.15.162:6000
2: Rob
3: 5
```

```
4: false
Tempo minimo para receber o token: 5
Digite o apelido da maquina destino:
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
Recebi uma mensagem!
123 Cliente:('192.168.15.162', 60202)
Novo token gerado aleatoriamente!!!
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
Recebi uma mensagem!
Encontrei um erro na mensagem! Coloquei NAK
Vou receber um pacote
Recebi o Token!
Nenhuma mensagem na fila, vou enviar o Token adiante
Vou receber um pacote
Recebi uma mensagem!
Encontrei um erro na mensagem! Coloquei NAK
Vou receber um pacote
Recebi uma mensagem!
Pacote nao e para mim enviando para a proxima maquina
Vou receber um pacote
Recebi o Token!
```