Protocolo de Enlace: Entrega parcial 2

Marcelo Bittencourt e Osvaldo Neto

Relatório apresentado para a disciplina de Projeto de Protocolos do curso Engenharia de Telecomunicações, Instituto Federal de Santa Catarina - Campus São José.

1. Como utilizar

A aplicação de teste disponível é capaz de realizar a transmissão de dados capturados do sistema operacional a partir de uma interface de rede gerada. Para realizar o teste do protocolo desenvolvido será necessário configurar o ambiente de acordo com o sistema operacional em utilização. Considerando um ambiente com Linux, primeiramente gere duas portas seriais virtuais através do programa Serialemu e em uma máquina virtual aplique uma das portas geradas em **Configurações de Portas Seriais**. Feito isso, siga os passos abaixo para executar o protocolo:

1. Verificando a porta serial padrão do sistema operacional da sua máquina virtual. No caso do sistema Linux o nome dessa porta é ttyS0. **Anote esse nome**.

```
dmesg | grep tty
```

2. Comando para ver as opções de parâmetros do protocolo.

```
sudo python3 main.py -h
```

3. Executando o protocolo na máquina virtual, sendo -s a porta serial do sistema, -ipd o número IP de destino e -ipo o número IP de origem.

```
sudo pytyhon3 main.py -s=/dev/ttyS0 -ipd=10.0.0.2 -ipo=10.0.0.1
```

4. Executando o protocolo na máquina real, perceba a alternância dos números IP. Nesse caso, as portas seriais virtuais geradas pelo programa Serialemu foram /dev/ttys03 e /dev/ttys04. A porta /dev/ttys04 foi configurada na máquina virtual e a /dev/ttys03 está sendo utilizada no sistema real.

```
sudo pytyhon3 main.py -s=/dev/ttys03 -ipd=10.0.0.1 -ipo=10.0.0.2
```

5. Realize, por exemplo, uma operação de *ping* entre as máquinas e visualize o recebimento e o envio dos pacotes sendo realizados pelo protocolo.

Observação: Dentro do diretório com todos os arquivos necessários para a execução correta do programa é necessário instalar as dependências do projeto, para isso crie um ambiente virtual, e nele faça o comando abaixo para instalar as dependências do mesmo:

```
sudo pip3 install -r requirements.txt
```

2. Inserindo novas funcionalidades

Caso o usuário queira implementar uma nova funcionalidade na estrutura do protocolo em questão, o mesmo deve seguir as seguintes etapas.

1. Criar uma classe para a nova funcionalidade (subcamada). Seguindo o padrão do projeto, essa classe pode se chamar de **CallbackFuncionalidade.py**;
2. Essa nova classe deve especializar a classe existente **Subcamada.py,** a qual é responsável por referenciar a subcamada superior e inferior através do método *conecta()*, assim como define os métodos abstratos de *envia()* e *recebe()* que são utilizados para a comunicação intercamadas (Obs: A Subcamada.py especializa a classe Callback que possui métodos que monitoram o recebimento, envio e tempo de resposta da funcionalidade);
3. Na classe referente a nova funcionalidade deve-se implementar os seus métodos específicos e os métodos abstratos (*envia() e recebe()*) herdados da classe Subcamada;

```python
import sys
from Subcamada import Subcamada

class CallbackFuncionalidade(Subcamada):
    '''Classe responsável pela nova funcionalidade'''

    def __init__(self):
        Subcamada.__init__(self, sys.stdin, 0)

    def handle(self):
        '''Método de Callback'''

    def recebe(self, dados):
        '''Recebe da camada superior'''

    def envia(self, msg):
        '''Envia para a camada superior'''
```

4. Na arquivo **main.py** ocorrerá o instanciamento dessa nova funcionalidade (subcamada), a qual será passada como parâmetro para a sua camada inferior. Abaixo está um exemplo de como inserir uma nova aplicação, sendo a subcamada Aplicação a mais "alta" e Enquadramento mais "baixa".

```python
# Callback Aplicação
app = CallbackAplicacao()

# Callback Nova Funcionalidade
func = CallbackFuncionalidade()

# Callback Enquadramento
enquadra = CallbackEnquadramento(cnx, Timeout)
```

5. Realiza a conexão entre as subcamadas e adiciona elas ao objeto *Poller* para o seu monitoramento.

```python
# Conecta os Callbacks
func.conecta(app)
enquadra.conecta(func)

# Adicionando ao Poller
sched = poller.Poller()
sched.adiciona(enquadra)
sched.adiciona(app)
sched.adiciona(func)
sched.despache()
```