Projeto 1 - Protocolo de Enlace

Daniel Cabral Correa
Luísa Machado
Professor: Marcelo Maia Sobral.
Disciplina: Projeto de protocolos.
Engenharia de Telecomunicações

05 de outubro de 2018

1 Introdução

Este documento apresenta o primeiro projeto da disciplina de Projetos de Protocolos. O projeto consiste em realizar um protocolo de enlace ponto-a-ponto com transmissão confiável por um meio sem fio, além disso o protocolo deve garantir a entrega e ter controle e tratamento de erro. Para implementação deste projeto foi utilizada a linguagem *Python*[1].

2 Especificação

Para esta atividade o professor Marcelo Sobral solicitou o desenvolvimento de um protocolo de enlace com comunicação ponto-a-ponto em uma rede sem-fio composta por um *transceiver RF* que será utilizado como interface serial do tipo *UART*. Com isso o *transceiver RF* poderá estabelecer enlaces de média distância (até 1 km) com baixas taxas de transmissão.

O Protocolo deverá garantir:

- Encapsulamento de mensagens entre 8 e 256 bytes.
- Transmissão confiável.
- Recepção de mensagens livres de erros.
- Garantia de entrega
- Estabelecimento de sessão

Contudo o protocolo deve ser interoperável, ou seja, ele deverá se comunicar com os protocolos desenvolvidos pelo demais grupos.

3 Análise do protocolo

- Serviço: protocolo de enlace ponto-a-ponto para camada física do tipo *UART* com encapsulamento de mensagens entre 8 e 256 bytes garantindo recepção de mensagens livres de erros além de garantia a entrega.

[1] Python é uma linguagem de programação

- Ambiente de execução: rede sem-fio composta por um *transceiver RF* capaz de transmitir em média distância (até 1 km) e baixas taxas de transmissão. Esse *transceiver* será usado como uma interface serial do tipo *UART*.
- Vocabulário: *Frames* de informação e confirmação *(ACK)*.
- Codificação: Bytes.

4 Descrição

4.1 Enquadramento

O enquadramento é responsável por delimitar os quadros na interface com a camada física. Sabendo que a camada física faz o envio e a recepção de sequências de bytes sem qualquer estrutura, a camada de enlace deve delimitar as unidades de dados do protocolo *(PDU)* dentro dessas sequências de bytes.

Existem várias técnicas de enquadramento, porém esse protocolo utilizará a técnica do tipo sentinela. A técnica insere no início e no fim do arquivo uma *flag* 7E (01111110)), além disso também usa a *flag* 7D (01111101) para preenchimento de octeto. Caso apareça no conteúdo do quadro alguma das duas *flags* o mesmo deve ser alterado por meio de um *XOR* com o byte 20.

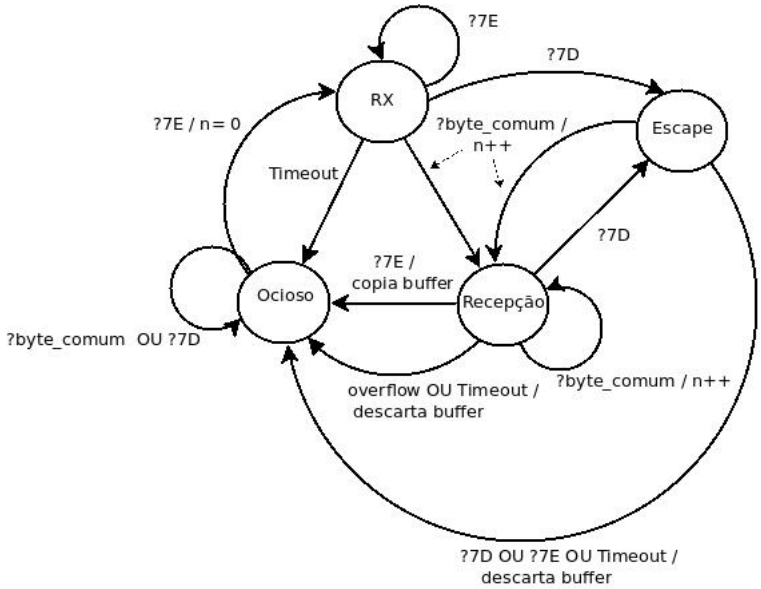

Figura 1 - Máquina de estado finita da classe Enquadramento.

A imagem acima mostra a máquina de estados utilizada para implementar a classe enquadramento.

4.2 Controle de erros

Uma das funções implementadas no protocolo é o recebimento das mensagens sem erros, o mesmo é feito implementando o método de detecção de erros (*CRC*[2])

O *CRC* tem como função adição de *bits* redundantes no *payload*. Os *bits* são determinados com base no conteúdo da mensagem usando uma forma de álgebra polinomial. Para checar a integridade das mensagens, é feito, na recepção o processo novamente, obtendo dessa forma somente os *bits* redundantes e comparando-os com os calculados na hora do envio.

Neste protocolo, foram determinados dois *bytes* para o *CRC*, sendo assim oito *bits* redundantes no *payload*. A implementação foi feita na classe *Enquadramento*, que segue os valores do *CRC* da versão do algoritmo da *RFC 1662*[3].

4.3 Garantia de entrega (ARQ)

É um serviço que certifica ao transmissor se a mensagem foi entregue ou não ao seu destino. Neste protocolo foi utilizado o mecanismos ARQ *(Automatic Repeat reQuest)* do tipo pare-e-espere que funciona da seguinte maneira, enquanto o transmissor não receber a confirmação, o *(ACK)*, de entrega, a mensagem será mantida na fila de saída pelo protocolo, caso fique algum tempo na fila essa mensagem será reenviada.

Abaixo a máquina de estados utilizada para o desenvolvimento da classe responsável pela garantia de entrega.

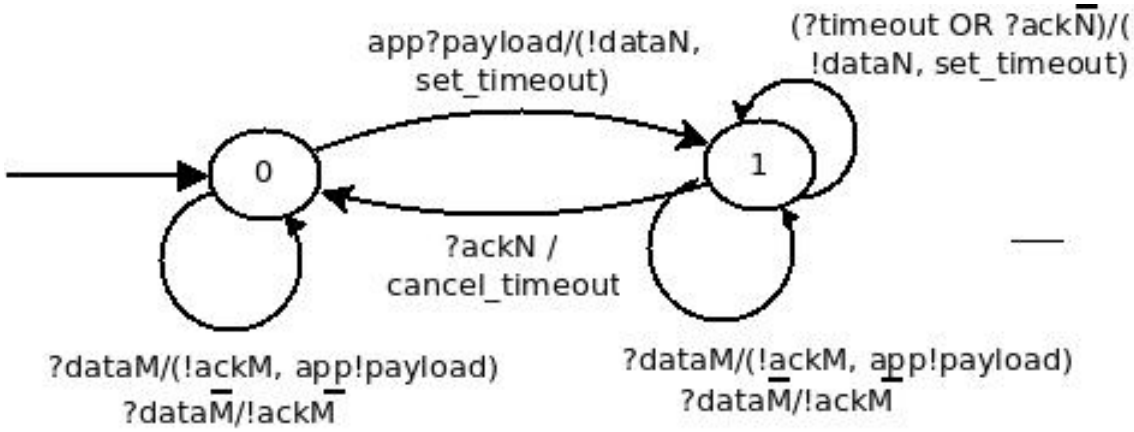

Figura 2 - Máquina de estado finita da classe *ARQ*.

A figura abaixo mostra o cabeçalho sendo adicionado ao quadro. O cabeçalho em questão possui dois campos, o primeiro é o controle, responsável por identificar se a mensagem contém informações ou se é uma mensagem de controle, um *(ACK)*, e também tem a função de identificar o número de sequência do quadro (sendo 0 ou 1). O segundo campo está identificado na imagem como Proto mas atualmente não tem uma utilidade específica.

Ao fim do quadro existe mais um campo que ocupa dois bytes, esse campo contém o valor do código do CRC, sendo o primeiro byte o *LSB (Least Significant Byte)* do código CRC, e o último o *MSB (Most Significant Byte)*, responsável pelo controle de erro do quadro.

[2]CRC - Cyclic Redundancy Check.

[3]RFC 1662 - https://tools.ietf.org/html/rfc1662

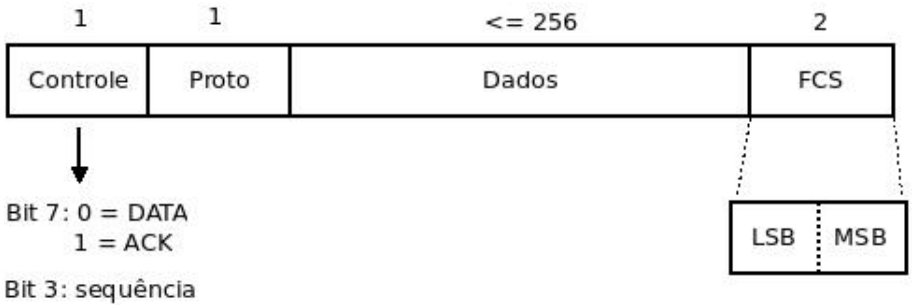

Figura 3 - Quadro da classe *ARQ*.

4.4 Timeout

Este protocolo possui limites de tempo para efetuar algumas tarefas, esse prazo é comumente chamado de *timeout*. O estouro do *timeout* implica em um evento para as máquinas de estados, normalmente esses eventos encerram ou repetem algumas operações.

No protocolo desenvolvido deveriam exister alguns *timeout*, como por exemplo no *ARQ* quando o transmissor aguarda a confirmação de entrega do quadro.

5 Comportamento

O Transmissor funciona da seguinte maneira, começa com a aplicação informando uma mensagem, em seguida a mensagem é encaminhada para o *ARQ* onde recebe o cabeçalho com um byte de controle, contendo o número de sequência e a identificação do tipo do quadro, e o Proto. Após isso, o quadro é enviado ao enquadramento onde é calculado o CRC para verificação de erro formado por dois bytes o *LSB* e o *MSB*, esses bytes são adicionados no fim do quadro. Por fim, o Enquadramento analisa o *payload* checando se existe algum 7E ou 7D, caso exista é necessário fazer uma modificação destes bytes, e insere as *flags* 7E no início e fim do *payload*. Após isso a mensagem é enviada ao seu destino e o protocolo aguarda a confirmação da entrega.

O Receptor inicialmente fica em modo de espera até receber um *payload*, quando chega ele é enviado ao enquadramento, em seguida as *flag* 7E são retiradas do início e fim da mensagem, após isso é feita a verificação de integridade do quadro com o CRC e são retirados os dois bytes *LSB* e *MSB*, após isso a mensagem é enviado ao *ARQ* aonde será verificado se a mensagem trata-se de informação ou controle. Ao final a mensagem é enviada para a aplicação e um *ACK* é transmitido para sinalizar que o arquivo foi recebido com sucesso.

6 Manual do Usuário

Para iniciar a execução deste protocolo é necessário ter pré-instalado a versão 3 ou superior do *Python*, além do pacote *Pyserial*. Também é preciso instalar o módulo *CRC16* disponibilizado pelo professor no site da disciplina.

Atendidos os requisitos acima, pode-se iniciar a execução, para isto, deve-se ter os arquivos a seguir em um mesmo diretório:

- LD.py
- ARQ.py
- Enquadramento.py

- tx.py
- rx.py
- crc.py

Para executar este projeto, primeiro executa-se o receptor em seguida o transmissor, da seguinte forma:

- **Receptor:** python3 rx.py porta proto
- **Transmissor:** python3 tx.py porta proto

Onde:

- **porta:** é nome da porta da serial, exemplo: */dev/pts/6*
- **Proto:** é o número do Proto, exemplo: *5*

Ao executar o Transmissor será solicitado ao usuário que digite a informação que deseja-se enviar ao receptor.

7 Conclusão

O protocolo obteve sucesso na transmissão e recepção de dados, atendendo a maior parte das especificações, exceto pelo gerenciamento de sessão e o *timeout* que não foram implementados em tempo hábil. O principal desafio dessa atividade, foi o entendimento e a codificação das máquinas de estados, que refletiram com o atraso na finalização do projeto.

8 Referências

1. Sobral, Marcelo. PTC29008, 2018.
   Disponível em:
   <https://wiki.sj.ifsc.edu.br/wiki/index.php?title=PTC-EngTel_(página)>
   Acesso em: 29/09/2018