Redes I – TP - Implementação de pilha de protocolos

DNS

Jonathan Henrique, Rúbio Torres e Igor Miranda

Camada Física

A implementação da camada física foi feita em Python, usando a biblioteca de socket. De acordo com o RFC, ela opera com PDUs no seguinte formato:



O código recebe um payload de dados da camada superior, e coloca o cabeçalho com todos os dados necessários, os converte em binário, e envia para o destinatário, ou faz o processo inverso ao receber mensagens de fora.

Implementação

Código do Servidor

Ao executar o servidor, ele fica constantemente “escutando” por conexões. Quando uma máquina solicita uma conexão, ele aceita e espera o recebimento de dados. Ao receber os dados, a máquina verifica a origem para saber se os dados são provenientes de uma camada superior ou do ambiente externo, e toma a ação apropriada (monta um quadro para enviar caso seja da camada superior, ou desmonta e decodifica o quadro para mensagens vindas de fora).

Montagem e decodificação de Quadros

Para montar o quadro, o servidor começa, inicialmente, pelo *preamble*, uma sequencia de 7 bytes de 1s e 0s alternados, e mais um byte terminando com 11 (10101011) para demarcar o fim desse trecho. Depois, temos o endereço MAC de origem e de destino, codificados em binário. O código recebe o MAC de origem como uma configuração interna, e calcula o MAC de destino a partir do IP. Para fazer esse cálculo, o código confere se o MAC existe na tabela local, e, caso não exista, faz o ARP (via linha de comando), encontra o ARP e registra na tabela local. Depois, ele coloca 2 bytes para representar o tamanho total da mensagem (EtherType). Esse valor pode ter outro significado acima de 1500, mas, como as mensagens de DNS são curtas, esse valor é sempre o tamanho da mensagem. Depois, se segue o binário do payload, convertido de string, e, por final, o FCS (Frame Check Sequence), que é um valor usado para verificação do pacote.

Quando o servidor recebe uma mensagem de fora, ele faz o processo reverso, ou seja, ele separa os bits da mensagem de acordo com o tamanho de cada campo, e faz a conversão desses bits para valores legíveis. O servidor também faz a verificação do FCS, mas, como essa parte não fazia parte da especificação do trabalho, não poderíamos esperar que todas as mensagens dos outros grupos tivessem o FCS implementado, e, portanto, ele aceita as mensagens mesmo quando a verificação falha (mas ainda avisa quando falha).

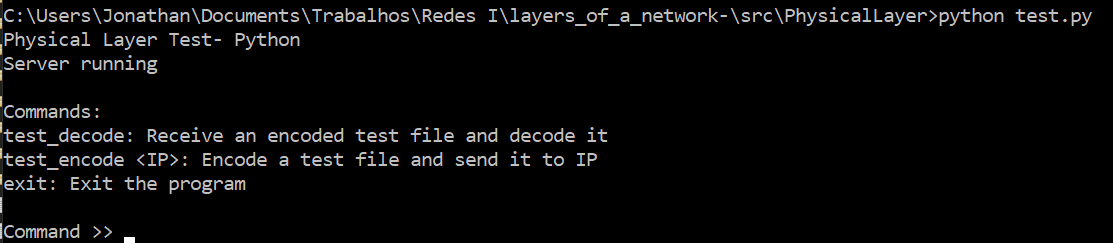
Envio de Dados

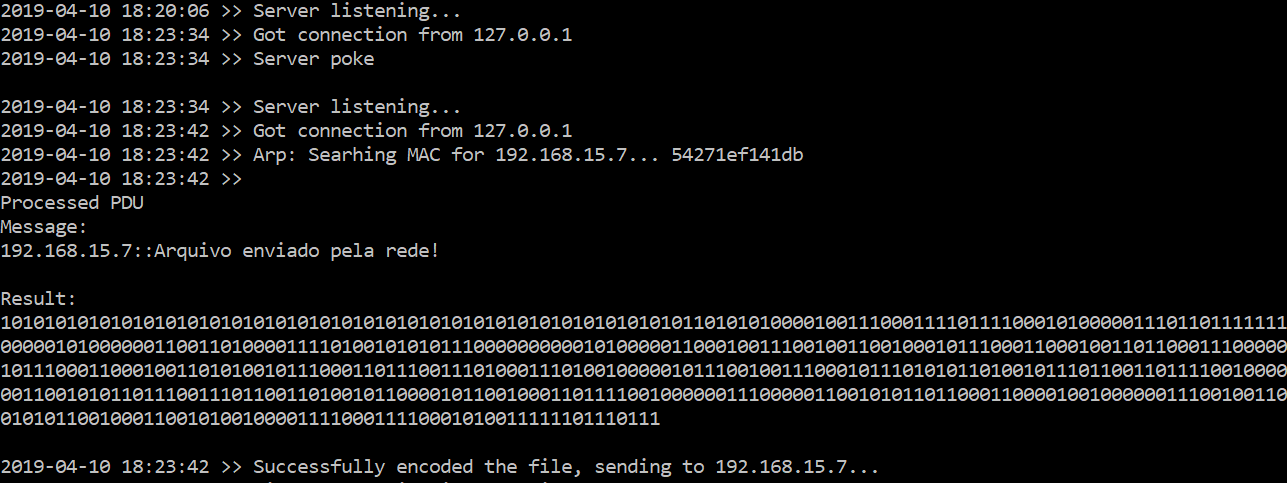
Ao finalizar a montagem de quadro, o servidor simplesmente envia para o destinatário via socket, usando o IP do destinatário, pois, para essa implementação, não seria viável colocar a mensagem diretamente no meio físico. Porém, antes de enviar, o código simula a probabilidade de colisão (5%), e, em caso de colisão, ele espera um tempo aleatório (entre 0.01 e 1 seg) e tenta enviar novamente.

FCS

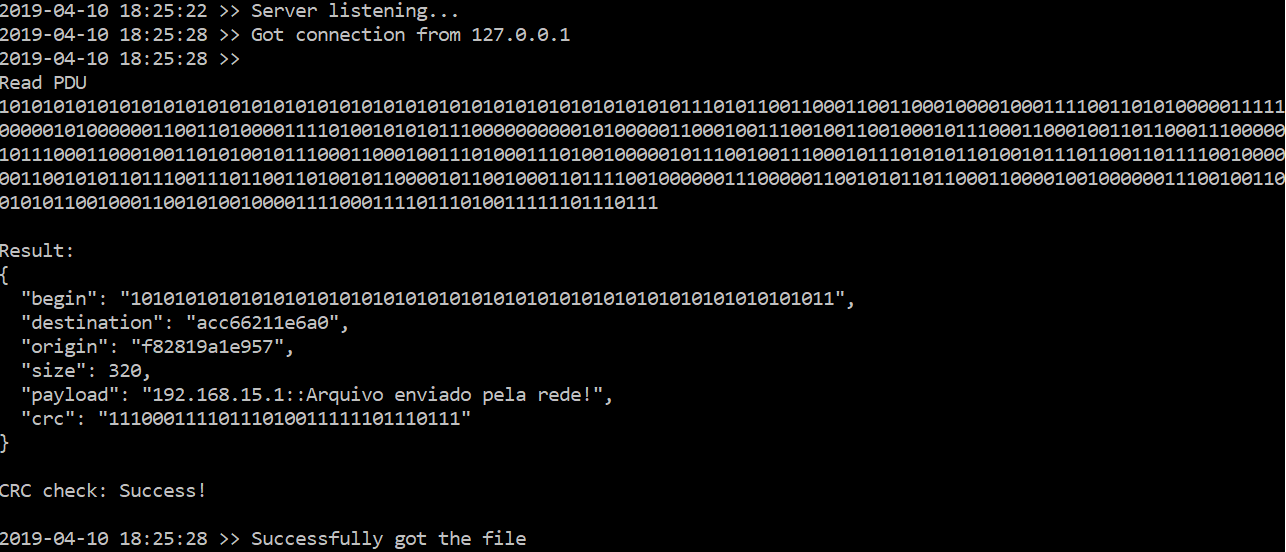
Como foi mencionado anteriormente, o código faz a verificação de pacotes. Para isso, o código gera um valor baseado no dado do payload, e verifica esses dados quando recebe pacotes externos. Como o código gerador de FCS é um algoritmo usual, foi usado um código da internet para a geração e checagem do valor. O código gera um valor de 32 bits, usando um polinômio gerador “101010101010101010101010101010101” (o polinômio gerador deve ter 1 bit a mais que o resultado desejado)

Testes

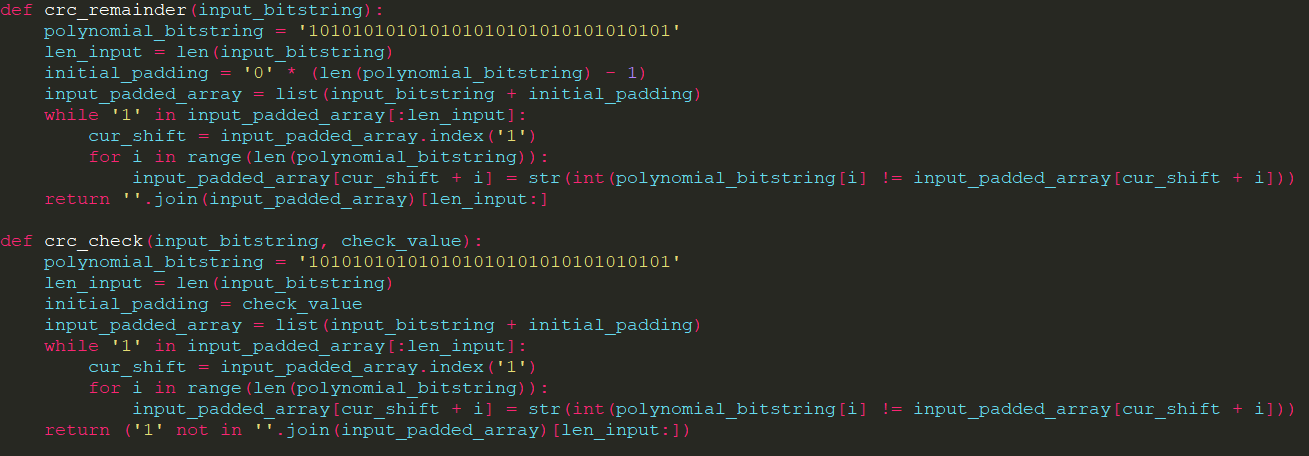
Para testar o funcionamento do servidor sem as camadas superiores implementadas, foi desenvolvida uma pequena aplicação Python para fazer comunicação com esse servidor e testar as funcionalidades. 

Essa aplicação envia mensagens para o servidor, sendo essa uma mensagem do mundo externo ou de camada superior, além de, na inicialização, enviar uma mensagem de conferência do status do servidor (Uma mensagem cujo conteúdo é somente “Hello”), que o servidor foi programado para responder.

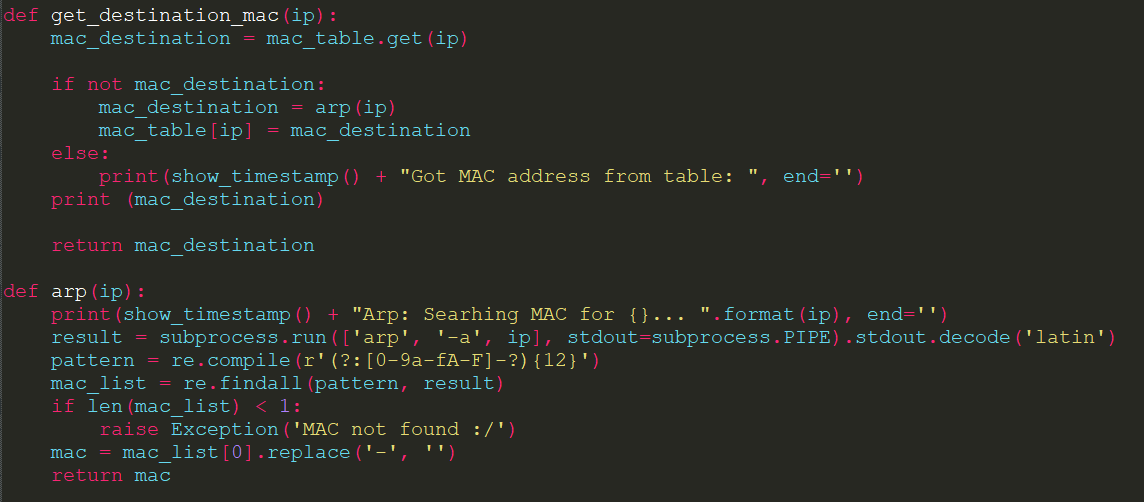
Resposta do servidor ao receber o ping (apelidado “*poke*”) e codificação de mensagem para binário



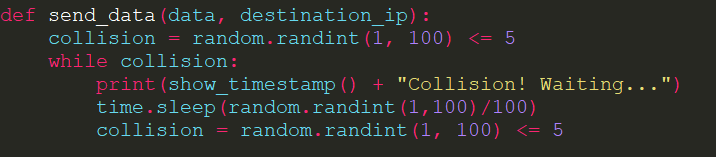
Resposta do servidor ao receber uma mensagem externa, fazendo a decodificação e separando os campos



Código do CRC, disponível em <https://en.wikipedia.org/wiki/Cyclic_redundancy_check> (com modificações)



Código do ARP



Código de colisão



Montagem e “desmontagem” de quadros

Execução

Para executar o programa, simplesmente entre em um terminal, vá para a pasta do código e execute “python server.py”. Para executar a aplicação de testes, execute “python test.py” em um outro terminal. Para testar a codificação de mensagens, digite na aplicação de testes “test\_encode IP”, sendo IP o endereço da máquina de destino. Para testar a decodificação, basta enviar ao servidor um arquivo codificado, e ele exibirá todo o processo, conforme mostrado nos *prints* acima.

Considerações Finais

A camada física foi implementada com um funcionamento que pode ter pequenas alterações para se adequar à implementação das camadas superiores, ou para se adequar às implementações dos outros grupos, visando a comunicação mútua. Tudo foi implementado de acordo com a especificação, exceto pela adição da conferência do FCS, e pelo fato de que o log do processamento e geração de PDUs é exibido por padrão, sem necessidade de um comando para exibí-los.