Etapa 4 - Verificar o endereço MAC e o Protocolo encapsulados no frame

Grupo:
Guilherme Nishi Kanashiro - 628298
Leonardo Utida Alcântara - 628182
Rodolfo Krambeck Asbahr - 628042
Tiago Bachiega de Almeida - 628247

Como executar o programa:

Primeiramente, é necessário ajustar o mtu dele, pois o tamanho padrão é de 65536 bytes, o que impossibilita qualquer forma de fragmentação automática e queremos testar o que foi feito na Etapa 3 integrada com o código da Etapa 4. Para fazer esse ajuste, é necessário usar o seguinte comando:

```
$ ip link set lo mtu 1500
```

Feito isso, é necessário ajustar dentro do código os seguintes parâmetros de acordo com sua rede e dispositivo. Para o grupo, foi utilizado o seguinte:

```
# Coloque aqui o endereço de destino para onde você quer mandar o ping
# Pode ser obtido com o comando arp -a | grep _gateway
dest_ip = '192.168.1.1'

# Coloque abaixo o endereço IP do seu computador na sua rede local
# Pode ser obtido com o comando ifconfig
src_ip = '192.168.1.116'

# Coloque aqui o nome da sua placa de rede
# Pode ser obtido com o comando ifconfig
if_name = 'wlp2s0'

# Coloque aqui o endereço MAC do roteador da sua rede local (arp -a | grep _gateway)
# Pode ser obtido com o comando arp -a | grep _gateway
dest_mac = '7c:8b:ca:3d:01:8e'

# Coloque aqui o endereço MAC da sua placa de rede (ip link show dev wlan0)
# Pode ser obtido com  ip link show dev <nomePlacaDeRede>
src_mac = '28:e3:47:b0:6f:fa'
```

A implementação da interpretação e reconstrução dos datagramas IPs foram feitas utilizando python 3. Para rodar o código em um terminal é preciso executar o seguinte comando:

```
$ sudo python3 etapa4.py
```

Como funciona o programa:

Verificação do MAC Address

Para verificar se o MAC Address de destino do quadro é condizente com o MAC Address da placa de rede, foi criada uma função que obtém os 6 primeiros bytes do quadro, que são o MAC Address de destino e que converte para bytes o valor inserido na variável *src_mac,* que é o da placa de rede. Com isso, basta comparar se os valores são equivalentes e utilizar o resultado na função *raw_rcv()*, como será mostrado a seguir.

```
def verify_MAC(frame):
    print("****Verificando MAC addr****")

    dest_mac_bytes = frame[0:6]
    src_mac_bytes  = mac_addr_to_bytes(src_mac)

    print("MAC de destino : ", ':'.join('%02x'%x for x in dest_mac_bytes))
    print("Meu MAC : ", ':'.join('%02x'%x for x in src_mac_bytes))

    if dest_mac_bytes == src_mac_bytes:
        return True
    else:
        return False
```

Verificação do Protocolo:

Além disso, precisa-se verificar se o protocolo do quadro recebido é IP, ou seja ETH_P_IP, que possui o valor 0x0800. Para isso, criou-se outra função que obtém os bytes 12 e 13 do quadro, que especificam o protocolo, e comparou-se estes bytes com o atribuído a ETH_P_IP. A função que faz essa verificação também é usada no *raw_rcv()*.

```
def verify_protocol(frame):
    print("****Verficando o protocolo****")
    protocol_bytes = frame[12:14]

    print("Protocolo do Frame : ", protocol_bytes)
    print("Protocolo IP:  ", struct.pack('!H', ETH_P_IP))

    if protocol_bytes == struct.pack('!H', ETH_P_IP):
        return True
    else:
        return False
```

Verificação no *raw_rc* e integração com a Etapa 3:

Na função *raw_rcv()* o programa verifica as duas condições definidas por *verify_mac()* e *verify_protocol()*. Caso ambas sejam verdadeiras, remove o cabeçalho da camada de enlace do quadro, obtendo então o datagrama e o passa para a função *defrag()*, que realiza o processamento da Etapa 3.

```
def raw_recv(fd):
    frame = fd.recv(12000)

    print("\n ========================== \n")

    #Verifica MAC e Protocol
    if verify_MAC(frame) and verify_protocol(frame):
        print('recebido quadro de %d bytes' % len(frame))
        print(repr(frame))
        #Obtem o Datagram
        datagram = frame[14:]
        defragMSG = defrag(datagram)
    else:
        print("Nao passou na verificacao")
```

Teste:

A imagem a seguir exibe dois casos: um em que foi recebido um frame que passou na verificação do MAC Address e do Protocolo e outro que não passou na verificação do MAC Address. Para o primeiro caso, é possível verificar que os MAC Addresses são idênticos, assim como o Protocolo corresponde ao IP. Também é possível verificar que o quadro possui 58 bytes e também a sua mensagem. Logo em seguida, o datagrama interno a este quadro entra no processamento da Etapa 3, onde a mensagem é recebida e desfragmentada.

Para o segundo caso, é visível que os MAC Addresses não são correspondentes. Assim, o quadro não passou na verificação e não será processado.

```
 ==========================

****Verificando MAC addr****
MAC de destino :  28:e3:47:b0:6f:fa
Meu MAC :  28:e3:47:b0:6f:fa
****Verficando o protocolo****
Protocolo do Frame :  b'\x08\x00'
Protocolo IP:   b'\x08\x00'
recebido quadro de 58 bytes
b'(\xe3G\xb0o\xfa|\x8b\xca=\x01\x8e\x08\x00E\x00\x00,\xdc\xec\x00\x00@\x01\x1a\x
1f\xc0\xa8\x01\x01\xc0\xa8\x01t\x00\x00\t\xba\xba\xdc\x0f\xfe\xba\xdc\x0f\xfe\xb
a\xdc\x0f\xfe\xba\xdc\x0f\xfe\xba\xdc\x0f\xfe'
***Entrando no Processamento do Frame - Etapa 3***
Recebidos todos os 1 pacotes da mensagem 56556 vinda de 192 . 168 . 1 . 1
Mensagem desfragmentada!

 ==========================

****Verificando MAC addr****
MAC de destino :  01:00:5e:7f:ff:fa
Meu MAC :  28:e3:47:b0:6f:fa
Nao passou na verificacao
```