Scenario

Consider a network with a Maximum Transmission Unit (MTU) of 80 bytes. A sender needs to transmit data of size 300 bytes using the UDP protocol. The UDP header is 8 bytes, and the IP header is 20 bytes.

- Each IP packet includes the IP header (20 bytes).
- Each UDP packet includes the UDP header (8 bytes).
- The payload size of each fragment must be adjusted according to the given MTU.

Questions

1. Given the network conditions, how many fragments are required to transmit the 300-byte data?

จาก MTU 80 bytes ต้องการส่ง Data 300 bytes ต้องแบ่งส่งทั้งหมด 4 fragments

Fragment 1:	8 (UDP) + 72 (Data)
Fragment 2 - 3:	80 (Data) each
Fragment 4:	68 (Data)

2. Implement manual fragmentation using Scapy and Netcat based on the given scenario. Capture the transmitted packets using Wireshark, analyze them, and explain how fragmentation was performed.

Code:

```python
#!/usr/bin/python3
from scapy.all import *

ID     = 1000
dst_ip = "10.9.0.5"

# Fragment No.1 (Fragment offset: 0)
udp = UDP(sport=7070, dport=9090, chksum=0)
udp.len = 8 + 72 + 80 + 80 + 68

ip = IP(dst=dst_ip, id=ID, frag=0, flags=1)
payload = "A" * 71 + "\n"
pkt1 = ip/udp/payload

# Fragment No.2 (Fragment offset: (8 + 72)/8 = 10)
ip = IP(dst=dst_ip, id=ID, frag=10, flags=1)
ip.proto = 17
payload  = "B" * 79 + "\n"
pkt2 = ip/payload

# Fragment No.3 (Fragment offset: (8 + 72 + 80)/8 = 20)
ip = IP(dst=dst_ip, id=ID, frag=20, flags=1)
ip.proto = 17
payload  = "C" * 79 + "\n"
pkt3 = ip/payload

# Fragment No.4 (Fragment offset: (8 + 72 + 80 + 80)/8 = 30)
ip = IP(dst=dst_ip, id=ID, frag=30, flags=0)
ip.proto = 17
payload  = "D" * 67 + "\n"
pkt4 = ip/payload

# Sending fragments
send(pkt1)
send(pkt2)
send(pkt3)
send(pkt4)
```

Attacker:

```
root@21ea68d7f732:/volumes# ./fragment_spoof2.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
root@21ea68d7f732:/volumes#
```

Victim:

```
root@e915b82aca09:/# nc -lu 9090
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBB
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
```

Wireshark:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	02:42:0a:09:00:69	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.105
2	0.000145	02:42:0a:09:00:05	02:42:0a:09:00:69	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
3	0.132024	10.9.0.105	10.9.0.5	UDP	342	7070 → 9090 Len=300

```
▸ Frame 3: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface -, id 0
▸ Ethernet II, Src: 02:42:0a:09:00:69 (02:42:0a:09:00:69), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
▸ Internet Protocol Version 4, Src: 10.9.0.105, Dst: 10.9.0.5
▸ User Datagram Protocol, Src Port: 7070, Dst Port: 9090
▾ Data (300 bytes)
    Data: 414141414141414141414141414141414141414141414141…
    [Length: 300]
```

```
0020  00 05 1b 9e 23 82 01 34  00 00 41 41 41 41 41 41   ····#··4 ··AAAAAA
0030  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0040  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0050  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0060  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0070  41 0a 42 42 42 42 42 42  42 42 42 42 42 42 42 42   A·BBBBBB BBBBBBBB
0080  42 42 42 42 42 42 42 42  42 42 42 42 42 42 42 42   BBBBBBBB BBBBBBBB
0090  42 42 42 42 42 42 42 42  42 42 42 42 42 42 42 42   BBBBBBBB BBBBBBBB
00a0  42 42 42 42 42 42 42 42  42 42 42 42 42 42 42 42   BBBBBBBB BBBBBBBB
00b0  42 42 42 42 42 42 42 42  42 42 42 42 42 42 42 42   BBBBBBBB BBBBBBBB
00c0  42 0a 43 43 43 43 43 43  43 43 43 43 43 43 43 43   B·CCCCCC CCCCCCCC
00d0  43 43 43 43 43 43 43 43  43 43 43 43 43 43 43 43   CCCCCCCC CCCCCCCC
00e0  43 43 43 43 43 43 43 43  43 43 43 43 43 43 43 43   CCCCCCCC CCCCCCCC
00f0  43 43 43 43 43 43 43 43  43 43 43 43 43 43 43 43   CCCCCCCC CCCCCCCC
```

Data (data), 300 bytes | Packets: 3 · Displayed: 3 (100.0%) · Dropped: 0 (0.0%) | Profile: Default

Data 300 bytes ถูกตัดแบ่งส่งเป็น 4 fragments โดย

- fragment ที่ 1 ส่ง UDP Header 8 bytes กับ data 72 bytes (A 71 ตัว + \n) รวมเป็น 80 bytes

- fragment ที่ 2 และ 3 ส่ง data 80 bytes (fragment 2: B 79 ตัว + \n และ fragment 3: C 79 ตัว + \n)

- fragment ที่ 4 ส่ง data ส่วนที่เหลือ 68 bytes (D 67 ตัว + \n)

ทุก fragment มี IP header ระบุปลายทาง ระบุ ID เป็น 1000 และเลขลำดับ fragment offset ที่ fragment นั้นๆ เริ่มต้น เนื่องจาก unit ของ offset มีค่าเป็น 8 bytes การระบุตำแหน่งจึงต้องหารค่าด้วย 8 คือ

- fragment ที่ 1 เริ่มที่ 0

- fragment ที่ 2 เริ่มที่ 80 bytes ค่า offset คือ 80/8 = 10

- fragment ที่ 2 เริ่มที่ 160 bytes ค่า offset คือ 160/8 = 20

- fragment ที่ 2 เริ่มที่ 240 bytes ค่า offset คือ 240/8 = 30

ปลายทาง: 10.9.0.5 จะรอรับทุก fragment ให้ครบก่อน จึงจะแสดงผลบน terminal