

Demo-report

Let's see what the server received and parse the packets (before fixing them):

```
C:\Python27\python.exe E:/GoogleDrive/Projects/PyCharmProjects/NetworkSecurity/Server.py
Socket created
Socket bind complete
```

The packets that were received (**Before fixing**):

{e₁₋₅, **1**, 2, 3, 4, 5}

```
packet 0:( packet index - 0 )
[1, 1, 5, '0110000101100001']
packet 1:( packet index - 2 )
[2, '0110001001100010']
packet 2:( packet index - 3 )
[2, '0110001101100011']
packet 3:( packet index - 4 )
[2, '0110010001100100']
packet 4:( packet index - 5 )
[2, '0110010101100101']
```

{e₇₋₁₁, **7**, 8, 9, 10, 11}

```
packet 5:( packet index - 6 )
[1, 7, 11, '0110101001101010']
packet 6:( packet index - 8 )
[2, '0110011101100111']
packet 7:( packet index - 9 )
[2, '0110100001101000']
packet 8:( packet index - 10 )
[2, '0110100101101001']
packet 9:( packet index - 11 )
[2, '0110101001101010']
```

{e₁₃₋₁₅, **13**, 14, 15}

```
packet 10:( packet index - 12 )
[1, 13, 15, '0110101001101010']
packet 11:( packet index - 14 )
[2, '0110110001101100']
packet 12:( packet index - 15 )
[3, '0110110101101101']
```

The packets that were received (**After fixing**):

$\{e_{1-5}, 1, 2, 3, 4, 5\}$

```
packet 0:( packet index - 0 )  
[1, 1, 5, '0110000101100001']  
packet 1:( packet index - 1 )  
[2, '0110000101100001']  
packet 2:( packet index - 2 )  
[2, '0110001001100010']  
packet 3:( packet index - 3 )  
[2, '0110001101100011']  
packet 4:( packet index - 4 )  
[2, '0110010001100100']  
packet 5:( packet index - 5 )  
[2, '0110010101100101']
```

$\{e_{7-11}, 7, 8, 9, 10, 11\}$

```
packet 6:( packet index - 6 )  
[1, 7, 11, '0110101001101010']  
packet 7:( packet index - 7 )  
[2, '0110011001100110']  
packet 8:( packet index - 8 )  
[2, '0110011101100111']  
packet 9:( packet index - 9 )  
[2, '0110100001101000']  
packet 10:( packet index - 10 )  
[2, '0110100101101001']  
packet 11:( packet index - 11 )  
[2, '0110101001101010']
```

$\{e_{13-15}, 13, 14, 15\}$

```
packet 12:( packet index - 12 )  
[1, 13, 15, '0110101001101010']  
packet 13:( packet index - 13 )  
[2, '0110101101101011']  
packet 14:( packet index - 14 )  
[2, '0110110001101100']  
packet 15:( packet index - 15 )  
[3, '0110110101101101']
```

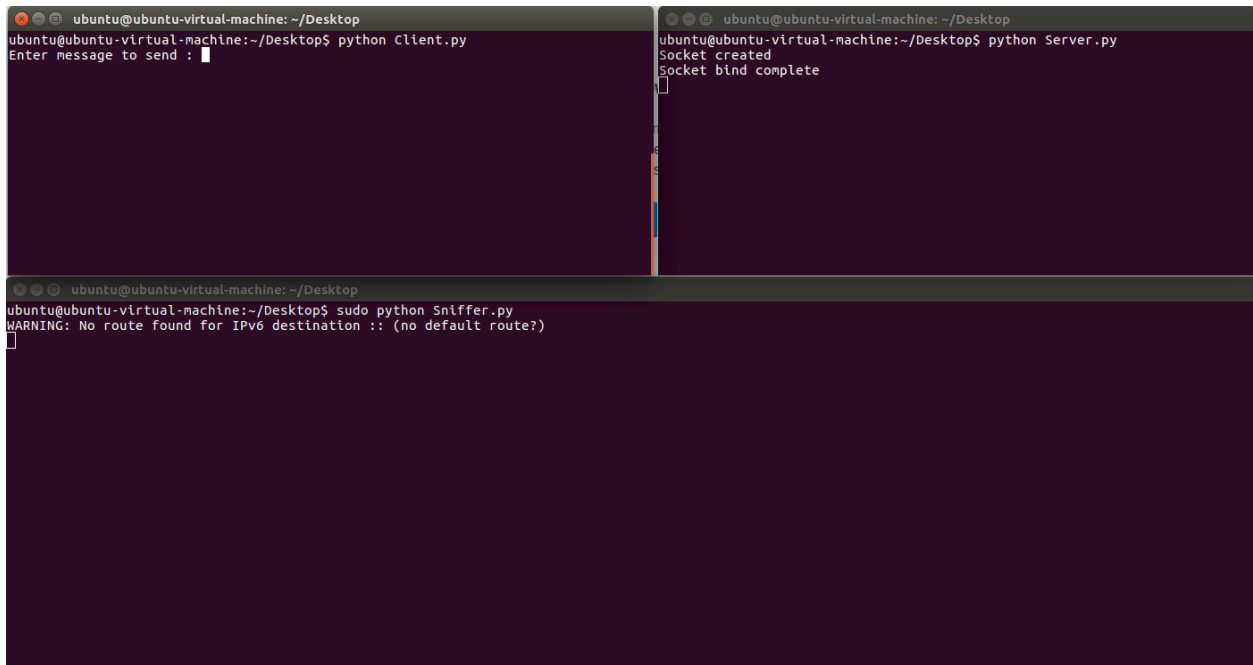
As we can see, from each d group, the first packet wasn't received, but the server returned the full message:

```
Message[127.0.0.1:58614] - aabbccddeeffgghhiijjkkllmm
```

We see that the server recovered despite the missing packets.

The sniffer build on Linux platform using scapy.

The Client, Server and the Sniffer working together .



```
ubuntu@ubuntu-virtual-machine: ~/Desktop
ubuntu@ubuntu-virtual-machine:~/Desktop$ python Client.py
Enter message to send :

ubuntu@ubuntu-virtual-machine:~/Desktop$ python Server.py
Socket created
Socket bind complete

ubuntu@ubuntu-virtual-machine:~/Desktop$ sudo python Sniffer.py
WARNING: No route found for IPv6 destination :: (no default route?)
```

For input: "aabbccdd"

```
ubuntu@ubuntu-virtual-machine: ~/Desktop
ubuntu@ubuntu-virtual-machine:~/Desktop$ python Client.py
Enter message to send : aabbccdde
Server reply : aabbccdde
Enter message to send :

ubuntu@ubuntu-virtual-machine:~/Desktop$ python Server.py
Socket created
Socket bind complete
Message[127.0.0.1:34355] - aabbccdde

ubuntu@ubuntu-virtual-machine:~/Desktop$ sudo python Sniffer.py
WARNING: No route found for IPv6 destination :: (no default route?)

Packet type: 1
Packet index: 0
First index of e XOR: 1
Last index of e XOR: 5
Packet XOR data: 0110000101100001

Packet type: 1
Packet index: 0
First index of e XOR: 1
Last index of e XOR: 5
Packet XOR data: 0110000101100001

Packet type: 2
Packet index: 2
Packet data: bb

Packet type: 2
```

The double prints are due the sniffing to the default connection 'lo' and not to 'eth0'.