Samarth Vats
CED17I020

Exercise-8
Device Drivers LAB

Short overview of SNULL

SNULL(Simple Network Utility for Loading Localities)

1. Characteristics and overview of SNULL:

- Driver of the network device
- Device that does not talk to the 'actual' devices
- Works like a loopback device
- Simulates actual operations
- Simulates communication with the actual servers
- Does not send hardware requests
- Only support IP protocol

2. Working and behaviour of SNULL:

SNULL basically works by creating two interfaces. The first, and most important, design decision was that the sample interfaces should remain independent of real hardware, just like most of the sample code used in this book. This constraint led to something that resembles the loopback interface.The messages transmitted in one interface loops back to the other interface. ***snull* is not a loopback interface**; however, it simulates conversations with real remote hosts in order to better demonstrate the task of writing a network driver. The Linux loopback driver is actually quite simple; it can be found in *drivers/net/loopback.c*.

To be able to establish such a communication through the SNULL interface the source and destination addresses need to be modified during data transmission. In other words, packets sent through one of the interfaces should be received by the other, but the receiver of the outgoing packet shouldn't be recognised as the local host. The same applies to the source address of received packets.

How SNULL does this is simple. It toggles the least significant bit of the 3rd octet of the IP(a part of network id) of the source and destination. Changing the source is important otherwise when the packet comes back, the user would be able to observe the sender as himself / herself, but the most important part is changing the destination, otherwise we wouldn't even receive the packet.

3. Example- working and initialization of SNULL:

Ref: https://www.oreilly.com/library/view/linux-device-drivers/0596005903/ch17.html
Let the IP[3] - represent the 32 bits. *IP - "IP[0] . IP[1] . IP[2] . IP[3]"*
IP[0] - part 1, *IP[1]* - part 2, *IP[2]* - part 3, *IP[3]* - part 4.

```
Snull() {
// Initializations
IP[2] ^= 1;
// Rebuild checksum
// Packet ready to be sent
}
```

The packet is now sent. The operation is done by swapping the last bit of the 3rd octet of IP using an XOR operator.

And thus for the user, a packet seems to magically come from some other network so that he / she can test other codes which takes care of incoming packets from other networks.