Device Drivers Practice Lab

CED17I046 - Venkata Ramanan A

Exercise 8

Write on how SNULL(**S**imple **N**etwork **U**tility for **L**oading **L**ocalities) works.

SNULL

Simple **N**etwork **U**tility for **L**oading **L**ocalities

What is SNULL?

- **The driver of the network device.**
- **The driver that does not talk to the ”actual” devices.**
- **Works like a loopback device.**
- **Simulates actual operations.**
- **Simulates communication with actual servers.**
- **It does not send hardware requests.**
- **It only supports IP protocol.**
- **The driver modifies the packets because there are no remote servers.**
- **It must know the protocol.**
- **Changes content (changes source / target addresses, etc)**

How is SNULL designed?

The design decision was that the sample interfaces should remain **independent of real hardware**. This constraint led to something that resembles the loopback 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*.

Another feature of SNULL is that it supports **only IP traffic**. This is a consequence of the internal workings of the interface that SNULL has to look inside and **interpret the packets** to properly **emulate** a pair of **hardware interfaces**. Real interfaces don't depend on the protocol being transmitted.

Assigning IP Numbers

The **SNULL** module **creates two interfaces**. These interfaces are different from a simple loopback, in that whatever you **transmit through one** of the **interfaces loops back** to the **other one**, **not** to **itself**. It looks like you have two external links, but actually, your computer is replying to itself.

Unfortunately, this effect can't be accomplished through IP number assignments alone, because the kernel wouldn't send out a packet through interface A that was directed to its own interface B. Instead, it would use the loopback channel without passing through SNULL. To be able to **establish communication** through the **SNULL interfaces**, 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 recognized as the localhost. The same applies to the source address of received packets.

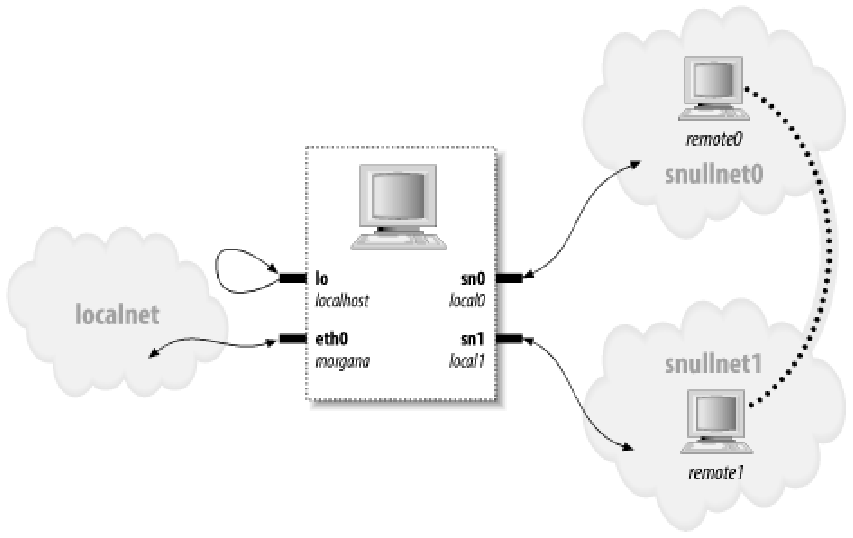
To achieve this kind of **"hidden loopback"**, the **SNULL** interface **toggles** the **least significant bit** of the **third octet** of both the **source** and **destination addresses**, i.e, it changes both the **network number** and the host number of **class C IP numbers**. The net effect is that packets sent to network A (connected to sn0 which is the first interface) appear on the sn1 interface as packets belonging to network B.

To avoid dealing with too many numbers, let's assign symbolic names to the IP numbers involved:

- **snullnet0** is the network that is connected to the **sn0** interface.
- Similarly, **snullnet1** is the network connected to **sn1**.
- The addresses of these networks should **differ** only in the **least significant bit** of the **third octet**.
- These networks must have **24-bit netmasks**.
- **local0** is the IP address assigned to the **sn0** interface, it belongs to **snullnet0**.
- The address associated with **sn1** is **local1**.
- **local0** and **local1** must **differ** in the **least significant bit** of their **third octet** and in **the fourth octet**.
- **remote0** is a host in **snullnet0**, and its fourth octet is the same as that of **local1**.

- Any packet sent to **remote0** reaches **local1** after its network address has been modified by the interface code.
- The host **remote1** belongs to **snullnet1**, and its fourth octet is the same as that of **local0**.

The operation of the SNULL interfaces in which the hostname associated with each interface is printed near the interface name.



Here are possible values for the network numbers. Once you put these lines in */etc/networks*, you can call your networks by name. The values were chosen from the range of numbers reserved for private use.

snullnet0	192.168.0.0
snullnet1	192.168.1.0

The following are possible host numbers to put into */etc/hosts*:

192.168.0.1	local0
192.168.0.2	remote0
192.168.1.2	local1
192.168.1.1	remote1

The important feature of these numbers is that the host portion of **local0** is the same as that of **remote1**, and the host portion of local1 is the same as that of **remote0**.

If the computer is already connected to a network, the numbers chosen must be real Internet or intranet numbers, and assigning them to your interfaces prevents communication with the real hosts. For example, although the numbers are just shown are not routable Internet numbers, they could already be used by your private network.

With the chosen number, set up the interfaces for operation by issuing the following commands:

```
ifconfig sn0 local0
ifconfig sn1 local1
```

Might need to add the **netmask 255.255.255.0** parameter if the address range was chosen is not a class C range.

At this point, the **"remote"** end of the interface can be reached. The following screen dump shows how a host reaches **remote0** and **remote1** through the SNULL interface:

```
morgana% ping -c 2 remote0
64 bytes from 192.168.0.99: icmp_seq=0 ttl=64 time=1.6 ms
64 bytes from 192.168.0.99: icmp_seq=1 ttl=64 time=0.9 ms
2 packets transmitted, 2 packets received, 0% packet loss
```

```
morgana% ping -c 2 remote1
64 bytes from 192.168.1.88: icmp_seq=0 ttl=64 time=1.8 ms
64 bytes from 192.168.1.88: icmp_seq=1 ttl=64 time=0.9 ms
2 packets transmitted, 2 packets received, 0% packet loss
```

Note that it won't be able to reach any other "host" belonging to the two networks because the **packets are discarded** by your computer **after the address** has been **modified** and the **packet** has been **received**. For example, a **packet** aimed at **192.168.0.32** will **leave through sn0** and **reappear at sn1** with a **destination** address of **192.168.1.32**, which is **not** a **local** address for the **host computer**.