Device Drivers Lab - 8

CED17I031

SNULL

SNULL (Simple Network Utility for Loading Localities) works by creating 2 interfaces. The messages transmitted in one interface loops back to the other interface.

It is not exactly a loopback interface, but 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**

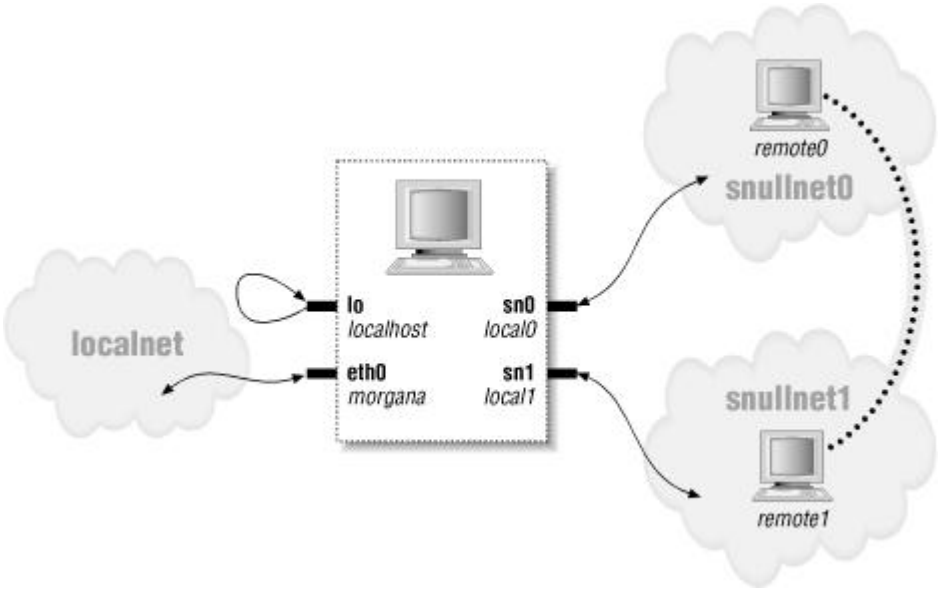
To be able to establish such a communication through the SNULL interface the source and destination addresses need to be modified during data transmission.

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.

-To simplify the discussion, the interface uses the Ethernet hardware protocol and transmits IP packets. The knowledge you acquire from examining *snull* can be readily applied to protocols other than IP, and writing a non-Ethernet driver is different only in tiny details related to the actual network protocol.

Example:



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 shown 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. You can use completely different numbers as long as this relationship applies.

Whatever numbers you choose, you can correctly set up the interfaces for operation by issuing the following commands:

```
ifconfig sn0 local0

ifconfig sn1 local1

case "`uname -r`" in 2.0.*)

   route add -net snullnet0 dev sn0

   route add -net snullnet1 dev sn1

esac
```

The following screendump 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

The Physical Transport of Packets

As far as data transport is concerned, the snull interfaces belong to the Ethernet class.

snull emulates Ethernet because the vast majority of existing networks—at least the segments that a workstation connects to—are based on Ethernet technology, be it 10baseT, 100baseT, or gigabit. Additionally, the kernel offers some generalized support for Ethernet devices, and there's no reason not to use it. The advantage of being an Ethernet device is so strong that even the plip interface (the interface that uses the printer ports) declares itself as an Ethernet device.

The last advantage of using the Ethernet setup for snull is that you can run tcpdump on the interface to see the packets go by. Watching the interfaces with tcpdump can be a useful way to see how the two interfaces work. (Note that on 2.0 kernels, tcpdump will not work properly unless snull's interfaces show up as ethx. Load the driver with the eth=1 option to use the regular Ethernet names, rather than the default snx names.)

As was mentioned previously, snull only works with IP packets. This limitation is a result of the fact that snull snoops in the packets and even modifies them, in order for the code to work. The code modifies the source, destination, and checksum in the IP header of each packet without checking whether it actually conveys IP information. This quick-and-dirty data modification destroys non-IP packets. If you want to deliver other protocols through snull, you must modify the module's source code.