Project 1. Let's Analyze Some Packets

Due: 02/06/2023

Goal

Write a network packet analyzer called **_pktsniffer_** that reads packets and produces a detailed summary of those packets. It works like tcpdump and Wireshark combined, and runs as a shell command. Note that pktsniffer does not support Boolean expressions unlike tcpdump. It first reads packets, not from a network interface, but from a specified file with **the -r flag** (like tcpdump). The pktsniffer program then extracts and displays the different headers of the captured packets. Specifically, it displays the Ethernet header fields of the captured frames. Then, if the Ethernet frame contains an IP datagram, it prints the IP header. Finally, it prints the packets encapsulated in the IP datagram. TCP, UDP, or ICMP packets can be encapsulated in the IP packet.

Turnin

Submit your program through myCourses with readme that contains how to compile and execute.

Input

Capture packets using Wireshark or a similar tool and store them into a .pcap file. See the supplemental files for details.

Output

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet size = 98 bytes
ETHER:  Destination = 0:c0:4f:79:ed:a2,
ETHER:  Source     = 0:0:c:b:47:64,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:       xxx. .... = 0 (precedence)
IP:       ...0 .... = normal delay
IP:       .... 0... = normal throughput
IP:       .... .0.. = normal reliability
IP:   Total length = 84 bytes
IP:   Identification = 32501
IP:   Flags = 0x4
IP:       .1.. .... = do not fragment
```

```
IP:        ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 254 seconds/hops
IP:   Protocol = 1 (ICMP)
IP:   Header checksum = 48f8
IP:   Source address = 129.21.30.37, queeg.cs.rit.edu
IP:   Destination address = 129.21.22.196, glados.cs.rit.edu
IP:   No options
IP:
ICMP:  ----- ICMP Header -----
ICMP:
ICMP:  Type = 8 (Echo request)
ICMP:  Code = 0
ICMP:  Checksum = f369
ICMP:


ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet size = 178 bytes
ETHER:  Destination = 8:0:20:1c:f:ee,
ETHER:  Source      = 0:c0:4f:79:ed:87,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:        xxx. .... = 0 (precedence)
IP:        ...0 .... = normal delay
IP:        .... 0... = normal throughput
IP:        .... .0.. = normal reliability
IP:   Total length = 68 bytes
IP:   Identification = 44605
IP:   Flags = 0x4
IP:        .1.. .... = do not fragment
IP:        ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = dac6
IP:   Source address = 129.21.30.37, queeg.cs.rit.edu
IP:   Destination address = 128.10.3.10, (hostname unknown)
IP:   No options
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 1022
UDP:  Destination port = 2049 (NFS)
UDP:  Length = 48
UDP:  Checksum = 8D93
```

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet size = 178 bytes
ETHER:  Destination = 8:0:20:1c:f:ee,
ETHER:  Source     = 0:c0:4f:79:ed:87,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:       xxx. .... = 0 (precedence)
IP:       ...0 .... = normal delay
IP:       .... 0... = normal throughput
IP:       .... .0.. = normal reliability
IP:   Total length = 164 bytes
IP:   Identification = 32484
IP:   Flags = 0x4
IP:       .1.. .... = do not fragment
IP:       ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = e6f5
IP:   Source address = 129.21.30.37, queeg.cs.rit.edu
IP:   Destination address = 128.10.3.10, (hostname unknown)
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 708
TCP:  Destination port = 2049 (NFS)
TCP:  Sequence number = 1438029708
TCP:  Acknowledgement number = 3430040415
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x18
TCP:       ..0. .... = No urgent pointer
TCP:       ...1 .... = Acknowledgement
TCP:       .... 1... = Push
TCP:       .... .0.. = No reset
TCP:       .... ..0. = No Syn
TCP:       .... ...0 = No Fin
TCP:  Window = 8760
TCP:  Checksum = 0x921e
TCP:  Urgent pointer = 0
TCP:  No options
```

Filtering

Extend pktsniffer with the following ***pcap-filter*** expression:
- host
- port
- ip
- tcp
- udp
- icmp
- net

Check the pcap-filter manual page for details. In addition, implement **the -c flag** for limiting the number of packets to be analyzed.

Examples

pktsniffer -r file host 2.2.2.2

pktsniffer -r file -c 5

pktsniffer -r file port 80

pktsniffer -r file -net 123.1.2.0

FAQ

1. Is there a way to check whether my output is correct?
   A. Run Wireshark with a given pcap file, and compare your output with its output.

2. What is the net flag for?
   A. It filters packets with a given network address. For example, pktsniffer -r file -net 123.1.2.0 displays all the packets that have 123.1.2.x in either the source or destination IP addresses.

3. Should my output exactly match the sample output or tcpdump output?
   A. No. As long as your output has correct values, you'll be fine. For example, if you cannot get the hostname of a given IP address because it is a private address, you don't have to print the hostname.