What Are Raw Sockets?

Last updated: August 4, 2023

Written by:Maxmilián Otta

Networking

Sockets

1. Introduction

In this tutorial, we'll discuss what so-called raw sockets are, their differences from traditional sockets, and how we can use them.

In general, a socket is an abstract representation of an endpoint of a communication link, just like a telephone socket mounted on a wall. If we want to call someone, we simply plug in the (of course, an old-fashioned landline) phone and dial a phone number. We don't have to care about the network's topology, call routing, and how our voice is transmitted.

2. A Brief History of Sockets

The release of the 4.2BSD Unix operating system in 1983 included a TCP/IP stack implementation and a C programming language API for inter-process communication (IPC) using Internet protocols and Unix domain sockets.

The Unix operating system developers wanted to design the network API using the traditional I/O approach of open, read, write, and close operations. But soon, they realized network communication is more complex than I/O operations on classical block devices.

The BSD sockets API became very popular and was adopted by other operating systems like Windows or Linux.

Finally, in 2000, sockets became part of the POSIX standard IEEE Std 1003.1g-2000, Part 1: System Application Program Interface (API) – Amendment 6 – Protocol-Independent Interfaces (PII).

3. The Sockets API

The socket API provides functions for inter-process communication over a network. This API is designed to be independent of the underlying network and information exchange patterns like connection-less unreliable message transport or connection-oriented reliable data streams.

When opening a socket, we should determine the address family (the underlying network type), the socket type (exchange pattern), and, optionally, the communication protocol. If we don't specify the protocol, the kernel will choose the default one for the given address family and socket type.

Let's now look in more detail at how a socket can be created by calling the *socket()* function:

The function takes three arguments and returns a file descriptor (socket descriptor) on success or -1 on error.

The first argument specifies the address family. The name "domain" was originally used in the BSD sockets API because it determines the "communication domain" or "protocol family". In C header files, we still find protocol family constants starting with *PF_*, but the POSIX standard uses the term address family and defines the constants starting with *AF_*, as we can see in the following table:

The second argument specifies the socket type. The socket type determines the communication pattern. The most common types are defined in the table next:

The last argument shows the protocol. The protocol is used to determine the communication protocol. If the protocol is set to zero, the kernel will choose the default protocol for the given address family and socket type. The following table indicates the most common protocols for the *AF_INET* and *AF_INET6* address families:

4. Raw Sockets

Commonly, the socket API is used for inter-process communication at the transport layer (OSI layer 4). However, some special socket types can be employed to access the network layer (OSI layer 3) and the data link layer (OSI layer 2). These socket types are called raw sockets.

Raw sockets are used when we want to access the network layer and the data link layer directly. For instance, we can use raw sockets to implement a network protocol below the transport layer, like ICMP, ARP, or OSPF. Furthermore, we can use them for monitoring a network, sniffing packets, or hacking, by sending forged packets.

Raw sockets are an optional part of the POSIX.1 standard and are only available on Linux and BSD systems. On Windows, raw sockets are available through the WinSock API.

In summary, raw sockets have the following differences from traditional sockets:

- It can be used to send and receive packets at the network layer (OSI layer 3) and the data link layer (OSI layer 2)
- It can be opened only by root users (or on Linux also when the executable file has the CAP_NET_RAW capability set)
- It can be opened only for the SOCK_RAW socket type

- The programmer can't use traditional file I/O operations like *read()* or *write()*, but only "single message" operations like ***send(), sendto() and sendmsg()*** or ***recv(), recvfrom() and recvmsg()***
- The programmer can construct the network layer packet header or must build the **data link frame** header (for instance, an Ethernet frame) manually

4.1. Raw Sockets at the Network Layer

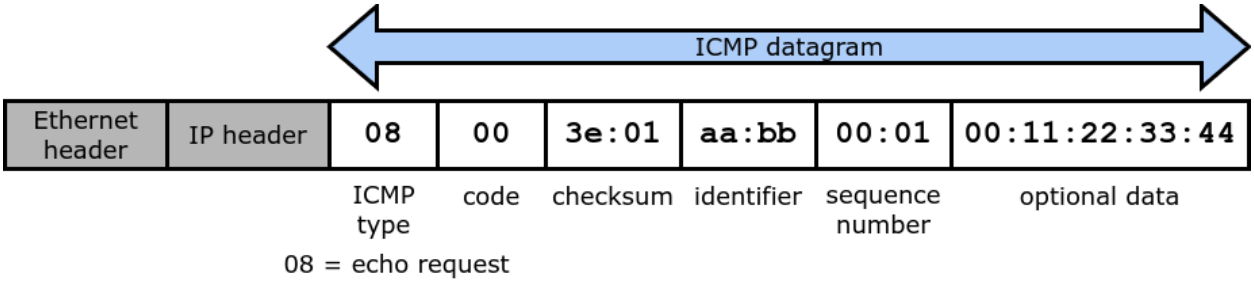
If we want to open a raw socket at the network layer (OSI layer 3), we must specify the AF_INET address family and the SOCK_RAW type. In the last argument, we need to define the Internet protocol number, which is included in the IP header field "Protocol":

Moreover, at the network layer (Internet Protocol), we have two options regarding how to handle the IP packet header construction:

- We can let the kernel construct the IP header for us (the *IP_HDRINCL* option is not set using *setsockopt()*)
- We can build the IP header ourselves (the *IP_HDRINCL* option is set using *setsockopt()*)

If we opt for the second option, we have total control over the IP header fields like Time-to-Live (employed, for instance, by *traceroute*) and source IP address (used for IP address spoofing), but we have to calculate the checksum ourselves.

This way works the well-known *ping* command, by sending an ICMP echo request message:

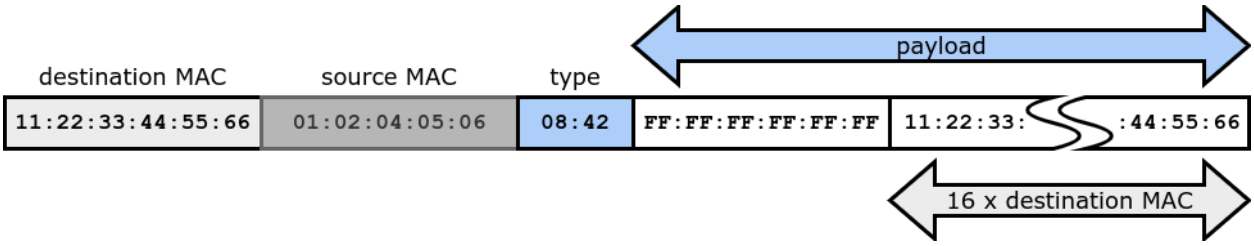


4.2. Raw Sockets at the Link Layer

If we want to open a raw socket at the data link layer (OSI layer 2), we need to specify the *AF_PACKET* address family and the *SOCK_RAW* socket type:

At the data link layer, we need to build the frame by ourselves. Moreover, on a computer having more than one network interface controller (NIC), we should also specify the interface at which we want the frame to send or receive.

Let's explore how we can use a raw socket at the link layer to send a Wake On LAN message, which turns on a device connected to the local network. **This function is often used in various smart home assistant software to turn on a TV using a voice command.** In order to do this, we open a raw socket at OSI layer 2, build a frame according to the Wake On LAN magic packet specification and send it to the target device:



5. Summary

To summarize the differences between raw sockets at OSI layers 2 and 3 and traditional sockets at OSI layer 4, we provide a table comparing the essential features side by side:

6. Conclusion

In this article, we discussed what raw sockets are, how to create them, and their application areas. Furthermore, we provided examples of the practical use of raw sockets at the OSI layers 2 and 3.

Raw sockets give the programmer total control over the PDU formation and access to information hidden when using traditional sockets.