1 Introduction

First of all we want to define the basic wordings and concepts used. Then the reader is given a short overview about the possibilities and restrictions the available APIs impose on the user.
Internet sockets are the common way to perform network communication implemented in most operating systems. They are usually provided by a socket API and are based upon the same principles as reading and writing a file. A program can get a socket via a function provided by the operating system. This function then returns a socket descriptor, usually a simple integer, similar to the ones provided by most operating systems for Read- and Write-Operations on files. This socket descriptor then can be used to write or read data from the socket. The data written to the socket is encapsulated by the operating system by adding headers and trailers and then the complete packet is sent via the network interface over the network to the target host. The data that has been received from the socket is presented to the program without the headers and trailers, only the user data is presented to the user. The sockets that work this way are the DATAGRAM- or STREAM-sockets provided via the Berkeley socket API. The user is unaware of the communication between the lower layers. All network communication steps, like for example the connection establishment, are taken care of without knowledge of the user. The user is only responsible for creating the socket and then providing the data that he wants to send to the correct function.

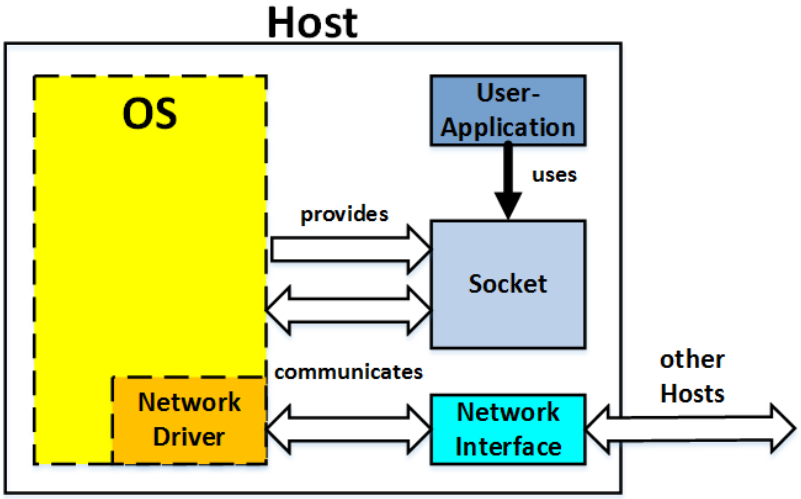

Figure 1: Socket provided by the operating system

The only parameters the user has to set are:

- The source socket address, a combination of IP address and port number can be set usually via the `bind()` function or defined directly with one of the `send()` functions.
- By choosing either the DATAGRAM- or STREAM-socket we can choose if we want to have a separate datagrams or a byte stream. This also chooses the Transport Layer Protocol that is being used (UDP or TCP).

If we want to gain access to the data of the lower layers we have several possibilities: RAW-sockets, PACKET-sockets, network drivers and Data Link layer APIs. The programming of Network Drivers will not be discussed further in this work, since we want to have a look at portable solutions that work for different operating systems. What can be accessed by the APIs we will present is shown in figure 2 on the following page.

With these APIs it is possible for an application to change and access the fields of the Network layers that are used for sending the data. This might be seen as a break with the traditional layering model, since we can influence the service the lower layers provide.

1.1 RAW-sockets

RAW-sockets are part of the standard Berkeley sockets and the socket API that is based upon it [1]. They are another option in addition to the already mentioned DATAGRAM- or STREAM-sockets to create data packets with the socket API [1]. In addition to simply sending data and defining address information the RAW-socket allows the user to access and manipulate the header and trailer information of the lower layers, more specifically with RAW-sockets to the Network and Transport layer (layer 3 and 4 of the OSI model). Since RAW-sockets are part of the Internet socket API they can only be used to generate and receive IP-Packets.

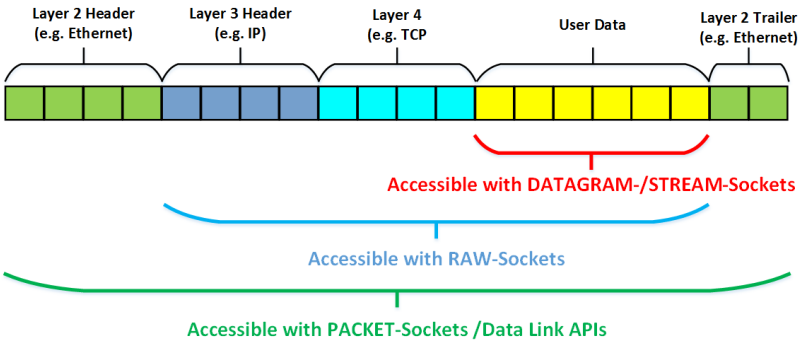

Figure 2: Overview over the layers and access possibilities

The biggest problem with RAW-sockets (also the PACKET-sockets we discuss later) is that there is no uniform API for using RAW-sockets under different operating systems:

- The provided APIs differ in regard to the used byte order. Depending on the operating system the fields of the packets have to be filled with data in network- or host-byte-order.
- Differences also exists in the types of packets and protocols that can be created using the RAW-socket API.
- The usage and functionality of the APIs is also different for each operating system.
- Some operating systems do not allow certain packet types to be received using the RAW-socket API.
- There are also differences in the definitions and paths of the necessary header files provided by each operating system.
- The required access levels for using the RAW-socket API can also differ. However most operating systems require root or admin access permissions to use them.

The user has to keep these things in mind if he wants to use the RAW-socket API, especially if he plans to use them across different operating systems.

1.2 PACKET-sockets and Data Link Layer APIs

If the user also wants to access the header and trailer of the Data Link layer then a Data Link Layer API has to be used. Under Linux (and Unix as well) this a functionality is provided by the operating system as the so called PACKET-socket and can be seen as a special form of RAW-sockets, so the limitations and problems of the previous section also apply to them [1]. But since this is not a standardized functionality there might be a different API provided by the operating system to perform these operations. For example under BSD the so called BSD Packet Filter (BPF) not the PACKET-socket API provides access to the Data Link Layer [2]. In general all of these APIs usually allow access to the whole packet from the Data Link Layer (OSI Layer 2), the Network Layer (OSI Layer 3) up to the Transport layer (OSI Layer 4). Not all operating systems provide a simple interface to access the Data Link Layer. As an alternative we introduce a library that can be used to provide this functionality.