**Module 2.1 – Intro to Socket Programming.**

**Goals:**

* Define socket programming.
* Introduce the different types of sockets covered in the rest of the modules.

**Required reading material:**

* Brian “Beej Jorgensen” Hall, “Beej's Guide to Network Programming, v3.1.11”. April 2023. <https://beej.us/guide/bgnet/html/split/>
  + Chapter 2.
    - If you have a solid foundation of the OSI layer, IP addresses, and port numbers you could skim through the reading materials.
  + Chapter 3, sections 3.1, Pages: 10-12.
* Jon Erickson, “Hacking the Art of Exploitation 2nd ed”. No Starch Press. February 2008. ISBN: 978-1593271442. <https://learning.oreilly.com/library/view/hacking-the-art/9781593271442/>
  + Read chapter 0x04, section 0x420 -Sockets.
* Lewis Van Winkle, “Hands-On Network Programming with C". Packt Publishing. May 2019. ISBN: 9781789349863. <https://learning.oreilly.com/library/view/hands-on-network-programming/9781789349863/>
  + Read chapter 2: [What are Sockets? page: 41][Two Types of sockets: pages: 44-45]
* W. Richard Stevens, Bill Fenner, Andrew M. Rudoff, “The Sockets Networking API: UNIX® Network Programming Volume 1, Third Edition”. Addison Wesley. November 2003. ISBN: 0-13-141155-1. <https://learning.oreilly.com/library/view/the-sockets-networking/0131411551/>
  + This book provides a more in-depth/technical explanation for the topics covered in this module.
  + Read Chapter 3 – Sockets Introduction. <https://learning.oreilly.com/library/view/the-sockets-networking/0131411551/ch03.html>

In this module we will cover the basics of socket programming and define what is meant by Socket Programming.

What is meant by network programming?

“Network programming is the act of using computer code to write programs or processes that can communicate with other programs or processes across a network. Programmers use various programming languages, code libraries, and protocols to do the work.”

[ <https://www.cisco.com/c/en/us/solutions/enterprise-networks/what-is-network-programming.html> ]

What is a socket?

A socket is a communications connection point (endpoint) that you can name and address in a network. Socket programming allows us to use sockets to establish communication links between remote and local processes.

“The processes that use a socket can reside on the same system or different systems on different networks. Sockets are useful for both stand-alone and network applications. Sockets allow you to exchange information between processes on the same machine or across a network, distribute work to the most efficient machine, and they easily allow access to centralized data. Socket application program interfaces (APIs) are the network standard for TCP/IP. A wide range of operating systems support socket APIs. i5/OS sockets support multiple transport and networking protocols. Socket system functions and the socket network functions are thread safe.” <https://www.ibm.com/docs/en/i/7.1?topic=communications-socket-programming>

In simple terms, a socket is one endpoint of a communication link between systems. Your application sends and receives all of its network data through a socket. A socket is a standard way to perform network communication through the Operating System. A socket can be thought of as an endpoint to a connection, similar to the telephone system.

Sockets are programmer's abstractions that take care of all the low-level protocol / layer details; think back to the OSI or TCP/IP layer models. A programmer uses a socket to send or receive data over a network and this data is then transmitted through the OSI layer following a similar path of:

Session layer-> Transport layer -> Application Layer -> Network layer -> Data link layer, and -> Physical layer. All these steps are abstracted and transparent to a programmer through the Socket API.

Although there are multiple types of sockets, the two most common are stream and datagram sockets, corresponding to TCP and UDP, respectively. There are also Raw sockets, while not common in standard applications, these are used in applications such as Scapy, Wireshark and similar applications. Raw sockets allow a programmer to send, receive, inspect, and manipulate network traffic at a lower level than standard network applications. We will cover Raw sockets in a later module.

**Type of Sockets**

We will focus on two types of sockets: Stream and Datagram sockets.

Stream Sockets

Stream sockets provide reliable two-way communication where one side initiates the connection to the other, and after the connection is established, either side can communicate with the other. In addition, there is immediate confirmation that what you transmitted reached its destination.

Stream sockets are implemented in the Transmission Control Protocol (TCP), which maps to layer 4 (Transport Layer) of the OSI model.

Connection-oriented, Reliability, error-free, and all other benefits of stream sockets are provided by the Transmission Control Protocol (TCP).

Common Stream Socket protocols: HTTP/S, SSH, Telnet, and so on. If it requires reliability, error-free, etc. it likely uses stream sockets.

Datagram Sockets

Communicating with a datagram socket is more like mailing a letter than making a phone call. Commonly referred to as "connectionless" sockets because they do not maintain an open connection like stream sockets do. They are more of the fire and forget variety, allowing us to build a packet and send it out to its destination. As such, datagram sockets provide less overhead when compared to stream sockets. Datagram sockets are better suited for applications such as gaming, video, and voice applications where loss of small number of packets is not important.

Datagram sockets are implemented in the User Datagram Protocol (UDP), which maps to layer 4 (Transport Layer) of the OSI model.

**Socket APIs**

There are a few different socket application programming interfaces (APIs). The first were Berkeley sockets, which were released in 1983 with 4.3BSD Unix. The Berkeley socket API was widely successful and quickly evolved into a de facto standard. From there, it was adopted as a POSIX standard with little modification. The terms Berkeley sockets, BSD sockets, Unix sockets, and Portable Operating System Interface (POSIX) sockets are often used interchangeably.

Windows' socket API is called Winsock. It was created to be largely compatible with Berkeley sockets. In the book “Hands-On Network Programming with C", the author makes use of cross-platform code that is valid for both Berkeley sockets and Winsock. We will focus on the Linux implementation in this course to avoid the added complexity of cross-platform development and focus on learning about socket programming instead.