There are multiple paradigms of communication on the internet. The two we are looking at will be stream-oriented and message-oriented.

- Stream paradigm
    - A connection-oriented communication service that requires the sender and the listener to both be connected.
    - This is a 1-to-1 communication paradigm.
    - Streams are sequences of individual bytes; the bytes do not necessarily arrive in order.
    - These streams can be of arbitrary length.
    - Stream messaging is used by most applications.
    - Stream messaging is built on the TCP protocol.
- Message paradigm
    - A connectionless service. The sender broadcasts, but does not receive confirmation that the listener has received the information, or even if there is anyone listening at all.
    - Many-to-many communication is possible; the connectionless service can be listened to by many machines at once.
    - The message paradigm is a sequence of individual messages, as opposed to a stream.
    - These messages are limited to only 64 kilobytes in size.
    - This is used for multimedia applications.
    - It is built on the UDP protocol.

What is TCP? TCP is a Transmission Control Protocol. It guarantees the delivery of uncorrupted packets by using flow control, error detection, congestion protocol, and re-transmission of any data that is corrupted or lost. A sequence of numbers is used to ensure that the packets are reassembled in the correct order once the receiving host gets the data.
The data from multiple transmissions can be transmitted as one large, single chunk; alternatively, it can be divided into many segments when it is being transmitted. Larger chunks of data can be transmitted faster due to reduced overhead, but smaller chunks of data take less time to replace if they are corrupted.

A TCP header contains information related to the message. This includes:

- Source port, the port the packet came from.
- Destination port, the port the packet is going towards.
- Sequence number, the number in the sequence of packets that this one stands in. Once all packets are received, the packets are reassembled based on their sequence number.
- Acknowledgement number, which few programmers use.
- Checksum, which is used to confirm that the data inside of the packet is intact. If it's not intact, it's sent again.
- Several other pieces of information.

What is UDP? UDP is a User Datagram Protocol. It is a connectionless transmission form that accepts and delivers individual messages. If the sender sends N bytes in a message, the receiver will receive exactly N bytes. This allows unicast, multicast, or broadcast transmission (for one destination, multiple destinations, or all destinations, respectively).

UDP is faster than TCP. Because UDP does not need to confirm that all messages arrive intact, headers do not need to be as large in a UDP transmission. Additionally, they do not need to wait for confirmation messages, and therefore can transmit much faster than TCP. However, the packets can be lost, duplicated, or received out of order; the application using UDP is responsible for handling the error conditions.

A UDP header contains information related to the message. This includes:

- Source port, the port the packet came from.
- Destination port, the port the packet is going towards.
- Message length.
- Checksum, which is used to confirm that the data inside of the packet is intact. If it's not intact, it is lost, and is NOT sent again.

If UDP is unreliable, why use it? If a computer network is congested, UDP will not notice; it will just keep dumping transmissions into the network, without any sense of how slowly information is moving. UDP is, however, great for real-time applications such as audio or video streaming.

On an abstract level, a socket is a mechanism to allow communication using different protocol families. For example, a socket will translate a TCP protocol into an OS protocol, allowing a computer (using its Operating System) to understand information that streams in from a network (using TCP or UDP).

Sockets first appeared in BSD 4.2, around 1983. They are now a POSIX standard. Microsoft uses WINSOCK API, which is fairly old. Most other computers use more modern sockets. Because sockets have been in use for so long, it can be inferred that they are useful and serve their purpose well.

What kind of tools can we develop with sockets and packets? One of these tools is "ping". Ping is a utility that sends a (user-adjustable) set of bytes of data to a destination address, asking the address to send a message back. Ping also times how long it took for the message to return.
While this may seem simple, ping is much more complicated than just this. Ping sends a lot of information, including the size of the packet, the place it came from, and other things. One can decode the rest of the header file for information.
Running a ping listening application like "tcpdump" can grant you a lot of power. Listening to the ping data from a machine lets you hear everything the user is sending and receiving. You can pick up passwords, messages, and even spoof someone else's appearance online. For this reason, you cannot perform this action on machines that you do not have privilege on.