Socket:

socket and socket API are used to send messages across a network.

The most common type of socket applications are client-server applications, where one side acts as the server and waits for connections from clients.

The socket is primarily a **concept** used in the Transport layer of the Internet model or Session layer of the OSI model.

Strictly speaking, Sockets is not a protocol, but rather a programming **method**.

A socket is externally identified to other hosts by its socket address, which is the triad of transport protocol, IP address, and port number.

Datagram Socket (connectionless):
They use “User Datagram Protocol” (UDP).

Stream Socket (two way, error-free):
Telnet (Terminal Network) — uses —> stream socket — uses —> Transmission Control Protocol (TCP) which makes sure your data arrives sequentially and error-free. (“TCP/IP” (Internet Protocol): IP deals primarily with Internet routing and is not responsible for data integrity).

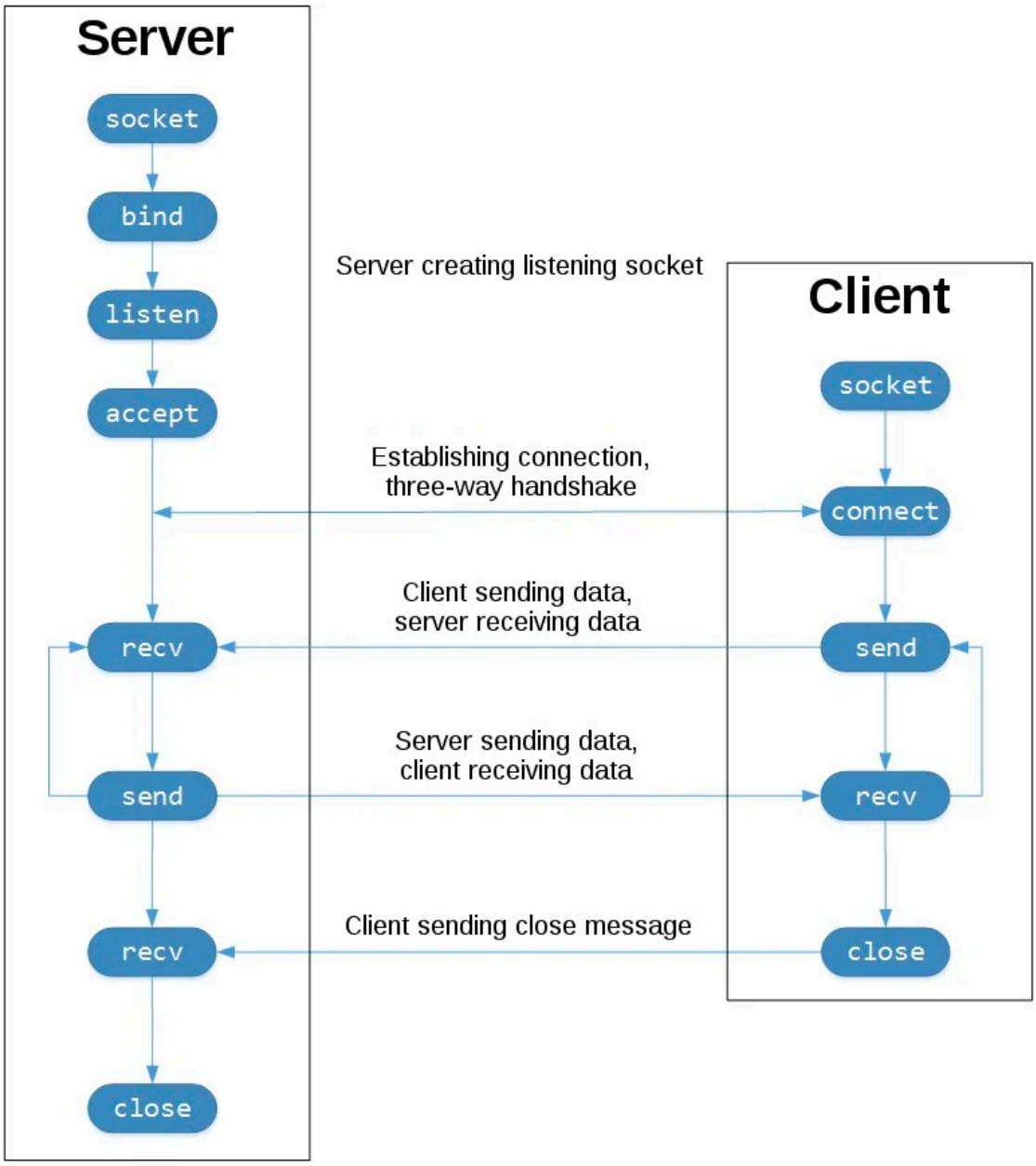
Web browsers get pages — uses —> Hypertext Transfer Protocol (HTTP) — uses —> stream sockets — uses —> TCP

TCP:
- Is reliable: packets dropped in the network are detected and retransmitted by the sender.
- Has in-order data delivery: data is read by your application in the order it was written by the sender.

HTTP:
Client/Server protocol,
intended for downloading HTML files,
can be generalized to download any kind of files
HTTP 3-digit responses status codes:
1xx - informational
2xx - success (200 OK)

3xx - redirection (301 moved permanently; 303 moved temporarily)
4xx - error (404 Not Found)
5xx - server error (505 http version not supported)

The sequence of socket API calls and data flow for TCP:



Network devices (for example, routers and switches), have finite bandwidth available and their own inherent system limitations. They have CPUs, memory,

buses, and interface packet buffers, just like our clients and servers. TCP relieves you from having to worry about packet loss, data arriving out-of-order, and many other things that invariably happen when you're communicating across a network.

Viewing Socket State:

$ netstat -an:

To see the current state of sockets on your host

after starting the server:
Proto: tcp4
Local Address : 127.0.0.1.65432
state: LISTEN

If EchoServer.py had used HOST = '' instead of '127.0.0.1', netstat would show:
Proto: tcp4
Local Address : *.65432
state: LISTEN

It means all available host interfaces that support the address family will be used to accept incoming connections. In EchoServer.py, we used socket.AF_INET,
so in the Proto column: tcp4.

$ lsof -i -n

lsof (List open files) gives you the COMMAND, PID(process id), and USER(user id) of open Internet sockets when used with the -i option.

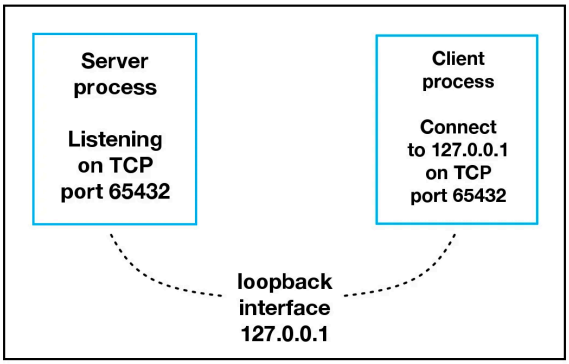
netstat and lsof have a lot of options available, check out the man page.
They're definitely worth spending a little time with and getting to know.
You'll be rewarded.

A common error when a connection attempt is made to a port with no listening socket:
1: the specified port number is wrong.
2: the server is not running.

3: there is a firewall in the path that's blocking the connection.

Communication Breakdown:

closer look on how the client and server communicate with each other:



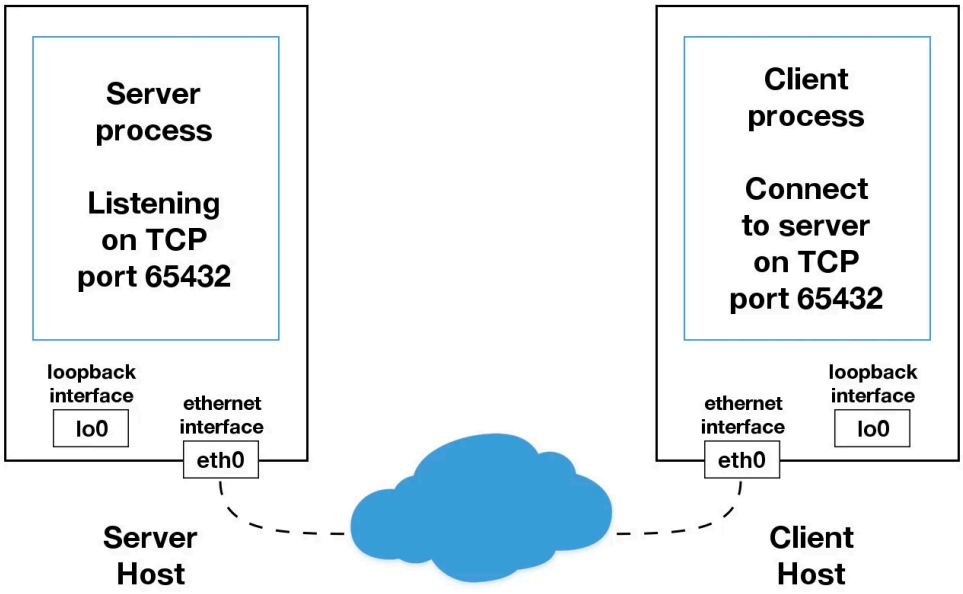
When using the loopback interface (IPv4 address 127.0.0.1 or IPv6 address ::1), data never leaves the host or touches the external network. In the diagram above, the loopback interface is contained inside the host; the connections and data transiting are local to the host. This is why the loopback interface and IP address 127.0.0.1 or ::1 referred to as “localhost”.

Applications use the loopback interface to communicate with other processes running on the host; for security and isolation from the external network. It’s not exposed since it’s internal and accessible only from within the host.

You can see this in action if you have an application server that uses its own private database. The database is configured to listen for connections on the loopback interface only, so other hosts on the network can’t connect to it.

When you use an IP address other than 127.0.0.1 or ::1 in your applications, it’s probably bound to an Ethernet interface that’s connected to an external network. This is your gateway to other hosts outside of your “localhost” kingdom.
Be careful out there. It’s a nasty, cruel world:)

Handling Multiple Connections:


Server
process
Listening
on TCP
port 65432
loopback
interface
lo0
ethernet
interface
eth0
Server
Host
Client
process
Connect
to server
on TCP
port 65432
loopback
interface
lo0
ethernet
interface
eth0
Client
Host