# What Is A Socket?

Sockets are a way to enable **communication between programs** running on a device, or between programs running on separate devices.

* For the first case, the socket is called *UNIX socket* (check this tutorial: *Tutorials\Linux\IPCs.docx - UNIX Socket*).
* For the second case, the socket is called *network socket*. It uses the Internet Protocol (IP) to encapsulate and handle sending and receiving data.

In this tutorial, we'll mainly talk about network socket.

Sockets configured as streams which are **bidirectional** and are controlled following a **client/server pattern**: The client connects to the server which will accept the connection. If everything works, requests from the client and responses from the server then flow through the channel until this is closed on either end, thereby breaking the connection.

# Socket Components

When creating a socket with the standard Linux API int socket(int domain, int type, int protocol), there are three components which we have to define:

## Domain

There are several options for domain, but the most used are:

* AF\_LOCAL(orAF\_UNIX): Local to the filesystem and can be used for internal communications in local machine.
* AF\_INET: IP socket and can be used for external communications across different machines.

## Socket Types

There are several options for socket *types*, but the most used are:

* SOCK\_STREAM: Connection-oriented sockets which uses **TCP** as the underlying protocol (reliable, bidirectional, byte streams, arbitrary-sized payload).
* SOCK\_DGRAM: Datagram-oriented sockets which uses **UDP** as the underlying protocol (unreliable, unidirectional, fixed-sized payload).
* SOCK\_RAW: Raw low-level network protocol access. Not a processed packet, but a raw packet. The application using the packet must be responsible for stripping off the headers, analyzing the packet, and other stuff that the TCP/UDP normally does for you. Check this [guide](https://www.opensourceforu.com/2015/03/a-guide-to-using-raw-sockets/).

## Underlying Protocols

There are several options for underlying protocol, but the most used are:

* **TCP**:
* **UDP**:

**Note**: If you set protocol as 0, you'll let the system picks underlying protocol. Normally, only a single protocol exists to support a particular *socket type* within a given protocol family:

* Socket type SOCK\_STREAM uses TCP
* Socket type SOCK\_DGRAM uses UDP

However, it is possible that many protocols may exist, in which case a particular protocol must be specified.

For more insights to these IPs (Internet Protocols), read this tutorial:

[*Tutorials\Network\Communication Protocols.docx*](../Network/Communication%20Protocols.docx) *- IPs (Internet Protocols)*

# Socket Properties

## States

Read this tutorial: [*Tutorials\Network\Communication Protocols.docx*](../Network/Communication%20Protocols.docx) *- IPs (Internet Protocols) / TCP / States*

## Binding

### What Is Binding?

**Each socket needs a unique address**. Keep in mind that **an** **address = an IP address + a port number**.

When a socket is created, it assumes the IP address of the network node that created it.

If a socket has an IP address but not a port number, it is said to be 'unbound'. An unbound socket cannot receive data because it does not have a complete address.

When a socket has both an IP address and a port number, it is said to be 'bound to a port', or 'bound to an address'. A bound socket can receive data because it has a complete address.

**The process of allocating a port number to a socket is called 'binding'.**

### What to Consider when Binding?

#### Choose IP Address

The IP address range is huge, but you cannot pick any number as you like. There're rules for that!

Otherwise, you’ll get error "*cannot assign requested address*" (Linux error number: EADDRNOTAVAIL 🡪 *A nonexistent interface was requested, or the requested address was not local*).

So basically, a **valid IP address must be: existent or localhost**.

* To check all existent IP addresses on your machine, run command: $ ifconfig | grep 'inet addr'. The result usually includes 127.0.0.1 and your machine's public IP address.
* Localhost IP address (also called *loopback IP address*) ranges from 127.0.0.0 to 127.255.255.255. You can pick any number in this range.

In addition, there is another special IP address, which is 0.0.0.0 (in Linux, defined by macro INADDR\_ANY). When binding to this IP, bind() does NOT generate a random IP, but it **binds the socket to all available interfaces**. This way, the server socket listens and accepts connections from all the valid IP addresses of the machine.

For a server, you typically want to bind to all interfaces – not just "localhost".

#### Choose Port

There are many options for port:

* Well-known ports range from 0 to 1023.
* Registered ports (or reserved ports) are from 1024 to 49151.
* Dynamic ports (or private ports) are from 49152 to 65535.

Registered ports are used by standard services. So, **our custom port should be dynamic ports**.

### What If You Don't Want To Bind?

Port 0: You should use 0 as the port. The OS will choose a random available port for you. This is the same way it would do if you had not called bind().

Raw socket is not bound to any port (only address and protocol).

## Blocking vs Non-Blocking

# Linux Comands

## Check Open Ports

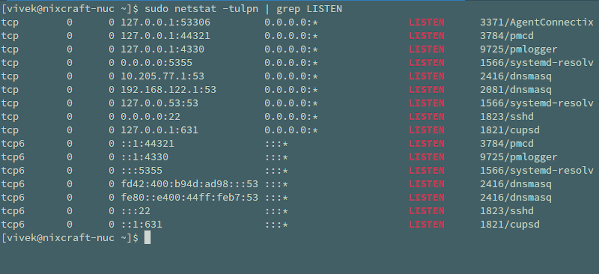
### Using 'netstat'

$ netstat -tulpn | grep LISTEN

Where:

* **-t** : All TCP ports
* **-u** : All UDP ports
* **-l** : Display listening server sockets
* **-p** : Show the PID and name of the program to which each socket belongs
* **-n** : Don’t resolve names
* **| grep LISTEN** : Only display open ports by applying grep command filter.

For examples:



Where:

* TCP port 22 opened by *sshd* process and *sshd* listing on all IP address for ssh connections.
* TCP port 631 opened by *cupsd* process and *cupsd* only listing on the loopback address (127.0.0.1).

### Using 'ss'

The ss command is used to dump socket statistics. It allows showing information similar to netstat. It can display more TCP and state information than other tools. The syntax is:

$ sudo ss -tulpn

Sample outputs:

Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port

udp UNCONN 0 0 224.0.0.251:5353 0.0.0.0:\* users:(("chromium-browse",pid=12893,fd=419))

udp UNCONN 0 0 224.0.0.251:5353 0.0.0.0:\* users:(("chromium-browse",pid=12938,fd=395))

udp UNCONN 0 0 224.0.0.251:5353 0.0.0.0:\* users:(("chrome",pid=10111,fd=178))

udp UNCONN 0 0 224.0.0.251:5353 0.0.0.0:\* users:(("chrome",pid=10111,fd=139))

udp UNCONN 0 0 224.0.0.251:5353 0.0.0.0:\* users:(("chrome",pid=10111,fd=48))

udp UNCONN 0 0 224.0.0.251:5353 0.0.0.0:\* users:(("chrome",pid=10161,fd=43))

udp UNCONN 0 0 0.0.0.0:5353 0.0.0.0:\* users:(("avahi-daemon",pid=1590,fd=15))

udp UNCONN 0 0 0.0.0.0:5355 0.0.0.0:\* users:(("systemd-resolve",pid=1566,fd=12))

udp UNCONN 0 0 0.0.0.0:55204 0.0.0.0:\* users:(("avahi-daemon",pid=1590,fd=17))

udp UNCONN 0 0 0.0.0.0:49112 0.0.0.0:\* users:(("openvpn",pid=18342,fd=8))

udp UNCONN 0 0 10.205.77.1:53 0.0.0.0:\* users:(("dnsmasq",pid=2416,fd=8))

udp UNCONN 0 0 192.168.122.1:53 0.0.0.0:\* users:(("dnsmasq",pid=2081,fd=5))

udp UNCONN 0 0 127.0.0.53%lo:53 0.0.0.0:\* users:(("systemd-resolve",pid=1566,fd=17))

udp UNCONN 0 0 0.0.0.0%lxdbr0:67 0.0.0.0:\* users:(("dnsmasq",pid=2416,fd=4))

udp UNCONN 0 0 0.0.0.0%virbr0:67 0.0.0.0:\* users:(("dnsmasq",pid=2081,fd=3))

udp UNCONN 0 0 0.0.0.0:68 0.0.0.0:\* users:(("dhclient",pid=18263,fd=7))

udp UNCONN 0 0 127.0.0.1:323 0.0.0.0:\* users:(("chronyd",pid=1652,fd=6))

udp UNCONN 0 0 [::]:5353 [::]:\* users:(("avahi-daemon",pid=1590,fd=16))

udp UNCONN 0 0 [::]:5355 [::]:\* users:(("systemd-resolve",pid=1566,fd=14))

udp UNCONN 0 0 [::]:60302 [::]:\* users:(("avahi-daemon",pid=1590,fd=18))

udp UNCONN 0 0 [fd42:400:b94d:ad98::1]:53 [::]:\* users:(("dnsmasq",pid=2416,fd=12))

udp UNCONN 0 0 [fe80::e400:44ff:feb7:3233]%lxdbr0:53 [::]:\* users:(("dnsmasq",pid=2416,fd=10))

udp UNCONN 0 0 [::1]:323 [::]:\* users:(("chronyd",pid=1652,fd=7))

udp UNCONN 0 0 [::]%lxdbr0:547 [::]:\* users:(("dnsmasq",pid=2416,fd=6))

tcp LISTEN 0 5 127.0.0.1:44321 0.0.0.0:\* users:(("pmcd",pid=3784,fd=0))

tcp LISTEN 0 5 127.0.0.1:4330 0.0.0.0:\* users:(("pmlogger",pid=9725,fd=9))

tcp LISTEN 0 128 0.0.0.0:5355 0.0.0.0:\* users:(("systemd-resolve",pid=1566,fd=13))

tcp LISTEN 0 5 10.205.77.1:53 0.0.0.0:\* users:(("dnsmasq",pid=2416,fd=9))

tcp LISTEN 0 32 192.168.122.1:53 0.0.0.0:\* users:(("dnsmasq",pid=2081,fd=6))

tcp LISTEN 0 128 127.0.0.53%lo:53 0.0.0.0:\* users:(("systemd-resolve",pid=1566,fd=18))

tcp LISTEN 0 128 0.0.0.0:22 0.0.0.0:\* users:(("sshd",pid=1823,fd=5))

tcp LISTEN 0 5 127.0.0.1:631 0.0.0.0:\* users:(("cupsd",pid=1821,fd=10))

tcp LISTEN 0 5 [::1]:44321 [::]:\* users:(("pmcd",pid=3784,fd=3))

tcp LISTEN 0 5 [::1]:4330 [::]:\* users:(("pmlogger",pid=9725,fd=10))

tcp LISTEN 0 128 [::]:5355 [::]:\* users:(("systemd-resolve",pid=1566,fd=15))

tcp LISTEN 0 5 [fd42:400:b94d:ad98::1]:53 [::]:\* users:(("dnsmasq",pid=2416,fd=13))

tcp LISTEN 0 5 [fe80::e400:44ff:feb7:3233]%lxdbr0:53 [::]:\* users:(("dnsmasq",pid=2416,fd=11))

tcp LISTEN 0 128 [::]:22 [::]:\* users:(("sshd",pid=1823,fd=7))

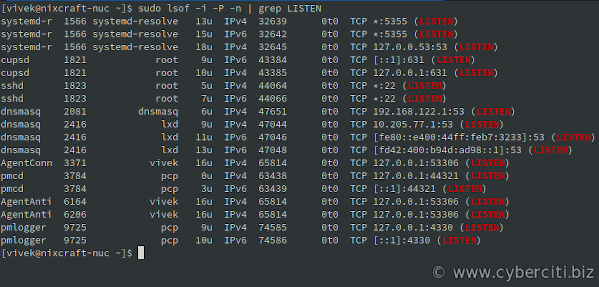
tcp LISTEN 0 5 [::1]:631 [::]:\* users:(("cupsd",pid=1821,fd=9))

### Using 'lsof'

$ sudo lsof -i -P -n | grep LISTEN

Where:

* **-i** : Look for listing ports
* **-P** : Inhibits the conversion of port numbers to port names for network files. Inhibiting the conversion may make lsof run a little faster. It is also useful when port name lookup is not working properly.
* **-n** : Do not use DNS name
* **| grep LISTEN** : Only show ports in LISTEN state.



### Using 'nmap'

The [nmap command](https://www.cyberciti.biz/networking/nmap-command-examples-tutorials/) which is an open-source tool for network exploration and security auditing.

To find and list open ports with nmap, run:

$ sudo nmap -sT -O localhost

$ sudo nmap -sU -O 192.168.2.254 # list open UDP ports

$ sudo nmap -sT -O 127.0.0.1 # list open TCP ports

$ sudo nmap -sTU -O 192.168.2.24

Sample outputs:

Starting Nmap 7.70 ( https://nmap.org ) at 2019-07-22 23:49 IST

Nmap scan report for localhost (127.0.0.1)

Host is up (0.00024s latency).

Other addresses for localhost (not scanned): ::1

Not shown: 998 closed ports

PORT STATE SERVICE

22/tcp open ssh

631/tcp open ipp

Device type: general purpose

Running: Linux 2.6.X

OS CPE: cpe:/o:linux:linux\_kernel:2.6.32

OS details: Linux 2.6.32

Network Distance: 0 hops

## Create Sockets

TCP: <https://www.digitalocean.com/community/tutorials/understanding-sockets#creating-tcp-based-stream-sockets>

UDP: <https://www.digitalocean.com/community/tutorials/understanding-sockets#creating-datagram-sockets>

UNIX: <https://www.digitalocean.com/community/tutorials/understanding-sockets#creating-unix-domain-sockets>

# Coding

## Linux

### UNIX Socket

Code sample at: *Tutorials\Linux\Code\IPCs\UnixSocket*

### Network Socket

APIs:

<https://manpages.ubuntu.com/manpages/bionic/man2/send.2.html>

<https://stackoverflow.com/a/1100524>

## Windows

### UNIX Socket

### Network Socket