Topic -1 Basic

- System Call
- Context Switching
- File Descriptor
- NAT
- NAT Gateway

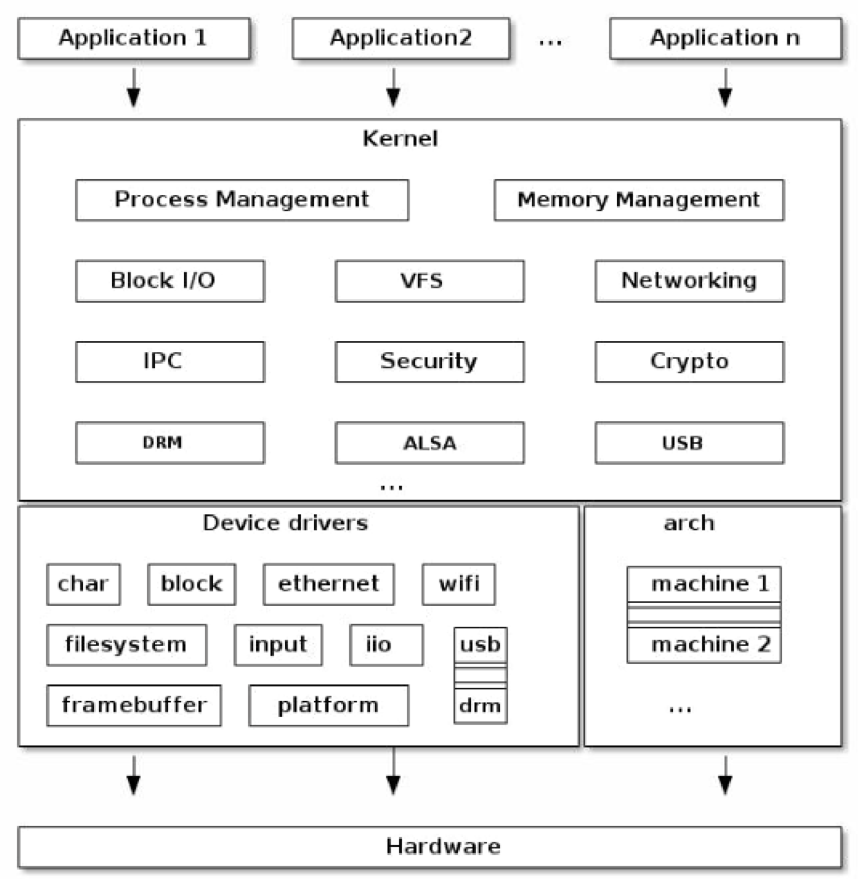
System Call :

- **Syscalls** are **functions in the kernel** that **provide services to a user space application**. They are the API that the kernel exposes to user space programs, which allow a program to utilise the functionality the kernel offers.
- Examples include starting new processes, disk I/O, and networking.
- All the modern computer operating systems generally divide the **virtual memory**(vMem) into two following spaces:
    - Kernel space
    - User space

If a program executes in "user space" then that program does not have direct access to the memory, the hardware and such resources.

But if a program is executing in kernel mode, then that program has the direct access to the memory, the hardware and such resources. So, if a program is executing in kernel mode, then it is in a privileged mode because it is having direct access to many of the resources.

But when a program happens to crash during its execution in kernel mode then the entire system would crash, or it comes to a halt.



But when a program happens to crash during its execution in user mode then the entire system would crash, or it comes to a halt.
So, user mode is safer for execution though kernel mode is privileged mode.
When a program is executing in user mode and needs to be switched to kernel mode for a particular time then "system call" is generated.

Context Switching:

When a program switched from user mode to kernel mode (or vise-a-versa) is known as "context switching".

System Call Type:

System calls are divided into 5 categories mainly:

1. Process Control
2. File Management
3. Device Management
4. Information Maintenance
5. Communication

Process Control:

- This system calls perform the **task of process creation, process termination**, etc.
- The Linux System calls under this are fork() , exit() , exec().

a) fork()
   - A new process is created by the fork() system call.

b) exit()
   - The exit() system call is used by a program to **terminate its execution**.

c) exec()
   - A **new program will start** executing after a call to exec()
   - The currently running program is immediately terminated, and the new program starts executing in the context of the existing process.

File Management:

- File management system calls handle file manipulation jobs like **creating a file, reading, and writing**, etc. The Linux System calls under this are open(), read(), write(), close().

a) open():

   - It is the system call to **open a file**.
   - This system call just opens the file, to perform operations such as read and write

b) read():
   - This system call opens the file in **reading mode**
   - We can not edit the files with this system call.
   - **Multiple processes can execute the read()** system call on the same file simultaneously.

c) write():
   - This system call **opens the file in writing mode**
   - We can edit the files with this system call.
   - **Multiple processes can not** execute the **write() system** call on the same file simultaneously.

d) close():
   - This system call **closes the opened file**.

Device Management :

- Device management does the job of **device manipulation like reading from device buffers, writing into device buffers**, etc.
- The Linux System calls under this is ioctl().

a) ioctl():
   - ioctl() is referred to as Input and Output Control.
   - ioctl is a system call for device-specific input/output operations and other operations which cannot be expressed by regular system calls.

Information Maintenance:

- It **handles information and its transfer between the OS and the user program**. In addition, OS keeps the information about all its processes and system calls are used to access this information. The System calls under this are getpid(), alarm(), sleep().

a) getpid():
   - getpid stands for Get the Process ID.
   - The getpid() function shall return the process ID of the calling process.
   - The getpid() function shall always be successful and no return value is reserved to indicate an error.

b) alarm():
   - This system call sets an alarm clock for the delivery of a signal that when it has to be reached.
   - It arranges for a signal to be delivered to the calling process.

c) sleep():
   - This System call suspends the execution of the currently running process for some interval of time
   - Meanwhile, during this interval, another process is given chance to execute

Communication :

- These types of system calls are specially **used for inter-process communications**(IPC).
- Two models are used for inter-process communication
- Message Passing(processes exchange messages with one another)
- Shared memory(processes share memory region to communicate)
- The system calls under this are pipe() , shmget() ,mmap().

a) pipe():
   - The pipe() system call is used to **communicate between different Linux processes**.
   - It is mainly used for inter-process communication.
   - The pipe() system function is used to **open file descriptors**.

b) shmget():
   - shmget stands for shared memory segment.
   - It is mainly used for **Shared memory communication**.
   - This system call is used to access the shared memory and access the messages in order to communicate with the process.

c) mmap():
   - This function call is used to **map or unmap files or devices into memory**.

The mmap() system call is responsible for mapping the content of the file to the virtual memory space of the process.

########################

File Descriptor:

In simple words, when you open a file, the operating system creates an entry to represent that file and store the information about that opened file. So if there are 100 files opened in your OS then there will be 100 entries in OS (somewhere in kernel). These entries are represented by integers like (...100, 101, 102....). This entry number is the file descriptor. So it is just an integer number that uniquely represents an opened file for the process. If your process opens 10 files then your Process table will have 10 entries for file descriptors.

On a Unix-like operating system, the first **three file descriptors**, by default, are **STDIN** (standard input-0), **STDOUT** (standard output-1), and **STDERR** (standard error-2).

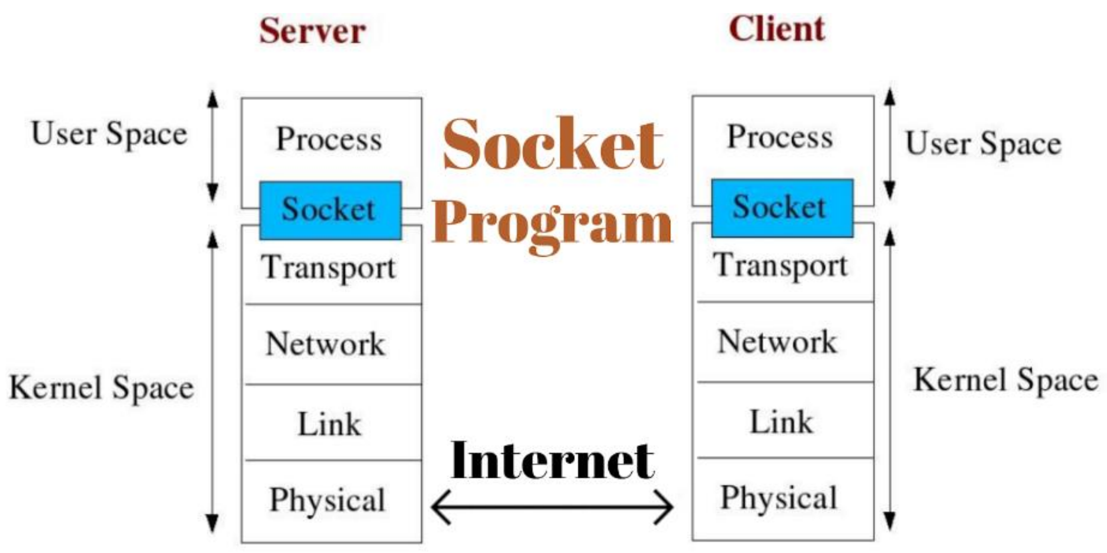
Similarly, when you open a network socket, it is also represented by an integer and it is called Socket Descriptor.

########################

Sockets:

Sockets **allow communication between two different processes** on the same or different machines. To be more precise, it's a way to talk to other computers using standard Unix file descriptors. In Unix, every I/O action is done by writing or reading a file descriptor.

A Unix **Socket is used in a client-server application framework**. A server is a process that performs some functions on request from a client. Most of the application-level protocols like **FTP**, **SMTP**, and **POP3** make use of sockets to establish connection between client and server and then for exchanging data.



Socket Types :

There are four types of sockets available to the users. The first two are most commonly used and the last two are rarely used.

a) Stream Sockets
Delivery in a networked environment is guaranteed. If you send through the stream socket three items "A, B, C", they will arrive in the same order − "A, B, C". These sockets use TCP (Transmission Control Protocol) for data transmission. If delivery is impossible, the sender receives an error indicator. Data records do not have any boundaries.

b) Datagram Sockets
Delivery in a networked environment is not guaranteed. They're connectionless because you don't need to have an open connection as in Stream Sockets − you build a packet with the destination information and send it out. They use UDP (User Datagram Protocol).

c) Raw Sockets
These provide users access to the underlying communication protocols, which support socket abstractions. These sockets are normally datagram oriented, though their exact characteristics are dependent on the interface provided by the protocol. Raw sockets are not intended for the general user; they have been provided mainly for those interested in developing new communication protocols, or for gaining access to some of the more cryptic facilities of an existing protocol.

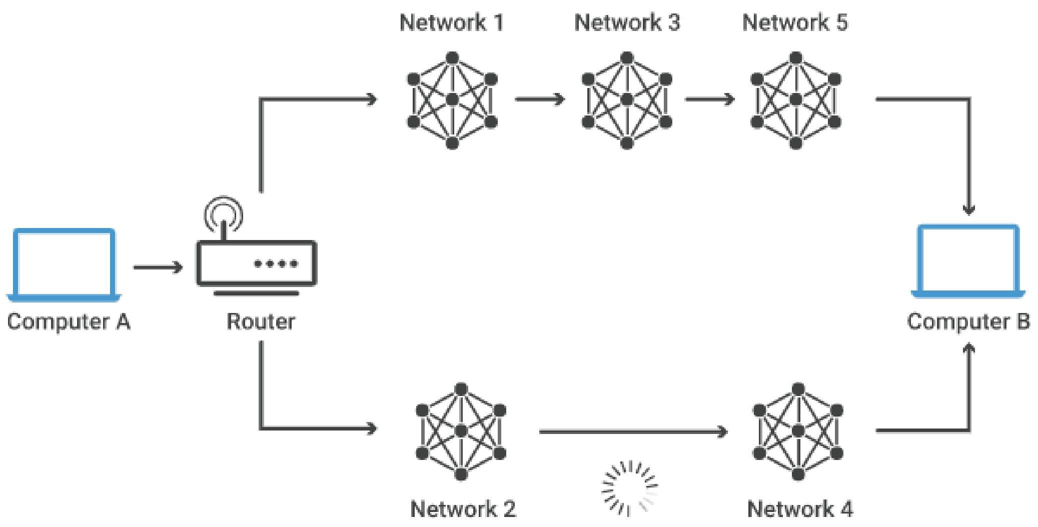
d) Sequenced Packet Sockets
They are similar to a stream socket, with the exception that record boundaries are preserved. This interface is provided only as a part of the Network Systems (NS) socket abstraction, and is very important in most serious NS applications. Sequenced-packet sockets allow the user to manipulate the Sequence Packet Protocol (SPP) or Internet Datagram Protocol (IDP) headers on a packet or a group of packets, either by writing a prototype header along with whatever data is to be sent, or by specifying a default header to be used with all outgoing data, and allows the user to receive the headers on incoming packets.

#####################

Network Routing:

Network routing is the **process of selecting a path across one or more networks**. In packet-switching networks, such as the Internet, routing selects the paths for Internet Protocol (IP) packets to travel from their origin to their destination. These Internet routing decisions are made by specialized pieces of network hardware called routers.

A router is a piece of network hardware responsible for forwarding packets to their destinations. Routers connect to two or more IP networks or subnetworks and pass data packets between them as needed.



A routing protocol is a protocol used for identifying or announcing network paths.

The following protocols help data packets find their way across the Internet:

- IP: The Internet Protocol (IP) specifies the origin and destination for each data packet. Routers inspect each packet's IP header to identify where to send them.
- BGP: The Border Gateway Protocol (BGP) routing protocol is used to announce which networks control which IP addresses, and which networks connect to each other. (The large networks that make these BGP announcements are called autonomous systems.) BGP is a dynamic routing protocol.

The below protocols route packets within an AS:

- OSPF: The Open Shortest Path First (OSPF) protocol is commonly used by network routers to dynamically identify the fastest and shortest available routes for sending packets to their destination.
- RIP: The Routing Information Protocol (RIP) uses "hop count" to find the shortest path from one network to another, where "hop count" means number of routers a packet must pass through on the way. (When a packet goes from one network to another, this is known as a "hop.")

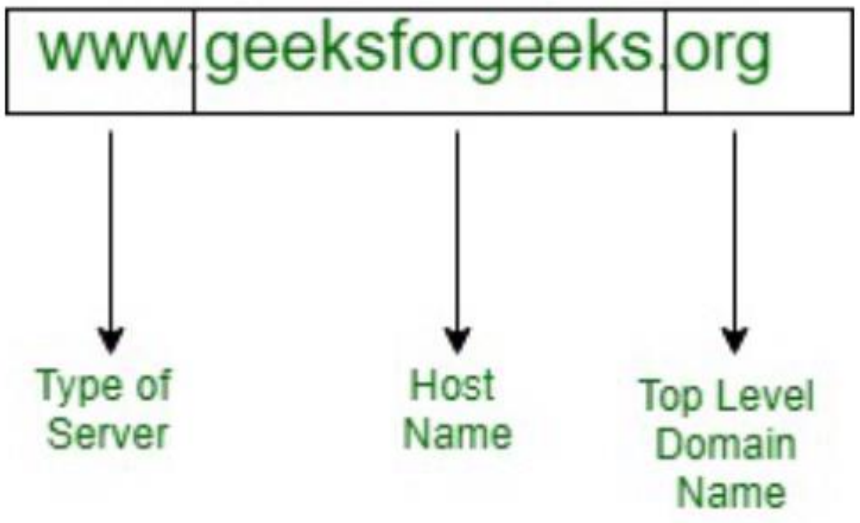
Other interior routing protocols include EIGRP (the Enhanced Interior Gateway Routing Protocol, mainly for use with Cisco routers) and IS-IS (Intermediate System to Intermediate System).

Difference between Domain Name and URL based routing:

a) Domain Name :

- It is very hard to remember the IP address of the server. Hence, the domain name is the text form of the IP address and it is more human friendly and easy to learn.
- All the parts of the domain name are separated by the periods or dots. An example of the domain name is www.geeksforgeeks.org. A domain name can be divided into the following parts.

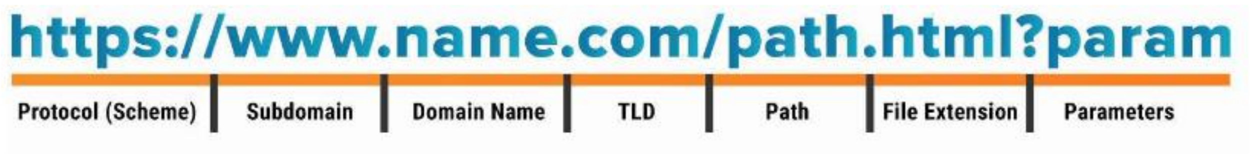
- Type of Server : The www or World Wide Web represents the web server.
- Host Name : It is generally the main name of the domain like google , geeksforgeeks etc.
- Top-level Domain : The last part of the domain name is top-level domain. Examples are .com,.in,.au etc.



b) Uniform Resource Locator(URL) :

- Uniform Resource Locator is known as a URL. It is a string that provides the complete address of the web page i.e, this is also known as a web address. In order to access a particular website you just need to put the URL in the search bar to find the website. For example, the web address of the google home page is https://www.google.com/.

- A URL can be divided into different parts.
  - Method or Protocol : This is the method or protocol used to retrieve the file from the server.
  - Host Name : It is also known as the domain name or a human-friendly text form for the IP address of the server where the file or document is located.
  - Port : It is just a protocol number. It is just a communication endpoint.
  - Path : It is the location of the file on the server.



################

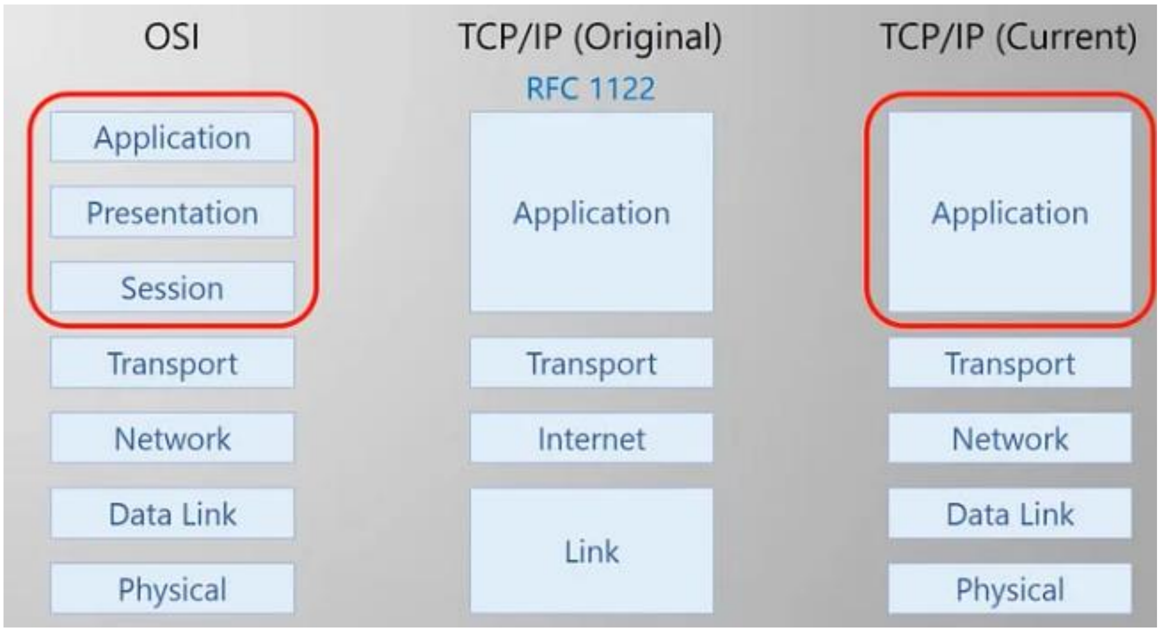
Networking Stack/Model:

The Network Stack is what allows the applications to be able to access a network through a physical networking device. Networking devices can be modems, cable modems, ISDN, Wi-Fi devices, Ethernet cards, Token Ring cards, etc.

The OSI model divides the network interconnection framework into the following 7 layers.

- L7- Application: Responsible for providing a unified interface for applications.
- L8- Presentation: Responsible for converting the data into a format compatible with the receiving system.
- L-5 Session: Responsible for maintaining communication connections between hosts.
- L-4 Transport: Responsible for adding a transport header to data to form a data packet. **port**
- L-3 Network: Responsible for the routing and forwarding of data. **IP**
- L-2 Data link: Responsible for MAC addressing, error detection and correction. **MAC address**.
- L-1 Physical: Responsible for transmitting data frames in the physical network. **Wifi cable**

The TCP/IP network model has five layer model (old model is four layers).



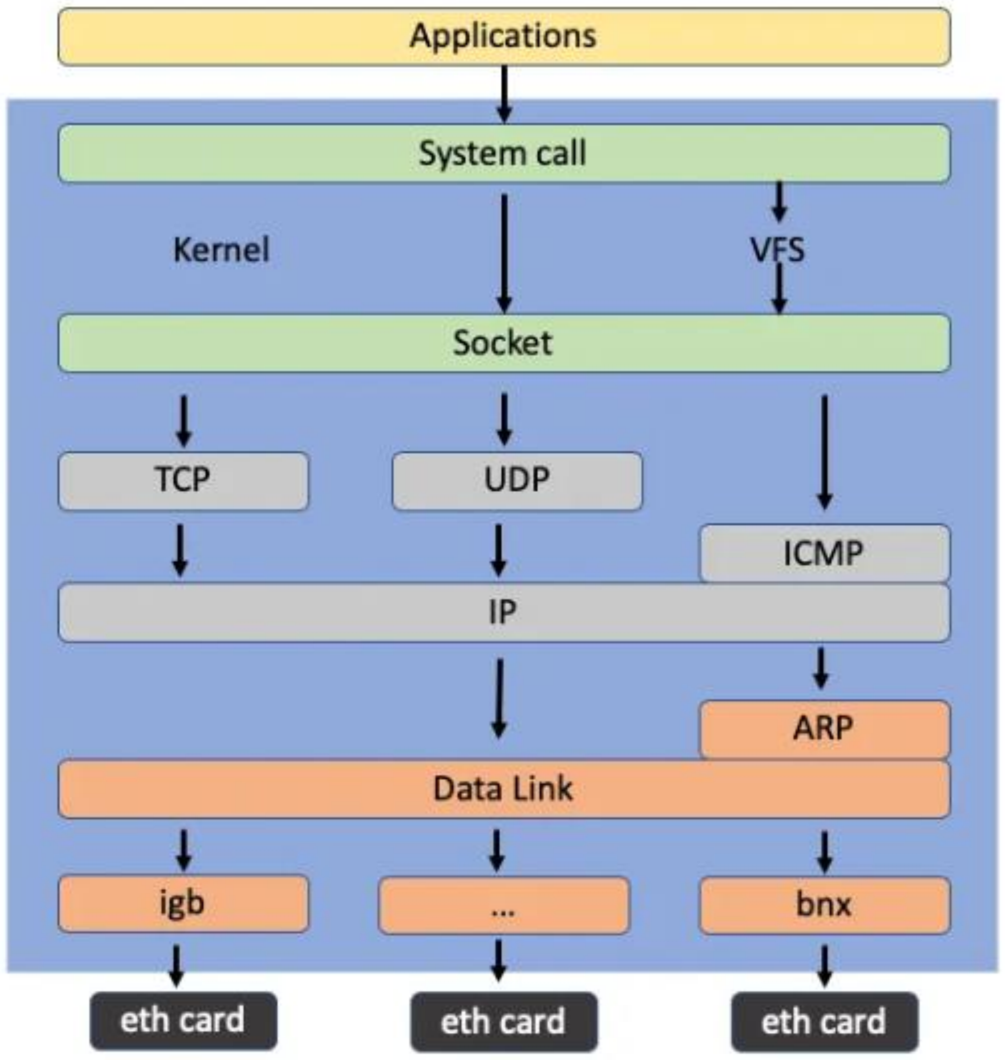
For example,

at application layer, an application that provides a REST API can use the HTTP protocol, encapsulate the JSON data it needs to transmit into HTTP protocol, then pass it down to the TCP layer.

The maximum transmission unit (MTU) configured on the network interface specifies the maximum IP packet size. The default MTU is usually 1500 bytes.

Once the packet exceeds the MTU size, it will be fragmented at network layer to ensure that the fragmented IP packet is not larger than MTU.After understanding the TCP/IP network model and the encapsulation principle of network packets, you can easily think that the network stack in the Linux kernel is actually similar to the five-layer structure of TCP/IP.

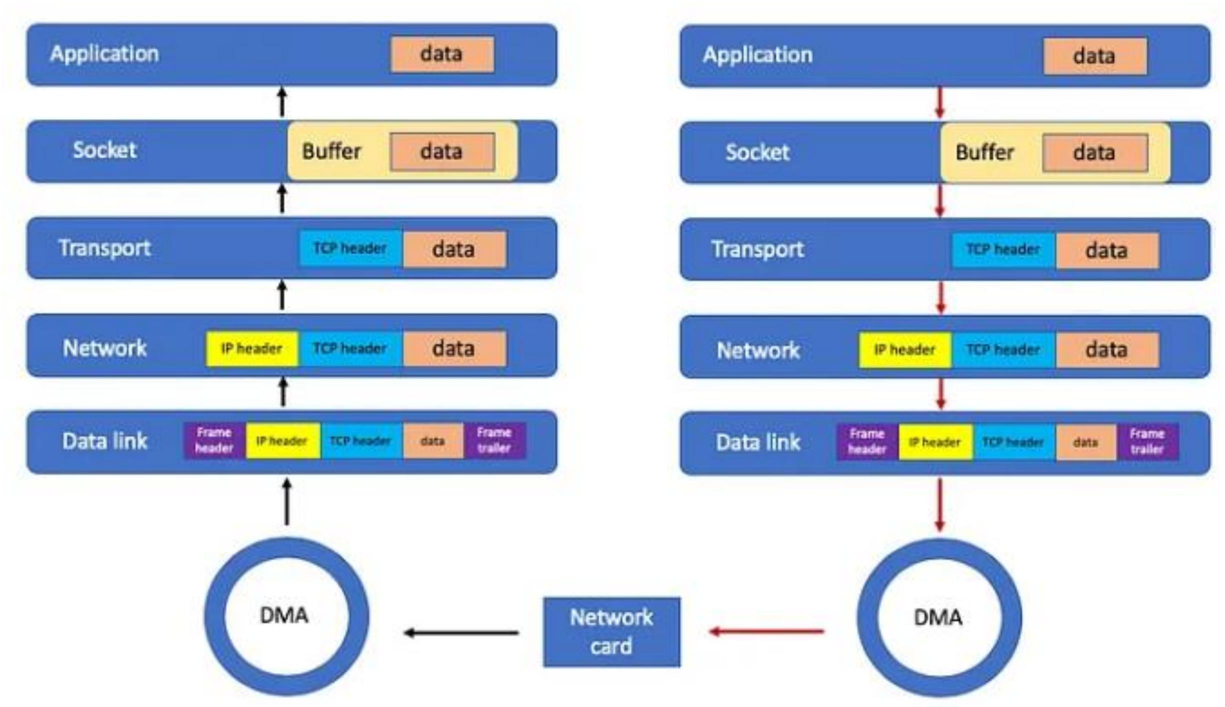
Linux general IP network stack



- The top level application needs to interact with the socket interface through system calls
- Below the socket is the transport layer, network layer
- The bottom layer is the network card driver and the physical network card device

A network card is the basic device for sending and receiving network packets. During system startup, the network card is registered with the system through the network card driver in the kernel. In the process of network transmission and reception, the kernel interacts with the network card through interrupts.

Sending Network Packets :



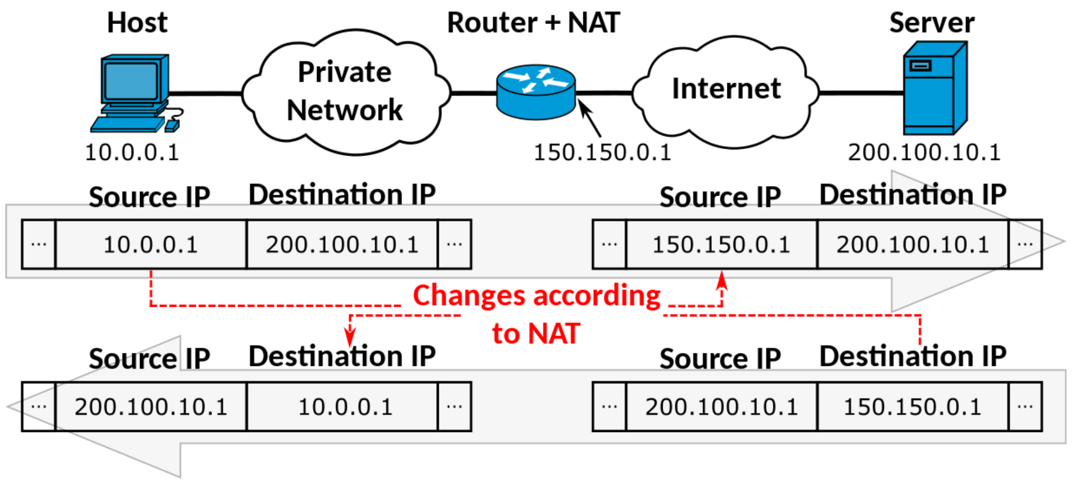
a) First, the **application calls socket API to send network packets**
b) Since this is a system call, it will be **trapped in the socket layer of kernel mode**. The socket layer will put the data packet into the socket send buffer
c) Next, **the network stack takes out the data packet from the socket send buffer**, and then process is layer by layer according to TCP/IP stack
d) The **transport layer adds TCP header** to the packet
e) The network layer **adds IP header to the packet**, and **perform fragmentation according** to the MTU size
f) The **fragmented network packets are sent to data link layer**, which addressing for MAC address and add the frame header and tailer, the the frame will be put into the sending queue
g) Then data link layer will trigger a **soft interrupt notify the network card driver** that there are new network frames in the packet queue
h) The network card driver reads the network frame from the packet sending queue through DMA and sends it through the physical network card

*** **port** is used to “**bind**” with “**process**” : Process >> bind >> Port
*** If the number of port is **existed**(>64000) then the server will drop packet
*** **routing table** is used to **send packet to “interface”**

##############################

NAT(Network Address Translation) :

- NAT stands for network address translation.
- It's a way to map multiple private addresses inside a local network to a public IP address before transferring the information onto the internet.
- Network address translation (NAT) is a method of **mapping an IP address space into another** by **modifying network address information in the IP header of packets** while they are in transit across a traffic routing device.



Network Address Translation (NAT) Types –

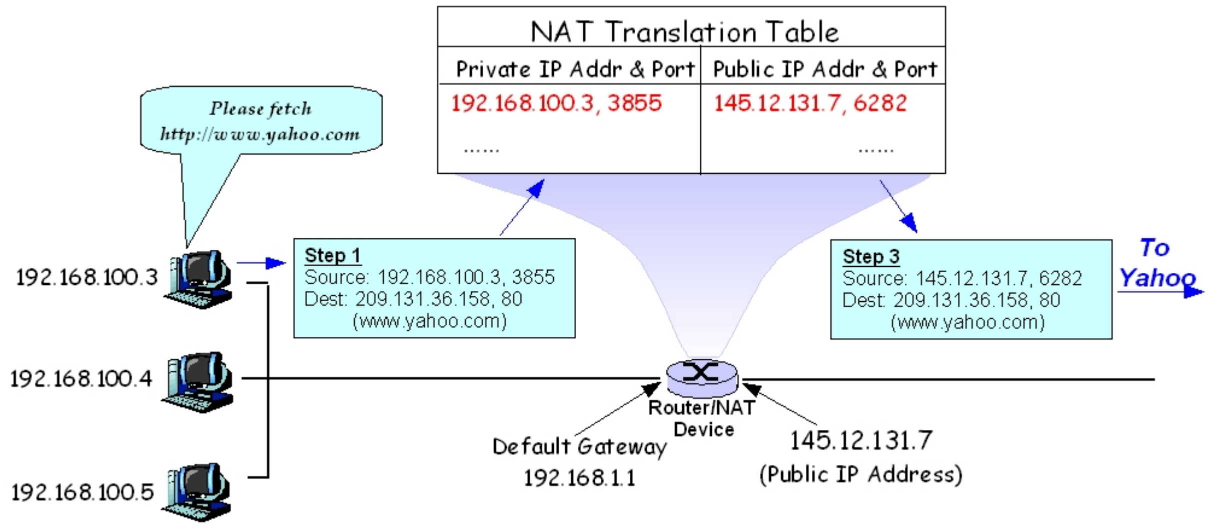
There are 3 ways to configure NAT:

a) Static NAT
In this, a single unregistered (Private) IP address is mapped with a legally registered (Public) IP address i.e one-to-one mapping between local and global addresses. This is generally used for Web hosting. These are not used in organizations as there are many devices that will need Internet access and to provide Internet access, a public IP address is needed.
Suppose, if there are 3000 devices that need access to the Internet, the organization has to buy 3000 public addresses that will be very costly.

b) Dynamic NAT
In this type of NAT, an unregistered IP address is translated into a registered (Public) IP address from a pool of public IP addresses. If the IP address of the pool is not free, then the packet will be dropped as only a fixed number of private IP addresses can be translated to public addresses.
Suppose, if there is a pool of 2 public IP addresses then only 2 private IP addresses can be translated at a given time. If 3rd private IP address wants to access the Internet then the packet will be dropped therefore many private IP addresses are mapped to a pool of public IP addresses. NAT is used when the number of users who want to access the Internet is fixed. This is also very costly as the organization has to buy many global IP addresses to make a pool.

c) Port Address Translation (PAT)
This is also known as NAT overload. In this, many local (private) IP addresses can be translated to a single registered IP address. Port numbers are used to distinguish the traffic i.e., which traffic

belongs to which IP address. This is most frequently used as it is cost-effective as thousands of users can be connected to the Internet by using only one real global (public) IP address.


NAT Translation Table
Private IP Addr & Port
Public IP Addr & Port
192.168.100.3, 3855
145.12.131.7, 6282
Please fetch http://www.yahoo.com
192.168.100.3
192.168.100.4
192.168.100.5
Step 1 Source: 192.168.100.3, 3855 Dest: 209.131.36.158, 80 (www.yahoo.com)
Step 3 Source: 145.12.131.7, 6282 Dest: 209.131.36.158, 80 (www.yahoo.com)
To Yahoo
Router/NAT Device
Default Gateway 192.168.1.1
145.12.131.7 (Public IP Address)

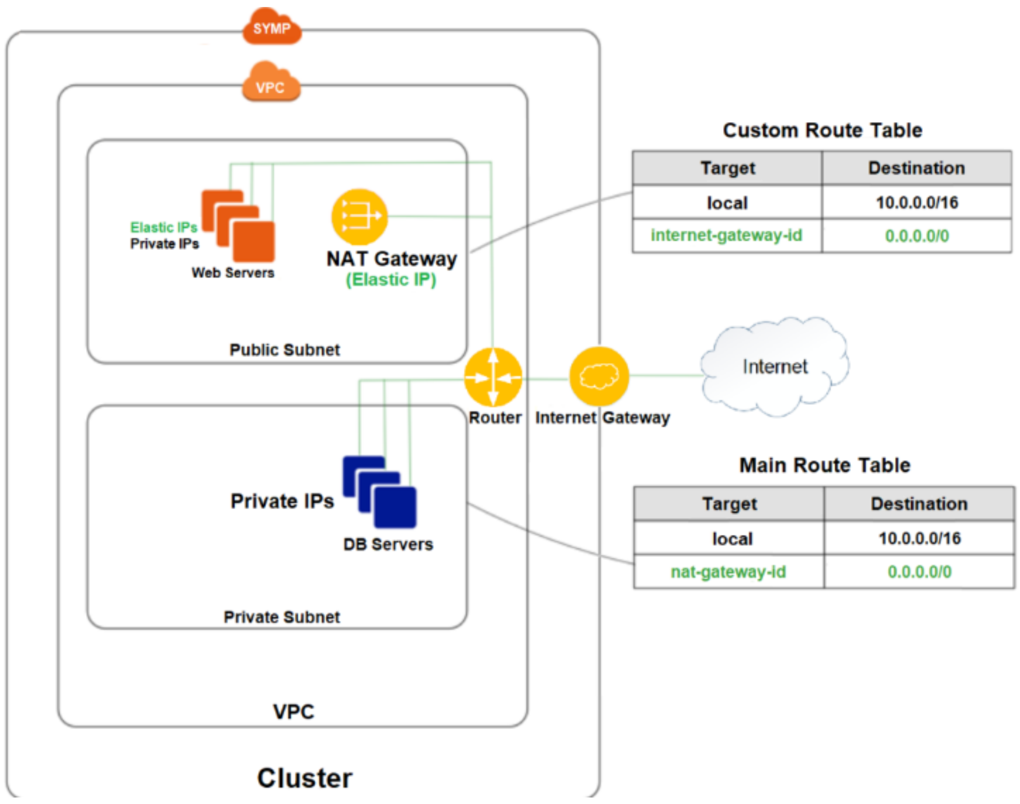
NAT Gateways:

- A NAT gateway is a Network Address Translation (NAT) service.
- You can use a NAT gateway so that instances in a **private subnet can connect to services outside your VPC** but **external services cannot initiate a connection with those instances**.

When you create a NAT gateway, you specify one of the following connectivity types:

a) Public – (Default)
- Instances in private subnets **can connect to the internet** through a public NAT gateway, but cannot receive unsolicited inbound connections from the internet.
- You create a public NAT gateway in a public subnet and **must associate an elastic IP address** with the NAT gateway at creation. You route traffic from the NAT gateway to the internet gateway for the VPC. Alternatively, you can use a public NAT gateway to connect to other VPCs or your on-premises network. In this case, you route traffic from the NAT gateway through a transit gateway or a virtual private gateway.

b) Private
- Instances in private subnets **can connect to other VPCs or your on-premises network** through a private NAT gateway.
- You can route traffic from the NAT gateway through a transit gateway or a virtual private gateway. You **cannot associate an elastic IP address** with a private NAT gateway. You can attach an internet gateway to a VPC with a private NAT gateway, but if you route traffic from the private NAT gateway to the internet gateway, the **internet gateway drops the traffic**.



Custom Route Table

Target	Destination
local	10.0.0.0/16
internet-gateway-id	0.0.0.0/0

Main Route Table

Target	Destination
local	10.0.0.0/16
nat-gateway-id	0.0.0.0/0

############################