ETHERNET AND NETWORK INTERFACE :

1. Every ethernet device is represented in the form of network interface for user space applications (eth0, eth1)
2. These interfaces must be registered via net\_device structure in kernel networking stack.
3. Thereby, kernel interacts with these with the help of sockets, driver api etc.
4. As per OSI layers,

* Physical layer -> NIC card acts as hardware doing encoding, sending and receiving data
* Data link layer -> Ethernet driver are defined under drivers/net/ethernet/ in linux which handles interrupts during transmission and reception, sending data from kernel memory to NIC card and vice-versa , interacts with kernel for all networking related functions of that particular device like registering, unregistering, getting properties of that device etc.
* Network layer -> IP is defined under net/ipv4/ip\_<output/input>.c
* Transport layer -> net/ipv4/<tcp/udp>.c
* Application layer -> ping , ssh etc via socket

1. Network device , drivers and interfaces are defined as below:

* Network device -> physical or virtual hardware component like ethernet NIC or virtual bridges etc. it will be appearing in the form of net\_device structure in linux kernel with all required properties like MAC , state (up/down) , etc.
* Network driver -> software component that interacts with kernel and network device and provides API for transmission, reception interrupt handling etc. registers the device and allocates memory for that in kernel. It has callback functions representing the operations of the device like ndo\_open() , ndo\_start\_xmit() , stop etc.
* Network interface -> logical representation of network device that user space applications can make use of to interact.

1. Linux has Kernel Networking Stack that completely takes over network communication

* It defines all the program (.c) kernel files for each OSI layers discussed above.
* High level user application level overview:

1. User space applications call socket implicitly (via ping or ssh) or explicitly
2. Socket structure will be allocated by KNS. Socket is the endpoint of communication between two processes. Three types of sockets are there. Stream socket works for connection oriented TCP , datagram socket for UDP and raw socket for custom packets or ICMP packets.
3. Process starts by calling or creating socket in user space. That creates socket structure which will be pointed to another structure holding protocol info (tcp, udp, ipv4 etc) , addresses (ip and mac) , socket buffer headers etc. (called binding)
4. Made to listen (in case of server) and accepts incoming client request
5. Sends and receives data using socket.
6. Closes the socket.

* KNS Handles packet flow as follows:

1. User space application calls socket
2. KNS creates associated structures for the socket call
3. Once socket buffer is initialized. Application data will be placed on it and followed by, all layers below will place their information in encapsulated form (tcp\_hdr , tcp\_sendmsg, udp\_sendmsg, ip\_output)
4. In network layer, kernel finds the route to default gateway and calls dev\_queue\_xmit() in its memory.
5. Network device driver via DMA access kernel memory and takes the data to NIC for transmission after adding data link layer header and trailer and it calls ndo\_start\_xmit() for NIC card to serialize data and encodes it and transmits as signal or wave according to the medium.
6. ethtool package is used in linux to handle ethernet interfaces and tune its parameters.
7. Ethernet (IEEE 802.3) frame format is as follows:

* Preamble – 7 bytes (series of 1 and 0)
* Start of frame – 1 byte (as per framing protocol)
* Destination Mac – 48 bits
* Source mac – 48 bits
* Length or type – 2 bytes (length and type encapsulation)
* Payload – 46 to 1500 bytes (considering jamming scenario, if not length encapsulation required)
* FCS – frame check sequence – 4 bytes (checksum for ethernet)

1. Supports basic 802.3 standard and also 802.1q (for vlan traffic), 802.1p (QoS – prioritization ) , 802.3af (power over ethernet) , 802.3d (link aggregation)
2. Quality of service in ethernet reduces congestion, utilizes the network bandwidth in efficient manner by prioritizing the ethernet frames. For this, 802.1p supports 3 bits as priority bits along with vlan tag to indicate 8 levels of priority (normal, tcp, udp, bulk, criticial, control and emergency frames) and also by traffic shaping in transmission side. It follows queueing disciplines (bandwidth sharing based on application, priority etc)
3. Link aggregation supports bonding many virtual and / or physical interfaces for improved bandwidth , load balancing and redundant paths for data in case one interface fails.
4. Linux KNS also defines CMSA-CD for half duplex ethernet to detect collision in the shared media especially bus topology (modern day full duplex start topology methods eliminates the basic need of csma-cd). It defines 1-persistent csma where in greedy approach, it continuously senses the channel and if found idle, immediately starts data transmission. However, chances of having collision in this case is higher than N- persistent csma which waits for some random time if channel if found busy and then senses again. If collision is detected by sender, it immediately abrupts the data transmission and sends jamming signal to notify all other participants to stop transmission. Now sender waits for random period (defined by exponential back-off algorithm with 51.2uS \* k where k ranges from 0 to 2^(n-1) where n = number of persistent collisions)