LINUX NETWORKING MODULE 3 AND 4 ASSESSMENT SOLUTIONS

-BY SAKTHI KUMAR S

8. Research the Linux kernel's handling of Ethernet devices and network interfaces. Write a short report on how the Linux kernel supports Ethernet communication (referencing kernel.org documentation).

The Linux network stack follows a layered architecture, where Ethernet operates at Layer 2 (Data Link Layer). The net_device structure represents a network interface, managed by the kernel.

Linux Kernel's Handling of Ethernet Devices

1. Ethernet Device Management in Linux Kernel

Ethernet devices in Linux are managed through the **Network Device Interface (net_device structure)** in the kernel. The process of handling Ethernet devices involves the following layers:

- **Network Interface Layer**: Manages network interfaces such as eth0, eth1, and wlan0.
- **Device Driver Layer**: Communicates with the hardware, handles data transmission and reception.
- **Networking Protocol Stack**: Supports TCP/IP, UDP, and other networking standards.
- **User Space Tools**: Commands like ip, ifconfig, and ethtool allow users to manage network interfaces.

2. Key Kernel Components for Ethernet Communication

- **net_device Structure**: The core data structure representing a network device. It defines attributes such as interface name, MAC address, and supported protocols.
- **Network Queue Disciplines (Qdisc)**: Manages packet scheduling.
- **SKB (Socket Buffers)**: The fundamental data structure used to store network packets.
- **Device Drivers**: Interact with physical NIC (Network Interface Card) and implement functions like:
    - ndo_open(): Initializes the network device.
    - ndo_start_xmit(): Handles packet transmission.
    - ndo_stop(): Stops the device when not in use.

3. Packet Transmission Flow

1. A user application sends data via a socket (send() system call).
2. Data is encapsulated into packets at the transport layer (TCP/UDP).

3. The network layer (IP) processes the packets. The IP layer adds an IP header to the data.
4. The Ethernet driver at Layer 2 (Data Link Layer) encapsulates it into an Ethernet frame.
5. The kernel networking stack places the frame in a network transmit queue (managed by qdisc scheduler).
6. The NIC driver fetches the frame using ndo_start_xmit().The driver sends the packet through the NIC to the Ethernet network.

4. Packet Reception Flow

1. The NIC receives data from the network. The NIC detects an incoming Ethernet frame, validates the MAC address, and checks CRC integrity.
2. The NIC places the frame in a Receive (RX) buffer using DMA (Direct Memory Access).
3. An interrupt (IRQ) is triggered to notify the Linux kernel. The driver processes the data and stores it in an SKB (Socket Buffer).
4. The kernel passes the packet to the network stack. The network driver reads the frame from the buffer using netif_rx().
5. The network stack inspects the EtherType field to determine the next protocol:
   - IPv4/IPv6 → Sent to IP layer
   - ARP → Handled at Layer 2
   - Other Protocols → Forwarded accordingly
6. If it's an IP packet, it is processed by the TCP/UDP stack. The transport layer (TCP/UDP) processes the data.
7. The kernel then forwards it to the user application via sockets (recv()) and application retrieves the data using system calls (recv()).

5.Kernel APIs for Ethernet Networking

- Important kernel APIs for Ethernet management:
  - **netdev_alloc_skb()** – Allocates socket buffers for packets.
  - **netif_rx()** – Passes received packets to the network stack.
  - **netif_carrier_on()/off()** – Manages link state.

6. Kernel Modules and Offloading

- Supports **modular network drivers** via modprobe and insmod.
- Enables **hardware offloading** (checksum, segmentation, GRO, TSO) for performance.

This is how the Linux kernel supports Ethernet communication