Assignment 7 : Design and Implement a Lightweight Custom Discovery Protocol (LCDP) using Raw Sockets in POSIX C

Deadline: March 31, 2025 EOD

In this assignment, you need to implement a **Custom Lightweight Discovery Protocol (CLDP)** using raw sockets in C (POSIX standard). The protocol is intended for a closed network environment where nodes announce their presence and query other nodes for specific application-level metadata (e.g., CPU load, system time, or hostname). You must implement both the *client* and the *server* in raw sockets, constructing and parsing packets manually at the IP level, using a custom protocol number.

Protocol Specification (Design Phase)

You must define the CLDP packet format. At minimum, your custom protocol should support the following message types:

- `0x01: HELLO` (used to announce the node is active)

- `0x02: QUERY` (request for metadata, such metadata can be any application-level performance counters, such as CPU Load, System Time, Hostname, etc. Your implementation should support at least THREE such metadata queries. Write in your documentation about which metadata are supported in your implementation.)
- `0x03: RESPONSE` (response with metadata)

Each message should include:
- Custom header (min 8 bytes): [Message Type, Payload Length, Transaction ID, Reserved]
- Optional payload based on type (e.g., hostname string, system time, etc.)

You need to submit a document specifying the protocol format along with the design details.

Raw Socket Implementation:

Implement both a **CLDP Server** and **CLDP Client**:
- The server listens for incoming CLDP packets, processes requests, and sends back responses using raw sockets.
- The client constructs CLDP packets manually, using raw sockets to send HELLO and QUERY messages to a known IP (e.g., broadcast or specific IP).

Packet Crafting & Parsing:

Manually craft and parse IP packets with your custom payload:
- Set appropriate fields in the IP header (e.g., protocol number = 253)
- Do not use transport-layer protocols like TCP or UDP
- Include checksum calculation for the IP header (if required)
- Parse incoming packets and extract only those matching your protocol number

Demonstration Scenario:

Write a demo program where:
- Multiple servers announce themselves via HELLO every 10 seconds
- A client sends a QUERY to all active nodes asking for their hostname and timestamp
- Servers respond with RESPONSE packets containing the requested information

Please maintain the following while implementing your protocol.

- Raw socket programming only (`AF_INET`, `SOCK_RAW`), do not use existing transport-layer protocols (UDP, TCP)
- The solution should run over a standard Linux OS, such as the one available at the software lab.
- Do not use `libpcap` or similar libraries; stick to `sendto()` and `recvfrom()` with raw sockets
- Consider using `setsockopt()` to set `IP_HDRINCL` for crafting IP headers
- Use `gethostname()`, `gettimeofday()`, or `sysinfo()` for metadata
- Run your programs with `sudo` since raw sockets require elevated privileges
- Validate your implementation by capturing traffic with `tcpdump` or `Wireshark`

Deliverables:

- Protocol Specification Document (`cldp_spec.pdf`)
- Source Code:
  - cldp_server.c
  - cldp_client.c
- README with:
  - Build and run instructions
  - Any assumptions or limitations
  - Demo Output (text capture or script)