Problem Statement 10
Hyperfast DNS Load Balancer

Reference: RFC 1034, 1035, 2308, 8198, 7766, 6891, 9715, 7858, 8484, 9250 etc.

Background

DNS is the **front door of the Internet**, resolving billions of queries per second. Modern applications and CDNs depend on DNS load balancing to distribute traffic across multiple servers, regions, and networks. However, traditional DNS load balancers face **latency, scalability, and efficiency bottlenecks**, especially under high query-per-second (QPS) conditions or during DDoS bursts.
A **Hyperfast DNS Load Balancer** is envisioned as a next-generation system capable of **sub-millisecond query handling**, **smart traffic steering**, and **resilience at scale**. It should integrate **Anycast routing, real-time telemetry, and AI/ML-based decision logic** for adaptive load distribution.

Problem Statement

Design and implement a **prototype Hyperfast DNS Load Balancer** that can:

1. **Process DNS queries at scale** (≥ 1M QPS in simulation) with low latency.
2. **Distribute load intelligently** across multiple authoritative backends or edge nodes.
3. **Incorporate adaptive policies** (e.g., geo-location, latency measurements, health checks).
4. **Demonstrate resilience** under stress conditions such as flash crowds or simulated DDoS.
5. Optionally, support **encrypted DNS protocols (DoH/DoT/DoQ)** while preserving speed.

Key Tasks

- Build a **high-performance DNS query engine** using efficient libraries or kernel-bypass techniques (e.g., DPDK, eBPF, XDP).
- Implement **load balancing strategies**: round-robin, latency-based, weighted, and geo-aware.
- Integrate **health monitoring** of backends and failover mechanisms.
- Benchmark query response latency and throughput under increasing load.
- Compare performance against a baseline (e.g., standard DNS load balancer like NGINX DNS or PowerDNS recursor).

Deliverables

- A working **prototype DNS load balancer** (code + configuration).
- **Performance benchmarking results** (QPS, latency, failover recovery).
- Documentation of architecture, algorithms used, and scalability considerations.
- A **demo scenario** (e.g., global web service served via multiple DNS backends).