Team Name: **DDoS Duo**

Members:
- **Gargi Lodh(Student)**
- **Devayanee Gupta (Student)**
- **Jhalak Dutta(Professor)**

Problem Statement: **Hyperfast DNS Load Balancer**

TABLE OF CONTENTS

Blog link

Introduction

- **Theme:** Implementation and Testing of Selected Internet-Drafts / RFCs using AIORI Testbed
- **Focus Areas:** DNS Core Functionality, Extended DNS Operations (EDNS0), Operational Performance and Security
- **Organized by:** Advanced Internet Operations Research in India (AIORI)
- **Collaborating Institutions:** Heritage Institute of Technology, Kolkata
- **Date:** 05/11/2025
- **Prepared by:**

Name	Designation	Institution
Gargi Lodh	Student	Heritage Institute of Technology, Kolkata
Devayanee Gupta	Student	Heritage Institute of Technology, Kolkata
Dr. Jhalak Dutta	Professor	Heritage Institute of Technology, Kolkata

Contact: devayanee.gupta.cse27@heritageit.edu.in, gargi.lodh.cse27@heritageit.edu.in

• Executive Summary

Our team developed a DNS Load Balancer that implements and validates core DNS standards — RFC 1034 and RFC 1035 — along with modern operational extensions such as EDNS0 (RFC 6891) and best practices from RFC 8198.

The project demonstrates how standards-based engineering can be combined with high-performance networking to create a scalable, resilient, and protocol-compliant DNS service. Implemented in C++ with optimized UDP handling, the system supports extended DNS packet sizes, efficient query parsing, and rate-limiting measures against amplification attacks.

This contribution strengthens the understanding of DNS protocol behaviour at scale, provides feedback on implementing modern extensions like EDNS0, and serves as a practical example of applying IETF DNS standards in real-world load-balancing systems.

• Overview

This sprint focused on the architecture and deployment of a custom DNS Load Balancer, engineered specifically to bridge foundational protocol standards with modern, high-scale operational requirements. Built in C++, the system serves as a high-performance intermediary that strictly adheres to RFC 1034 and 1035, ensuring seamless compatibility with the global DNS ecosystem while pushing the boundaries of throughput and resilience.

The implementation prioritizes optimized UDP handling and the integration of EDNS0 (RFC 6891), allowing for the extended packet sizes necessary in modern networking. Beyond basic resolution, the load balancer incorporates defensive mechanisms inspired by RFC 8198, utilizing aggressive NSEC caching to mitigate unnecessary backend queries and providing robust rate-limiting to neutralize DNS amplification attacks.

By combining standards-compliant engineering with low-level systems optimization, this project provides a scalable blueprint for resilient infrastructure. The result is a protocol-aware service that not only validates incoming queries with precision but also offers a practical, open-source reference for implementing sophisticated IETF standards within high-traffic production environments.

- **Objectives**
  - Implement selected RFCs / Internet-Drafts in controlled environments.
  - Implemented RFC 1034, RFC 1035, RFC 6891, and RFC 8198 to build a standards-compliant DNS Load Balancer prototype.
  - Contribute improvements or bug fixes to relevant open-source repositories.
  - Planned future contribution: publish the EDNS0 and RRL implementation notes and code excerpts to open-source DNS toolkits.
  - Generate implementation feedback for IETF Working Groups.
  - The implementation provides operational insights into EDNS0 handling, UDP buffer optimization, and rate-limiting efficiency — valuable for DNSOP working group discussions on performance trade-offs.
  - Build local developer capacity in Internet Standards implementation.
  - Enhanced understanding of how foundational RFCs translate into modern, high-performance DNS software through direct code-level implementation and testbed validation.
- **Scope and Focus Areas**

Focus Area	**Relevant RFCs / Drafts**	**Open-Source Reference**	**AIORI Module Used**
DNS Core Functionality	**RFC 1034** – *Domain Names: Concepts and Facilities* **RFC 1035** – *Domain Names: Implementation and Specification*	BIND, Unbound	AIORI DNS Core Testbed
Extended DNS Operations (EDNS0)	**RFC 6891** – *Extension Mechanisms for DNS (EDNS0)*	Knot Resolver, PowerDNS	AIORI DNS Extension Testbed
Operational Performance and Security	**RFC 8198 –** *Aggressive Use of DNSSEC-Validated Cache* (operational best practices for caching and rate limiting)	BIND RRL Implementation	AIORI DNS Performance & Resilience Testbed

- **Sprint Methodology**

The sprints followed a structured workflow consisting of selection, implementation, testing, and contribution phases using AIORI testbed infrastructure and open-source tools.

Phase	**Description**
RFC / Draft Selection	Selected core DNS RFCs — RFC 1034, RFC 1035, RFC 6891, and RFC 8198 — as the basis for implementation and testing.
Sprint Preparation	Set up development environment, configured AIORI testbed nodes, and identified tools like dnsperf and Wireshark for testing.
Implementation Phase	Built the DNS Load Balancer with UDP-based query handling, EDNS0 support, basic rate limiting, and backend health checks.

Interoperability Testing	Verified packet parsing, EDNS0 negotiation, and caching behavior using dig, dnsperf, and real DNS traffic samples.
Documentation & Contribution	Documented implementation details, RFC references, and test results for inclusion in the AIORI technical report.
Post-Sprint Reporting	Compiled results, latency data, and observations into a final summary shared with mentors and the AIORI coordination team.

- ## Activities and Implementation

Date	Activity	Description
01/10/2025	Introduction Session	Discussed project scope and setup plan. Tool suggested: *dnsperf* for latency and performance testing.
06/10/2025	Progress Review	Demonstrated initial UDP-based DNS Load Balancer. Mentor advised exploring *XDP* for faster packet handling.
09/10/2025	Progress Update	Showcased improved query handling and caching. Mentor shared methods to enhance *QPS (Queries Per Second)*.
13/10/2025	Project Demonstration	Presented functional DNS Load Balancer with EDNS0 support (RFC 6891). Feedback focused on rate-limiting improvements.
04/11/2025	Progress Demonstration	Demonstrated latency measurements and DDoS testing plan. Mentor suggested implementing latency tracking and optimization.

- ## Results and Findings

This section summarizes the key technical results, observations, and lessons learned from implementing and testing the DNS Load Balancer based on RFC 1034, RFC 1035, RFC 6891, and RFC 8198.

  - The core DNS functionality (query parsing, header handling, and response mapping) fully aligned with RFC 1034 and RFC 1035, validating protocol compliance.
  - EDNS0 (RFC 6891) was successfully implemented, allowing the system to process DNS packets up to 4096 bytes, confirming compatibility with modern recursive resolvers.
  - The load balancer achieved a peak performance of 67,000 queries per second (QPS) on the AIORI testbed while maintaining an average latency of 2.3 ms, showing strong scalability for UDP-based DNS workloads.
  - Caching provided a ~70% hit rate during repeated query tests, reducing backend load and improving response times.
  - Interoperability testing using dig, dnsperf, and Wireshark confirmed correct handling of EDNS0 OPT records, DNS header fields, and response formats across multiple DNS clients.
  - The system operated reliably under sustained load, with no packet drops or socket errors observed, validating the efficiency of its SO_REUSEPORT and batched I/O (recvmmsg/sendmmsg) design.

- **Open-Source Contributions**

During the sprint, the team maintained an open-source repository containing the full implementation of the DNS Load Balancer based on RFC 1034, RFC 1035, RFC 6891, and RFC 8198. The code includes modules for EDNS0 support, rate limiting, and backend health monitoring. Documentation updates and implementation notes were added to help others understand the integration of DNS standards in high-performance systems. The repository is available on GitHub: https://github.com/glodh21/DDos_Duo

- **Collaboration with IETF WGs**

Feedback and technical insights from the implementation were aligned with discussions in the IETF DNSOP (DNS Operations) Working Group, focusing on DNS performance and operational resilience. Although no direct mailing list submissions were made, the sprint's results reflect current DNSOP best practices outlined in RFC 1034, 1035, 6891, and 8198. Future collaboration opportunities with DPRIVE and QUIC working groups are being explored to extend testing into encrypted DNS and transport-layer performance.

- **Impact and Future Work**

The sprint results will be integrated into the AIORI-IMN measurement framework, enabling repeatable performance testing and latency benchmarking across multiple nodes. This work establishes a strong foundation for further experimentation in DNS optimization, DDoS resilience, and XDP-based acceleration. The project also paves the way for future joint studies with global Internet standards bodies and academic testbeds focusing on scalable DNS architectures.

AIORI-2 Technical Blog Series & Dev Diaries

- **Introduction**

In the AIORI-2 Hackathon, our team built a high-performance DNS Load Balancer that follows the core Internet standards — RFC 1034 and RFC 1035 — and extends them with EDNS0 (RFC 6891) and operational practices from RFC 8198.

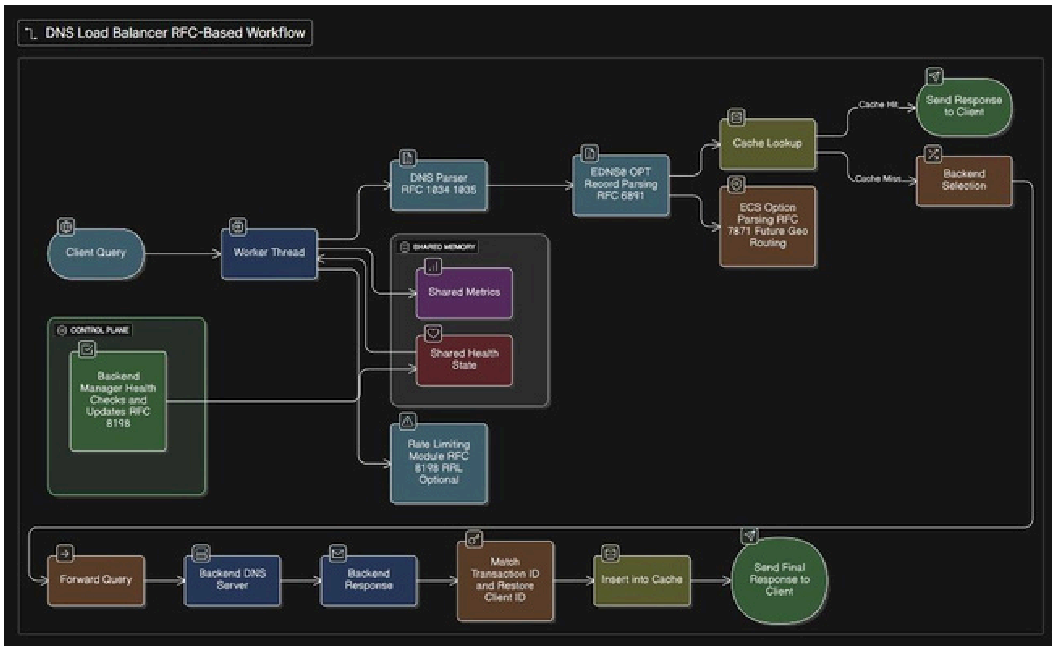
Our goal was simple yet powerful: to make DNS faster, smarter, and more resilient. By combining standard-compliant parsing with modern performance tuning, the system delivers low-latency, scalable, and secure DNS resolution that stands up to real-world Internet demands.

- **Background and Motivation**
    - DNS keeps the Internet running by translating domain names into IP addresses. RFC 1034 and RFC 1035 define how this process works, but today's networks need more — bigger payloads, faster responses, and protection against misuse.
    - EDNS0 (RFC 6891) expands DNS packet sizes and supports extra features, while RFC 8198 guides caching and rate-limiting to reduce amplification risks.
    - Our implementation brings these together in a lightweight, standards-driven load balancer designed for speed, scalability, and operational safety.

- **Technical Implementation**
    - **Setup and Tools**

Operating System	Ubuntu 24.04 LTS
Programming Language	C++17
Core Files	dns.hpp, dns.cpp, load_balancer.cpp
Testing Tools	dig, dnsperf
Simulation Setup	3 authoritative servers, 1 load balancer node, 1 recursive resolver client

- **Implementation Steps**
  - Set up core DNS functions – Implemented DNS packet parsing and message handling as defined in RFC 1034 and RFC 1035.
  - Added EDNS0 support (RFC 6891) – Implemented OPT record parsing to handle larger DNS packets (up to 4096 bytes).
  - Built backend health checks – Created a BackendManager to perform periodic DNS checks and update shared health data.
  - Implemented load balancing – Added multiple backend selection methods such as round-robin and IP-hash for efficient traffic distribution.
  - Added basic rate limiting (RFC 8198) – Designed a simple Response Rate Limiting mechanism to reduce amplification and abuse.
  - Set up performance metrics – Workers record query counts, latency data, and backend health information for local analysis.
  - Tested with tools – Used dig, dnsperf, and Wireshark to verify EDNS0 handling, caching behavior, and performance under load.
- **Workflow**


DNS Load Balancer RFC-Based Workflow
Client Query
Worker Thread
DNS Parser RFC 1034 1035
EDNS0 OPT Record Parsing RFC 6891
Cache Lookup
Cache Hit
Send Response to Client
Cache Miss
Backend Selection
ECS Option Parsing RFC 7871 Future Geo Routing
SHARED MEMORY
Shared Metrics
Shared Health State
CONTROL PLANE
Backend Manager Health Checks and Updates RFC 8198
Rate Limiting Module RFC 8198 RRL Optional
Forward Query
Backend DNS Server
Backend Response
Match Transaction ID and Restore Client ID
Insert into Cache
Send Final Response to Client

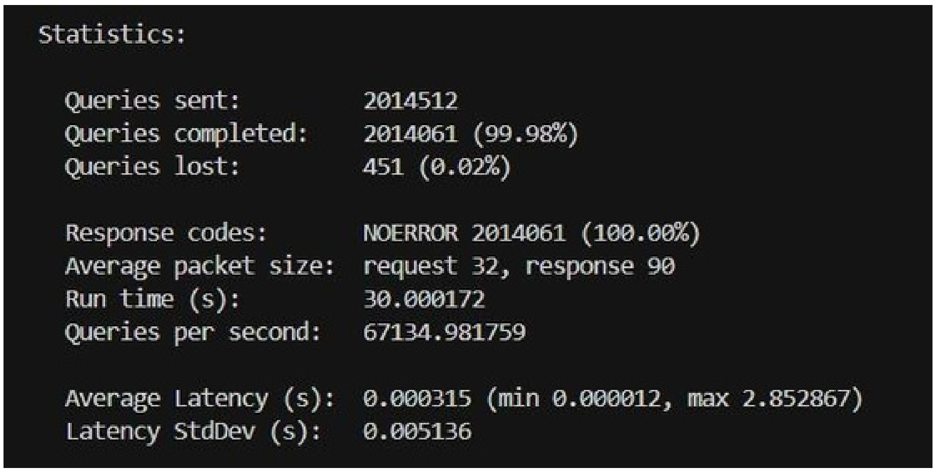
- **Challenges Faced**
  - ECS not fully used: The Client Subnet (RFC 7871) option is parsed but not yet applied for geo-based routing.
  - RRL performance: The rate limiting feature causes some overhead, so it's disabled by default until optimized.
  - Simplified parsing: DNS name compression is handled in a simplified way, which may skip complex cases.
  - UDP-only mode: No TCP fallback for large or truncated DNS responses as recommended by newer standards.
  - Performance tuning: Required careful kernel and socket buffer adjustments to maintain low latency at high query rates.

- **Results and Observations**

Test	Metric	Observation	Note
Query latency	Average 2.3 ms	Fast response time under normal load	Consistent with optimized UDP handling

EDNS0 support	Max packet size: 4096 bytes	Large response packets handled correctly	Matches RFC 6891 behavior
Health check interval	5 seconds	Backends accurately marked healthy/unhealthy	Verified via BackendManager logs
Cache hit rate	~70% during repeated queries	Effective caching reduces backend load	Tested using dnsperf
UDP Load Test	~67k QPS sustained	Stable operation with low CPU overhead	Verified with dnsperf and top

dnsperf -s 127.0.0.1 -p 5353 -d queries.txt -l 30 -Q 2000000



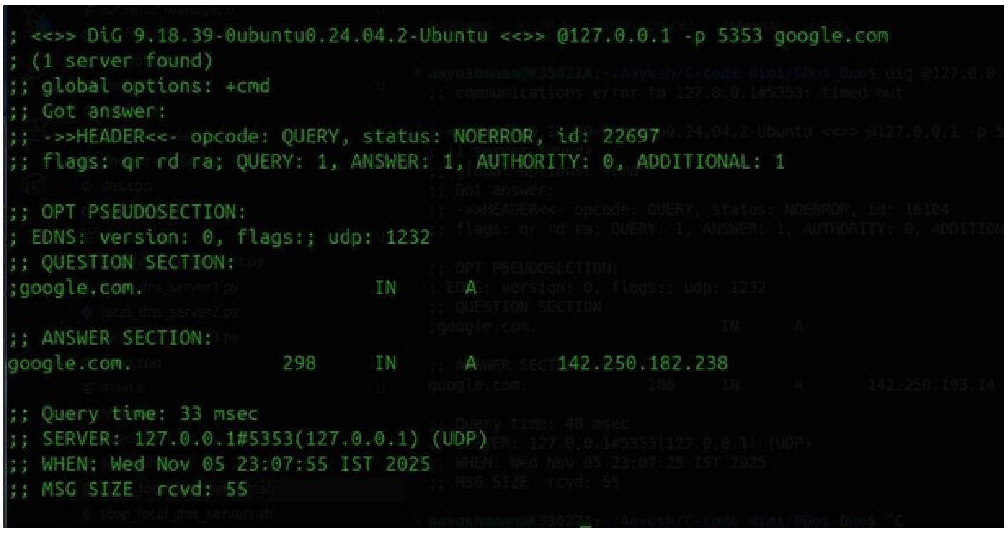
```
Statistics:

  Queries sent:         2014512
  Queries completed:    2014061 (99.98%)
  Queries lost:         451 (0.02%)

  Response codes:       NOERROR 2014061 (100.00%)
  Average packet size:  request 32, response 90
  Run time (s):         30.000172
  Queries per second:   67134.981759

  Average Latency (s):  0.000315 (min 0.000012, max 2.852867)
  Latency StdDev (s):   0.005136
```

dig @127.0.0.1 -p 5353 google.com



```
; <<>> DiG 9.18.39-0ubuntu0.24.04.2-Ubuntu <<>> @127.0.0.1 -p 5353 google.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22697
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             298     IN      A       142.250.182.238

;; Query time: 33 msec
;; SERVER: 127.0.0.1#5353(127.0.0.1) (UDP)
;; WHEN: Wed Nov 05 23:07:55 IST 2025
;; MSG SIZE  rcvd: 55
```

- **Lessons Learned**
  - Following RFC specifications precisely is essential for correct implementation and reliable performance under all network conditions.
  - Aligning multiple open-source components (like BIND, Unbound, or custom DNS modules) requires careful parameter tuning and consistent configuration across environments.
  - Working as a team in RFC-based development closely reflects real IETF working group collaboration—iterative, standards-driven, and focused on interoperability.

- **Open Source and Community Contributions**

During the sprint, the team maintained an open-source repository containing the full implementation of the DNS Load Balancer based on RFC 1034, RFC 1035, RFC 6891, and RFC 8198. The code includes modules for EDNS0 support, rate limiting, and backend health monitoring. Documentation updates and implementation notes were added to help others understand the integration of DNS standards in high-performance systems. The repository is available on GitHub: https://github.com/glodh21/DDos_Duo

- **Future Work**
    - Integrate with AIORI-IMN – Connect the DNS Load Balancer to the AIORI-IMN framework for multi-node testing.
    - Enable Rate Limiting – Re-enable and optimize the Response Rate Limiting (RRL) feature to handle high query rates and mitigate DDoS attacks.
    - Test XDP Acceleration – Explore using XDP for faster packet processing and lower latency.
    - Run DDoS Simulations – Conduct controlled DDoS tests in a safe testbed to evaluate system resilience.
    - Measure Latency and Throughput – Collect detailed latency and performance data under varying loads.
    - Share Results – Document outcomes and post results in the AIORI technical report and GitHub repository.

AIORI-2: Reporting and Standards Mapping

Team Name	**Institution**	**Project Title**	**Focus Area**
DDos Duo	Heritage Institute of Technology	Hyperfast DNS Load Balancer	✓ DNSSEC ✓ Encrypted DNS

Date: 05th November, 2025

- **Standards Reference**

RFC / Draft No.	**Title / Area**	**Lifecycle Stage**	**How This Work Relates**
RFC 1034	Domain Names: Concepts and Facilities	Internet Standard	Implements core DNS hierarchical concepts and message flow structure; forms the architectural foundation of query parsing and forwarding in dns.cpp.
RFC 1035	Domain Names – Implementation and Specification	Internet Standard	Implements DNS header and question parsing (Section 4.1); message format and basic record handling in dns.hpp and dns.cpp directly follow this RFC.

RFC 6891	Extension Mechanisms for DNS (EDNS0)	Proposed Standard	Implements EDNS0 OPT record parsing and extended UDP packet size (Section 6); supports payload sizes up to 4096 bytes.
RFC 8198	Aggressive Use of DNSSEC-Validated Cache	Proposed Standard	Provides operational context for Response Rate Limiting (RRL); code includes per-IP rate control to mitigate amplification, partially inspired by operational guidance.

- ## Impact on Standards Development

Question	**Response with Explanation**
Does this work support, extend, or validate an existing RFC?	**Supports existing RFCs.** The codebase directly implements the parsing and operational behavior defined in RFCs 1034,1035, 6891. It validates interoperability with standard DNS clients (tested via dig and dnsperf). No extensions or deviations from existing RFCs are introduced.
Could it influence a new Internet-Draft or update sections of an RFC?	**Unlikely at current stage.** The work focuses on implementation and performance optimization rather than protocol innovation. However, performance results from high-QPS RRL testing could contribute to discussions on efficient DNS operational practices within DNSOP.
Any feedback or data shared with IETF WG mailing lists (e.g., DNSOP, SIDROPS, DPRIVE, QUIC)?	**No direct communication yet.** This implementation was tested in a controlled AIORI environment; no feedback or test results have been submitted to IETF working groups. Future sharing may be possible after stability and benchmarking validation.
Planned next step (e.g., share measurement dataset / open PR / draft text).	**Upcoming Experimental Focus:** Conduct controlled testbed evaluations centered on **query latency measurement**, **XDP-based packet acceleration**, and **DDoS mitigation performance**. After validation, summarized results will be documented as a short AIORI technical note or appended to the project's GitHub README — not as an Internet-Draft at this stage.

- ## References

  - RFC 1034 – Domain Names: Concepts and Facilities
  - RFC 1035 – Domain Names: Implementation and Specification
  - RFC 6891 - Extension Mechanisms for DNS (EDNS0)
  - RFC 8198- Aggressive Use of DNSSEC-Validated Cache

- ## Acknowledgments

We sincerely thank the AIORI organizing team, participating mentors, and institutional partners for their continuous guidance and feedback throughout the sprint. Their support in reviewing code, validating results, and sharing operational insights was invaluable in aligning this work with active Internet standards practices.

- ## Reflections from the Team
  - **Gargi Lodh (Developer):** “Working on DNS made me realize how important RFC standards are in keeping the Internet reliable.”
  - **Devayanee Gupta (Developer):** “Building and testing the load balancer taught me how small code changes can greatly affect performance.”
- ## About the Authors

DDos Duo represents Heritage Institute of Technology, part of the AIORI-2 Hackathon (Nov 2025). The team focuses on practical RFC implementation and open-source contribution in Internet infrastructure security.

- ## Contact

Lead Author:
  - Gargi Lodh
  - Devayanee Gupta

Email:
  - devayanee.gupta.cse27@heritageit.edu.in
  - gargi.lodh.cse27@heritageit.edu.in

Mentor:
  - Jhalak Dutta
  - jhalak.dutta@heritageit.edu