Key Implementation Details

DNS Cache

The DNSCache class is responsible for storing DNS query results with a TTL (Time to Live). It uses a mutex for thread safety.

DNS Query

The DNSQuery class handles the actual DNS resolution using the Poco library. It supports both normal and recursive DNS queries.

DNS Resolver

The DNSResolver class integrates the cache and query functionalities. It provides options for using the cache, recursive queries, and setting timeouts.

Performance Optimization

- **Caching**: Implemented a DNS cache to store query results and reduce redundant network requests.
- **Thread Safety**: Used mutexes to ensure thread-safe access to the cache.
- **Retry Mechanism**: Added retries for DNS queries to handle transient network issues.
- **Efficient Data Structures**: Used unordered_map for O(1) average-time complexity for cache lookups.

Response Time

- **Asynchronous Queries**: Consider implementing asynchronous DNS queries to improve response time further.
- **Parallel Queries**: For recursive resolution, send parallel queries to multiple root servers.

Resource Utilization

- **Memory Management**: Efficient use of memory by limiting the cache size and periodically cleaning up expired entries.
- **Timeouts**: Configurable timeouts for DNS queries to prevent hanging.

Challenges and Key Decisions

Challenges

- **Handling IDNs**: Ensuring correct resolution of Internationalized Domain Names (IDNs).

  Problems Encountered While Handling Internationalized Domain Names (IDNs) and Their Solutions

  Handling Internationalized Domain Names (IDNs) can be tricky due to the need to correctly encode and decode domain names that contain non-ASCII characters. Here are the key problems encountered and their solutions:

1. **Problem: Incorrect Domain Encoding**
    - **Description**: IDNs need to be encoded using Punycode before they can be used in DNS queries. If the domain name is not correctly encoded, the DNS resolver will fail to find the domain.
    - **Solution**: Use functions provided by libraries such as Poco to encode IDNs before performing DNS queries. For example, Poco::Net::DNS::encodeIDN can be used to convert an IDN to its Punycode representation.
2. **Problem: Non-Existent or Incorrect IDNs**
    - **Description**: Using non-existent or incorrect IDNs can cause the resolver to fail with an NXDOMAIN status, indicating that the domain does not exist.
    - **Solution**: Verify the correctness and existence of the IDN using tools like dig or nslookup before including them in test cases. Ensure the domains being used are valid and reachable.
3. **Problem: Network or DNS Configuration Issues**
    - **Description**: Network configuration or DNS server settings might block DNS queries or cause incorrect resolutions.
    - **Solution**: Ensure that the network settings allow DNS queries and that the DNS servers being used are reliable. Use public DNS servers like Google's (8.8.8.8) or Cloudflare's (1.1.1.1) for testing.
4. **Problem: Cache Handling**
    - **Description**: Cached entries may become stale or provide incorrect results if not properly managed.
    - **Solution**: Implement cache expiration logic to ensure that cached entries are

- **Recursive Resolution**: Balancing the depth and breadth of recursive queries to optimize performance without overloading the network.

Key Decisions

- **Use of Poco Library**: Chose Poco for its robust networking capabilities.
- **Caching Strategy**: Decided to use a simple TTL-based caching mechanism to balance simplicity and performance.
- **Retry and Timeout**: Implemented retries and timeouts to handle network variability and improve reliability.

Setup Instructions

Prerequisites

- C++ compiler (e.g., g++)
- CMake

- Poco C++ Libraries (libpoco-dev)

Installation

1. **Install Dependencies:**

   sudo apt-get update

sudo apt-get install g++ cmake libpoco-dev

2. **Clone the Repository:**

   git clone <repository-url>

   cd <repository-directory>

3. **Build the Project:**

   mkdir build

   cd build

   cmake ..

   make

Usage Guidelines

Running the DNS Resolver

To run the DNS resolver, execute the DNSResolverTest binary:

    ./DNSResolverTest

To run resolver

    ./dns_resolver