COMP3331/9331 Assignment Report

Divya Tyagi - z5514961

1 Language and Code Organization

This project is implemented in Python 3. The code is organized as follows:

- `client.py`: Contains the DNSClient class and main client logic
- `server.py`: Contains the DNSServer, DNSCache, DNSRecord, and MasterFileLoader classes, as well as the main server logic
- `master.txt`: The master file containing DNS records (not included in submission, but read by the server)

No additional directories or Makefile are required for this Python implementation.

2 Program Design and Data Structures

2.1 Client Design

The client (`DNSClient` class) is designed to:

1. Parse command-line arguments
2. Construct and send a DNS query message
3. Wait for a response with a specified timeout
4. Parse and print the response

2.2 Server Design

The server consists of several key components:

- `DNSServer`: Main class that handles incoming requests, processes queries, and sends responses
- `DNSCache`: In-memory cache for storing DNS records
- `DNSRecord`: Represents a single DNS resource record
- `MasterFileLoader`: Loads DNS records from the master file

The server uses multi-threading to handle multiple clients concurrently and implements a random delay for query processing.

2.3 Key Data Structures

- DNS Records: Represented by the `DNSRecord` class, storing domain name, record type, and data.
- Cache: Implemented as a Python dictionary in the `DNSCache` class, with domain names as keys and lists of `DNSRecord` objects as values.
- DNS Messages: Constructed and parsed as strings, following a custom format that includes query ID, question section, and answer/authority/additional sections.

3 Known Limitations

1. The implementation does not support all DNS record types, focusing only on A, CNAME, and NS records.
2. Error handling for malformed master files or network issues could be more robust.
3. The caching mechanism is simple and does not implement TTL or cache invalidation strategies.
4. The random delay is implemented using Python's `time.sleep()`, which may not be precise for very short delays.

4 Borrowed Code

The basic structure for UDP socket programming in Python was adapted from the official Python documentation: https://docs.python.org/3/library/socket.html

Specifically, the following pattern was used for both client and server socket creation and communication:

```
with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as sock:
    sock.sendto(message.encode(), (host, port))
    data, addr = sock.recvfrom(buffer_size)
```

No other significant code segments were borrowed from external sources.