Assignment 2: Simple DNS

Due date: *November 25, 2018 at 22:00*

> *This is an individual assignment. You may discuss it with others, but your code and documentation must be written on your own. You might receive a bonus for an outstanding solution.*

Implement a simple local DNS server called *LocalDNS*. *LocalDNS* accepts queries from clients and tries to resolve them, first by checking its local cache and in case of a miss, by asking other name servers. *LocalDNS* must reply to A, NS, and CNAME requests, and it may drop queries of all other types.

Before you start to implement your server, you should have a clear understanding of the DNS protocol and, specifically, of the message format. You can find good information information on-line,[1] in addition to the textbook and the lecture slides.[2]

LocalDNS must be able to resolve queries *iteratively* starting from a given root name server. In other words, *LocalDNS* must work even with a root server that does not support recursive queries. So, to respond to a particular query, the server might have to query multiple servers, starting from the given root server for the root domain, and then iterating through the relevant authoritative name servers for each specific subdomain. For example, the root, then "ch", and then "usi" for "www.usi.ch". *LocalDNS* must also correctly handle canonical names. This means that a server (the root server or others) might reply with a CNAME record in response to a request for the A record for a particular name. In this case, *LocalDNS* should first resolve the CNAME record, and then reply to the original request with a DNS message that includes both the canonical-name record (CNAME) for the original name, and the address record (A) for the canonical name.

LocalDNS must implement and use an internal cache. For each client request and also for each intermediate request (e.g., a request for the "ch." domain), *LocalDNS* must first consult its cache and use a valid cached record if one exists. *LocalDNS* would then issue a query to an external server only if the needed record is not in the cache, or if the one in the cache is no longer valid. The validity of a record is determined by the TTL value for that record. *LocalDNS* must then insert every fresh record received from an external server in its cache. *LocalDNS* does that by either replacing the same (invalid) record in the cache, or by creating a new record in the cache if the cache is not full, or by evicting and replacing another existing record. In this latter case, *LocalDNS* must replace any *invalid* record in the cache before replacing a valid one.

In order to process DNS queries and replies, *LocalDNS* must be able to parse DNS messages. To do so, you may use the attached Java DNS library.[3] However, you should also feel free to implement your own parser.

LocalDNS must accept two command-line arguments: the UDP port to listen on, and the IP address of a root DNS server. For example:

Server
```
./LocalDNS 1234 8.8.8.8
```

You can easily test your implementation using the *dig* command with the proper arguments. For example:

Testing
```
dig @localhost -p 1234 research.inf.usi.ch A
```

To grade your implementations, we will run automated tests using *dig* to submit queries to your DNS server. Below is a summary of the most important requirements for your implementation:

[1] For example, http://www.networksorcery.com/enp/protocol/dns.htm

[2] http://www.inf.usi.ch/carzaniga/edu/ntw18f/dns.pdf, starting from page 35.

[3] http://www.inf.usi.ch/carzaniga/edu/ntw/dns_parser.zip derived, with slight modifications, from the course *Introduction to Computer Networks* by Prof. Aaron Gember-Jacobson

- Your code must compile.
- Your server must reply to DNS requests of types A, NS, and CNAME. Other types are optional.
- If your server receives a CNAME response from a name n while looking for the A record of n, your server should first recursively resolve the name returned in the CNAME record, and then reply to the client with both the CNAME for the original name n, and also the A record for the CNAME name.
- Your server should not send a DNS request if there exists a corresponding valid entry in the local cache.
- The cache entries should be updated for each response received from external DNS servers.

Submission Instructions

Submit all your source files. Add comments to your code to explain sections of the code that might not be clear. You may use an integrated development environment (IDE) of your choice. However, *do not submit any IDE-specific file,* such as project description files, and *make absolutely sure that the files you submit can be compiled and tested with a simple invocation of the standard* javac *compiler and the standard* java *virtual machine.*

In addition to the source files, submit a text file called *README* containing a brief description of your implementation, including a description of the communication protocol you developed, and also a list of all the limitations and errors you are aware of were unable to fix.

Package all the files you need to submit in an archive file named

`a02-<lastname>-<firstname>`

and submit that file through the iCorsi system.