**CSCE 855 Programming Assignment 2**

**Design Document**

Dongpu Jin

3/30/2013

The goal of this project is to build a DNS resolver that takes a hostname as input and resolve for its IP addresses. If the given hostname is invalid, the resolver reports error messages and aborts. If the given hostname is an alias to the actual hostname, the resolver will report the alias name and starts over from the root to resolve the IP addresses for the actual hostname.

The resolver uses the iterative approach to query the name servers starting from the root servers. First, the resolver builds the query message following the RFC 1035 protocol and sends the message to a name server via a UDP socket. During the first iteration, the message is sent to one of the root servers. Note that there are total 13 root servers. The resolver will choose the first available root server. If all root servers are down. The resolver will report error message and abort.

Upon receiving the message, the name server responses the sender with zero or more Resource Records (RR) for Answer section, Authority section, and Additional section, respectively. Then, the resolver looks at the information stored within each section and determine what to do next.

The Answer section contains the answers to the question. If this section is not empty, it indicates that either the IP addresses are found or the hostname is an alias. In the former case, we can just print those found IP addresses and done. In the later case, the hostname turns out to be an alias. The Answer section contians the actual hostname (CNAME) this alias points to. The resolver will start over the query from the root with the actual hostname. Note that when we trace down the CNAME chain, the chain may form a cycle. To solve this problem, we can keep track of the CNAME we have seen along the path. If we encounter a CNAME that we have seen before, it means there is a cycle and we can just abort the program.

However, if the Answer section is empty, which means that this name server that is being queried does not have the IP addresses for the given hostname. But the name server will point to some authoritative name servers that may contain the answers. Therefore, the resolver needs to further analyze the Authority section and Additional section to find the next authoritative name server to query.

In most cases, only analyzing the Additional section is sufficient since this section contains both the hostname and IP addresses for the authoritative name servers. However, this is not always the case. For instance, when iteratively query name servers with "nasa.org" start from the root server, the root server "a.root-server.net" points to an authoritative name server "a0.org.afilias-nst.info". When query the authoritative name server "a0.org.afilias-nst.info", there is no Additional section in the response message. Therefore, whenever the resolver finds that the Authority section is empty, it will look at the Authority section instead. Note that the Authority section only contains the hostname for the authoritative name servers without their IP addresses. Thus, before moving forward, the resolver needs to resolve the IP addresses for the authoritative name servers first by iteratively query the name servers start from the root.

Currently, the resolver does not cache the previous search results. Every time, it starts the search from the root, even for the same searches. One possible improvement is to make the resolve cache previous search results. Thus, the resolver can simply looks up the cache for the answers for the same queries without querying any name servers. However, this can be a design trade off because name servers may update their content, which may lead to inconsistency to the local cache. Therefore, we may not get the out dated IP addresses if using cache.

This resolver is based on the code on http://www.binarytides.com/dns-query-code-in-c-with-linux-sockets/ with none trivial modifications.