CMSC 417 Computer Networks	Spring 2000

Project 3 — Simple Name Service

Assigned: February 16 — *Due: February 25, Midnight.*

1 Introduction

In this project, you will write a simple name server and a resolver. The real domain name service in the Internet is used to resolve the mapping between a hostname and an IP address.

As usual, the server will run on a machine at a well-known port number. The clients, called resolvers, will contact the server with queries about hostnames and IP addresses. The server has a resident database of hostname to IP address mappings. The database and routines to access it will be provided to you. On receiving a name service query, the server will lookup the query in the database and respond with an appropriate answer.

The server and the client (resolver) will use UDP —the User Datagram Protocol— to communicate with each other. UDP is an unreliable protocol. Thus, for various reasons, UDP packets may be lost in the network.

In this assignment, you will need to implement two new mechanisms:

- **Re-transmissions from the client:** Since data transfer in UDP is not reliable, you will need to create a re-transmission mechanism in the client for queries that do not lead to a response from the server.

- **Checksum computation:** To ensure that the data received at either end has not been corrupted in the network, on receiving a packet from the network, you should first compute the checksum to see if the data is correct. The server and the resolvers will silently ignore all corrupted packets.

2 Protocol Specification

The protocol works as shown in Figure 1. The client will construct queries of different types and send them to the server. The server will query its internal database for each client query and send back a response. The client is responsible for re-transmitting any query that gets lost or corrupted in the network. Details of the protocol follow.

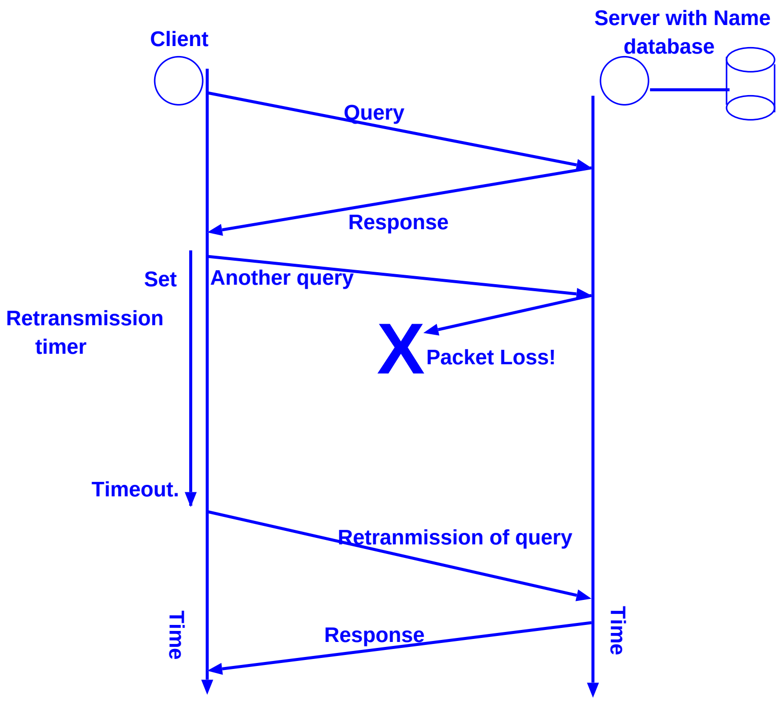

Figure 1: The basic protocol operation. The response to the second query is lost and the query is retransmitted by the client.

2.1 Packet Header

Both query and response packets have the same header structure shown below.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Version|Type |X|   Length      |             Query-Id          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    Data : HostName / IP Address (padded to 32 bit boundary)   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |            Checksum           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

`Version` (4 bits) The version field for the protocol should be set to `0110`.

`Type` (3 bits) The contents of the type field is given below.

`X` (1 bit) This field must be set to zero in the `query` packets (i.e., packets from the clients to the server). In the `response` packets, if the X-bit is set it means that a match was found for the query in the server's database. Otherwise, the X-bit is cleared to indicate that no match was found for the query. (If a match is not found, the data field size will also be zero bytes).

`Length` (8 bits) Contains the length of the packet in number of octets (bytes). For packets that contain a hostname in the data, the packet length would depend on the length of the hostname.

`Query-Id` (16 bit) This is an unsigned 16 bit query id assigned by the resolver to the query. The name server copies the Query-Id from the query packet into the response packet.

`Data` (Variable length) This field contains the hostname or the IP address as would be required, e.g., for a query for IP address and a response for a hostname packets, this field would contain the host name of the network host. In the other two packets it carries the IP address.

The IP address of a packet is stored in the network format as a 32-bit integer (see `inet_aton(3)` and `inet_ntoa(3)` for conversion between dotted decimal format and integer format of the addresses). No padding is necessary in this case.

A hostname may be of variable length. If the length of the hostname is not a multiple of 32 bits, then the remaining octets should be padded with zeros to a 32 bit boundary. The hostname should be terminated by at least one
`0` character.

`Checksum` (16 bits) The checksum is computed over all the other fields. The value of the checksum is one's complement of the one's complement sum of the 16-bit words of the other data fields. If the data was padded, the padding is also included in the sum. More details about the checksum computation will be posted on the newsgroup.

There are two different types of packets that are sent in our protocol:

1. Query Packets : These contain queries sent by the resolvers to the name server.
   - Query For IP Address `(Type 000)` : This query contains the IP address for which the resolver is seeking the hostname.
   - Query For Hostname `(Type 001)` : This query contains the hostname of a network host for which the resolver is seeking the IP address.
2. Response Packets: These are sent by the server in response to the queries from the resolver. As you would expect, there are two different response packets.

- Response For IP Address `(Type 010)` : It contains the IP address as queried by the resolver.
- Response For Hostname `(Type 011)` : It contains the host name of the network host as queried by the resolver.

The server will receive packets on a well-known port. It will ignore any packet whose type it does not know about. It will also ignore any packet that contains a bad checksum. In general, both the server and the resolver will ignore ill-formed packets. Upon receiving a valid query, the server will query its internal database for the information requested and will return what it finds in the database.

Note: the server will (re-)construct a new packet with its answer. For queries that do not have an entry in the database, the data part is 0 bytes and the X-bit is set to 0. For queries that do have an entry in the database, the answer to the query will be sent back. The client is responsible for mapping the answer to the original query using a `Query Id` (defined in the next section).

If the server does not reply within a pre-defined time period, the client will re-try its query. The server is completely stateless, i.e., it does not *remember* which queries it has served, from whom, and when. If it receives the same query from the same resolver, the server will answer as appropriate.

Note: Clients *must* be able to disambiguate stale responses from the server. Thus, if a client gets a response which does not match the latest query-id, it should ignore this response.

3 Executables

The name server and the client executables will accept the following command line arguments:

- Server:

  `server -p ⟨port⟩ -d ⟨database-file-name⟩`

  - port : Port of the host at which the server will run.
  - database-file-name : Name of the database file name to use for mapping the IP addresses to hostnames.

- Resolver:

  ```
  client [-h ⟨hostname⟩] [-p ⟨port⟩]  \
               [-t ⟨timeout⟩] [-i ⟨max-retries⟩] -T ⟨query-type⟩ -d ⟨data⟩
  ```

  - hostname: Hostname of the host at which the server is running. (Optional). Default server is at `wolfe.umd.edu`

- Port : Port of the host at which the server is running. (Optional). Default port is 9999.
- timeout: Indicates in seconds, how long to wait before regenerating an un-answered query. (Optional). The default timeout is 5 seconds.
- max-retries: Indicates the number of times the resolver will re-generate the query, before it quits, if no response is received from the server. (Optional). The default is 5 times.
- query-type: This is the type of query. Can only be set to 0 for IP address query, and 1 for hostname query.
- data: This is a string that will form the data portion of the query. Hostname queries *must* specify a dotted quad here, while IP address queries will specify a hostname. Note that the hostname should not be more than 256 bytes long.

In the client, if the hostname and the port number of the server is not specified, the defaults will be used.

4 Database Interface Functions

The server will query entries in the database using the following interface routines :

- `void open_database (char *file_name);`
  Returns `NULL` if no such database was found.

- `char * lookup_ip_address (char *host_name);`
  The hostname is passed as an argument. Returns the IP address as a character string in the dotted decimal format. `NULL` is returned if no match is found.

- `char * lookup_host_name (char *ip_address);`
  The IP address is passed in the dotted decimal format. The hostname string is returned. `NULL` is returned if no match is found.

- `void close_database (void);`
  Closes the database.

Note, in `lookup_ip_address` and `lookup_host_name`, the functions will allocate space for these strings. It is up to the calling functions to `free` the space allocated here.