מבוא לתקשורת מחשבים – תרגיל מעשי #2

מגישים: 322522244 אלון גולדנברג, 313419265 תם אריכא

מסמך דוקומנטציה – nsclient

בתרגיל זה, מימשנו תוכנה שמתשאלת שרת DNS עבור כתובת IP של דומיין הניתן ע"י המשתמש.

לשם כך, מימשנו פונקציה בשם dnsQuery, שמחקה את אופן פעולת הפונקציה gethostbyname.

התוכנה תקבל כארגומנט את כתובת הIP של שרת הDNS, ותפתח חלון פקודה, בו המשתמש מזין כתובת domain ע"פ התקן RFC 1035, או את הפקודה quit, שסוגרת את התוכנית. במידה והוזן דומיין, התוכנה מוודאת את תקינות הקלט, ואם הקלט תקין, מתשאלת באמצעות הפונקציה dnsQuery את שרת הDNS, שמחזיר בתורו את כתובת ה- IPV4 של הדומיין במידה וקיים, או שגיאה – BAD NAME. במידה והשרת החזיר קוד 0, כלומר שאילתא תקינה, הפונקציה מקצה, ממלאת ומחזירה מבנה מסוג hostent, ואחרת היא מחזירה NULL. במידה ומוחזר NULL, מודפס NONEXISTENT.
אחרת, כתובת ה- IP הראשונה במבנה ה- hostent המוחזר מודפסת כמחרוזת.

דוגמת הרצה:

```
$ nsclient.exe 8.8.8.8
```

```
Microsoft Visual Studio Debug Console
nsclient> gaga..com
ERROR: BAD NAME
nsclient> ynet.co.il
23.211.6.111
nsclient> google.com
142.250.185.174
nsclient> ynet..com
ERROR: BAD NAME
nsclient> ynet..co.il
ERROR: BAD NAME
nsclient> Google.com
142.250.185.142
nsclient> Google.co.il
172.217.16.131
nsclient> google.co.il
172.217.16.131
nsclient> gooGle.com
142.250.185.142
nsclient> Quit

C:\Users\alongold\Github\networks_project_nslookup\x64\Debug\nsclient.exe (process 38104) exited with code 0.
```

בעמודים הבאים, תיעוד טכני של החלקים העיקריים בפרויקט.

בכל קבצי ה- headers, וגם ברוב הפונקציות, אפשר לראות docstrings מלאים, אז מומלץ להסתכל שם.

שם קובץ: main.c

```c
int main(const int argc, const char* argv[]);
/*
 * Main loop, as described in README.docx
 * INPUT: argc: number of console arguments
 * INPUT: argv: console arguments
 * OUTPUT: command prompt, IP address of host requested by user, exits if “quit”
 * RETURN: 0 if exited program cleanly (via "quit")
 *         1 if could not connect to DNS server
 */
```

שם קובץ: dns_utils.c

```c
int validateHost(const unsigned char* hostname);
/*
* INPUT: hostname
* RETURN: STATUS_SUCCESS if hostname is valid, STATUS_ERR_BAD_NAME otherwise
* DESCRIPTION:
* Follows RFC 1035 specification.
* we consider hostname to be valid <=> hostname matches definition of <subdomain>
* We are intentionally ignoring the " " case
* Definitions, from the specification:
* Octet ::= 8 bits (1 char)
* <domain> ::= <subdomain> | " "
* <subdomain> ::= <label> | <subdomain> "." <label>
* <label> ::= <letter> [ [ <ldh-str> ] <let-dig> ]
* <ldh-str> ::= <let-dig-hyp> | <let-dig-hyp> <ldh-str>
* <let-dig-hyp> ::= <let-dig> | "-"
* <let-dig> ::= <letter> | <digit>
* <letter> ::= any one of the 52 alphabetic characters A through Z in
*              upper case and a through z in lower case
* <digit> ::= any one of the ten digits 0 through 9
*/

struct hostent* dnsQuery(const char* hostname);
/*
* INPUT: "hostname": e.g. "google.com", "www.ynet.co.il"
* RETURN: a hostent object == gethostbyname(hostname)
*/

size_t change_question_name(const unsigned char* hostname, unsigned char* qname);
/*
* INPUT: "_hostname": e.g. "www.abcd.com"
* OUTPUT: "dst": e.g. [0x3,"www",0x4,"abcd",0x3,"com",0x0]
* RETURN: size of dst written
*/

char* createDnsQueryBuf(const char* hostname, size_t* p_sizeof_query, size_t*
sizeof_qname);
/*
* INPUT: "hostname": e.g. "google.com", "www.ynet.co.il"
* RETURN: "query[]":
*         This string is sent to the DNS server
*         It contains the request "give me the IP address for <hostname>"
*/

struct hostent* parseDnsResponseBuf(const unsigned char* response, size_t sizeof_reponse,
size_t sizeof_qname);
/*
* INPUT: "response": fetched from DNS server through recvfrom()
*        sizeof_response
*        sizeof_qname : size of the name **sent** to DNS
* RETURN: hostent object with returned IP
*/

size_t read_qname(const unsigned char* reader, char far** h_name);
/*
```

```
* INPUT: "reader": pointer to location in which we can expect the hostname in the
response
*                   we expect 3www6google3com0 style formatting
* INPUT: "h_name": pointer to pointer in which we want to store the extracted hostname
* OUTPUT: "h_name": will be in www.google.com format
* RETURN: size of name as stored within reader (3www6google3com0 -> 16)
*/

int read_qname_wrapper(const unsigned char* reader, size_t sizeof_qname, size_t
sizeof_response,
                                        const unsigned char* response, struct hostent*
remoteHost);
/*
 * Wrapper for read_qname, to account for different location possibilities for hostname
*/

void printRemoteHost(struct hostent* remoteHost);
/*
* INPUT: "remoteHost": a hostent struct
* OUTPUT: prints first IP address listed in "remoteHost->h_addr_list" if nonnull
*/

void parseDnsHeaderFromResponse(dns_header_t* dns);
/*
 * (enabled iff FLAG_DEBUG == 1)
 * If DNS server returned status code >= 1, displays error
 */

void assertDnsQueryResultIsValid(const struct hostent* remoteHost, const char* hostname);
/*
* (enabled iff FLAG_DEBUG == 1)
* Verifies correctness of remoteHost through comparison against gethostnameaddr() result
*/
```

שם קובץ: socket_utils.c

```c
int socket_initialize(WSADATA* wsaData);
/*
* INPUT: "wsaData": WSADATA pointer
* RETURN: int: same as WSAStartup documented in winsock2.h
* Wrapper for winsock2.h:WSAStartup with predefined parameters
*/

int socket_connect(SOCKET* sock, const char* dest, const u_short port);
/*
* INPUT: "sock": socket pointer to initialize and connect through
* INPUT: "dest": destination IP address
* INPUT: "port": destination port
* RETURN: connect() status code as defined by winsock2.h
* Attempts to create UDP connection to dest:port, sets timeout of 2sec for recvfrom
*/
```

שם קובץ: common_types.c

```c
typedef struct DnsHeader {
    unsigned short id;
    unsigned char rd : 1;
    unsigned char tc : 1;
    unsigned char aa : 1;
    unsigned char opcode : 4;
    unsigned char qr : 1;
    unsigned char rcode : 4;
    unsigned char cd : 1;
    unsigned char ad : 1;
    unsigned char z : 1;
    unsigned char ra : 1;
    unsigned short q_count;
    unsigned short ans_count;
    unsigned short auth_count;
    unsigned short add_count;
} dns_header_t;
// Contains DNS header struct, used for writing query buffer, and reading from response

typedef struct Question {
    unsigned short qtype;
    unsigned short qclass;
} question_t;
// Contains question, used for reading from response buffer after header

typedef struct Response {
    unsigned short rtype;
    unsigned short rclass;
    unsigned int rttl;
    unsigned short rdlength;
    struct in_addr addr;
} response_t;
// Contains response
```