How hostname resolution works

- DNS? 인터넷에 연결하기 위한 주소
  Domain(Host name) <-> IO
- getaddrinfo() hostname을 IP로 바꿔주는 함수
  DNS 처리 API

getaddrinfo() 실제 실행

1. www.example.com에 접속하려고 한다.
2. os는 해당 도메인의 IP가 로컬캐시에 있는지 확인한다.
3. TTL이 끝나 없으면, os는 DNS 서버에 쿼리를 보낸다.
4. DNS 서버도 해당 쿼리를 받으면 자신의 로컬캐시를 확인한다.
5. 있으면 리턴, 없으면 다른 DNA 서버에 물어보라고 한다.

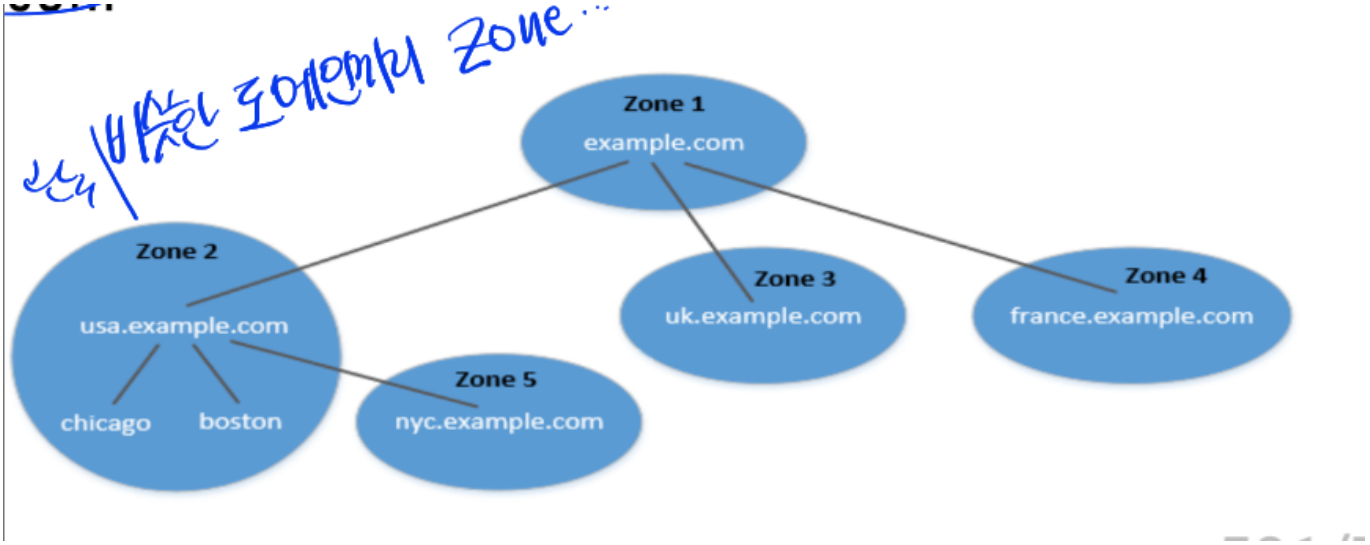
DNS Domain Hierarchy

도메인은 계층구조로 이루어져있다.

- Root
- Top-Level Domain(TLDs): .com, .net, .edu..
- Second-Level Domain: google, example..

DNS Zone

비슷한 성격의 도메인들끼리 존을 이룬다.



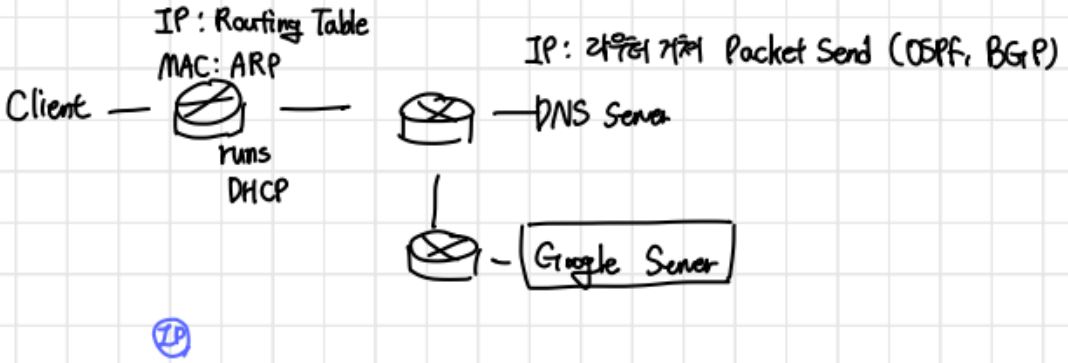
authoritative nameserver

각 존은 최소 하나의 **authoritative nameserver**를 갖고있으며, 거기서는 존에 대한 정보를 제공한다.
이를 통해 DNS 쿼리에 대한 답을 제공한다.

또한, 모든 존을 관리하는 master server이거나 존에 대한 정보가 존재하는 slave server로 나뉜다.

DNS Protocol

Google 접속.



(IP)

내장 컴퓨터 (목적지의 IP주소 check)

① DNS cache에 있는지 확인

② DNS IP & 본인 IP
→ DHCP로 받아오기
UDP 데몬 사용
└ 브로드캐스트이기에 신뢰성 없어도됨. & 빠름.

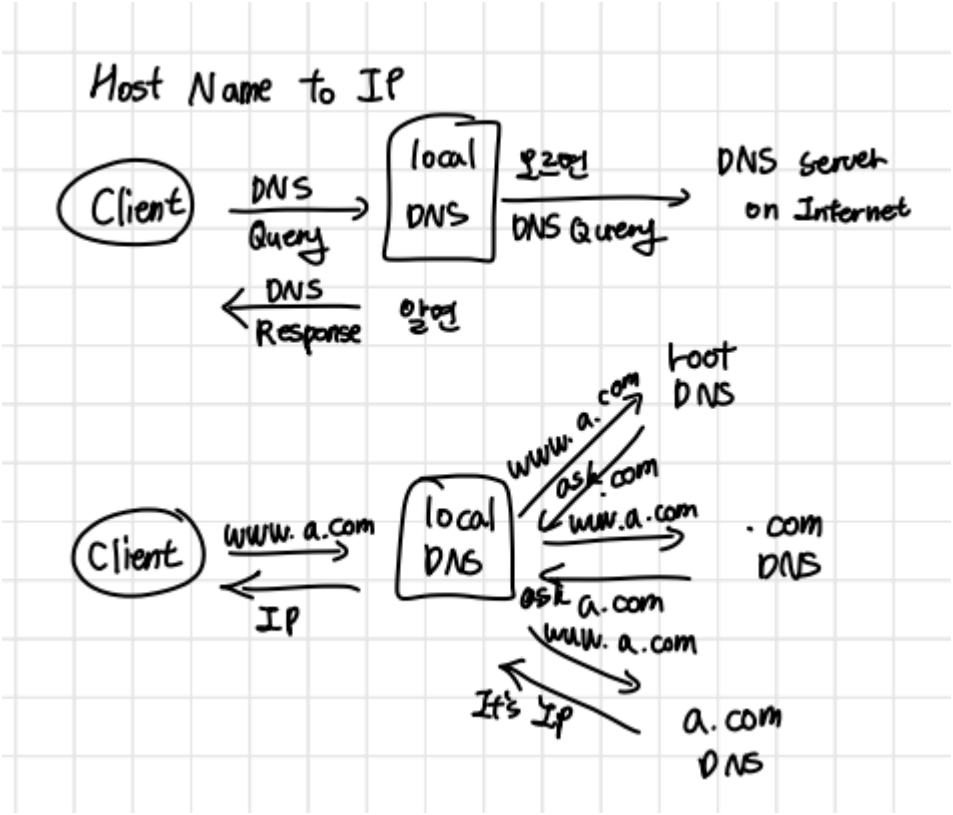
③ DHCP Response
1. 본인 IP
2. DNS IP
3. 가장 가까운 라우터의 IP.
↳ 라우터의 MAC주소를 알기위해 (ARP)

④ Google IP 알기 — Google IP 주세
DNS IP 알기때문에 DNS에 Request.
Client → DNS
SIP : Client SMAC : Client
DIP : DNS DMAC : 가까운 라우터

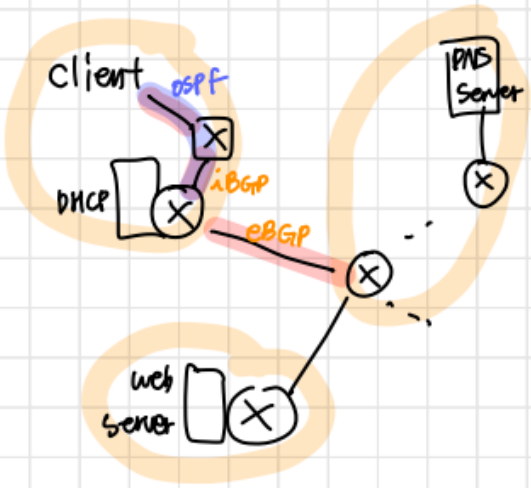
⑤ HTTP통신. — Client - Google.
TCP 덕분
1) 3-way handshaking
2) request하면 (Google)

- DHCP

Client IP, 가까운 Router IP, DNS server IP 얻어와야 (할당) 해야 할때
동적 IP 할당
UDP로 캡슐화



Client to Google.



1. Client IP 할당받기

DHCP request: UDP로 캡슐화, IP, 802.3 Ethernet

Ethernet broadcast (MAC: fff··) on LAN,
→ DHCP 프로토콜을 동작중인 Router가 받음

Ethernet → IP / UDP → DHCP
demultiplex demultiplex

DHCP Response → 라우터의 MAC도 알 수 있음 (ARP)
: Client IP, 인접 라우터 IP, DNS Server IP

2. Host name (www.google.com) to IP

ARP로 라우터 MAC 알고, 라우터를 통해 DNS Query

RIP · OSPF · BGP ·· 로

DNS Response Google IP

3. Google IP와 TCP Connection → 연결 (3-way-handshaking) HTTP 통신 (TCP socket을 통해)
↳ web page 내려옴

1. DNS 서버의 IP는 기본으로 세팅되어 있다.

2. hostname으로 root DNS query
3. 모르면 TLD DNS query (.com)
4. 또 모르면 하위 DNS query (example.com)
5. IP 받음

DNS Response

- Question Section: client가 물어본 것
- Answer section: DNS가 해당 도메인 IP 알고있으면 IP 출력
- Authority section: 모르면 다른 DNS Server에게 물어보라고함
- Additional section: 쿼리 관련 추가적인 섹션

```
nsp@nsp-VirtualBox:~/Desktop$ dig @e.gtld-servers.net www.example.net
                                                    Ask a .net nameservers.
; <<>> DiG 9.11.3-1ubuntu1.15-Ubuntu <<>> @e.gtld-servers.net www.example.net
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59932
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 5
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.net.               IN      A

;; AUTHORITY SECTION:
example.net.            172800  IN      NS      a.iana-servers.net.
example.net.            172800  IN      NS      b.iana-servers.net.

;; ADDITIONAL SECTION:
a.iana-servers.net.     172800  IN      A       199.43.135.53
a.iana-servers.net.     172800  IN      AAAA    2001:500:8f::53
b.iana-servers.net.     172800  IN      A       199.43.133.53
b.iana-servers.net.     172800  IN      AAAA    2001:500:8d::53
                                                    Go ask them!
;; Query time: 116 msec
;; SERVER: 192.12.94.30#53(192.12.94.30)
;; WHEN: Mon May 03 03:12:48 KST 2021
;; MSG SIZE  rcvd: 177
```

답 찾으면 Authority Section말고 Answer section출력

DNS message format

- header
- question
- answer (return IP)
- Authority
- Additional

DNS message: header

- 12바이트
- 각 16비트로 구성됨

필드	설명
ID	16bit 식별자
기타	
QDCOUNT	DNS 쿼리 개수
ANCOUNT	DNS 응답 개수 -> 하나의 도메인이 여러 IP를 가질 수 있기 때문에
NSCOUNT	레코드의 수
ARCOUNT	레코드의 수

- 기타 필드

필드	설명
QR	1bit) 0이면 query, 1이면 response 나타내는 비트
Opcode	4bit) Query Type 0이면 standary 쿼리, 1이면 reverse query (IP to name), 2면 서버 상태 요청
AA	1bit) 응답이 인증된 응답인 경우 1, 인증되지 않은 경우 0
TC	1bit) 메세지 축약된 형태 / TCP사용
RD	1bit) 재귀 요청 / 최종 IP 얻기 위해 계속 DNS Query
RA	1bit) DNS Server가 재귀 가능한지
Z	2bit) 사용되지 않음, 0세팅
RCODE	4bit) 응답 코드로, 응답의 상태를 나타냄

RCODE	설명
0	no error
1	format error
2	server fail
3	name error
4	not implemented
5	refused

DNS message: Question

필드	설명

필드	설명
NAME	Host name
QTYPE	질의 유형(예: A, AAAA, MX 등)
QCLASS	질의 클래스(주로 1(Internet) 클래스 사용)

- QTYPE A: IPv4 AAAA: IPv4
- Name field... www.example.com을 기준으로

length	value
3	w
w	w
7	e
x	a
m	p
l	e
3	c
o	m
0	name 끝나면 0byte

원래는 TLV Rule이지만 (Type, Length, Value)
여기서는 LV만 사용한다.

3 - www 7 - example 3 - com 0 - end 알림

DNS Format: answer

필드	설명
NAME TYPE CLASS	Question Format과 동일
TTL	32bit) host의 대응 IP가 머물 시간
RDLEN	RDATA(리소스 데이터) 필드의 길이
RDATA	리소스 데이터, 질의에 대한 실제 응답 값 / Type에 따라 달라짐 (A, AAAA)

DNS Program

1. DNS Query(Header + Question)
2. DNS Response (Header + Question + Answer + etc..)
3. DNS Response print

Question과 Answer에 LV encoding Rule로 된 동일한 Name 필드가 존재한다. (호스트 네임) pointing을 통해 패킷 압축 필요

Pointing

원래는 Name 필드
3 w w w 7 e x a m p l e 3 c o m 형식

하지만 상위 2byte가 0xc0으로 시작하면 포인팅으로 인식한다.
-> 1 1(압축 flag 2bit) + 14bit pointer bit

dns_query.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define ISVALIDSOCKET(s) ((s) >= 0)
#define CLOSESOCKET(s) close(s)
#define SOCKET int
#define GETSOCKETERRNO() (errno)

//DNS 메세지의 시작, NAME 필드 가리키고 있는 포인터, DNS 메세지의 끝부분
//big endian형식이기에 시작 주소가 끝 주소보다 크다
const unsigned char *print_name(const unsigned char *msg,
        const unsigned char *p, const unsigned char *end) {

    //L, V 최소 2bit존재해야하지만 아니라면 에러처리
    if (p + 2 > end) {
        fprintf(stderr, "End of message.\n"); exit(1);}

    //포인터형식
    if ((*p & 0xC0) == 0xC0) {
        //0x3f: 0011 1111
        //최상위 2bit 제외한 값이 포인터의 주소이다
        const int k = ((*p & 0x3F) << 8) + p[1];
        //포인터를 의미하는 상위 2비트 뛰어넘기
        p += 2;
        printf(" (pointer %d) ", k);
        //msg+k -> Question의 NAME 필드의 것을 가져옴
        print_name(msg, msg+k, end);
        return p;
    }
    //LV형태 NAME 필드 형식
    else {
```

```
        //length 저장
        const int len = *p++;
        //lendl 0이면 문장의 끝
        if (p + len + 1 > end) {
            fprintf(stderr, "End of message.\n"); exit(1);}

        //p에서 len만큼 읽기
        printf("%.*s", len, p);
        p += len;
        //p!=0이면
        if (*p) {
            printf(".");
            return print_name(msg, p, end);
        } else {
            //p==0이면
            return p+1;
        }
    }
}

void print_dns_message(const char *message, int msg_length) {

    //메세지 헤더는 무조건 12byte
    if (msg_length < 12) {
        fprintf(stderr, "Message is too short to be valid.\n");
        exit(1);
    }

    //msg[0]이 8byte
    //ID = msg[0]+msg[1]
    //msg[2] = QR, OPcode, AA, TC, RD
    const unsigned char *msg = (const unsigned char *)message;

    printf("ID = %0X %0X\n", msg[0], msg[1]);

    //8bit 중 해당하는 비트만 살려서 shift연산해 사용

    //1000 0000(0x80)과 AND연산 -> 상위 1bit만 보기
    //상위 1bit를 7bit shift -> QR 알기
    const int qr = (msg[2] & 0x80) >> 7;
    //QR 1이면 response, 0이면 query
    printf("QR = %d %s\n", qr, qr ? "response" : "query");

    //0111 1000(0x78) -> type
    const int opcode = (msg[2] & 0x78) >> 3;
    printf("OPCODE = %d ", opcode);
    switch(opcode) {
        case 0: printf("standard\n"); break;
        case 1: printf("reverse\n"); break;
        case 2: printf("status\n"); break;
        default: printf("?\n"); break;
    }

    const int aa = (msg[2] & 0x04) >> 2;
```

```
    printf("AA = %d %s\n", aa, aa ? "authoritative" : "");

    const int tc = (msg[2] & 0x02) >> 1;
    printf("TC = %d %s\n", tc, tc ? "message truncated" : "");

    const int rd = (msg[2] & 0x01);
    printf("RD = %d %s\n", rd, rd ? "recursion desired" : "");

    //QR:1 response일때만..
    if (qr) {
        //rcode: response에만 존재한다
        const int rcode = msg[3] & 0x0F;
        printf("RCODE = %d ", rcode);
        switch(rcode) {
            case 0: printf("success\n"); break;
            case 1: printf("format error\n"); break;
            case 2: printf("server failure\n"); break;
            case 3: printf("name error\n"); break;
            case 4: printf("not implemented\n"); break;
            case 5: printf("refused\n"); break;
            default: printf("?\n"); break;
        }
        if (rcode != 0) return;
    }

    //int는 4byte이기 때문에 msg[4]를 일단 앞으로 당기고 남은 8bit msg[5]
    //count 하나당 16bit(2byte)
    const int qdcount = (msg[4] << 8) + msg[5];
    const int ancount = (msg[6] << 8) + msg[7];
    const int nscount = (msg[8] << 8) + msg[9];
    const int arcount = (msg[10] << 8) + msg[11];

    printf("QDCOUNT = %d\n", qdcount);
    printf("ANCOUNT = %d\n", ancount);
    printf("NSCOUNT = %d\n", nscount);
    printf("ARCOUNT = %d\n", arcount);


    //헤더 크기 더하기 -> 다음 메세지인 Question을 가리킨다.
    const unsigned char *p = msg + 12;
    //msg_length: 함수 호출시 받은 값
    const unsigned char *end = msg + msg_length;

    //query가 하나 이상이라면
    if (qdcount) {
        int i;
        for (i = 0; i < qdcount; ++i) {
            //초기 p: Question Format 포인팅
            if (p >= end) {
                fprintf(stderr, "End of message.\n"); exit(1);}

            printf("Query %2d\n", i + 1);
            printf("  name: ");
```

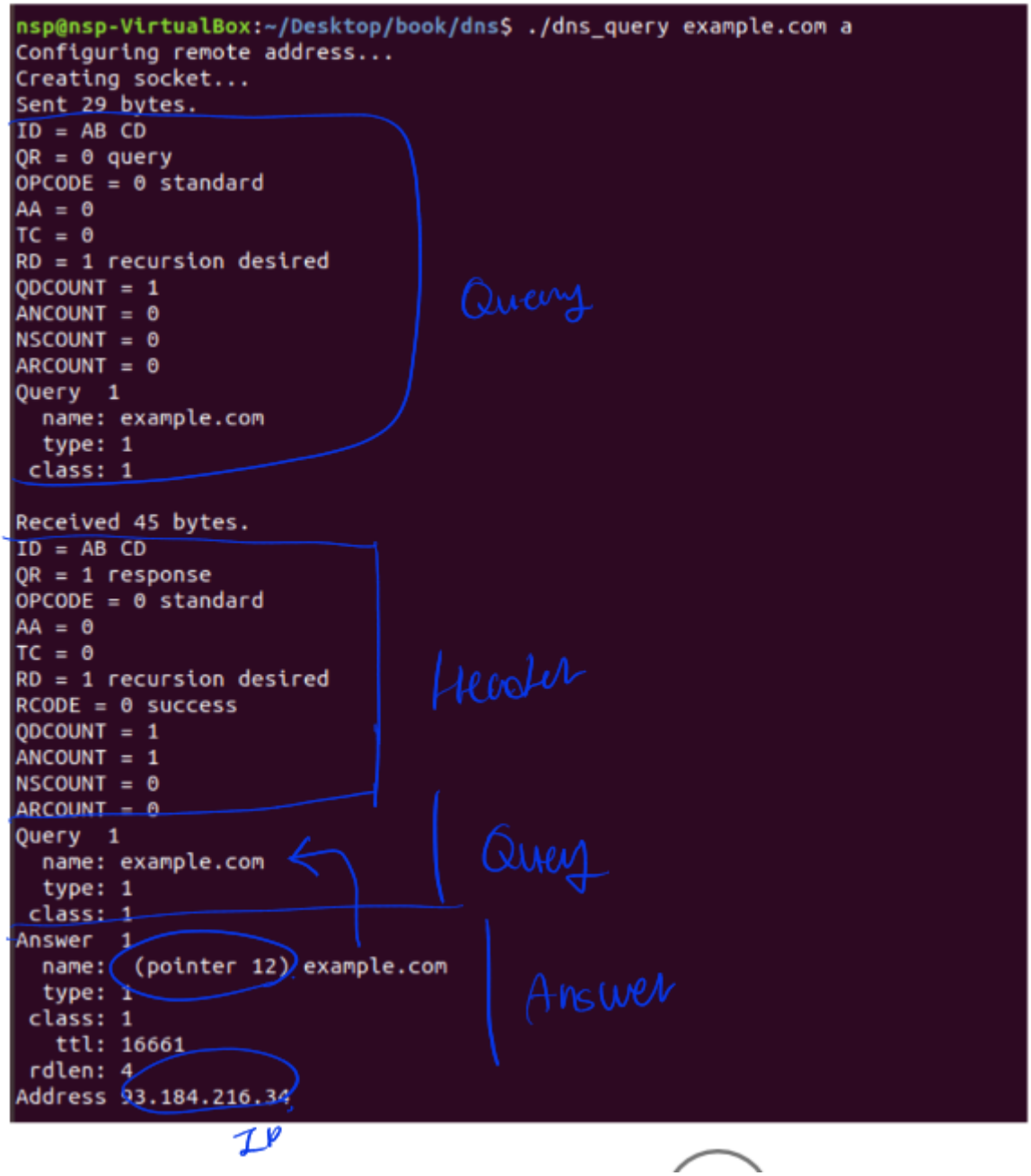
```
            //이름을 다 읽고 QTYPE을 가리키는 p
            p = print_name(msg, p, end); printf("\n");

            if (p + 4 > end) {
                fprintf(stderr, "End of message.\n"); exit(1);}

            //qtype읽음
            const int type = (p[0] << 8) + p[1];
            printf("  type: %d\n", type);
            p += 2; //p는 qclass가리킴

            const int qclass = (p[0] << 8) + p[1];
            printf(" class: %d\n", qclass);
            p += 2;
            //마지막가르킴
        }
    }

    //answer section까지 있으면...
    if (ancount || nscount || arcount) {
        int i;
        for (i = 0; i < ancount + nscount + arcount; ++i) {
            if (p >= end) {
                fprintf(stderr, "End of message.\n"); exit(1);}

            printf("Answer %2d\n", i + 1);
            printf("  name: ");

            //실제로는 pointing한 주소로 가서 이름을 가져오는 것
            p = print_name(msg, p, end); printf("\n");

            if (p + 10 > end) {
                fprintf(stderr, "End of message.\n"); exit(1);}

            const int type = (p[0] << 8) + p[1];
            printf("  type: %d\n", type);
            p += 2;

            const int qclass = (p[0] << 8) + p[1];
            printf(" class: %d\n", qclass);
            p += 2;

            const unsigned int ttl = (p[0] << 24) + (p[1] << 16) +
                (p[2] << 8) + p[3];
            printf("   ttl: %u\n", ttl);
            p += 4;

            const int rdlen = (p[0] << 8) + p[1];
            printf(" rdlen: %d\n", rdlen);
            p += 2;

            if (p + rdlen > end) {
                fprintf(stderr, "End of message.\n"); exit(1);}
```

```
            //IPv4 A:1
            if (rdlen == 4 && type == 1) {
                /* A Record */
                printf("Address ");
                printf("%d.%d.%d.%d\n", p[0], p[1], p[2], p[3]);

            }
            //IPv6
            else if (rdlen == 16 && type == 28) {
                /* AAAA Record */
                printf("Address ");
                int j;
                for (j = 0; j < rdlen; j+=2) {
                    printf("%02x%02x", p[j], p[j+1]);
                    if (j + 2 < rdlen) printf(":");
                }
                printf("\n");

            }
            /*까지는 구현하고 가능해야 한다*/
            else if (type == 15 && rdlen > 3) {
                /* MX Record */
                const int preference = (p[0] << 8) + p[1];
                printf("  pref: %d\n", preference);
                printf("MX: ");
                print_name(msg, p+2, end); printf("\n");

            } else if (type == 16) {
                /* TXT Record */
                printf("TXT: '%.*s'\n", rdlen-1, p+1);

            }
            //하나의 alternative name
            else if (type == 5) {
                /* CNAME Record */
                printf("CNAME: ");
                print_name(msg, p, end); printf("\n");
            }

            p += rdlen;
        }
    }

    if (p != end) {
        printf("There is some unread data left over.\n");
    }

    printf("\n");
}


int main(int argc, char *argv[]) {

    if (argc < 3) {
```

```
        printf("Usage:\n\tdns_query hostname type\n");
        printf("Example:\n\tdns_query example.com aaaa\n");
        exit(0);
    }

    if (strlen(argv[1]) > 255) {
        fprintf(stderr, "Hostname too long.");
        exit(1);
    }

    unsigned char type;

    //RDATA - 표준
    if (strcmp(argv[2], "a") == 0) {
        type = 1;
    } else if (strcmp(argv[2], "mx") == 0) {
        type = 15;
    } else if (strcmp(argv[2], "txt") == 0) {
        type = 16;
    } else if (strcmp(argv[2], "aaaa") == 0) {
        type = 28;
    } else if (strcmp(argv[2], "any") == 0) {
        type = 255;
    } else {
        fprintf(stderr, "Unknown type '%s'. Use a, aaaa, txt, mx, or any.",
                argv[2]);
        exit(1);
    }

    //8.8.8.8 53 -> 구글에서 제공하는 DNS Server
    printf("Configuring remote address...\n");
    struct addrinfo hints;
    memset(&hints, 0, sizeof(hints));
    hints.ai_socktype = SOCK_DGRAM;
    struct addrinfo *peer_address;
    if (getaddrinfo("8.8.8.8", "53", &hints, &peer_address)) {
        fprintf(stderr, "getaddrinfo() failed. (%d)\n", GETSOCKETERRNO());
        return 1;
    }


    printf("Creating socket...\n");
    SOCKET socket_peer;
    socket_peer = socket(peer_address->ai_family,
            peer_address->ai_socktype, peer_address->ai_protocol);
    if (!ISVALIDSOCKET(socket_peer)) {
        fprintf(stderr, "socket() failed. (%d)\n", GETSOCKETERRNO());
        return 1;
    }


    char query[1024] = {0xAB, 0xCD, /* ID */
                        0x01, 0x00, /* Set recursion */
                        0x00, 0x01, /* QDCOUNT */
```

```
                        0x00, 0x00, /* ANCOUNT */
                        0x00, 0x00, /* NSCOUNT */
                        0x00, 0x00 /* ARCOUNT */};

    char *p = query + 12;
    char *h = argv[1];

    //Query name에 hostname 넣어주기
    //LV 형태
    while(*h) {
        char *len = p;
        p++;
        if (h != argv[1]) ++h;

        while(*h && *h != '.') *p++ = *h++;
        *len = p - len - 1;
    }

    *p++ = 0;
    *p++ = 0x00; *p++ = type; /* QTYPE */
    *p++ = 0x00; *p++ = 0x01; /* QCLASS */
    //세팅끝

    const int query_size = p - query;

    int bytes_sent = sendto(socket_peer,
            query, query_size,
            0,
            peer_address->ai_addr, peer_address->ai_addrlen);
    printf("Sent %d bytes.\n", bytes_sent);

    print_dns_message(query, query_size);
    //쿼리 보내기

    char read[1024];
    int bytes_received = recvfrom(socket_peer,
            read, 1024, 0, 0, 0);
    //응답 읽기

    printf("Received %d bytes.\n", bytes_received);

    print_dns_message(read, bytes_received);
    printf("\n");
    //출력

    freeaddrinfo(peer_address);
    CLOSESOCKET(socket_peer);

    return 0;
}
```

실행 결과


nsp@nsp-VirtualBox:~/Desktop/book/dns$ ./dns_query example.com a
Configuring remote address...
Creating socket...
Sent 29 bytes.
ID = AB CD
QR = 0 query
OPCODE = 0 standard
AA = 0
TC = 0
RD = 1 recursion desired
QDCOUNT = 1
ANCOUNT = 0
NSCOUNT = 0
ARCOUNT = 0
Query 1
name: example.com
type: 1
class: 1
Query
Received 45 bytes.
ID = AB CD
QR = 1 response
OPCODE = 0 standard
AA = 0
TC = 0
RD = 1 recursion desired
RCODE = 0 success
QDCOUNT = 1
ANCOUNT = 1
NSCOUNT = 0
ARCOUNT = 0
Header
Query 1
name: example.com
type: 1
class: 1
Query
Answer 1
name: (pointer 12) example.com
type: 1
class: 1
ttl: 16661
rdlen: 4
Address 93.184.216.34
Answer
IP