CENG393 Computer Networks
Labwork 2

HTTP, NTP, DNS, FTP, SSH Protocols

1 HTTP - Hyper Text Transfer Protocol

HTTP (Hyper Text Transfer Protocol) is a standard protocol used mainly for transmission of web page contents. By default, it operates on TCP port 80. HTTP operates by request and reply messages. The requests are usually sent by web browsers and specifies which resource is requested from the server. The server responds to this request with a header and reply content.

1.1 Request

A typical HTTP request is of the following form:

```
GET /index.html HTTP/1.0
```

This request is made of three fields: HTTP method (Table 1), requested source (index.html, media/logo.jpg...) and the HTTP protocol (HTTP/1.0, HTTP/1.1, HTTP/2.0). Every HTTP request must be followed by two newlines (\n\n or sometimes \r\n\r\n).

1.2 Reply

```
HTTP/1.1 200 OK
Date: Wed, 27 Dec 2017 20:06:59 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 892
Connection: close
Content-Type: text/html; charset=UTF-8

<html>
  <head><title>Sample Web Page</title></head>
  <body><h1>Hello World!</h1></body>
</html>
```

This reply has two fields: reply header and reply content. HTTP headers usually contain information such as reply status code [1], reply length, server information, etc. Some of the most popular status codes are “200 OK” and “404 Not Found”. Reply content may contain the requested source or error messages if a problem occurs. Headers are seperated by contents with double newlines.

The code httpclient.c in Appendix section demonstrates basic HTTP usage.

For more information, you may refer to the textbook and also RFC (Request for Comments) document 2616 [2].

2 NTP - Network Time Protocol

NTP (Network Time Protocol) is a standard web protocol which is used for clock synchronization of devices that are connected to network. It can check and correct clock of a device even for milliseconds if it has been found to be incorrect.

There are clock servers around the earth, which always provide the correct universal time (UTC). These servers are organized in a hiearchy named "strata". Stratum 0 contains devices such as atomic clocks or GPS. Stratum 1 devices are synchonized to Stratum 0 devices within microseconds. Devices in lower stratum levels always synchonize themselves to devices in one higher level.

When a clock synchronization request to any of these servers is sent via UDP port 123, the server returns a reply containing the correct clock, that is used by the requesting device to correct its clock if necessary.

The initial version of NTP is proposed in 1985 [3]. The latest version, NTPv4 has been proposed in 2010 [4] and is backward compatible with previous NTP versions. Figure 1 shows the packet structure for NTPv4:

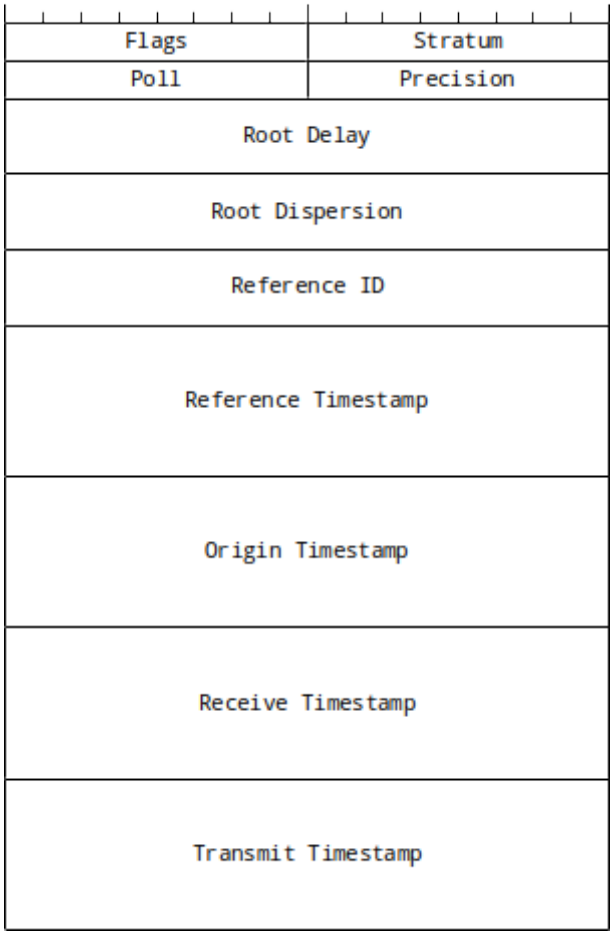

Figure 1: NTP Packet

- **Flags:** Leap indicator (2 bits), Version number (3 bits), Mode (3 bits)
- **Stratum:** Stratum level (0 - unspecified, 1 - primary, 2-15 - secondary, etc)
- **Poll:** Maximum interval between successive messages
- **Precision:** Precision of the system clock in log2 seconds.
- **Root Delay:** Total round-trip delay to the reference clock.

- **Root Dispersion:** Total dispersion to the reference clock.
- **Reference ID:** 32-bit code identifying the server.
- **Reference Timestamp:** Time when system clock was last set/corrected.
- **Origin Timestamp:** Time at the client when the request departed.
- **Receive Timestamp:** Time at the server when the request arrived.
- **Transmit Timestamp:** Time at the server when the response departed.

The code ntpclient.c in Appendix section demonstrates basic NTP usage.

3 DNS - Domain Name System

DNS (Domain Name System) is a naming service for devices connected over a network. It stores information such as IP addresses for stored domain names and returns these information when requested. When a request to a server is sent via its domain name (e.g. www.cankaya.edu.tr), this request is first forwarded to a DNS server; which receives the domain name and returns the IP address associated with it. The client then open a socket connection between itself and the server by using its IP address. Therefore, DNS can be imagined as the phonebook of the internet.

DNS operates on UDP port 53 on a request and reply basis. A request created by a client must obey the following scheme (Figure 2):

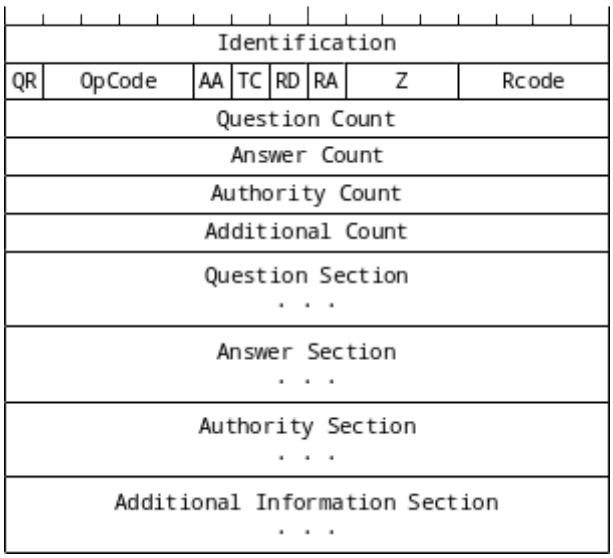

Figure 2: DNS Packet

- **Identification:** A unique identifier used for matching requests and replies.
- **QR:** Request (0) or Reply (1)
- **OpCode:** Request type
- **AA:** Authoritative Answer: Reply from authoritative (1) or from cache (0)
- **TC:** Truncated (1) or not (0)
- **RD:** Recursion desired: recursive (1) or iterative (0) response
- **RA:** Recursion available: server manages recursion (1) or not (0)

- **Z:** Reserved
- **Rcode:** Error codes
- **Count:** Number of Question / Answer / Authority / Additional Info entries.

For detailed information about DNS packet structure and how the protocol operates, you may refer to the textbook and also RFC documents 1034 [5], 1035 [6] and 2181 [7].

4 FTP - File Transfer Protocol

FTP (File Transfer Protocol) is a standard network protocol used for transmitting files between a server and clients. The computer running the FTP server software has a directory where all files that are served are stored in. When a client connects to the server, either by username authentication or anonymously, the client may browse the files published by the server, transfer them to their computer or upload files to the server.

FTP operates on two socket connections: one for command queries and one for data transmission. Command socket usually operates on TCP port 21. Some common FTP commands are user, pass, list, retr and stor.

Over years, FTP has proven to be an insecure protocol and many different attack types have target the protocol. In order to ensure safe and reliable file transfer, alterations (FTP over SSH, FTPS) to the protocol have been proposed.

For detailed information about FTP protocol and how it operates, you may refer to RFC document 959 [8].

5 SSH - Secure Shell

SSH (Secure Shell) is a standard server-client model network protocol used for operating remote systems using a secure private connection. It was designed to replace earlier unsecure protocols. It uses both symmetric and assymetric key based cryptography schemes to authenticate and establish connections between clients and the server.

Although the most common use for SSH is connecting to a remote server and execute commands on it, some of its other popular uses are forwarding, tunneling and secure file transfer.

For detailed information about SSH protocol and how it operates, you may refer to RFC documents 4251 [9], 4252 [10], 4253 [11] and 4254 [12].

6 Exercises

1. Study and execute httpclient.c in Appendix section. Modify it such that contents of a requested source are stored in appropriate files.
2. Study and execute ntpclient.c in Appendix Section. Modify it such that the time at client is written to origin timestamp in request packet. Then calculate the difference between received and transmitted timestamps.

Appendices

A httpclient.c

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

#define SIZE sizeof(struct sockaddr_in)

int main() {
    int sockfd, nread;
    short int port = 80;
    char buf[24000], address[256], request[256];
    struct sockaddr_in server;

    printf("Enter address of the server: ");
    scanf("%s", address);
    server.sin_family = AF_INET;
    server.sin_addr = *((struct in_addr*) gethostbyname(address)->h_addr);
    server.sin_port = htons(port);
    memset(&(server.sin_zero), 0, 8);

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        perror("socket");
        return 0;
    }
    if ((connect(sockfd, (struct sockaddr *) &server, SIZE)) == -1) {
        perror("connect");
        return 0;
    }

    strcpy(request, "GET / HTTP/1.0\r\n\r\n");
    send(sockfd, request, strlen(request) + 1, 0);

    nread = recv(sockfd, buf, 24000, 0);
    buf[nread] = '\0';
    printf("The response received is:\n%s", buf);
}
```

B ntpclient.c

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <time.h>
#define NTP_TIMESTAMP_DELTA 2208988800ull // seconds between Jan 01, 1900 and Jan 01, 1970

typedef struct _ntppacket {
    unsigned char flags;
    unsigned char stratum;
    unsigned char poll;
    unsigned char precision;
    unsigned int rootdelay;
    unsigned int rootdispersion;
    unsigned int referenceid;
    unsigned long int refts;
    unsigned long int orts;
    unsigned long int rcvts;
    unsigned long int trts;
} NTPPacket;

int main() {
    NTPPacket request, reply;
    char IP[] = "91.189.89.199"; // ntp.ubuntu.com
    struct sockaddr_in server;
    time_t received;
    int sockfd, port = 123, socklen = sizeof(struct sockaddr*);

    request.flags = 0x23; // leap=00, version=100, mode=011 (client)
    request.stratum = 0;
    request.poll = 0;
    request.precision = 0;
    request.rootdelay = 0;
    request.rootdispersion = 0;
    request.referenceid = 0;
    request.refts = 0;
    request.orts = 0;
    request.rcvts = 0;
    request.trts = 0;

    server.sin_family = AF_INET;
    server.sin_addr.s_addr = inet_addr(IP);
    server.sin_port = htons(port);
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);

    sendto(sockfd, &request, sizeof(NTPPacket), 0, (struct sockaddr*) &server,
            sizeof(struct sockaddr_in));
    recvfrom(sockfd, &reply, sizeof(NTPPacket), 0, (struct sockaddr*) &server,
            (socklen_t*) &socklen);

    received = (time_t) (ntohl((unsigned int) reply.trts) - NTP_TIMESTAMP_DELTA);
    printf("Time received: %s\n", ctime((const time_t*) &received));

    return 0;
}
```

C HTTP Request Methods and Response Codes

Option	Description
GET	The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.
HEAD	The HEAD method asks for a response identical to that of a GET request, but without the response body.
POST	The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server.
PUT	The PUT method replaces all current representations of the target resource with the request payload.
DELETE	The DELETE method deletes the specified resource.
CONNECT	The CONNECT method establishes a tunnel to the server identified by the target resource.
OPTIONS	The OPTIONS method is used to describe the communication options for the target resource.
TRACE	The TRACE method performs a message loop-back test along the path to the target resource.
PATCH	The PATCH method is used to apply partial modifications to a resource.

Table 1: HTTP Request Methods [13]

Codes	Description
100 - 199	Informational responses
200 - 299	Successful responses
300 - 399	Redirects
400 - 499	Client errors
500 - 599	Server errors

Table 2: HTTP Response Status Codes [1]

References

[1] MDN. Http response status codes. https://developer.mozilla.org/en-US/docs/Web/HTTP/Status, 2019.

[2] IETF Tools. Rfc2616 - hypertext transfer protocol. https://tools.ietf.org/html/rfc2616, 1999.

[3] IETF Tools. Rfc958 - network time protocol (ntp). https://tools.ietf.org/html/rfc958, 1985.

[4] IETF Tools. Rfc5905 - network time protocol version 4: Protocol and algorithms specification. https://tools.ietf.org/html/rfc5905, 2010.

[5] IETF Tools. Rfc1034 - domain names. https://tools.ietf.org/html/rfc1034, 1987.

[6] IETF Tools. Rfc1035 - domain names - implementation and specification. https://tools.ietf.org/html/rfc1035, 1987.

[7] IETF Tools. Rfc2181 - clarifications to the dns specification. https://tools.ietf.org/html/rfc2181, 1997.

[8] IETF Tools. Rfc959 - file transfer protocol. https://tools.ietf.org/html/rfc959, 1985.

[9] IETF Tools. Rfc4251 - the secure shell (ssh) protocol architecture. https://tools.ietf.org/html/rfc4251, 2006.

[10] IETF Tools. Rfc4252 - the secure shell (ssh) authentication protocol. https://tools.ietf.org/html/rfc4252, 2006.

[11] IETF Tools. Rfc4253 - the secure shell (ssh) transport layer protocol. https://tools.ietf.org/html/rfc4253, 2006.

[12] IETF Tools. Rfc4254 - the secure shell (ssh) connection protocol. https://tools.ietf.org/html/rfc4254, 2006.

[13] MDN. Http request methods. https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods, 2019.