*Eduardo Sanchez*

*Benjamin Rayner*

*ECE 358 Lab 2*

Socket Programming using Python

Table of Contents

[Task 1: Web Server 2](#_Toc119072567)

[Task 2: Authoritative DNS Server 4](#_Toc119072568)

[Task 3: Questions 8](#_Toc119072569)

# Task 1: Web Server

Webserver.py sets up a TCP socket on local host that accepts and parses all incoming GET/HEAD requests and attempts to respond with the requested file. This is done by first setting up the server TCP socket with the needed parameters (IP, port, IPv4, TCP, etc). Once we begin listening to the new socket, we wait in a while loop for incoming connections. When a connection with a client is accepted, the server returns a new socket to establish communication with the client. The server receives a HTML request header through this new socket and parses the first two strings for the type of request and file requested. Given this information, the server attempts to open the requested file. If the file exists, construct an appropriate HTML response header. If GET request send this header + file and if HEAD request just send the header. If the file does not exist, a 404 Not Found is sent back. Once the transaction is complete, close the connection and wait for the next client.

Text

Description automatically generated  
Fig 1: Webserver.py

Graphical user interface, text, application, chat or text message

Description automatically generated  
Figure 2: HTML sent to client

# Task 2: Authoritative DNS Server

Server.py sets up a UDP socket on local host that parses incoming DNS queries, retrieves the requested DNS entry, and sends a DNS response to the client. A dictionary is used to store DNS entries. Upon receiving data from a client, it saves their IP address so that it can respond later. If the server finds the domain requested in the database it will append its data to the received DNS question header, change the QR bit (to response), and change the ANCOUNT (num of entries).

Text

Description automatically generated  
Fig 3: Server.py

Client.py sets up a UDP socket that sends a constructed DNS query based on input from the user. It then parses the response from the server and displays the received DNS entry. Since every DNS answer is 16 bytes and has a standard structure, we are able index into each answer and parse TTL and the translated IP.

Text

Description automatically generated  
Fig 4: Client.py

Text

Description automatically generated  
Fig 5: Output of “google.com” (Client Side)

Text

Description automatically generated  
Fig 6: Output of “google.com” (Server Side)

|  |  |  |
| --- | --- | --- |
| **Type** | **Key** | **Value** |
| **DNS Header** | ID | 92 c4 |
| FLAGS | 04 00 |
| QDCOUNT | 00 01 |
| ANCOUNT | 00 00 |
| NSCOUNT | 00 00 |
| ARCOUNT | 00 00 |
| **Query** | QNAME | 06 67 6f 6f 67 6c 65 03 63 6f 6d 00 |
| QTYPE | 00 01 |
| QCLASS | 00 01 |

Table 1: “google.com” query color code

|  |  |  |
| --- | --- | --- |
| **Type** | **Key** | **Value** |
| **DNS Header** | ID | 92 c4 |
| FLAGS | 84 00 |
| QDCOUNT | 00 01 |
| ANCOUNT | 00 02 |
| NSCOUNT | 00 00 |
| ARCOUNT | 00 00 |
| **Query** | QNAME | 06 67 6f 6f 67 6c 65 03 63 6f 6d 00 |
| QTYPE | 00 01 |
| QCLASS | 00 01 |
| **Response** | NAME | c0 0c |
| TYPE | 00 01 |
| CLASS | 00 01 |
| TTL | 00 00 01 04 |
| RDLENGTH | 00 04 |
| RDATA | c0 a5 01 01 |

Table 2: “google.com” response color code

Text

Description automatically generated  
Fig 7: Output of “wikipedia.org” (Client Side)

Text

Description automatically generated  
Fig 8: Output of “wikipedia.org” (Server Side)

|  |  |  |
| --- | --- | --- |
| **Type** | **Key** | **Value** |
| **DNS Header** | ID | 61 36 |
| FLAGS | 04 00 |
| QDCOUNT | 00 01 |
| ANCOUNT | 00 00 |
| NSCOUNT | 00 00 |
| ARCOUNT | 00 00 |
| **Query** | QNAME | 09 77 69 6b 69 70 65 64 69 61 03 6f 72 67 00 |
| QTYPE | 00 01 |
| QCLASS | 00 01 |

Table 3: “wikipedia.org” query color code

|  |  |  |
| --- | --- | --- |
| **Type** | **Key** | **Value** |
| **DNS Header** | ID | 61 36 |
| FLAGS | 84 00 |
| QDCOUNT | 00 01 |
| ANCOUNT | 00 01 |
| NSCOUNT | 00 00 |
| ARCOUNT | 00 00 |
| **Query** | QNAME | 09 77 69 6b 69 70 65 64 69 61 03 6f 72 67 00 |
| QTYPE | 00 01 |
| QCLASS | 00 01 |
| **Response** | NAME | c0 0c |
| TYPE | 00 01 |
| CLASS | 00 01 |
| TTL | 00 00 00 a0 |
| RDLENGTH | 00 04 |
| RDATA | c0 a5 01 04 |

Table 4: “wikipedia.org” response color code

# Task 3: Questions

1. What is a socket?

A *socket* is one endpoint of a two-way communication link between two programs running on the network. (Oracle)

It is a door between application process and end-end-transport protocol. (Huang)

2. How do sockets work? Why is socket required?

A socket has a typical flow of events. In a connection-oriented client-to-server model, the socket on the server process waits for requests from a client. To do this, the server first establishes (binds) an address that clients can use to find the server. When the address is established, the server waits for clients to request a service. The client-to-server data exchange takes place when a client connects to the server through a socket. The server performs the client's request and sends the reply back to the client. (IBM)

Sockets allow you to exchange information between processes on the same machine or across a network, distribute work to the most efficient machine, and they easily allow access to centralized data. (IBM)

3. What are the types of Internet Sockets? Briefly explain the characteristics of each type

The types of sockets are:

* **Stream Sockets**
  + Reliable (packets received in order)
  + Connection-oriented communication
  + Less efficient
  + TCP as default
* **Datagram Sockets**
  + Unreliable (packets received out of order)
  + Connectionless communication
  + More efficient
  + UDP as default

(Huang)

4. Mention at least one application of each type of socket

* Stream sockets are used for implementations such as the File Transfer Protocol (FTP). (Microsoft)
* Datagram sockets can be used for sending and receiving broadcast messages. (GeeksforGeeks)

5.

socket() – creates a socket

bind() –  assign a name to the socket

connect() – initiate a connection

listen() – establish a queue for connections

accept() – extract a connection from the queue

close() – close the socket

(Huang)

6. TCP Flow Diagram

Diagram

Description automatically generated with low confidence

(Huang)

7. On which layer socket will execute in the Internet protocol stack?

Application Layer.

8.

|  |  |  |
| --- | --- | --- |
| Protocol | Port # | Common Function |
| HTTP | 80 | Helps users load websites and applications without typing in a long list of IP addresses. (Haber) |
| FTP | 20, 21 | Port 20 for data channel and port 21 for control channel. (Slattery) |
| IMAP4 | 143 | Used by eMail clients for the retrieval of their eMail from designated eMail "post office" servers. (Gibson) |
| SMTP | 25 | Used for relaying email on the internet. (Sobowale) |
| Telnet | 23 | Provides remote access to a variety of communications systems.  (*Cyber Security Certifications)* |
| POP3 | 110 | Used for unencrypted access to electronic mail. (Chapple) |

9. What is DNS and how does it work?

DNS (Domain Name System) is a distributed database implemented in hierarchy of many name servers. It is an application layer protocol where hosts and DNS servers communicate to resolve names (address/name translation). (Kurose)

# References

[1] J. F. Kurose and K. W. Ross: Computer Networking, A Top-Down  
Approach. 8th Edition.

[2] Oracle. (n.d.). *What is a socket?* What Is a Socket? (The Java™ Tutorials > Custom Networking > All About Sockets). Retrieved November 11, 2022, from <https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html>

[3] IBM. (n.d.). *How Sockets Work*. How sockets work. Retrieved November 11, 2022, from <https://www.ibm.com/docs/en/i/7.3?topic=programming-how-sockets-work>

[4] IBM. (n.d.). Socket programming. Retrieved November 11, 2022, from <https://www.ibm.com/docs/en/i/7.1?topic=communications-socket-programming>

[5] Microsoft. (n.d.). *Windows sockets: Stream sockets*. Microsoft Learn. Retrieved November 11, 2022, from <https://learn.microsoft.com/en-us/cpp/mfc/windows-sockets-stream-sockets>

[6] *Java.net.DatagramSocket class in Java*. GeeksforGeeks. (2021, November 30). Retrieved November 11, 2022, from <https://www.geeksforgeeks.org/java-net-datagramsocket-class-java>

[7] Haber, L. (2021, July 16). *What are port numbers and how do they work?* SearchNetworking. Retrieved November 11, 2022, from <https://www.techtarget.com/searchnetworking/definition/port-number>

[8] Slattery, T. (2022, August 16). *Understanding the FTP port command*. SearchNetworking. Retrieved November 11, 2022, from <https://www.techtarget.com/searchnetworking/tip/Understanding-the-FTP-PORT-command>

[9] Gibson, S. (n.d.). *GRC :  Port Authority, for internet port 143*. GRC | Port Authority, for Internet Port 143  . Retrieved November 11, 2022, from [https://www.grc.com/port\_143.htm](https://www.grc.com/port_143.htm#:~:text=Internet%20Message%20Access%20Protocol&text=Imap%20protocol%20is%20used%20by,of%20a%20remote%20eMail%20server)

[10] Sobowale, O. (2020, April 20). *WHAT SMTP port should I use?* SparkPost. Retrieved November 11, 2022, from https://www.sparkpost.com/blog/what-smtp-port

[11] *Cyber Security Certifications | GIAC certifications*. (n.d.). Retrieved November 11, 2022, from <https://www.giac.org/paper/gcih/328/tcp-port-23/103233>

[12] Chapple, M. (2009, January 26). *What are the security risks of opening port 110 and Port 25?* SearchSecurity. Retrieved November 11, 2022, from <https://www.techtarget.com/searchsecurity/answer/What-are-the-security-risks-of-opening-port-110-and-port-25>

[13] Huang, I. Y. (2022) *Socket Programming in Python* [PowerPoint slides]. Engineering, University of Waterloo. <https://learn.uwaterloo.ca/d2l/le/content/840942/viewContent/4631568/View>