Name: _____

IB Computer Science 2
Application-layer protocols worksheet

Application-layer protocols are how user-facing programs communicate. Most protocols are based around a client-server model where clients request information and servers respond with it. In this activity, we will look at two common examples.

HTTP

The World Wide Web is built on HyperText Transfer Protocol (HTTP), which it uses to transfer documents between Web servers and users' Web browsers.

1. Enter `localhost/file` in your Web browser's location bar. What happens and why?

2. Close the tab. Open a terminal and start the command `nc -l http`. Open a new browser tab to `localhost/file`. What does the browser do differently now?

3. You should see new output in your terminal. You are looking at the HTTP request that your browser sent `nc`, which is acting as a Web server. Copy down the first two lines.

   ```
   % nc -l http
   ```

4. The Uniform Resource Locator (URL) you visited was: `http://localhost/file`
   Draw an arrow from each of its components to the corresponding terminal text above.
   How would a request for `http://dev.java/learn/modernio` begin?

5. The `localhost/file` page should still be loading in your Web browser. In the same terminal as before, type the following into `nc`:
   ```
   HTTP/1.1 200 OK
   Content-Length: 11

   Hello world
   ```
   Hit enter at the end. What happens?

6. Reload the browser tab. Back in `nc`, try sending back some other text of your choice. What do you have to change about the HTTP response and why?

The first lines you typed were **headers**, or protocol metadata. The blank line separates the data.

If you have extra time…

7. The `200 OK` is a status code. You have probably seen others such as `404 Not Found` before. Reload the browser tab and try responding with the following:
   ```
   HTTP/1.1 302 Found
   Content-Length: 0
   Location: http://dev.java/learn/modernio
   ```
   Hit enter twice at the end. What happens?

8. What HTTP requests does your browser send when you visit a `bit.ly`-style shortlink?

DNS

The Domain Name System (DNS) protocol allows computers to find each other by name instead of number. To resolve a domain name, your operating system sends a request to a DNS server, which looks up the name in its database and responds with the corresponding numeric address.

Have you ever connected to public Wi-Fi without a network password, but then encountered a network login page when you tried to browse the Web? Such “captive portal” networks use a fake DNS server to resolve queries for any domain name to the address of the login page.

Visit `detectportal.firefox.com/success.txt` in your Web browser. This page is intentionally simple because the Mozilla Firefox browser uses it to detect captive portals.

9. Run the terminal command `nslookup -query=a detectportal.firefox.com` and look at the last line of output. What address is associated with this domain name?

10. Back in your browser, erase everything before the `/success.txt` and type `http://` followed by the address. What do you see?

Name: _____

If you have extra time…

11. Run `nslookup dev.java`. In your Web browser's location bar, enter `http://` followed by the address from the last line of output, followed by `/learn/modernio`. What happens?

12. Run `nslookup github.io`. What do you notice about the address(es) at the end of the output?

13. Visit `github.io` in your Web browser. How does the site compare to the Java Developer site? What must be going on?

14. Look back at the HTTP response you wrote down earlier. Like any networked computer, a Web server already knows its own address. What is the purpose of the `Host` header?

15. Open `nc` in client mode by running the command `nc github.io 80`. Looking back at the HTTP request you wrote down earlier, write a request for `http://dev.java/learn/modernio`. What do the `HTTP/1.1` and `Location:` lines of the response say?

16. How was the Web server at `github.io` able to answer a request meant for `dev.java`?