CY350 - Computer Networks Project

Milestone D –Web Application Enhancement (Comprehensive)

By now, you've built a TCP-based client-server web application with routers and basic routing functionality. In this comprehensive final milestone, you will apply what you have learned in CY350 to extend the web application to handle **HTTP POST requests, updating** the **Go-Back-N protocol** for reliable data transfer with **cumulative acknowledgements**, implementing **dynamic routing updates** based on topology changes, and designing a **Switch** class to simulate link-layer services between hosts and routers. You will also incorporate **encryption** to secure communications. This milestone will bring together concepts from all layers of the network stack and introduce improved features for both performance and security.

Skills and Concepts:

- Extending HTTP functionality to support POST requests.
- Implementing cumulative acknowledgements for reliable data transfer.
- Handling dynamic routing updates to adjust to network changes.
- Understanding the role of switches in link-layer services.
- Encrypting and decrypting data to ensure secure communication.

General Information

An autograder will **not** be used for Milestone D. The primary focus of the comprehensive project for CY350 is to assess your knowledge of the networking concepts highlighted above. Therefore, you must submit a written report that describes your approach to accomplishing the tasks (described in detail below) and the results of your efforts.

Project Milestone D Points (200 total)

- 1. Application Layer: Support for POST Requests (50 pts)
- 2. Transport Layer: Go-Back-N with cumulative acknowledgements (50 pts)
- 3. Network Layer: Handling Dynamic Routing Updates (35 pts)
- 4. Link Layer: Implementing Switches (35 pts)
- 5. Network Security: Encryption (30 pts)
- 6. Bonus: Open HTML Response on Client (20 pts)

Code: When required to develop code, you must use Python 3. We highly recommend that you use the programming environment configured in your EECSNet Virtual Machine (VM) for the course. Since the milestone requires raw sockets, the code is best run on Linux and must be run using administrative privileges.

The starter file (*milestone_d_starter_files.zip*) includes a full implementation of the web application and network described in the instructions for Milestone C.

---

1. Application Layer: Support for POST Requests

In Milestone A, you implemented GET requests. Now, you will extend your application to support **HTTP POST requests**, allowing the client to send data to the server.

Client-Side Modifications:

- Modify the client to generate and send HTTP POST requests. The POST request should allow the client to send data (e.g., form data) to the server.
- Example POST request format:

```
POST /resource HTTP/1.1
Host: <server>
Content-Length: <length of data>
<data>
```

Server-Side Modifications:

- The server should handle POST requests by receiving the data sent by the client and processing it. Recall that the POST request should result in an update to the resources available to the server. The server should then send an appropriate response back to the client (e.g., confirming the data was received).
- Example POST response:

```
HTTP/1.1 200 OK
Content-Type: text/plain
POST request successfully received.
```

1.a. In your report, provide one (1) paragraph describing your approach for implementing POST requests.

1.b. In the next subsection of the report, provide figures with the code you implemented to accomplish the task.

1.c. Explain (in 1-2 paragraphs) how you tested your implementation and provide evidence (in figures or tables) of it's correctness. The presence of flaws in your code is acceptable and will receive full credit so long as you demonstrate a reasonable approach to testing and identify that bugs still exist.

---

2. Transport Layer: Go-Back-N with cumulative acknowledgements

In Milestone B, you implemented the **Go-Back-N protocol** for reliable data transmission. Now, you will update Go-Back-N with cumulative acknowledgements, a more efficient means for communicating the successful receipt of data. The change only needs to be made for sending the response from the server and receiving it at the client.

Client-Side Modifications:

- Use cumulative acks when receiving the response from the server. The client should collect data segments for a prescribed period of time and, if they are consecutive, acknowledge them in one acknowledgement.
- *Hint*: the time period that the client collects data segments for must be less than the timeout on the server.

Server-Side Modifications:

- The server must handle the cumulative acknowledgement when sending a response. Be sure to appropriately manage 'base' as acknowledgements are received.

2.a. In your report, provide one (1) paragraph describing your approach for implementing cumulative acknowledgements.

2.b. In the next subsection of the report, provide figures with the code you implemented to accomplish the task.

2.c. Explain (in 1-2 paragraphs) how you tested your implementation and provide evidence (in figures or tables) of it's correctness. The presence of flaws in your code is acceptable and will receive full credit so long as you demonstrate a reasonable approach to testing and identify that bugs still exist.

---

3. Network Layer: Handling Dynamic Routing Updates

In Milestone C, you implemented basic routing functionality using a link-state algorithm. Now, you need to consider how the router must function when a change to the network topology or link cost occurs.

3.a. In your report, describe how the Router class and, potentially, the NetworkApp class need to be modified to achieve this.

3.b. In the next subsection of your report, provide an updated diagram for the Router Class**.** Update the attributes and methods in Router class diagram provided in the Milestone B instructions to create the ability to update the forwarding table upon a change to network topology or link cost, including any necessary intermediate steps.

---

4. Link Layer: Implementing Switches

In this milestone, you will design a **Switch** class to simulate link-layer services, sitting between the hosts (client and server) and the routers.

4.a. In your report, provide a class diagram for the Switch class. Be sure to include any attributes and methods required for the functionality of a switch. The format for the class diagram should follow that of the diagrams provided in previous milestones.

4.b. In the next subsection of your report, select the most significant method for the operation of the Switch, justify your selection, and provide pseudocode for implementing the method.

---

5. Network Security: Encryption

To secure data transmission, you will implement **encryption** at server and **decryption** at the client. All response data transmitted between the server and client should be encrypted via **symmetric encryption** to ensure confidentiality. Hint: Use the cryptography library for python. You should perform this task last so you can continue to see the payload in plaintext for troubleshooting the other tasks.

Server-Side Modifications:

- Implement encryption for the payload of outgoing packets (response) before sending them.

Client-Side Modifications:

- Implement decryption for the payload of incoming packets (response).

5.a. In the project report, explain your choice for encryption type.

5.b. In the next subsection of the report, provide figures with the code you implemented to accomplish the task.

5.c. Explain (in 1-2 paragraphs) how you tested your implementation and provide evidence (in figures or tables) of it's correctness. The presence of flaws in your code is acceptable and will receive full credit so long as you demonstrate a reasonable approach to testing and identify that bugs still exist.

---

6. Bonus: Open HTML Response on Client

Upon receiving the response to a request for 'index.html', the client should open the contents in the browser. Update the client so that it does so once the full response is received.

6.a. In the project report, provide the code and evidence (in figures or tables) that confirms your client opens 'index.html' in the browser after receiving it from the server.
