**CSC/CIS 419 Computer Networks**

**Programming Project: A Forward Proxy Agent**

**By: Jacob Bykowski**

**Introduction:**

Socket programming is a concept in computer networking that provides communication between processes running on different devices within a network. Essentially socket programming enables the establishment of endpoints (sockets) that facilitate the exchange of data, which allows diverse networked applications.

**Basic Principles:**

Socket programming operates on the client-server model, one side acts as a server, waiting for incoming connections, and another as a client, initiating communication. This model allows for seamless data exchange between devices. Communication is achieved through the use of sockets, which are software endpoints representing the communication channels between processes. There are two main types of sockets, TCP (Transmission Control Protocol) and UDP (User Datagram Protocol), each offer different reliability and performance. TCP ensures reliable, connection-oriented communication, while UDP provides speed over reliability.

**Main Procedures:**

The main procedures involved in socket programming include creating, binding, listening, accepting, connecting, and sending/receiving data through sockets.

**Creating Sockets:**

Sockets are created using the Socket class in Java, specifying the IP address and port number.

**Binding:**

The bind method associates a socket with a local address and port number, allowing it to receive incoming connections.

**Listening:**

A server socket enters a listening state, awaiting incoming connection requests from clients.

**Accepting:**

Upon receiving a connection request, the server socket accepts the connection, creating a new socket dedicated to that client.

**Connecting:**

Clients initiate connections by creating a socket and connecting it to the server's address.

**Sending/Receiving Data:**

Input and output streams associated with sockets facilitate the transmission of data between connected devices.

**Possible Applications:**

Common applications include web servers, email servers, instant messaging systems, online gaming, and, as demonstrated in this project, the implementation of a forward proxy. The forward proxy serves as an intermediary between clients and servers, forwarding client requests and fetching responses. This project showcases how socket programming enables the creation of such networking tools.

**Methods:**

**ProxyAgent Side:**

* **ForwardProxy Class:**

Listens on port 60000 for incoming client connections.

Accepts client connections and delegates handling to ProxyHandler instances.

* **ProxyHandler Class:**

Manages communication with individual clients.

Validates the special flag and processes client requests.

Utilizes WebPageFetcher to retrieve web pages from the Internet.

* **WebPageFetcher Class:**

Fetches web pages from the Internet.

Returns an HTTP response.

* **DisableSSLCertificateValidation Class:**

Disables SSL certificate validation with help of the TrustAllManagerClass

* **TrustAllManagerClass:**

Implements the X509TrustManager interface, which allows it to trust any SSL certificate without verification.

**Client Side:**

* **ProxyClient Class:**

Simulates a client sending a special command string to the forward proxy.

Receives and prints the response from the forward proxy.

**Results:**

**Proxy Agent Side (MIT.edu):**

**A black background with white text

Description automatically generated**

**Client Side (MIT.edu):**

**A screen shot of a computer

Description automatically generated**

\*Received full html of the MIT.edu website but couldn’t fit all in a screenshot

**Proxy Agent Side (Brockport.edu):**

A black screen with white text

Description automatically generated

**Client Side (Brockport.edu):**

**A black screen with white text

Description automatically generated**

\*Received full html of the Brockport.edu website but couldn’t fit all in a screenshot

**Contribution:**

In this project, I contributed to the design and implementation of the forward proxy application using socket programming in Java. Specifically:

* Developed the ForwardProxy class to handle incoming client connections.
* Implemented the ProxyHandler class to manage communication with clients and process requests.
* Designed the WebPageFetcher class to simulate fetching web pages from the Internet.
* Designed the DisableSSLCertificateValidation class to be able to work with the TrustAllManager class to establish a connection without having a verified certificate.
* Created the TrustAllManager class which allows it to trust any SSL certificate without verification, effectively disabling SSL certificate validation globally for the Java application.
* Created the ProxyClient class for testing client-server interactions.

**Conclusion:**

In conclusion, this application is designed to act as an intermediary between clients and web servers and uses socket programming to facilitate communication between entities.

The introduction showed the fundamental concepts of socket programming, and its basic principles and potential applications. Through the implementation of a forward proxy, I enabled secure and efficient communication in a networked environment.

The integration of a custom WebPageFetcher class demonstrated the capability to fetch web pages via HTTP, enhancing the overall functionality of the proxy.

The experimental results, obtained through the local host test provided insights into the reliability of the forward proxy. Identifying and resolving issues, such as SSL certificate challenges, showcased the practical problem-solving aspects of the development process.

My contribution to this project can be demonstrated by the design and implementation of the ProxyHandler class, which manages client-server communication, and the integration of SSL certificate validation mechanisms. The resolution of issues related to URL handling and SSL handshakes were key challenges in the project.

In conclusion, the forward proxy application represents a practical implementation of socket programming. The project not only increased my knowledge of network programming but also showed real-world scenarios in a networked environment.

**GitHub Link:** <https://github.com/Beefandbean/ProxyServerProject>