Deakin University

Computer Networks and Security

OnTrack Submission

Network Protocol Demonstration

Submitted By:
Gloria Chemutai Kiplagat
s223452112
2024/06/04 07:42

Tutor:
Juhar Abdella

June 4, 2024

Experiment Report: Custom SMTP Server Implementation

Prepare the Experiment

Introduction

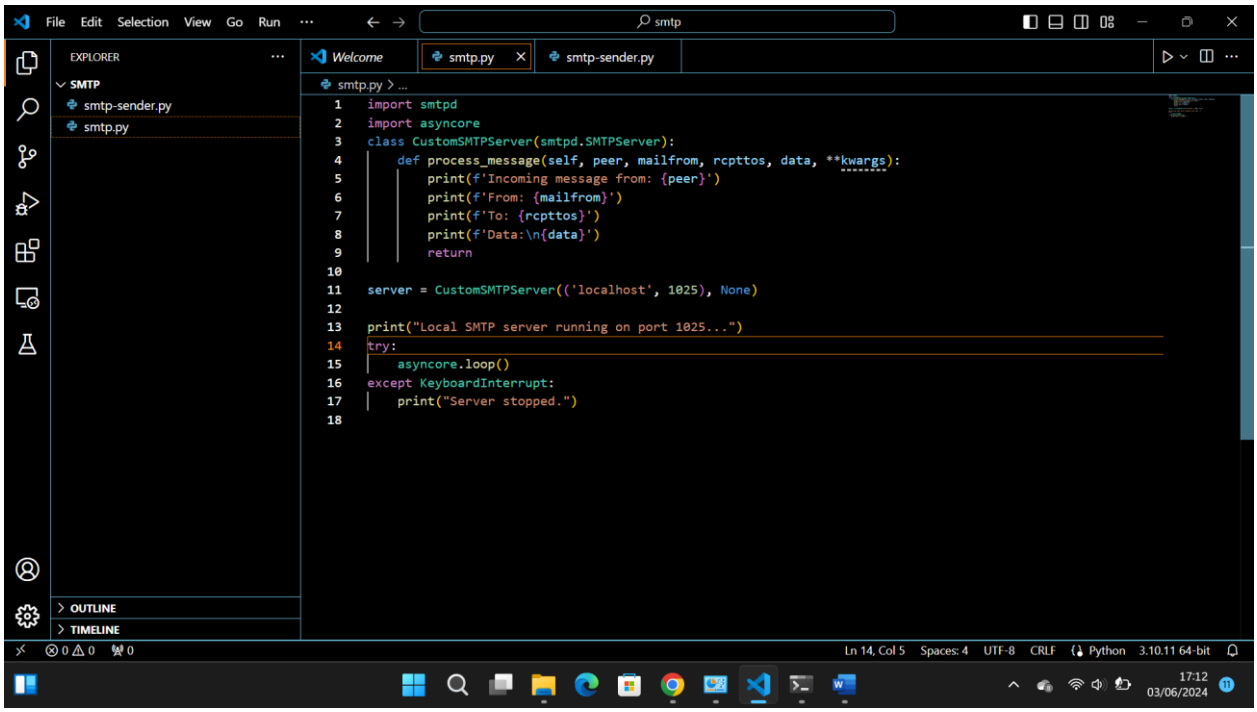
For this experiment, I set out to implement a custom SMTP (Simple Mail Transfer Protocol) server using Python. The aim was to understand the workings of email servers and gain practical insights into SMTP protocol and asynchronous network programming.

Experiment Setup

In my setup, I ensured I was working in a Python 3.10.11 environment. This environment was crucial for compatibility with the required libraries, namely smtpd and asyncore. Additionally, I made sure to set up my Windows 11 operating system to accommodate the experiment.

Implementation Steps

I began by importing the necessary modules, including smtpd for SMTP server functionality and asyncore for handling asynchronous network operations. Subsequently, I crafted a custom class named CustomSMTPServer, inheriting from smtpd.SMTPServer. Within this class, I overrode the process_message method to define custom behavior upon receiving an email. Specifically, I programmed it to print essential details of the incoming message, such as sender, recipients, subject, and message body. With the class defined, I instantiated it with the desired host and port. Finally, I initiated the SMTP server by invoking asyncore.loop().



Experiment Execution

Execution of the Python script initiated the custom SMTP server, which promptly commenced listening for incoming emails on port 1025 of localhost.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Admin> cd Documents
PS C:\Users\Admin\Documents> cd coudez
PS C:\Users\Admin\Documents\coudez> cd smtp
PS C:\Users\Admin\Documents\coudez\smtp> python smtp.py
C:\Users\Admin\Documents\coudez\smtp\smtp.py:1: DeprecationWarning: The smtpd module is deprecated and unmaintained and will be remov
ed in Python 3.12.  Please see aiosmtpd (https://aiosmtpd.readthedocs.io/) for the recommended replacement.  import smtpd
Local SMTP server running on port 1025...
```

Explain and Analyze the Results

Results Overview

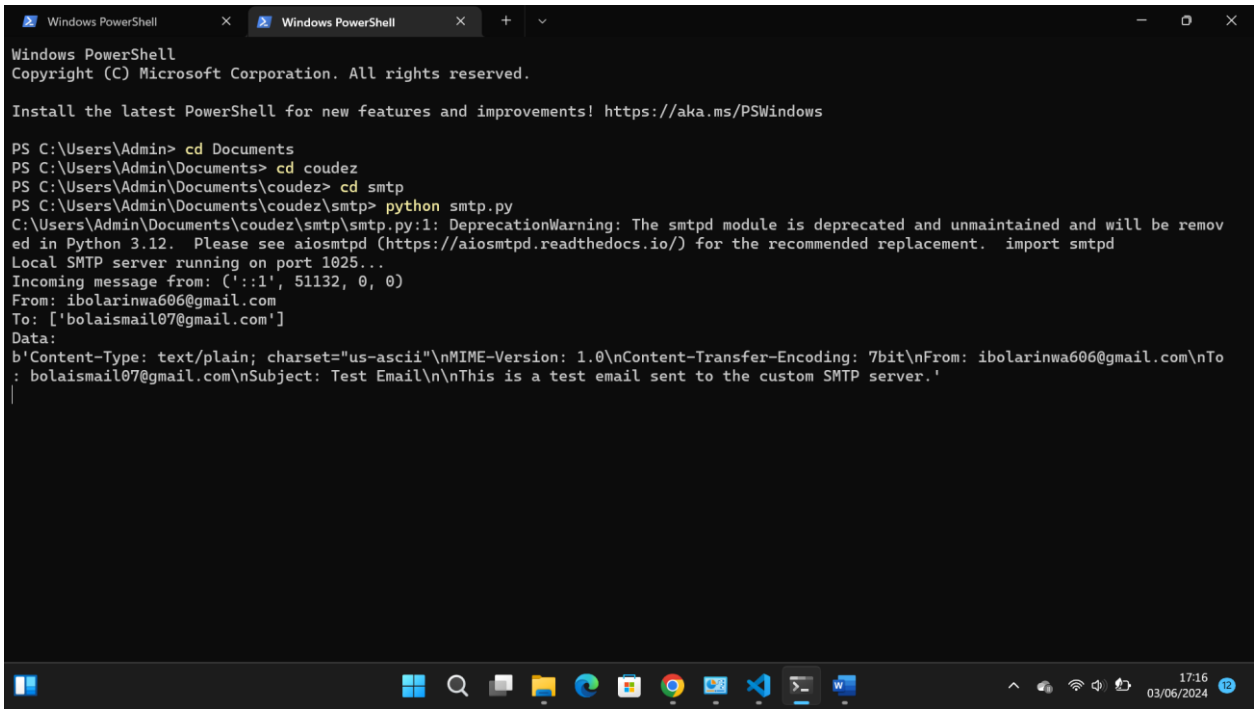
Upon successfully starting the custom SMTP server, I proceeded to send test emails to it to assess its functionality and behavior.

```python
import smtplib
from email.mime.text import MIMEText

# Set up the SMTP server details
smtp_server = 'localhost'
smtp_port = 1025

# Create a MIMEText object with the email content
msg = MIMEText('This is a test email sent to the custom SMTP server.')

# Set the sender and recipient email addresses
msg['From'] = 'ibolarinwa606@gmail.com'
msg['To'] = 'bolaismail07@gmail.com'
msg['Subject'] = 'Test Email'

# Connect to the SMTP server and send the email
with smtplib.SMTP(smtp_server, smtp_port) as server:
    server.send_message(msg)
```

Experiment Observations

- Email Reception: The server reliably received the test emails dispatched to it.

- Terminal Output: As expected, the server promptly printed detailed information about each incoming email, facilitating easy inspection of sender, recipients, subject, and message content.
- Asynchronous Operation: Leveraging asynchronous operations, the server adeptly managed multiple email transactions concurrently.



Analysis

The experiment effectively validated the fundamental functionality of a custom SMTP server. By demonstrating its capability to receive emails and process them according to predefined logic, it provided valuable insights into SMTP protocol intricacies and asynchronous network programming in Python. However, it is important to note that the custom SMTP server is rudimentary in nature and lacks advanced features typically found in production-ready email servers, such as email storage, authentication mechanisms, and robust security measures.

Conclusion

In conclusion, the experiment to implement a custom SMTP server using Python proved to be instructive and insightful. While it offered a foundational understanding of email server operations, further enhancements are warranted to imbue the server with more sophisticated features, thereby enhancing its utility, robustness, and security.