PA1 for CS305 2023 Fall: SMTP Server

Author: Ziyang LENG, Jinyi CUI

Inspector: Qing WANG

Introduction

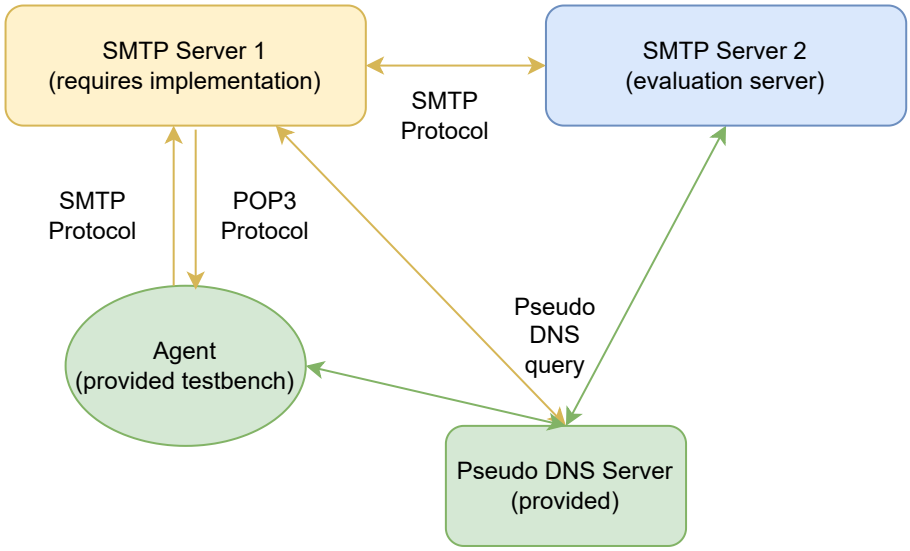
SMTP:

The **Simple Mail Transfer Protocol** (**SMTP**) is an Internet standard communication protocol for electronic mail transmission. Mail servers and other message transfer agents use SMTP to send and receive mail messages. User-level email clients typically use SMTP only for sending messages to a mail server for relaying, and typically submit outgoing email to the mail server.

POP-3:

The **Post Office Protocol** (**POP**) provides access via an Internet Protocol (IP) network for a user client application to a mailbox maintained on a mail server. The protocol supports list, retrieve and delete operations for messages. **POP3** is the version in most common use. Comparing with the original version, POP3 clients have an option to leave mail on the server after retrieval, and in this mode of operation, clients will only download new messages which are identified by using the UIDL command (unique-id list).

In this assignment, your goal is to implement a simple **SMTP Server** in Python, which should be able to support e-mail sending and retrieving in a proper way. In order to give a clearer understanding, the figure below shows the overall structure of this assignment, the green parts are provided and **YELLOW** parts requires implementation.



Modules in the above figure covers:

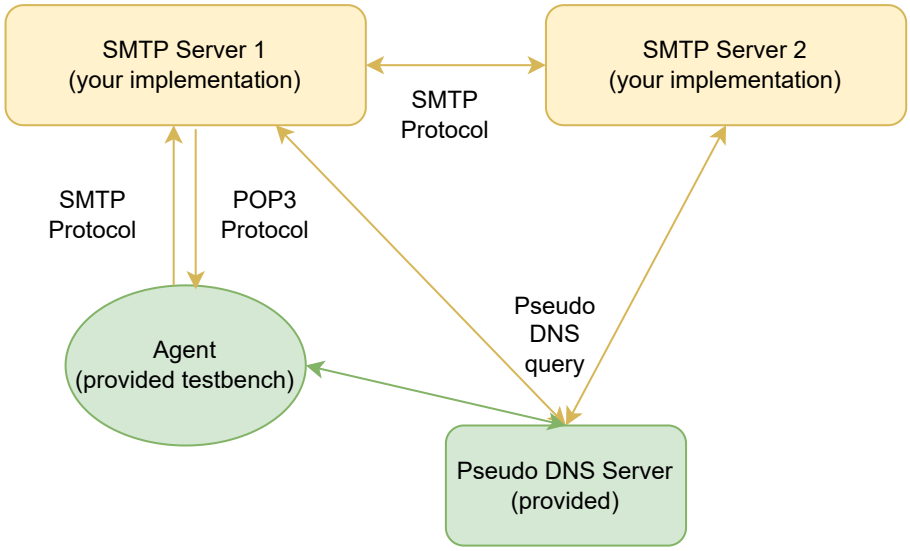
1. An **Agent** that can: send and receive email, perform pseudo DNS query to lookup the servers.

2. A **SMTP Server** that support: POP3 & SMTP protocol transmission with Agent, SMTP protocol transmission with another SMTP Server, and performs pseudo DNS query to lookup servers.
3. A **Pseudo DNS Server** that handles: pseudo DNS queries from Agent & Server.

Typically the SMTP server provider will assgin specific ports for their SMTP and POP3 service and deploy on separate servers. However, to adapt the DNS lookup process for local testing, we retain the *MX record* while replacing the *A record* with the *P record* which returns the port of the queried domain rather than ip address. Then we can use *localhost* and the returned *port number* to establish TCP connections to different servers.

```
[MX]
"mail.sustech.edu.cn." = "mxbiz1.qq.com."
"gmail.com." = "gmail-smtp-in.l.google.com."

[P]
"smtp.exmail.qq.com." = "1025"
"smtp.gmail.com." = "2025"
"pop.exmail.qq.com." = "3110"
"pop.gmail.com." = "2110"
"mxbiz1.qq.com." = "1025"
"gmail-smtp-in.l.google.com." = "2025"

```

During implementation, you can test your server anytime with the structure below by simply start two SMTP server processes from your implementation. In this case, the structure comes to:



For more detailed information, please refer to section *Hands-on Tutorial*. If you have any questions about this assignment, you are welcomed to raise an issue in this GitHub repository. By the way, you can read these documents (online or offline. For the offline documents, you can click the "Offline" links below to open them, or check them in the ./references dictionary.) for a better understanding of this assignment:

- Wikipedia:
  - Simple Mail Transfer Protocol (Online)
  - Simple Mail Transfer Protocol(Offline)
  - Post Office Protocol (Online)
  - Post Office Protocol (Offline)
  - List of SMTP server return codes (Online)
  - List of SMTP server return codes (Offline)
- List of SMTP & POP3 commands
- RFC 1939: Definition of POP3

- RFC 5321: Definition of SMTP

Environment Setup

Python 3.7+ in Windows/MacOS/Linux is required to run all the modules, and your code submitted will be tested in **Python 3.10**. By the way, a few dependencies needs to be installed if you have not. To do this, you can use cmd under the **project directory** (default: .../assign1/src) and type this command:

```
pip install -r requirements.txt
```

Tasks

In the following tasks you can **ONLY** use the Python standard library, **excluding** *smtplib*, *poplib* and all the libraries provided by us. Failure to comply with this rule will lead to a 0 score for this assignment, and the result **CANNOT** be appealed.

We have provided a skeleton file called **server.py**, you can implement your code in this file. For the first step, you **MUST** replace the id in **"student_id()"** function with your own SID. Except for this, do not modify the function names and the corresponding parameter lists.

Task 1: Implement POP3 protocol transmission handling (30pts)

- The following are common commands and usage of POP3:

CMD	Description	Usage
USER	Authentication username.	USER **username**
PASS	Authentication password.	PASS **password**
STAT	Return mailbox status.	STAT
LIST	Returns the total number of bytes for the specified message. If not specified, returns all separately.	LIST x / LIST
RETR	Returns the text for the specified message.	RETR x
DELE	Pre delete the specified email. But it is actually deleted when QUIT is executed.	DELE x
RSET	Revoke all DELE commands.	RSET
NOOP	Return a positive response.	NOOP
QUIT	End the session.	QUIT

**In the above chart, x is the index of a specific e-mail. The bold parts in the above chart need to be replaced with specific values according to the actual situation.*

For the commands above, the command names are not case sensitive, but some parameters are case sensitive for authentication reasons. For example:* **USER usr1 == user usr1, *but* **USER usr1 != USER Usr1**.

In this task, you should implement a basic POP3 server that can:

- **Handle connections (5 pts):** In default, POP3 listens on port 110. However, for security concerns, you are supposed to use other ports. Successfully establishing connnection with agent and verifying user with password will lead to full marks in this part. For sample settings, please refer to ".../src/data/config.toml" and ".../src/data/fdns.toml". (i.e. support command: `USER` + `PASS`)
- **Return mailbox status (5 pts):** To simplify, you just need to return the number of emails and the total number of bytes in emails. (i.e. support command: `STAT`)
- **Testing and managing emails (10 pts):** In this part, you are supposed to list, retrieve, (pre)delete emails, and also reset the deletion. Further, a function that simply returns a positive response is required. (i.e. support command: `LIST` + `RETR` + `DELE` + `RSET` + `NOOP`)
- **End the session (10 pts)**: To end the session, you should first clear up the emails you marked as (pre)delete. After that, disconnect the server with the agent and return back. (i.e. support command: `QUIT`)

Task 2: Implement SMTP protocol transmission handling (60 pts)

- The following are common commands and usage of SMTP:

CMD	Description	Usage
HELO	Start a session.	helo/ehlo [**ip_addr of sender**]
MAIL	Start sending mails.	mail FROM:<**e-mail_addr of the sender**>
RCPT	Tells the mail address of the receptors.	rcpt TO:<**e-mail_addr of receptors**>
DATA	The content of the mail.	data
QUIT	End the session.	quit

**The bold parts in the above chart need to be replaced with specific values according to the actual situation.*

For the commands above, the command names are not case sensitive, but some parameters are case sensitive for authentication reasons. For example:* **mail FROM:<123abc@gmail.com> == mail from:<123abc@gmail.com>, *but* **mail FROM:<123abc@gmail.com> != mail FROM:<123ABC@gmail.com>**.

In this task, you should implement a basic SMTP server that can:

- **Handle connections (10 pts):** Successfully establish connection with agent and other SMTP servers (utilizing pseudo DNS query). (i.e. support command: `HELO`)
- **Anonymous logins (10 pts):** Verify the existence of sender if the connection is from an agent, and the existence of receiver if the connection is from another SMTP server. Generate a delivery failure message and mail it back to the sender if verification above failed. (i.e. support command: `HELO`)
- **Transfer mails (30 pts):** Successfully receive the data bytes and deliver to the correct mailbox or SMTP server. When sending the mail to all recipients, if the recipient's domain is same with the sender, put it into the mailbox. Otherwise, send it to the recipient's server. (i.e. support command: `MAIL` + `RCPT` + `DATA`)
- **End the session (10 pts):** Require the receptor to send a farewell message and stop the connection. Before the sender receives the farewell message, neither side can disconnect, even if there is an error in the transmission. (i.e. support command: `QUIT`)

Task 3: Advanced functions (up to 10 pts)

There are also some points for you if you finish this assignment more elegant or implement some advanced functions. The following are some tips for what you can do. You will receive no more than 10 points for this part.

- **Error handling (5-10 pts)**: Connection refused, illegal mail address, illegal command, ...
- **More commands (2 pts each)**: Implement more commands other than those required in Task 1

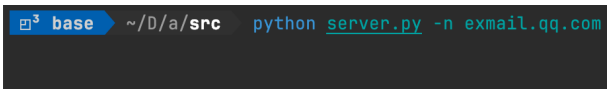
  and 2. e.g. `UIDL`, `HELP`, `VRFY`, `SOML`, `SAML`, etc.
- **Peer mailing (5 pts)**: Achieve the e-mail sending and receiving between you and your friends. If you choose one or more students in our class as your partner, all of you can get the 5 pts.
- Other work that you think is worth doing.

Hands-on Tutorial

To test your implementation through the structure in above figure, you can run the agent and server using the commands below,

```
# Agent
python agent.py -e <email_addr> -p <password>
# SMTP Server
python server.py -n <domain_name>
```

The way to simply start an agent and a server: (Just an example, not a test case.)



```
base ~/D/a/src python agent.py -e usr1@mail.sustech.edu.cn -p pass1
[smtp|pop|exit]>>> pop
b'+OK POP3 server ready'
b'+OK'
b'+OK user successfully logged on'
[pop]>>> stat
0 messages (0 bytes)
[pop]>>> list
[]
[pop]>>> noop
b'+OK'
[pop]>>> quit
b'+OK'
```

The demo test case is provided here. You can execute the instructions in the order shown in the example for testing. Also, there is a sample output for the server and wireshark when emailing myself. You can also refer to them for the **output format checking**. The final test script **includes but is not limited** to the commands displayed in the sample output. **All commands** listed above will be tested.

- Start two server, three agents and email each other:

1. Using the following command to start two server.

```
base ~/D/a/src python server.py -n gmail.com
```

```
base ~/D/a/src python server.py -n exmail.qq.com
```

2. Using the following command to connect agent to server, send or retrieve email.

```
base ~/D/a/src python agent.py -e usr1@mail.sustech.edu.cn -p pass1
[smtp|pop|exit]>>> smtp
To: usr1@mail.sustech.edu.cn
To:
Subject: Test
Content: This is a test emailing myself.
[smtp|pop|exit]>>> smtp
To: usr2@mail.sustech.edu.cn
To:
Subject: Test
Content: This is a test emailing other user in the same domain.
[smtp|pop|exit]>>> smtp
To: usr@gmail.com
To:
Subject: Test
Content: This is a test emailing other user in the different domain.
```

```
base ~/D/a/src python agent.py -e usr1@mail.sustech.edu.cn -p pass1
[smtp|pop|exit]>>> pop
b'+OK POP3 server ready'
b'+OK'
b'+OK user successfully logged on'
[pop]>>> stat
1 messages (195 bytes)
[pop]>>> list
[b'1 195']
[pop]>>> retr 1
b'Content-Type: text/plain; charset="utf-8"'
b'MIME-Version: 1.0'
b'Content-Transfer-Encoding: base64'
b'Subject: Test'
b'From: usr1@mail.sustech.edu.cn'
b''
b'VGhpcyBpcyBhIHRlc3QgZW1haWxpbmcgbXlzZWxmLg=='
```

```
base ~/D/a/src python agent.py -e usr2@mail.sustech.edu.cn -p pass2
[smtp|pop|exit]>>> pop
b'+OK POP3 server ready'
b'+OK'
b'+OK user successfully logged on'
[pop]>>> stat
1 messages (223 bytes)
[pop]>>> list
[b'1 223']
[pop]>>> retr 1
b'Content-Type: text/plain; charset="utf-8"'
b'MIME-Version: 1.0'
b'Content-Transfer-Encoding: base64'
b'Subject: Test'
b'From: usr1@mail.sustech.edu.cn'
b''
b'VGhpcyBpcyBhIHRlc3QgZW1haWxpbmcgb3RoZXIgdXNlciBpbiB0aGUgc2FtZSBkb21haW4u'
```

```
base ~/D/a/src python agent.py -e usr@gmail.com -p pass (base)
[smtp|pop|exit]>>> pop
b'+OK POP3 server ready'
b'+OK'
b'+OK user successfully logged on'
[pop]>>> stat
1 messages (233 bytes)
[pop]>>> list
[b'1 233']
[pop]>>> retr 1
b'Content-Type: text/plain; charset="utf-8"'
b'MIME-Version: 1.0'
b'Content-Transfer-Encoding: base64'
b'Subject: Test'
b'From: usr1@mail.sustech.edu.cn'
b''
b'VGhpcyBpcyBhIHRlc3QgZW1haWxpbmcgb3RoZXIgdXNlciBpbiB0aGUgZGlmZmVyZW50IGRvbWFp'
b'bi4='
```

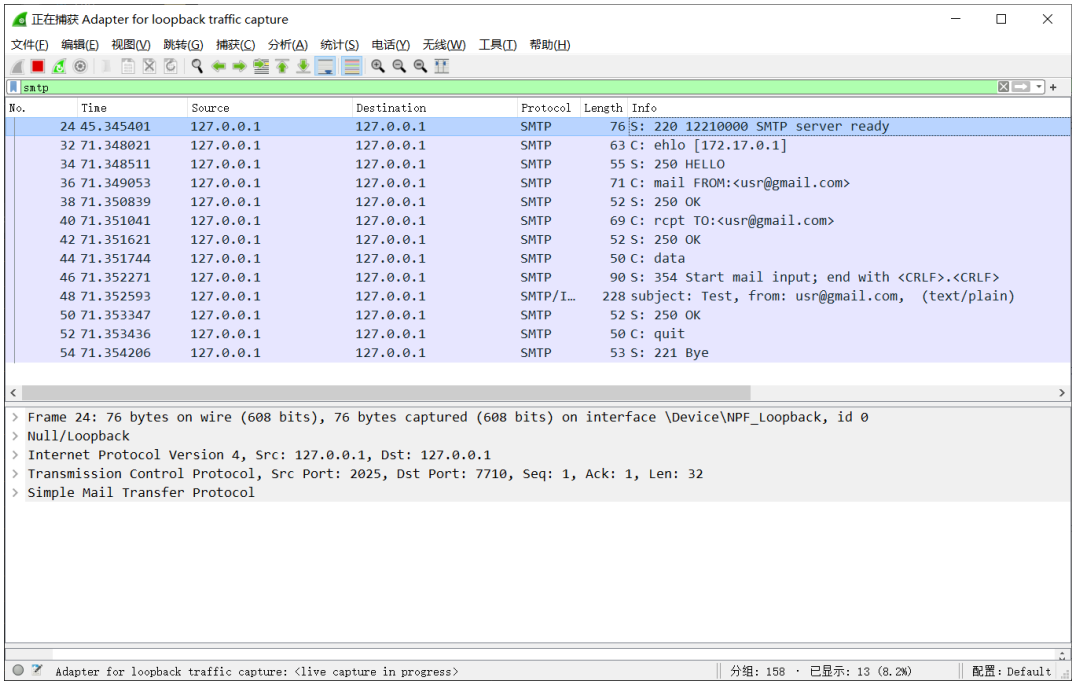
- Sample output for the server (Only for emailing myself):

```
管理员: C:\Windows\System32\cmd.exe - python  server.py -n gmail.com
Microsoft Windows [版本 10.0.19041.264]
(c) 2020 Microsoft Corporation. 保留所有权利。

E:\                                                    >python server.py -n gmail.com
>>> Received: b'ehlo [172.17.0.1]\r\n'
>>> Response: 250 HELLO

>>> Received: b'mail FROM:<usr@gmail.com>\r\n'
>>> Response: 250 OK

>>> Received: b'rcpt TO:<usr@gmail.com>\r\n'
>>> Response: 250 OK

>>> Received: b'data\r\n'
>>> Response: 354 Start mail input; end with <CRLF>.<CRLF>

b'Content-Type: text/plain; charset="utf-8"\r\nMIME-Version: 1.0\r\nContent-Transfer-Encoding: base64\r\nSubject: Test\r
\nFrom: usr@gmail.com\r\n\r\nVGhpcyBpcyBhIHRlc3QgZW1haWxpbmcgbXlzZWxmLg==\r\n.\r\n'
>>> Response: 250 OK

>>> Received: b'quit\r\n'
putting message into usr@gmail.com's mailbox which is in this server
>>> Response: 221 Bye

>>> Received: b'USER usr@gmail.com\r\n'
>>> Response: +OK

>>> Received: b'PASS pass\r\n'
>>> Response: +OK

>>> Received: b'STAT\r\n'
>>> Response: +OK

>>> Received: b'LIST\r\n'
>>> Response: +OK

>>> Received: b'RETR 1\r\n'
>>> Response: +OK
```

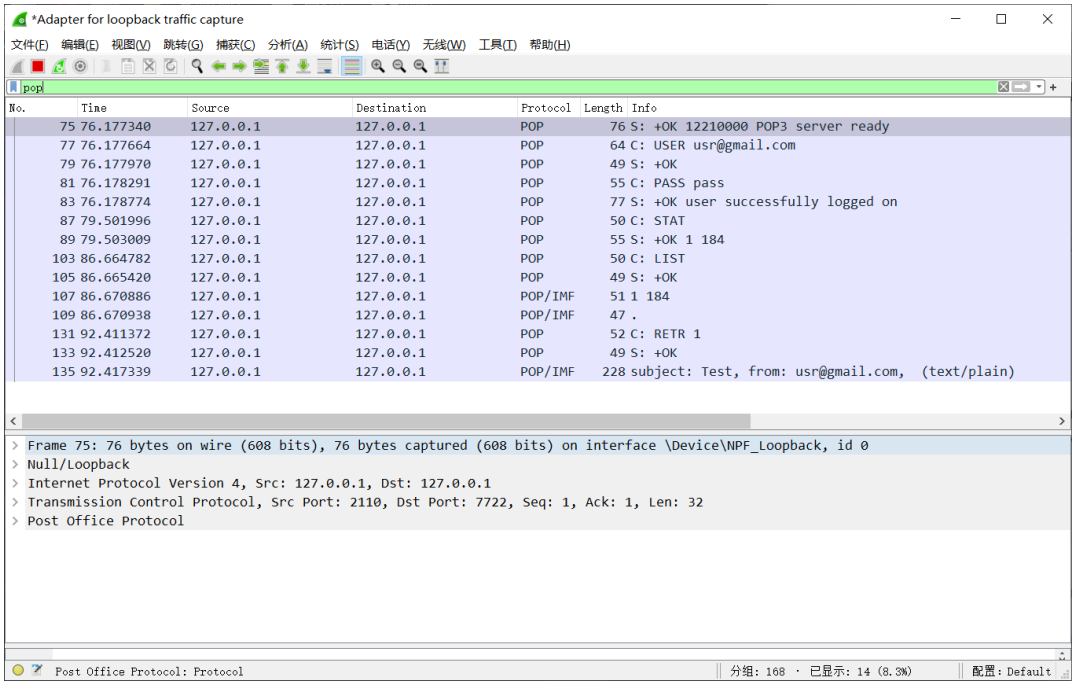
- Sample output for Wireshark (Only for emailing myself):

  (Should you have any question about the usage of Wireshark, you can refer to the appendix.)

  - SMTP packages:



  - POP3 packages (Only for emailing myself):



If you are confused with the response code and response body corresponding to the command, remember to refer to **wireshark result pictures** or **wikipedia documents** we provided above.

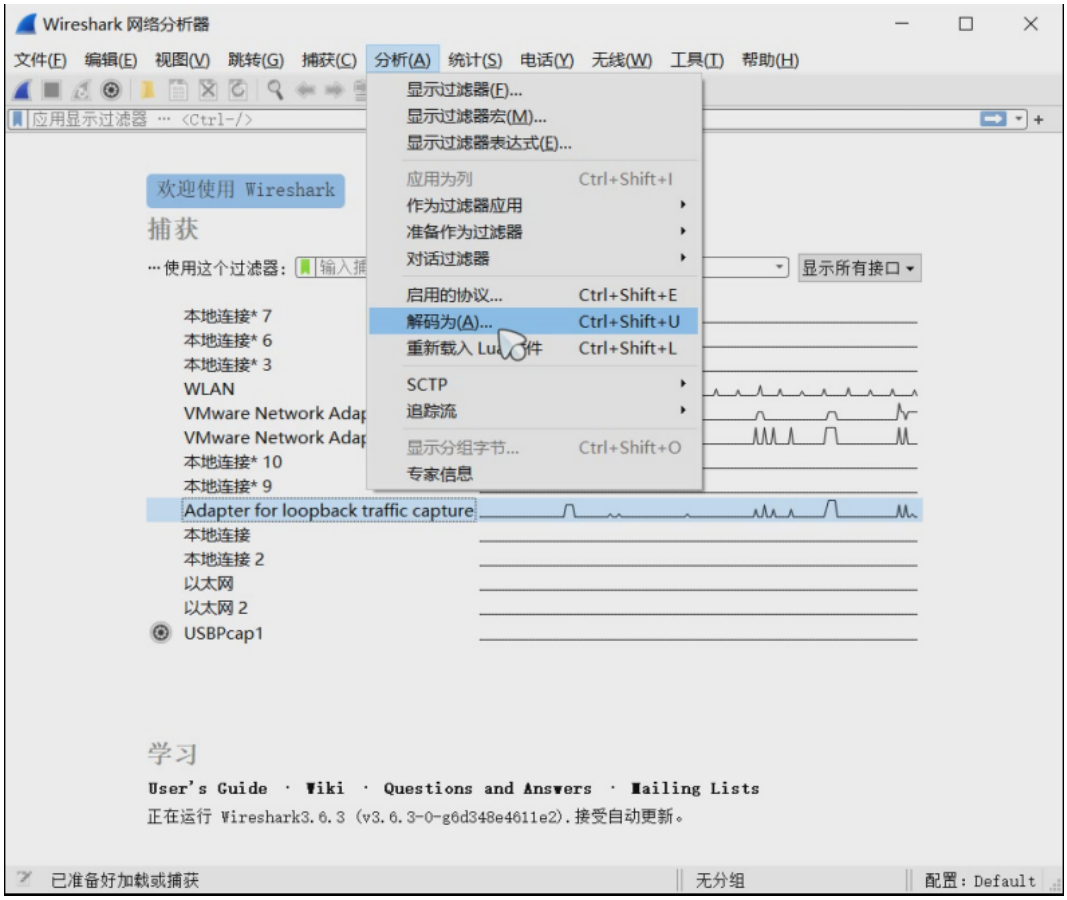
Grading Policies

- You should turn in a **zip** file *and* a **pdf** file. The zip file should include all of your code, and all of the functions should be implemented in the main file of the code named **server.py**. As for the pdf file, you should include the screenshot of the result of the testing script, which will be released shortly on the GitHub repository, AND the screenshot of the Wireshark packets captured during the testing procedure. You should properly set the filter so that only the packets related to this assignment are shown, otherwise we will **deduct 1~2 points** from your final score on Blackboard.
- If there are any points specified in section “Task 3: Advanced functions”, their functionalities and screenshots of the code should be included in the pdf file, **otherwise they will not be considered!** Your implementation will be scored according to the criteria listed in the Task section.
- Remember to change the welcome message of the server as instructed, otherwise you will get a **0** score for this assignment.
- You are not permitted to copy any code from other students or the Internet. **Your code will be checked for duplicates along with (not only) all submitted assignments**. If you are found plagiarizing, your score will be recorded as 0 for this assignment, and subject to punishment as shown in the rules.
- Should you have any questions, please raise an issue here. This is also the place where later materials, announcements, and clarifications will be published, so remember to check it out frequently.

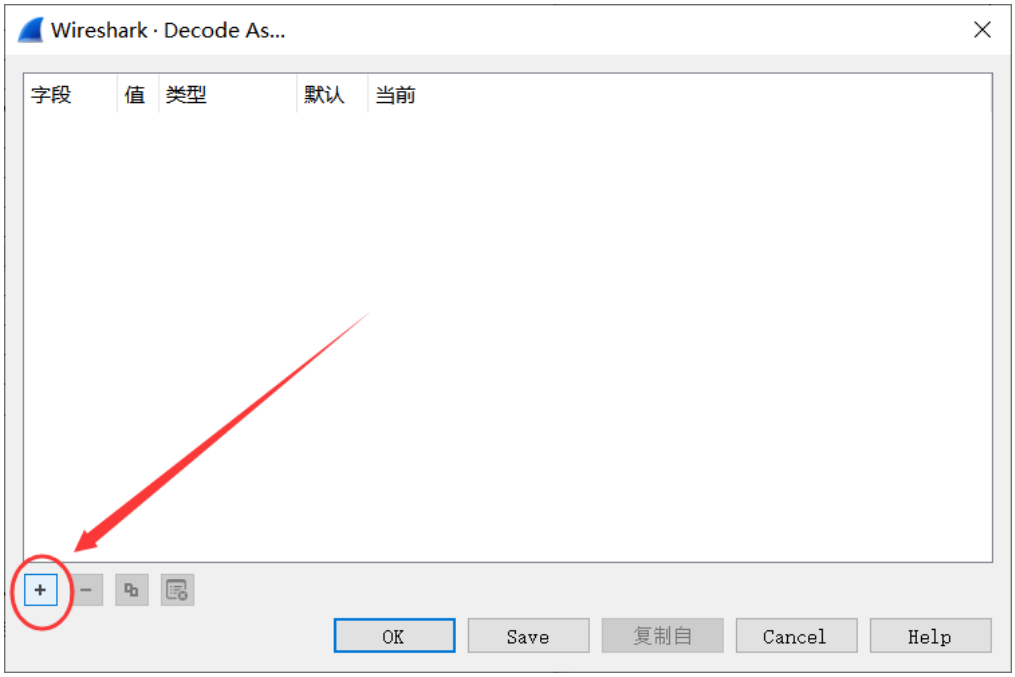
Appendix

Here is a detailed manual for how to use Wireshark to capture SMTP and POP3 packages.

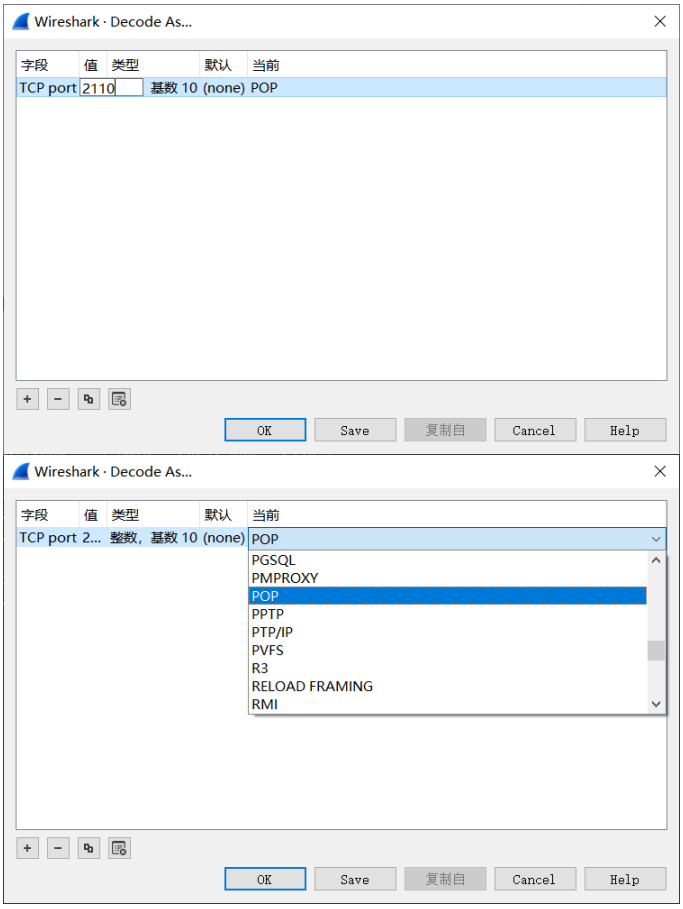
① Open Wireshark and choose **"Analysis" -> "Decode as"**:



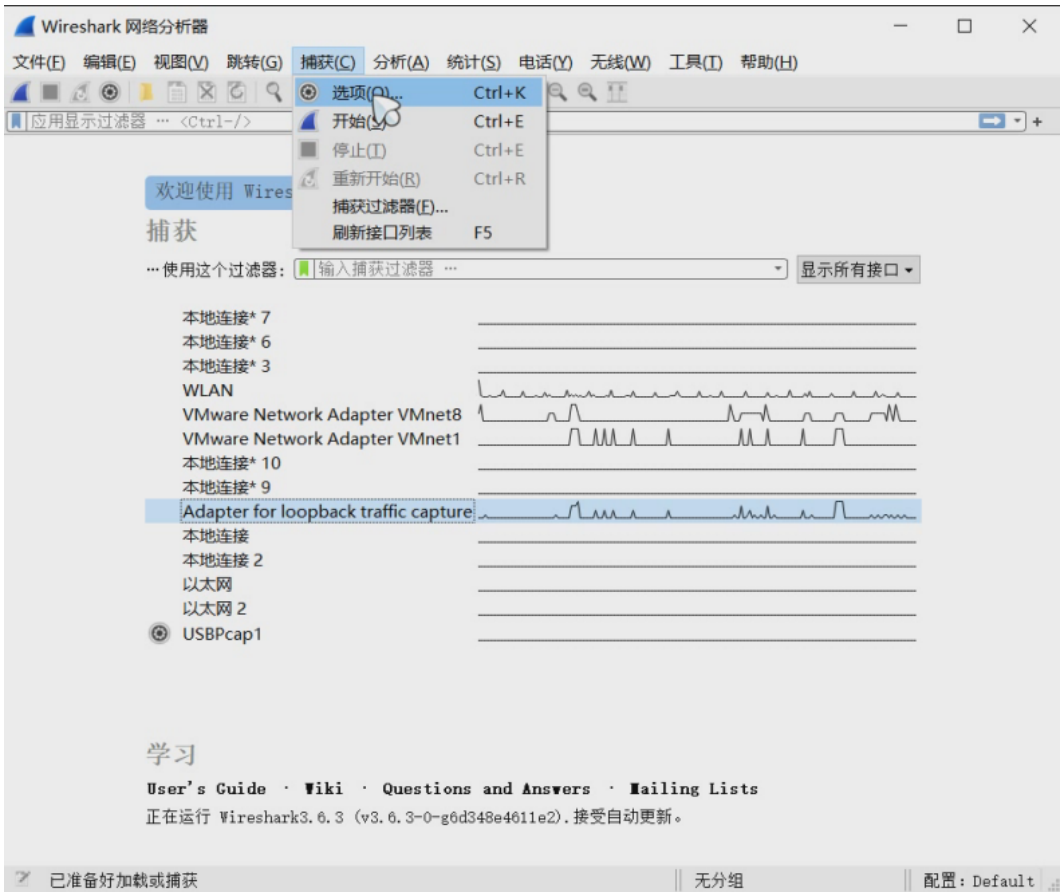
② Click "+".



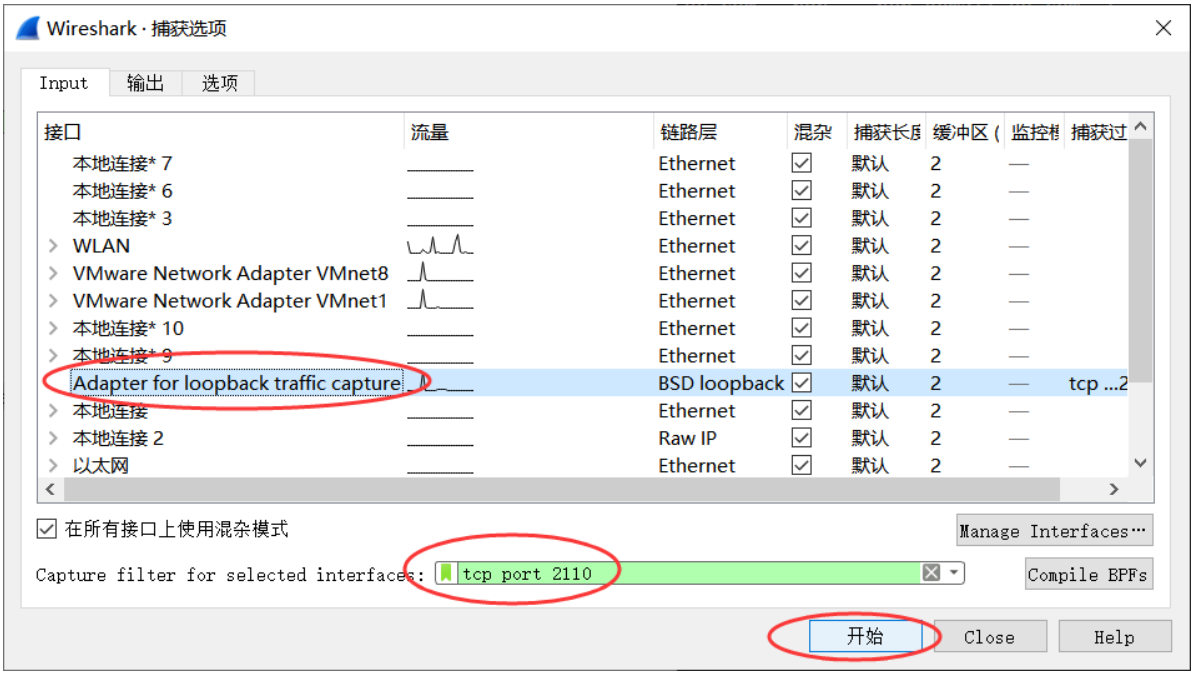
③ Take POP for gmail.com (port 2110) as an example, set **"Value"** as the port number 2110, and **"Current"** as POP, then click "OK":



④ Choose **Capture -> Options**:



⑤ Set the interface as **"Adapter for loopback traffic capture"**, filter as **"tcp port 2110"** in this example, then click "Start":



⑥ Execute your code in order, and you can capture the corresponding POP packets.

⑦ **For SMTP**, you only need to change the **"Current"** option as SMTP, and replace each of the above areas that involve setting port with the new SMTP port.