SMTP-клиент

Задача

Разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции клиента протокола SMTP.

Основные возможности. Приложение должно реализовывать следующие функции: 1. Создание нового письма, включающего такие поля, как From (отправитель), To (получатель), Subject (тема), Carbon copy (дополнительные адресаты), Blind copy (дополнительные скрытые адресаты), Body (текст) 2. Формирование всех необходимых заголовков письма, с тем, чтобы приёмная сторона не рассматривала данное письмо как спам. 3. Подключение к указанному SMTP-серверу и отсылка созданного письма 4. Подробное протоколирование соединения клиента с сервером.

Поддерживаемые команды. Разработанное приложение должно реализовывать следующие команды протокола SMTP: - HELO – передача серверу информации о домене пользователя - MAIL FROM – передача серверу адреса отправителя письма - RCPT TO – передача серверу адреса получателя письма - DATA – передача серверу тела письма - QUIT – завершение сеанса связи

Настройки приложения. Разработанное приложение должно обеспечивать настройку следующих параметров: 1. Собственное доменное имя для передачи в команде HELO 2. Адрес отправителя 3. IP-адрес или доменное имя сервера SMTP

Методика тестирования. Для тестирования приложения следует использовать почтовые серверы, имеющиеся в лаборатории, а также бесплатные почтовые серверы, имеющиеся в сети Internet (http://www.mail.ru, http://www.yandex.ru, http://www.rambler.ru, и т.п.). Средствами разработанного приложения осуществляется передача письма на указанные ящики электронной почты. Штатными клиентами электронной почты проверяется корректность доставки почты и правильность параметров письма.

Теоритические сведения протокола SMTP

Предоставленная информация соответсвует RFC 5321 — Протокол SMTP

Базовая структура

SMTP — требующий соединения текстовый протокол, по которому отправитель сообщения связывается с получателем посредством выдачи командных строк и получения необходимых данных через надёжный канал, в роли которого выступает TCP-соединение. SMTP-сессия состоит из команд, посылаемых SMTP-клиентом, и соответствующих ответов SMTP-сервера. Когда сессия открыта, сервер и клиент обмениваются её параметрами. Сессия может включать ноль и более SMTP-операций. После организации коммуникационного канала и согласования параметров клиент SMTP обычно инициирует почтовую транзакцию. Такая транзакция состоит из последовательности команд, задающих отправителя и получателя сообщения, а также передачи содержимого письма (включая все заголовки и прочие структуры).

Обзор процедур SMTP

Инициирование сеанса

Сеанс SMTP инициируется, когда клиент соединяется с сервером и сервер отвечает

соответствующим сообщением. Реализация сервера SMTP может включать идентификацию своих программ и сведения об их версии в отклик подтверждения соединения после кода 220. Протокол SMTP позволяет серверу формально отвергать транзакцию, не запрещая изначальные соединения: код 554 может возвращаться в открывающем сообщении взамен кода 220.

Инициирование клиента

После того, как сервер передал приглашающее сообщение (приветствие) и клиент получил его, последний передает серверу команду EHLO, идентифицирующую клиента. В дополнение к открытию сеанса использование EHLO показывает, что клиент способен работать с расширенным сервисом и запрашивает у сервера список поддерживаемых им расширений. Хост, передающий команду EHLO, идентифицирует в ней себя; команду можно интерпретировать как «Hello, I am » (Привет, я домен and I support service extension requests»

Почтовые транзакции

Почтовая транзакция SMTP состоит из трех этапов. Началом транзакции служит команда MAIL, дающая идентификацию отправителя. После этого следует одна или несколько команд RCPT, указывающих получателей сообщения. Последний этап транзакции начинается командой DATA, которая инициирует передачу почтовых данных и завершается индикатором end of mail, который также подтверждает транзакцию.

Порядок команд

Для порядка использования команд существуют некоторые ограничения. Сеанс, который будет включать почтовую транзакцию, должен быть сначала инициализирован командой EHLO. Команда EHLO может вводиться клиентом в действующем сеансе. Если команда EHLO неприемлема для сервера SMTP, он должен возвращать отклик 501, 500, 502 или 550. Клиент SMTP предоставляет в параметрах команд EHLO первичное доменное имя своего хоста.

AUTH – аутентификация и шифрование. Команда AUTH сообщает серверу механизм аутентификации. Если сервер поддерживает запрашиваемый механизм аутентификации, то он выполняет аутентификационный протокольный обмен, для того чтобы установить подлинность и идентифицировать пользователя. Опционально сервер также договаривается об уровне безопасности. В случае если запрашиваемый механизм аутентификации не поддерживается, сервер отклоняет команду с кодом ответа 504. Аутентификационный протокольный обмен состоит из серии запросов сервера и ответов клиента зависящих от механизма аутентификации. При получении команды аутентификации от клиента, сервер отправляет клиенту ответ с кодом 334 (ответ о готовности) и текстовой частью содержащей BASE64-закодированную строку. Ответ клиента состоит из BASE64-закодированной строки. В случае если сервер не может декодировать BASE64-аргумент, он отклоняет команду AUTH с кодом ответа 501. Если сервер отклоняет утентификационные данные, то ему СЛЕДУЕТ отклонить команду AUTH с кодом ответа 535. При успешном завершении клиентом аутентификационного обмена, SMTP сервер отвечает кодом ответа 235.

Команда MAIL начинает почтовую транзакцию. После начала транзакции последняя включает начальную команду, одну или несколько команд RCPT и команду DATA, вводимые в указанном порядке. Почтовая транзакция прерывается командой RSET, новой командой EHLO или командой QUIT. В сеансе может происходить множество последовательных транзакций или не быть транзакций вообще. Недопустимо передавать команду MAIL, если почтовая транзакция уже открыта, т. е., эту команду можно передавать только при отсутствии в сеансе продолжающейся почтовой транзакции — предыдущая транзакция должна быть завершена успешным выполнением команды DATA или прервана командой RSET или новой командой EHLO. Если аргумент начинающей транзакцию команды неприемлем, должен возвращаться отклик 501 и сервер SMTP должен сохранять свое состояние. Если в сеансе

нарушается порядок команд в такой степени, что это препятствует их выполнению сервером, последний должен возвратить отклик 503, сохраняя свое состояние. Последней командой сеанса должна быть команда QUIT. Клиентам следует использовать команду QUIT для разрыва соединения даже в тех случаях, когда команда организации сеанса не была передана и воспринята.

Архитектура приложения

Представленное приложение реализовано на языке Python v3.4.

Клиент поддерживает следующие команды:

- EHLO
- AUTH
- MAIL
- RCPT
- DATA
- QUIT

Команды клиента и положительные ответы сервера, реализованные на клиенте SMTP:

Команда	Описание	Положительный ответ сервера
EHLO {домен}	Инициализация сеанса	250
AUTH LOGIN {логин BASE64} {пароль BASE64}	Авторизация на сервере	334 - ответ на логин; 235 – авторизация пройдена успешно
MAIL FROM: <адрес отправителя>	Задание адреса отправителя	250
RCPT TO: <адрес получателя>,<...>	Задание адресов получателя	250
DATA {заголовки и текст} .	Задание письма со всеми заголовками и полями	250
QUIT	Разрыв соединения	221

Команда DATA имеет собственные заголовки:

Команда	Описание	
From:	Адрес отправителя	
To:	Адреса получателей	
Subject: адрес отправителя>	Тема сообщения	
Cc:	Адреса вторичных получателей, которым отправляется копия	250
Bcc:	Скрытые адреса получателей	250

При посылке сообщения без темы пользователь получает предупреждение, что писбмо может быть расценено как спам. *BODY* Тело письма отделяется от заголовка пустой строкой, а заканчивается строчкой, содержащей точку и CRLF.

Классы и методы

class SMTP() - класс, содержащий поля и методы, необходимые для организации работы SMTP-клиента.

В классе реализованы следующие методы:

__connect_to_server(self, host, port) - соединение с сервером

__send_cmd(self, cmd, args="") - посылка команды на сервер

__receive_code(self) - получение кода ответа от сервера

__disconnect_from_server(self) - рассоединение с сервером

__print_response(self, code=0) - вывод кода ответа с описанием пользователю

ehlo(self, domain) - реализация команды EHLO

auth(self, login, password) - реализация команды AUTH

mail_from(self, email) - реализация команды MAIL

rcpt_to(self, email) - реализация команды RCPT

data(self, email_from="", email_to="", subject="", cc="", bcc="", data="") - реализация команды DATA

quit(self) - реализация команды QUIT

class Message() - класс, содержащий поля и методы, требуемые для формирования заголовков и тела сообщения.

В классе реализованы следующие методы:

header(self, email_from, email_to, subject, cc, bcc) - формирование заголовка

body(self, body) - формирование тела сообщения

class SMTP_Error(BaseException) - класс, необходимый для формирования ошибок исключительных ситуаций

Для этого в файле *SMTP_client.py* описан словарь с кодами ответов сервера и их описание, а также все коды распределены на группы: 1. Коды ответов, которые разрешают повторной отправки команды без повторного соединения 2. Коды ответов, которые не разрешают повторной отправки команды, и при которых происходит разъединение связи

В файле *main.py* организовано непосредственное взаимодействие с пользователем.

Дизайн приложения

При запуске прилоэжения SMTP-клиент запрашивает у пользователя адрес сервера и номер порта, по к которому он хочет обратиться. Отметим, что данное приложение реализует зашифрованное SSL-содединение, поэтому в большинстве случаев требуется использование порта № 465. После успешного соединения с сервером, клиент запрашивает имя домена и отправляет команду EHLO.

В результате успешной инициализации соединения с сервером, клиент запрашивает у пользователя его логин, и в случае подтверждения сервером последнего, пароль (в незашифрованном виде, преобразование в Base64 осуществляет клиент). При первой попытке ввода логина и пароля разрешается использовать неполный логин, но в случае ошибочного ввода логина или пароля, последующая попытка требует имени пользователя вместе с доменом.

Пример:
Fully-Qualified LOGIN:
PASS:

После успешной авторизации начинается почтовая транзакция.

1. Клиент просит ввести e-mail отправителя, который будет виден получателям: (Почтовый адрес должен совпадать с логином, введенным при авторизации)

Enter your e-mail:

2. Клиент просит ввести адреса получателей через запятую.

   Enter target e-mails separated by ",": - основных получателей
   Enter cc e-mails separated by ",": - получателей в копию
   Enter bcc e-mails separated by ",": - скрытых получателей

Данные адреса проверяются сервером, и в случае успеха, клиенту отправляется код 250, который сообщает клиенту о том, что можно вводить тему и текст сообщения.

Сначала запрашивается тема сообщения. В случае, если пользователь пропускает ввод темы, клиент предупреждает его о том, что сообщение может быть расценено как спам, и просит ввести тему снова: После ввода темы, пользователь начинает ввод текста сообщения. В случае успешной отправки сообщения клиент получает код 250, и отправляет серверу команду QUIT.

Тестирование

Данное приложение тестировалось двумя способами. При этом были использованы различные SMTP-серверы, такие как: 1)mail.ru 2)yandex.ru 3)rambler.ru 4)gmail.com

1) *Ручное тестирование*

Для организации ручного тестирования было проведенено несколько тестов для проверки работоспособности отдельных команд, группы команд, и корректноых обработок ошибочных ситуаций.

Ниже представлены некоторые из этих тестов.

- Тест на проверку аутентификации Данный тест проверяет, что ошибка неверного ввода логина/пароля обрабатывается корректно, после чего клиент просит пользователя ввести логин/пароль еще раз. Заметим, что после повторных ошибок ввода нельзя использовать сокращенный логин (без домена). Однако при первой и удачной попытке - можно (это видно из следующего теста).

  Fully-Qualified LOGIN: knazarova9
  PASS: qqqqqqq
  AUTH LOGIN a25hemFyb3ZhOQ==
  cXFxcXFxcQ==
  535: Authentication credentials invalid
  Fully-Qualified LOGIN: knazarova9
  PASS: Q12345
  AUTH LOGIN a25hemFyb3ZhOQ==
  UTEyMzQ1
  535: Authentication credentials invalid
  Fully-Qualified LOGIN: knazarova9@mail.ru
  PASS: Q12345
  AUTH LOGIN a25hemFyb3ZhOUBtYWlsLnJ1
  UTEyMzQ1
  235: Authentication Succeeded

- Тест на успешную аутентификацию

  Fully-Qualified LOGIN: knazarova9

PASS: Q12345
AUTH LOGIN a25hemFyb3ZhOQ==
UTEyMzQ1
235: Authentication Succeeded

- Тест на проверку обработки ошибки при задании несуествующих получателей и доставке письма всем заданным получателям, в том числе получателей для копии и скрытых (поля заголовка cc: и bcc:).

Enter target e-mails separated by ",": k1
Enter cc e-mails separated by ",": k2
Enter bcc e-mails separated by ",": k3
['k2', 'k1', 'k3']
RCPT TO:
501: Syntax error in parameters
Enter target e-mails separated by ",": k-nazarova@mail.ru, knazaova9@rambler.ru
Enter cc e-mails separated by ",": knazarova9@gmail.com
Enter bcc e-mails separated by ",": knazarova9@yandex.ru
['k-nazarova@mail.ru', 'knazarova9@yandex.ru', 'knazarova9@gmail.com', ' knazaova9@rambler.ru']
RCPT TO:k-nazarova@mail.ru
250: OK
RCPT TO:knazarova9@yandex.ru
250: OK
RCPT TO:knazarova9@gmail.com
250: OK
RCPT TO:knazaova9@rambler.ru
250: OK

- Тест на проверку возможности отправки спама

Enter subject of your e-mail:
Your message may be mistaken for spam
Enter subject of your e-mail: Hello

2) *Автоматизированное тестирование*

Для тестирования реализованного SMTP-клиента были созданы автоматизированные тесты, описанные в файле test_SMTP.py. Для этого использовался фреймфорк unittest.py.

Тесты проводились с использованием SMTP-сервера mail.ru и порта 465:

HOST = "smtp.mail.ru"
PORT = 465
DOMAIN = "mail.ru"

Реализованные методы и классы.

class TestSMTP(unittest.TestCase) - класс, содержащий тест-кейсы для проверки общей работоспособности SMTP-клиента.

В классе реализованы следующие методы:
setUp(self) - инициализация соединения
test_mailbox_unavailable(self) - проверка обработки ошибки на недоступность сервера

test_smtp_successful(self) - успешная отправка письма
test_ehlo_failure(self) - проверка обработки ошибки исполнения команды EHLO
test_auth_failure_with_repetition(self) - проверка обработки ошибок при аутентификации
test_auth_success_after_failures(self) - успешное аутентификация после ошибок
test_mail_from_failre(self) - обработка ошибок при задании отправителя
test_for_wrong_input_sequence_of_cmd(self) - обработка ошибок при неверной последовательности команд
test_fail_with_unexpected_error(self) - обработка исключительных ситуаций
tearDown(self) - метод, инициирующий разъединение с сервером посредством команды QUIT
send_to_each_one(self, emails) - вспомогательный метод для отправки писем

class TestSMTPSendMsg(unittest.TestCase) - класс, содержащий тест-кейсы для проверки формирования сообщения

В классе реализованы следующие методы:
setUp(self) - инициализация соединения, авторизация, задание адреса отправителя
test_rcpt_to_failure(self) - проверка обработки ошибок при задании неверных получателей
test_rcpt_to_many_recipients_successfully(self) - успешная отправка письма нескольким получателям
test_create_msg_without_subj(self) - проверка обработки ситуации при создании сообщения без темы
test_create_msg_successful(self) - проверка на успешное создание сообщения
tearDown(self) - метод, инициирующий разъединение с сервером посредством команды QUIT
send_to_each_one(self, emails) - вспомогательный метод для отправки писем

TestSMTPConnection(unittest.TestCase) - класс, содержащий тест-кейсы для проверки соединения

В классе реализованы следующие методы:
test_connection_fail_with_OSError(self) - проверка обработки исключения OSError при попытке подключения к серверу
test_connection_fail_with_WinError(self) - проверка обработки исключений TimeoutError, WindowsError при попытке подключения к серверу
test_connection_fail_with_wrong_host_or_port(self) - проверка обработки ошибок при использовании несуществующего хоста и порта
test_connection_fail_after_success(self) - обработки разъединения с сервером во время активного сеанса

С помощью утилиты coverage было проверено покрытие тестами приложения. Результат:

Module	statements	missing	excluded	coverage
SMTP_client.py	**112**	**0**	**0**	**100%**
test_SMTP.py	**136**	**0**	**0**	**100%**
Total	**248**	**0**	**0**	**100%**

Вывод

В результате выполнения данной работы было разработано приложение для операционных систем семейства Windows, обеспечивающее функции клиента протокола SMTP:

1. Создание нового письма, включающего такие поля, как From (отправитель), To (получатель), Subject (тема), Carbon copy (дополнительные адресаты), Blind copy (дополнительные скрытые адресаты), Body (текст)
2. Формирование всех необходимых заголовков письма, с тем, чтобы приёмная сторона

не рассматривала данное письмо как спам.
3. Подключение к указанному SMTP-серверу и отсылка созданного письма
4. Подробное протоколирование соединения клиента с сервером.

Для организации корректной работы клиента были реализваны следующие команды протокола:

- EHLO – передача серверу информации о домене пользователя
- MAIL FROM – передача серверу адреса отправителя письма
- RCPT TO – передача серверу адресов получателя письма
- DATA – передача серверу тела письма
- QUIT – завершение сеанса связи

Готовое приложение успешно протестрировано с полным покрытием всех возможных ветвлений в коде.