Network protocol – Mail Application

By: Noam Mansur 305596785, Alon Kollman 203565650

|  |  |  |  |
| --- | --- | --- | --- |
| **General Comments** | | | |
| * The server can address only one client at a time. * The connection between the server and the client is created at the start of the client program run, and disconnects only after the client did a proper sign out or if the client process was killed. Only after that happens, the server can address a different client. | | | |
| **General Message Structure** | | | |
| Each Message consists of two ‘size’ bytes followed by the content of the message.  The size in the ‘size’ bytes is the size in bytes of the content (meaning we don’t take in account the two ‘size’ bytes in that number).  Maximum length of message is SMALL\_BUFFER\_SIZE (see under constants values) unless stated otherwise. | | | |
| **Constant Values** | | | |
| DEFAULT\_PORT | 6423 | The default port. | |
| SUCCESS\_MSG | “Success” | Message sent on success of certain operations. | |
| FAIL\_MSG | “Failure” | Message sent on failure of certain operations. | |
| SMALL\_BUFFER\_SIZE | 100 | Maximum length of most messages. | |
| BIG\_BUFFER\_SIZE | 5000 | Maximum length of some messages. | |
| MAX\_USERNAME | 50 | Maximum length of username. | |
| MAX\_PASSWORD | 50 | Maximum length of password. | |
| MAX\_SUBJECT | 100 | Maximum length of email subject. | |
| MAX\_CONTENT | 2000 | Maximum length of email content. | |
| TO\_TOTAL | 20 | Maximum users that one can send a message to. | |
| **Connection and authentication** | | | |
| After establishing connection between the two parties, the server should send a ‘greeting’ message.  After receiving the greeting, The client should send a message of the form: ‘USER;PASS’ where USER is the username, and password is PASS , both limited to MAX\_USERNAME and MAX\_PASSWORD respectively.  The server then returns a SUCCESS\_MSG or FAIL\_MSG according to whether the state of the authentication. | | | |
| Client Message: | ‘USER;PASS’ | Server Response: | SUCCESS\_MSG or FAIL\_MSG |
|  | | | |
| **Mail Operations** | | | |
| Composing a new email | | | |
| To compose a new email, the client should send two messages:  First one is ‘COMPOSE’ and after it a message of the form: ‘RECEIPIENTS;SUBJECT;CONTENT’ where:   * RECIEPIENTS is a string of usernames separated by commas (‘NAME,NAME,NAME…’) where each username is limited to MAX\_USERNAME and the number of users to be on the recipients list is limited to TO\_TOTAL. * SUBJECT is the subject of the email, limited to length MAX\_SUBJECT. * CONTENT is the content of the email, limited to MAX\_CONTENT.   RECIEPIENTS, SUBJECT and CONTENT should not contain semicolons or new line characters. The sever responds with SUCCESS\_MSG or FAIL\_MSG. | | | |
| Client Message: | ‘COMPOSE’ | Server Response: | No Response |
| Client Message: | ‘RECEIPIENTS;SUBJECT;CONTENT’ | Server Response: | SUCCESS\_MSG or FAIL\_MSG. |
| Checking inbox | | | |
| To check one’s inbox, the client sends the message: ‘SHOW\_INBOX’.  The server then responds by sending for every email in the user’s inbox a message of the form: ‘MSG\_ID. FROM “SUBJECT”’ followed by a single message: ‘END’ to inidicate there are no more messages at that user’s inbox.  MSG\_ID is the unique message id in the user’s inbox, FROM is the username that sent the message and SUBJECT is the message subject. | | | |
| Client Message: | ‘SHOW\_INBOX’ | Server Response: | Multiple messages of the form: ‘MSG\_ID. FROM “SUBJECT”’ followed by the message: ‘END’. |
| Fetching an email | | | |
| To get the content of a specific email, the client sends a message: ‘GET\_MAIL MSG\_ID’ where MSG\_ID is the id of the message the user wants to read.   * If message is found, the server responds with a message of the form: ‘FROM;TO;SUBJECT;CONTENT’ where FROM is the username who sent the email, TO is the recipients of that email, SUBJECT is the email’s subject and CONTENT is the email’s content.   Notice that this message is limited to BIG\_BUFFER\_SIZE.   * If message is not found, the server responds with a FAIL\_MSG. | | | |
| Client Message: | ‘GET\_MAIL MSG\_ID’ | Server Response: | ‘FROM;TO;SUBJECT;CONTENT’ or FAIL\_MSG |
| Deleting an email | | | |
| To delete a specific email, the client sends a message: ‘DELETE\_MAIL MSG\_ID’ where MSG\_ID is the id of the message the user wants to delete.   * If message is found, the server should delete and respond SUCCESS\_MSG. * If message is not found, the server responds with a FAIL\_MSG. | | | |
| Client Message: | ‘DELETE\_MAIL MSG\_ID’ | Server Response: | SUCCESS\_MSG or FAIL\_MSG |
| Signing out | | | |
| To sign out properly, the user should send the message: ‘QUIT’. | | | |
| Client Message: | ‘QUIT’ | Server Response: | No response |