Dynamic Allocation Of Mail Server Resources Among Users

Basti M. Aliyeva

Index: The article was dedicated to the solution of the problem on the dynamic allocation of mail server resources among users. It was noted that recently the electronic mail system had undergone serious changes and new features have been added to this system. The article examines the working principles of email and defines possible operating modes for subscribers. It was noted that mail server memory should be dynamically allocated among users so that the email can successfully perform its functions. In the article, the linear programming is applied to the problem of dynamic allocation of mail server memory. Known methods at every operation can resolve this issue. Proximity measure has been defined based on the Levenstein Distance (LD) for the determination of the renewal of documents on the server to improve the use of the server resources. This tool can be used to determine whether the documents have changed on the server. In the result, similar documents can be identified, and their number can be reduced up to one.

Key Words: email server, dynamic allocation of the memory, operating modes of subscribers, Levenstein Distance (LD), Measures of document proximity.

1. INTRODUCTION

It is known that e-mail is one of the most important information resources of the Internet, as well as it is the most massive communication tool. In recent years, e-mail system has undergone serious changes. Internet e-mail service had gone a long way from the time the users used the information exchange in the form of small, short text until the recent years when friends used pictures, music sharing, etc. Multiple protocols and software tools have been created and implemented to make email services to perform its new features. As a result, it was possible to exchange information not only in the form text but also binary files. The rapid growth of the Internet Network was the major driving force for the establishment and development of these protocols and program tools. In the past, the e-mail system consisted of smaller, secondary subsystems and it had to give priority to the corporate networks [1,2]. Email system has already been exploited quite a long time. Internet providers allocate separate independent systems to serve their clients’ email boxes. If we take into account that it is possible to send and receive information to and from dozens of international computer networks even some of them don’t have online service, it becomes clear that wide ranges of opportunities open up for the email users in a certain sense other than the information service of the Internet. It is known that in recent years, information has become a crucial factor in world politics, economy (trade, industry); it emerges as a product of outstanding scientific and research activities. The demand for structuring, collecting, storing, searching, and transmitting data is steadily increasing. Fulfilling these demands becomes the goal of creation and development of information networks. In this case, it is essential to pay special attention to data transmission issues. Numerous ways to transmit data and many software tools have been developed accordingly in this field. One of the most widespread means of data transmission is the transmission of data via e-mail. Email, being one of the vital exchange tools on the Internet does not reduce the relevance of the work done in this area. After a few seconds, the letter sent over the Internet reaches the addressee at another point of the planet – this indicates the advantage of the email over the usual post. Besides, the costs of sending such information are minimal. Thus, e-mail is an indispensable tool for correspondence between users in many organizations [3, 4]. Of course, it is difficult to find any organization in the current period that employees do not use or less use exchange advantages created by e-mail. The technology of electronic mail sending is that the sender sends the data to the addressee he/she knows by downloading a particular program, and the addressee also downloads a special software to verify and receive the data in its mailbox. At this time, a local mailbox is created for each user that stores all the user’s data in that box. There are special programs for working with mailboxes, which are called mail user agents. In short, such applications are called the MUA (Mail User Agent). Mail user agents do not receive data from other remote computers; they show the content of the user’s mailbox. Electronic mail works with email departments – mail servers as an ordinary post. In its turn, these servers also send and receive letters over the global network. They communicate with each other via mail protocols that provide data transmission and recognition – authentication on the web. Client computers of mail servers serve to email users. Each user must have a login and password to use the e-mail box. The user undergoes authentication procedure in mail server: on selected SMTP (Simple Mail Transfer Protocol) to send the information via mail clients such as Outlook Express while receiving email on selected POP (Post Office Protocol) or IMAP (Internet Message Access Protocol). If the authentication procedure is successful, prepared mail information is sent to the server (Figure 1).
Once the mail is sent to the server, the server creates communication with mail recipients for DNS addresses and transmits the data there. Then, the recipient's Client program reads its inbox folder. In this case, for the default port #25 is used as SMTP port, port #110 is used as a POP and IMAP port. Examples of allocated email servers are HMailserver, CommuniGate Pro, Hula, Sendmail, Postfix, MDaemon, Microsoft Exchange Server. From mail servers mentioned before MDaemon and Microsoft Exchange Server only work under Windows Operation System, others operate both with Unix and Windows Operation System. Message-information written in a particular format can be sent via email. MIME can be as an example of a common format. Any information in this format consists of the header and body. The header has four essential parts:

- sender’s address (From);
- recipient’s address (To);
- Subject of information (Subject);
- sending date, time (Date).

Here, the sender fills second and third sections. The first and fourth sections are filled by mail program automatically. A body contains the content of the information. Simple text can be put in the body. Moreover, one or more files (for example, graphics, sound, text files, etc.) can be attached to the text. The content of the email may affect only the amount of data sent but does not change the sending process of email. Any user must register with the corporate mail server to work with email. A registered user in the mail server receives an email address, and the mailbox is created for the user where the received messages collected. The email address is generally set as follows: Username@mail server address. For example, secretary@science.az; Username is the name created by the user while registering in the mail server. Thus, email communication consists of two interconnected processes: Receiving incoming mail and sending outgoing information. During these processes, the mail program performs data exchange functions with the mail server on both sides. As it is known, one of the critical issues in the design of the corporate mail system is the selection of equipment and proper memory allocation among corporate mail users. At this point, it is necessary to choose the equipment so that the flow of current emails into the organization is ensured. This equipment should also have specific production resources other than providing the current flow so that to enhance the work of e-mail when the number of correspondence increases. One of the main problems of selection of equipment is that the technical specifications of selected servers are not compatible with their productivity. Thus, the mail administrator has to refer to the previous operation experience of corporate mail systems or the recommendations of the mail software manufacturers. In many cases, such information is insufficient for the installation and operation of corporate mail systems and generally any system requirements. Therefore, solving the problem of dynamic allocation of server memory among users is topical to ensure that email server is used effectively. The article considers a solution to this issue.

2. PROBLEM STATEMENT

The following parameters can be taken as a measure of the unit of the productivity of mail servers: email processing intensity or email packages' processing intensity. In most cases, email processing intensity is indicated as the unit of productivity of the mail servers. It should be taken into consideration that emails of various volumes have its processing productivity. Thus, as another productivity unit for the mail server is email packages' processing intensity. This unit of productivity of the email server does not depend on the length of the electronic mail. The other main technical specifications of the mail server include the frequency of the central processor (MP), the size of the processor's memory, the size of the hard disk, and so on. In some cases, numerous e-mails may be collected on the server. If many users in the enterprise store a large number of emails in their mailboxes, a large memory is required for storing them on the server. According to some researchs, in 2019, it was expected that e-mail users will send approximately 250 billion emails [5]. This may be an example of the relevance of the problem. The following operation modes are set up to improve the reliability and efficiency of corporate e-mail as well as to prevent traffics that may occur here:

1. A limited mailbox memory is allocated for the subscribers, for example. 20MB. The system administrator decides on the more memory allocation in specific cases.
2. Subscribers should check and monitor their mailboxes everyday.
3. The data must be transferred to the hard disk after being viewed or sent (after being processed), and thus the mailbox must be freed.
4. Sending and receiving time and date of each data must be verified by notices; each subscriber's computer must be set up to request mail software receipts to achieve this goal.
5. Betty Liu. The 11 Worst Email Mistakes Everyone Makes

Figure 1. Working principle of email.
6. The period of storage of any data in the mail server memory must be at least five days. Subscriber must set this mode himself/herself. If the subscriber does not use the computer for a long time and does not look at the mailbox, the storage period can be prolonged no more than 30 days.

7. There should be a limit on the amount of data sent at a time. Large files should be archived and compressed through the WinRAR program or other means before sending. At this time, if the file size exceeds the required volume, the file should be transmitted in parts. An excessive volume of mailboxes can cause unpredictable problems which can affect the system's reliability and other features. Therefore, it is necessary to control the users’ mailboxes to avoid problems such as full memory failure. The article solves the problem of dynamic allocation of the email server memory among users.

In the article used optimization and linear programming methods, measurement theory, and specifically Levenshtein metric between the two texts. Experimental software An experimental software tool was developed using the DELPHI programming system.

3. SOLUTION OF THE PROBLEM.

The problem of dynamic allocation of mail server memory among corporate network users should be addressed to avoid the need to increase the memory of the mail server. The linear programming model is formulated for the solution of the problem [6-8]. In this case, the first approximation allocates the equal amount of memory to each electronic mail user:

\[ Q / n = Q_i^0, i = 1, \ldots, n \]

Here, \( n \) is a number of users; \( i \) is the current user, \( Q \) is total memory volume.

\( Q_i^0 \) is an initial memory volume per \( i \) user. Under these circumstances

\[ Q_1^0 + Q_2^0 + \ldots + Q_n^0 = Q \]

The size of the memory used by the mailbox varies depending on the user's intensity (\( \alpha \)) of requests to the server's memory.

Let's assume that, \( q_i^j \) is the volume of the email received by the \( i \)th user at the \( j \)th time.

\[ V_i^j \] is the volume of email kept in the mailbox by the \( i \)th user till the \( j \)th time. Then, the mailbox of the \( i \)th user at the \( j \)th time occupies the memory of

\[ Q_i^j = V_i^j + q_i^j \]

The user removes a portion of the e-mail received and keeps \( Q_i^j \) volume of the received email. Thus, there would be information in the volume of

\[ V_i^j = Q_i^j \]

at the \( j \)th time in the user memory.

Let's mark the \( i \)th mailbox usage coefficient at the \( j \)th time in \( [j - 1, j] \) with \( \alpha_i^j \):

\[ \alpha_i^j = \frac{V_i^j}{Q} \]

But let's point out the dynamics of change in memory usage coefficient in \( [j - 1, j] \) with \( k_i^j = \frac{V_i^j - V_i^{j-1}}{Q} \). Apparently, for the random \( j \) the following conditions must be met:

\[ k_i^j = 0, \sum_{i=1}^{n} \alpha_i^j = 1 \]

Let's suppose that the user receives an email in volume \( Q_i^{j+1} \) at the \( j+1 \) time. At this time, it is necessary to find such a new \( \alpha_i^{j+1} \) value so that the following issues will be solved under the conditions set out below:

\[ \sum_{i=1}^{n} \left( \alpha_i^{j+1} - \frac{1}{j} \sum_{k=0}^{j} \alpha_i^k \right)^2 \rightarrow \min \]

\[ \frac{1}{j} \sum_{k=0}^{j} \alpha_i^k < \alpha_i^{j+1} < \frac{1}{j} \sum_{k=0}^{j} \alpha_i^k + k_i^j V_i^{j+1} \]

\[ \sum_{i=1}^{n} \alpha_i^{j+1} = 1 \]

This problem is solved at every \( j \) discrete time. One of the crucial issues for improving the use of server resources is the determination of the updated information on the server. It is known that documents containing written texts are updated more frequently. Therefore, Levenshtein Distance-based measure was proposed as a criterion for updating text documents [9-13].

When the proximity of documents is determined with the Levenshtein Distance (measure) the number of basic editing operations required to obtain one string from another is calculated. Basic operations mean deletion, substitution, insertion for a single symbol [14, 15]. Each word should be imagined as a sum of symbols to define Levenshtein Distance.

Thus, the distance between the two \( a = (a_1, \ldots, a_N) \) and \( b = (b_1, \ldots, b_N) \) is calculated by the following formula:

\[ L((a_1, \ldots, a_N), (b_1, \ldots, b_N)) = \begin{cases} N; & M = 0 \\ M; & N = 0 \\ \min(L((a_1, \ldots, a_M), (b_1, \ldots, b_N)) + 1, L((a_1, \ldots, a_N), (b_1, \ldots, b_M)) + 1, L((a_1, \ldots, a_{M+N}), (b_1, \ldots, b_M), \delta(a_{M+N}, b_N))) \\ \end{cases} \]

Here \( \delta(x, y) \) is the negation of Kronecker symbol.
\[ \overline{\delta}(x, y) = \begin{cases} 0; & x = y \\ 1; & x \neq y \end{cases} \]

Short \( L_{ij} \) is used for each \( L((a_1, ..., a_i), (b_1, ..., b_j)) \), \( i = 1, ..., N \), \( j = 1, ..., M \) distance. All \( L_{ij} \), \( i = 0, 1, ..., N \), \( j = 0, 1, ..., M \), \( (i, j) \neq (N, M) \) distances should be reckoned to calculate \( L(a, b) = L_{NM} \). \( L_{NM} \) in the particular case.

It should be noted that it is convenient to calculate the \( L_{ij} \) distances by the iterative method. In this case, first of all, \( L_{i0} = i \), \( i = 1, ..., N \) and \( L_{0j} = j \), \( j = 1, ..., M \) distances are initialized. Then \( L_{ij} \), \( i = 1, ..., N \), \( j = 1, ..., M \) distances are calculated for \( L_{(i-1),j} \), \( L_{(i-1),j-1} \) and \( L_{(i-1),(j-1)} \) values.

Only deletion and insertion are used to ensure the adequacy of Levenstein Distance. Thus the substitution operation is not regarded as a serious modification by the human in most cases.

If \( L(D, D^*) \) value is the number of basic operations required to convert a document to another document, then proximity distance between \( D \) and \( D^* \) is

\[ \text{dist}(D, D^*) = \frac{L(D, D^*)}{\lambda + \lambda^*}. \]

Here \( \lambda \) and \( \lambda^* \) are the lengths of similar documents. Calculated distance meets \( 0 \leq \text{dist}(D, D^*) \leq 1 \) and is zero for similar documents.

Zero distance between two documents can ensure the deletion of one of the documents, and it can improve the operation of email servers.

There are various algorithms for determining the Levenstein distance between the two lines [16, 17].

The Levenshtein distance between the two lines can be calculated as in the DELPHI fragment given in the form of the following TForm1.Levenshtein. The s1 and s2 rows that are compared in this subprogram include the edit3 (first row) and edit4 (second row) windows. As you can see from the fragment, the length of the lines is written to the variables n and m respectively. The result of the fragment operation is indicated in the edit5 window. This fragment (the software was written and tested as a computer experiment)

The main and auxiliary variables used in the fragment are defined as follows:

```delphi
s1, s2: string;
d: array [0..100, 0..100] of integer;
m, n, d11, elave, e1, e2, e3, i, j: integer;
```

In the program created for the experiment, as in Figure 2, one of the rows is replaced by the first string and the second string. When you press the Levenshtein button, the distance between the lines is calculated and replaced by Dist = 6.

**Figure 2. Levenshtein distance program screen.**

```
procedure TForm1.LevenshteinClick(Sender: TObject);

begin
  s1 := edit3.Text;
s2 := edit4.Text;
n := length(s2);
m := length(s1);

  if n = 0 then
  begin
    d[0, j] := j;
    Goto lab1;
  end
  else
    if m = 0 then
    begin
      d[i, 0] := i;
      Goto lab1;
    end
    else
      if m > 0 then
      begin
        for i := 1 to m do // first
          begin
            d[i, 0] := i;
            for j := 1 to n do // second
              begin
                d[0, j] := j;
                if n > 0 then
                begin
                  var d11 := d[i, j];
                  for i := 0 to 99 do begin
                    d11 := d11 + elave;
                    for j := 1 to 99 do
                      begin
                        if s1[i-1] = s2[j-1] then
                          elave := 1;
                        else
                          elave := 0;
                        if elave = 1 then
                          begin
                            e1 := d[i-1, j] + 1;
                            e2 := d[i, j-1] + 1;
                            e3 := d[i-1, j-1] + elave;
                            if e2 < e1 then e1 := e2;
                            if e3 < e1 then e1 := e3;
                            d[i, j] := e1;
                          end;
                        end;
                      end;
                    end;
                  end;
                  d11 := d11 + elave;
                  d[i, j] := min(d11, e1, e2);
                  if m > 0 then
                  begin
                    for j := 0 to 99 do begin
                      if e1 < e2 then e2 := e1;
                      if e2 < e3 then e3 := e2;
                      d[i, j] := e1;
                    end;
                  end;
                end;
              end;
            end;
          end;
        d11 := d11 + elave;
        for i := 0 to 99 do begin
          d[i, j] := d[i, j] + elave;
          for j := 1 to 99 do
            begin
              if s1[i-1] = s2[j-1] then
                elave := 1;
              else
                elave := 0;
              if elave = 1 then
                begin
                  e1 := d[i-1, j] + 1;
                  e2 := d[i, j-1] + 1;
                  e3 := d[i-1, j-1] + elave;
                  if e2 < e1 then e1 := e2;
                  if e3 < e1 then e1 := e3;
                  d[i, j] := e1;
                end;
            end;
          end;
        end;
      end;
    end;
end;
```

lab1:
```
edit5.text := inttostr(d11);
```
From the definition of Levenshtein distance and the program shown, the distance L(s1, s2) between the lines s1 and s2 pays the following properties:

\[ L(s_1, s_2) \geq 0; \]
\[ L(s_1, s_2) = L(s_2, s_1); \]
\[ L(s_1 = s_2) = 0; \]
\[ L(s_1, s_2, s_3) \leq L(s_1, s_3) + L(s_2, s_3); \]

From the definition of Levenshtein distance and the program shown, the distance L(s1, s2) between the lines s1 and s2 regulated the following properties:

4. SUMMARY
The problem of dynamic allocation of mail server memory among corporate network users should be addressed to avoid the need to increase the memory of the mail server. Linear programming is applied to the problem stated to achieve the goal in the article. Real memory can be defined to be used by considering each user's memory usage coefficient at random time and the quality of memory usage increases. The Levenstein metric-based measure was proposed to update the documentation on the servers. The zero distance between the two documents can cause the deletion one of the documents and can improve the performance of e-mail servers as well. The conclusions achieved in the article can be used to solve the analogical problems.

REFERENCES