Efficient Management of Multimedia Attachments*

Itai Dabran[1], Philippe Klein[2], and Danny Raz[1]

[1] Computer Science Department, Technion, Hiafa 3200, Israel
[2] Telrad Mediagate Ltd, P.O.Box: 488, Rosh Haayin 48091, Israel

Abstract. In a modern heterogeneous environment, where users use a variety of devices such as smart phones, PDAs, and laptops, and a variety of network connections such as Wireless LAN, and GPRS to connect to the email system, handling of email and attachment messages and ensuring readability becomes an important management challenge. In some cases it is necessary to transcode the multimedia attachment to a format supported by the recipient device, and appropriate for the connection used, in order to allow the recipients to view the multimedia attachment. This transcoding could be done at the sender mail server, the receiver mail server, or at a proxy point in the middle. In addition the message may be addressed to several recipients in the receiver server, each may have different preferences and characterizations. This paper proposes an efficient scheme for handling email messages with multimedia attachments. We describe an architectural and algorithmic framework that allows service maximization to the end users (depending on their devices and connectivity) with minimum cost to the organizations in terms of storage, transcoding, and communication. We also provide a simulation study indicating that the proposed scheme is feasible and results in very efficient solutions.

1 Introduction

The increased importance of email in commercial applications, and the increased popularity of using rich media attachment files, makes efficient handling of email and attachment messages an important challenge. In addition to being the source for an endless increase in the demand for more storage resources, handling attachment files requires that the recipient of the message will have the ability to open the attachment file format. This problem becomes much more challenging when considering the modern heterogeneous environment, where users use a variety of devices such as smart phones, PDAs, laptops, and high-end desktops to connect to the email system. Each of these devices has a different capability in terms of color and screen resolution, memory, and available CPU. Moreover, the characterizations of the lines connecting the user to the mail server also vary in a significant way. A client that uses a PDA device and a GPRS network to

* This work is part of the STRIMM consortium operates under MAGNET program of the Chief Scientist of the Israeli Ministry of Trade and Industry (www.strimm.org).

connect to his email server, may have a limited bandwidth and a considerable loss on the link, while the same user may have better connectivity when the PDA is connected via, say, wireless LAN (WLAN). Handling multimedia attachments efficiently in such a heterogeneous environment presents significant management challenges. This paper deals with an efficient scheme for handling email messages with multimedia attachments in today's heterogeneous environments. Clients are usually connecting to their email server using the popular IMAP [1] or PoP3 [2] protocols. These protocols use the Multipurpose Internet Mail Extensions (MIME) [3] that support various attachment encoding methods. Email messages between mail servers across the Internet are mostly carried by the Simple Mail Transfer Protocol (SMTP) [4]. The objective of the SMTP is to transfer mail reliably and efficiently between mail servers, it is independent of a particular transmission subsystem and is capable of relaying mail over several transport services in a Wide Area Network (WAN). It runs between e-mail servers where the sender establishes a two-way transmission channel to a receiver that may be either the ultimate or an intermediate destination.

Consider a typical scenario in which a user wants to send an email message with a multimedia attachment to another user, or more generally to a set of users. These users may be connected to the same mail server, or they may be using different mail servers. Each of the recipients has its own preferences in term of the end device he uses (PDA or smart phone), and its connectivity characteristics (wireless, wire line). Thus, in order for the recipients to view the attachment, it may be necessary to transcode the attached file into a different media format (or in some case several different formats). This transcoding could be done at the sender mail server, the receiver mail server, or at a proxy point in the middle. Each of the receiving mail servers, can then decide how to store the attached file and in which format. This decision has a critical impact both on the storage capability of the server, and on the ability to stream the media to the client when the end user opens his mail message and wants to view the attached multimedia content.

In this paper we concentrate on the connection between two SMTP servers, that is, we focus on the case where an email message containing a multimedia attachment is sent from one mail server to another mail server. The message may be addressed to several recipients in the receiver server, each may have different preferences and characterizations as described above. Our goal is to describe an architectural and algorithmic framework that allows an efficient handling of the multimedia attachments, in a way that allows service maximization to the end users (depending on their devices and connectivity) with minimum cost to the organizations in terms of storage, transcoding, and communication.

The first step toward achieving this goal is to allow mail servers to exchange information regarding users' preference and storage limitations. This information can then be used by the sender mail server in order to decide how to handle the multimedia attachment.

The cost of sending a message with a multimedia attachment is combined of three components. The transcoding cost reflects the computation resources used

to perform file format transcoding. The communication cost reflects the use of network resources in order to send the information. Note that since different multimedia formats have different sizes, this cost may vary when the transcoding location changes from the sender mail server to the receiver mail server. The third component of the cost is the storage cost. In this paper we assume that the receiving mail server decides what multimedia formats are wanted, and this decision takes into account both the preference of the recipients, as well as storage constraints. Thus, the storage cost is not part of the algorithmic scheme but rather a part of the input.

There are few possible options to transfer the message: the format transcoding may be done at the sender server, or the multimedia attachment may be sent as is to the receiver server to be transcoded there. In addition, more than one format can be sent. In this paper we develop an algorithmic scheme that uses the information retrieved via the Capability Exchange phase in order to optimize the cost of the multimedia attachment sending process.

2 Related Work

The problem of efficient transcoding was addressed recently by several research papers ([5], [6], [7], and [8]). In [5] software and algorithmic optimizations for a real time MPEG-2 to MPEG-4 video transcoder is presented. This optimization results in a reduction of over 86% in the MPEG-4 transcoding time. The variation of the transcoding cost is also mentioned in [6] and [7]. In these papers the authors suggest to allow a caching proxy to perform the transcoding process. Variants of a video object can be delivered by transcoding the cached content to the appropriate format, instead of accessing it at the content origin. By this, heterogeneous clients with various network conditions, receive videos that are suited for their capabilities, as content adaptation can easily be done at the network edges. Another work [9] presents a complexity-quality analysis of various transcoding architectures for reduced spatial resolution, whereas to enable broadcast-quality video streams to be decoded and displayed on mobile devices, transcoding from MPEG-2 MP@ML to MPEG-4 Simple Profile is needed. This conversion implies a reduction in bit-rate from approximately 6Mbps to 384kbps and lower, as well as a reduction in spatial resolution from 720x480 interlace to 352x240 progressive.

The demand for multimedia information is increasing beyond the capabilities of a single storage device and may lead to the necessity of a new storage architecture [10], transcoding proxies [6] or server replications [11]. There are storage prototypes designed to address the real-time demands of digital video and audio. [10] presents a scalable storage architecture based on the replication of high performance storage instances, employing load balancing techniques of static file replication (whereas each server instance holds a copy of all other server instances' files and has a pre-allocated number of clients) and network striping (whereas the multimedia file is distributed over a number of cooperating server

instances in a highly capable network) to minimize the load on individual servers and interconnecting networks.

However all of the work mentioned above deals with real-time streaming and do not consider our problem of off-line transcoding cost of multimedia attachments.

3 Architecture

The SMTP protocol was defined in 1982 [4] and was initially designed to deliver short text messages coded in American ASCII characters. During the last 30 years several extensions such as DSN (Delivery Status Notifications) [12], and DRN (Delivery Report Notifications) [13] were added. When email messages contained only American ASCII characters the readability of the messages was 100% i.e. every successfully delivered message could be correctly read by the recipient. However, once users started sending messages coded in other character sets, and using attachment files, the readability of the message could not be assured any more. DSN and DRN give an indication of successful delivery of the message and a notification that the message was opened by the recipient, but there is no guaranty that the message could also be successfully read. In fact, the current heterogeneous environment, where mail clients can be located at different devices supported by various operating systems, creates a situation where in many cases the attached file format is not supported and cannot be opened by the mail recipient. The increasing use of rich-media attachments exacerbates this message readability problem. Therefore, there is a need to upgrade the protocol and model used, so that not only delivery but also readability is ensured. A first step in this direction is to allow the sender mail server to query the recipient mail server regarding the recipient capability or restriction before sending the message. In this way recipients can report for example, maximal attachment size accepted and different media format supported. The sender mail server can then either adjust the outgoing message according to the new information or notify the actual sender and ask for an appropriate format. This will be done using minimal storage and communication resources, as the large (unreadable) attachment is not sent.

In the current architecture, mail is sent using the SMTP protocol from the client to the outgoing mail server. The mail may contain an attachment file and the encoding is done using the Multipurpose Internet Mail Extensions (MIME) [3] format. The outgoing mail server then sends the message to the receiver mail server. This is done by establishing a two-way transmission channel between the sender and a receiver that may be either the ultimate or an intermediate destination. In most cases the current practice is to establish such a channel to the ultimate destination, i.e. the recipient mail server. The message is then accepted by the mail server, and the recipient client can access it via the popular IMAP [1] or PoP3 [2] protocols.

In our architecture we add an initial phase that allows the sender mail server to retrieve information regarding the capabilities of the recipient in terms of size

and supported format. This phase is termed Capability Discovery, and can be implemented within the SMTP protocol framework using a well defined text or XML messages.

When an email message is sent to a distribution list, the Capability Discovery information exchange becomes more challenging, because some of the information needed such as storage limits or supported format may be different for different members of the list and since individual information regarding members of the list cannot be exposed. When using multimedia attachments, and when transcoding at the mail servers is feasible, the information retrieved in the Capability Discovery phase can be used in order to decide the best format of this attachment. If the recipient list contains more than one receiver at the target mail server, then more than one format can be sent. Thus, in order to establish an efficient system, one needs to decide what are the best formats, and where to do the transcoding (at the sender or receiver server). We address this question in Section 4. However, in order to deploy the Capability Discovery phase, one needs to agree on a set of agreed parameters. In particular we need to define a set of profiles and formats that can be used. An example for such profiles for MPEG-4 can be found in Figure 1.

Profile / Level	L0	L1	L2	L3	L4
N-Bit			2 Mbps		
Advanced Coding Efficiency		384 Kbps	2 Mbps	15 Mbps	38.4 Mbps
Main		234 Kbps	2 Mbps	15 Mbps	38.4 Mbps
Core scalable		768 Kbps	1.5 Mbps	4 Mbps	
Advanced Core		384 Kbps	2 Mbps		
Core		384 Kbps	2 Mbps		
Simple scalable		128 Kbps	256 Kbps		
Advanced Real Time Simple		64 Kbps	128 Kbps	384 Kbps	2 Mbps
Simple	64Kbps	64 Kbps	128 Kbps	384 Kbps	

Fig. 1. Example of MPEG-4 Different Profile and levels

4 The Multimedia Attachments Problem

In this section we study the algorithmic aspects of our problem. We consider an outgoing mail server that has an email message with a multimedia attachment in a given format, and a list of receivers at another mail server. As explained in Section 3, we assume that capability discovery is possible and the outgoing mail server can exchange relevant information with the receiving mail server. We also assume, as described above, that transcoding is possible in both mail servers. In

the general case the attachment file is needed in several formats at the receiver, thus one needs to decide what transcoding is needed and where to perform the transcoding (i.e., at which mail server - sender or receiver). Once such a decision is made, the format of the multimedia attachment that is sent over the network is also decided. Note that in some cases, when the message is sent to a list of users with different preferences, more than one copy of the message, each with a different attachment format, may be sent from the sender mail server to the receiver mail server.

Our goal is to make the most efficient decision that is the one that requires minimum resources. Note that there are two different resource types we consider here. One is the CPU and time needed for the transcoding. The second resource we consider is the communication cost which depends on the amount of information (bits) we send over the network. A third resource type, which may be critical in this context, is the storage ability at the receiving server. However, in our framework this aspect is covered in the Capability Discovery protocol, and is part of the receiving server decision to determine what formats are needed. Note that comparing CPU utilization at the servers and communication over the network is hard and thus we need to define a cost for each of the operations, and try to find the minimal solution, i.e., the one with the minimal cost.

The first input to the algorithm is thus, the format of the multimedia attachment at the sender, and the requested format(s) at the receiver. As explained in the previous section, there could be many requested formats, and thus we assume a small number of relevant agreed formats, for examples the formats described in the tables of the previous section.

The second part of the input is the ability of the different servers to perform transcoding, and the relative cost of such a transcoding between two given formats in each server. The transcoding capability and the transcoding time depends on the hardware capabilities and the software in use, and the difference may be significant as described for example in [5, 7, 9]. We present this information for each server by a transcoding graph. In this graph every node represents a multimedia file format, and a link (directed edge) between format A and format B represents the ability of the server to perform transcoding between these formats. Each such link is associated with a cost; this cost represents the resources needed for that transcoding in terms of time and CPU. An example for such a graph is presented in the top part of Figure 2a. This figure shows two graphs that represent the source format in the vertexes upper level and the destination formats as "target vertexes" in the lower level. Note that for our problem we get two graphs, one for the sending server and one for the receiving server.

The third part of the input information represents the communication cost. For each relevant multimedia format we need to define the cost associated with sending an attachment in this format between the two mail servers. This is represented in our model by a link connecting the node representing the format in one server (sender), with the node representing the same format in the second server (receiver). The cost of the link is the communication cost of the specific format. Since different formats have different sizes, the communication cost between the

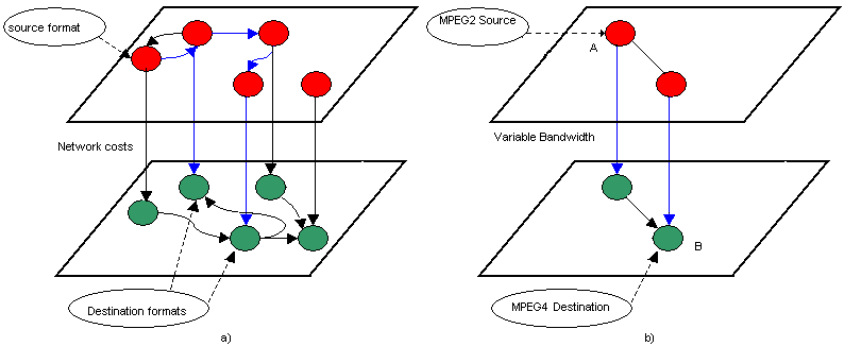

Fig. 2. Transcoding graph example

same sender and receiver varies according to the different formats. The required output is a list of transcoding needed to be performed in each of the servers, and a list of the different formats that are needed to be sent from the sender to the receiver server. We want, of course, to choose the output that induces minimal cost (i.e., minimal resource utilization). We formulate the Multimedia Attachment Problem in the following way:

Definition 1. *Given a set of formats V, two weighted graphs $G_S = (V, E_S)$, $G_R = (V, E_R)$, where $E_S(v_1, v_2)$ ($E_R(v_1, v_2)$) represents the cost at the sender (receiver) for transcoding format v_1 to format v_2, a sender format v_S, a set of required receiving formats $v_{r_1}, \ldots, v_{r_l}$, and communication cost $C(v_i)$ representing the cost to send format v_i, find the minimal tree, rooted at v_S in G_s, and connecting all v_{r_i} in G_R.*

The problem as defined above is an instance of the minimal Steiner tree problem [14], and is NP-Complete. Thus, an algorithm that finds an optimal solution with a polynomial time complexity is unlikely to exist. One can thus take one of the following two approaches. The first approach is to use heuristics methods or proven approximation algorithms. These algorithms will find in polynomial time a solution to the Multimedia Attachment Problem, however, the solution may be sub-optimal in terms of cost. That is, if we have an R-approximation algorithm, it can find a solution with cost up to R times larger than the optimal solution.

The second approach is to try to find the optimal solution to the problem in hand. In this case we will be able to complete the task only if the input graph is small. The exact meaning of 'small' here depends on the structure of the graph, and the actual values of the costs. We discuss several running examples in the next section. Since most practical cases are small, we chose at this point to try to find the optimal solution using the A-Star algorithm [15]. This is a general search algorithm which uses a heuristic function in order to create a "minimal weighted path tree". However, a good choice of the heuristic function can guarantee that the best (i.e. the lightest) tree will be found eventually. Thus, the appropriate

way to use this algorithm in this case is to run it for a limited amount of time, and if the optimal solution was not found during this time, to run an approximation algorithm and get a sub-optimal solution. In all examples we present in the next section the optimal solution was found by the A-star algorithm.

A good way to describe the algorithm is the following. At each step we have a list of 'open trees', and a list of 'closed trees'. All trees contain the root node (i.e. the sender format). The trees are ordered by their overall weight - the sum of the links weights. At each step we chose the first (i.e. lighter) tree in the open list. If it covers all the target nodes (i.e. all needed formats at the receiver), then we finish and output this tree. Otherwise, we move the tree to the 'closed trees' and add all the trees that can be created by adding one node to this tree, to the 'open trees' list. One can prove that this usage of the A-star algorithm guarantees finding the best solution. However, as explained above, in some cases the number of possible trees may be exponential, and the running time of the pure A-star algorithm may be infeasible. We implemented the above algorithm, where XML is used to describe and present the input graphs. The definition of an XML based notation helps to specify the dependencies and costs of all the components in our network in a similar way to the use of XML in [16].

5 Simulation Study

In order to demonstrate the benefit of our purposed architecture and algorithmic solution we present several realistic scenarios to study the various possible solutions for the multimedia attachment problem. We start with a very simple example. In this example the sender wants to send an email attachment that contains a 150Mbit MPEG-2 video clip (duration of 30 seconds, 5Mbps with a resolution of 720X480 (Format1)). The destination is a single user in the receiver server, which needs the attachment in 10 frames per second MPEG-4 format with a resolution of 352X240 (Format2). The size of the attachment needed by the destination is about 12Mbit.[3]

We evaluated the costs in terms of seconds for the process of transmission and transcoding to the needed format. We tested the transcoding process with and without our algorithm, over various bit rates. In our simulation we used 31.18 seconds as the transcoding time at the server, and 3.53 seconds as the transcoding time at the receiver. Such a transcoding was performed in [5] using a Pentium 4 Dell workstation with 1.8 GHz processor, 512MB memory and running Windows 2000, with and without MMX optimization software. The transmission time of the MPEG-2 format file and the MPEG-4 format file is derived from the file size that is approximately 150Mb for the MPEG-2 format and approximately 12Mbit for the MPEG-4 format. In Figure 3(a) we show the cost in terms of time for sending and transcoding the Format1 file from node A to Format2 file at node B, when each of the options is used, while in Figure 3(b) we show our solution for the needed format when transcoding is done in the most efficient way. We see

[3] These values are only similar to the values depicted in the tables of the previous section. This is done since we are using real measurements reported in [5].

that each of the transmission routes is not optimal, while our solution chooses to switch between them in order to find the optimal solution.

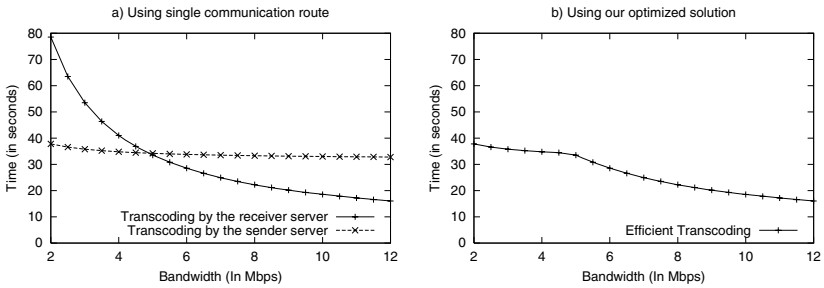

Fig. 3. The cost of multimedia transmission

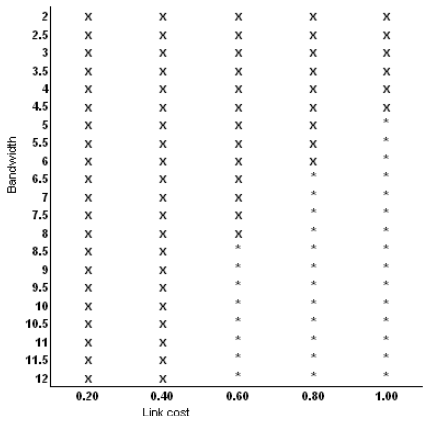

Fig. 4. A Communication Link with a variable cost

Suppose that the cost of the communication line is a function of more factors than the bandwidth. Maintaining a satellite communication link or using cellular communication may cost much more. In our second simulation we checked the problem described in figure 2b, but with a variable cost of the communication link. Figure 4 depicts two areas. The first area (marked by x) presents the range where a decision to transmit the multimedia file and afterwards to transcode it is taken and the second one (marked by $*$) presents the range where a decision to transcode the multimedia file and afterwards to transmit it is taken. Both areas present the most optimal way to handle the multimedia attachment. Next we consider a somewhat more complex example. In this case the same sender is sending again the same format (Format1) but this time there are two users

in the receiver server, whereas one of them needs the multimedia attachment in Format1, and the other in Format2. The sender server and the receiver server are connected in a 10Mbps communication link. Figure 5 presents this example. In

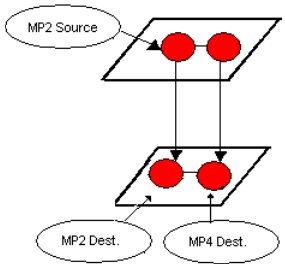

Fig. 5. When two formats are needed at the receiver server

this case there are two options: a) The sender server transcodes the multimedia attachment and sends a copy in Format1 and another copy in Format2 to the destination server, and b) The sender server sends an attachment in Format1 to the receiver server and the receiver server saves one copy in Format1, and a transcoded copy in Format2. Figure 6a depicts both options as a function of the transcoding time of the receiver server. We see that when the first option is executed and the source transcoding time is constant, the time it takes to transmit the multimedia attachment remains constant. When the second option is done, the time it takes to transmit the multimedia attachment is a linear function that depends on the receiving server transcoding time. Figure 6b depicts our solution which optimizes each of the options presented in Figure 6a. We see that as the transcoding process in the sender server increases, our proposed solution requires that the transcoding process will be at the destination server.

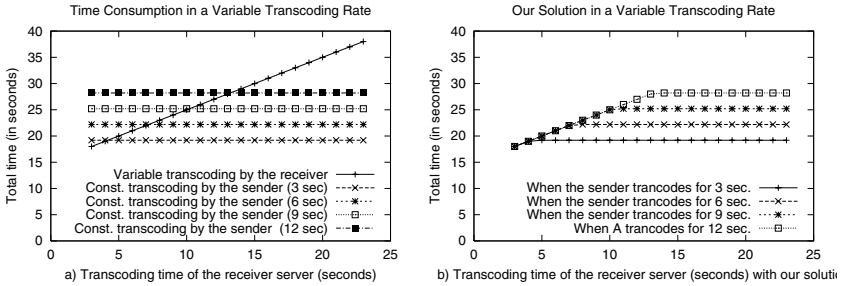

Fig. 6. A variable transcoding time of the receiver server with and without our solution

Figure 7a presents a much more complicated problem, where the source has an AVI format and needs to transmit it to a destination server, where 3 formats

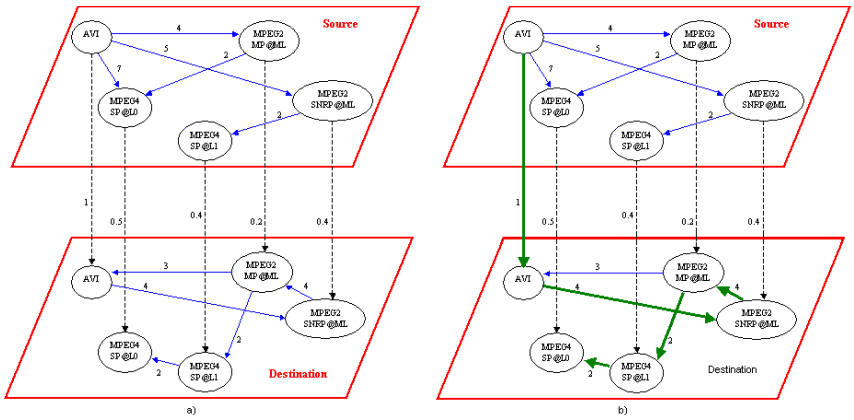

Fig. 7. A complex example

are needed: MPEG-2, MPEG-4 and AVI. The solution produced by our simulator shows that the most optimal way is as shown by the wide arrows on Figure 7b. Note that this is indeed the optimal solution which indicates that our algorithmic solution can handle rather complex input.

6 Conclusions

In this paper we addressed the problem of efficient management of email messages containing multimedia attachments. We identified the need for capacity exchange between the mail servers, and presented an efficient algorithmic solution. This framework allows a mail server to decide how to handle multimedia attachment files, if and where to perform transcoding, and to what format. However, the results presented here are only a first step and very many interesting problems are left open. A basic problem is the generalization of the algorithm to multiple receiving servers. A more complex problem is related to proxy servers. In many cases it will be more efficient to store the multimedia attachments at separate servers that might be collocated with the mail servers, or located elsewhere in the network. In such a case, the multimedia attachment can be replaced by a link to the data in the appropriate media storage and streaming server, as done in Corsoft Aileron (url: http://www.corsoft.com). In this case the problem of finding the appropriate multimedia format and the optimal location for the attachment files becomes much more interesting and complicated. Finding good algorithms for the more general problem is a very challenging topic for future research. One has also to take into account economic considerations such as who pays for the mail (sender or receiver) and what is the business model of the transcoding services.

Acknowledgments

The simulation tool was developed in the LCCN Lab by Assaf Karpel, Oren Kaminer and Guy Gurfinkel, CS Department, Technion, Haifa, ISRAEL.

References

1. Crispin, M.: Internet Message Access Protocol — Version 4 rev1. IETF RFC 2060 (1996)
2. Myers, J., Rose, M.: Post Office Protocol — Version 3. IETF RFC 1939 (also STD0053) (1996)
3. Borenstein, N., Freed, N.: MIME(Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies. RFC-1341 (1992)
4. Postel, J.: Simple Mail Transfer Protocol. IETF RFC 2821 (also STD0010) (2001)
5. Hari Kalva, Anthony Vetro and Huifang Sun: Performance Optimization of an MPEG-2 to MPEG-4 Video Transcoder . SPIE Conference on VLSI Circuits and Systems, Vol. 5117. (2003) 341-350
6. Bo Shen, Sung-Ju Lee: Transcoding-Enabled Caching Proxy System for Video Delivery in Heterogeneous Network Environments. In: Proceedings of IASTED IMSA 2002, Kauai, HI. (2002)
7. Bo Shen, Sung-Ju Lee, and Sujoy Basu: Performance Evaluation of Transcoding-enabled Streaming Media Caching System. Proceedings of the 4th International Conference on Mobile Data Management (1997) 363-368
8. Youssef Iraqi and Raouf Boutaba: Supporting MPEG video VBR traffic in wireless networks. Computer Communications 24 (2001) 1188-1201
9. Anthony Vetro, Toshihiko Hata, Naoki Kuwahara, Hari Kalva, and Shun-ichi Sekiguchi: Complexity-Quality Analysis of Transcoding Architectures for Reduced Spatial Resolution. IEEE Transactions on Consumer Electronics, ISSN: 0098-3063, Vol. 48, Issue 3, (2002) 515-521
10. D. Pegler, D. Hutchison, P. Lougher, A. Scott, and D. Shepherd: Scalability issues for a networked multimedia storage architecture. (Multimedia Tools and Applications) http://www.comp.lancs.ac.uk/computing/users/phillip/scams.html.
11. Raouf Boutaba and Abdelhakim Hafid: A generic platform for scalable access to multimedia-on-demand systems. IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 17, NO. 9 (1999)
12. Moore, K.: SMTP service extension for delivery status notifications. IETF RFC 1891 (1996)
13. Vaudreuil, G.: The multipart report content type for the reporting of mail system administrative messages. IETF RFC 1892 (1996)
14. Hwang, F.K., Richards, D.S., Winter, P.: The Steiner Tree Problem. Volume 53 of ADM. North-Holland, Amsterdam, Netherlands (1992)
15. Rabin, S.: AI Game Programming Wisdom. Charles River Media, Hingham, Massachusetts (2002)
16. Ensel, C., Keller, A.: Managing Application Service Dependencies with XML and the Re source Description Framework. In: Proceedings of the 7th International IFIP/IEEE Symposium on Integrated Management (IM 2001), IFIP/IEEE, IEEE Publishing (2001)