DSNs & X.400 assist in ensuring email reliability

Bassam Talib Sabri

(University of information Technology & Communications)

Abstract: *While email is fairly reliable, messages do get lost. This can be due to the store-and-forward nature of e- mail, which passes responsibility for delivering email from one mail server to another, but in doing so exposes messages to loss, should a server not fulfil its responsibility. Some end users are more sensitive to email unreliability than others, and seek tools with which they can detect and recover from email failures. This paper describes tools that allow email sending software to automatically respond to reports of undeliverability by persisting with retransmission or retransmitting to alternate recipients.*

I. Introduction

Email is one of the most important network applications, with about 31 billion messages sent daily in 2003, a figure which was expected to double by 2006 [1]. However, messages can get lost, resulting in non-delivery to intended receivers. Measurement studies suggest that about 0.69% of messages might be lost silently (without the originator receiving a bounce message) [2], and while some servers offend more than others, perhaps 6% of servers exhibit slight (0.03%) loss and 18% of servers exhibit higher loss rates [3]. Loss can be caused by many points in the path that an email takes from originator to recipient. (In this paper we emphasize the loss aspects of reliability, rather than other aspects such as whether messages are duplicated.) At the ends, the originator may use an incorrect address for the recipient, or the recipient may have exceeded their storage quota. Filtering, to defend against spam and malware such as worms and viruses, can also lead to loss both at the recipient's end system or at mail servers en route from originator to recipient. Filters have mixed impact on overall reliability, since they reduce the possibility of storage quotas being fully used, but at the risk of misclassifying some messages as unwanted which can lead to them being disregarded. Such filters particularly affect permission-based marketing messages which may somewhat resemble spam, and several companies (e.g. Pivotal Veracity, Lyris, Return Path and Message Labs) monitor the deliverability of such messages and guide sources of such messages as to how they can be tailored to maximize deliverability despite filters. Garriss et al [4] show how such filters can be improved to reduce unreliability due to filtering. More fundamentally, two intentional design features of email, the ability for servers to operate asynchronously and for recipients to forward messages, can harm reliability by allowing loss from servers to evade detection by end-users. First, email systems are intentionally designed to enable communication between entities that need not be available simultaneously, i.e. they can operate asynchronously. This is achieved by mail servers operating in a store-and-forward mode, whereby they accept messages from sources and attempt to relay them towards, or deliver them to, the recipient. This is particularly useful at the end of a path, where a server can hold a message in a mail store where it waits for a client to access the message. SMTP servers communicate with each other through TCP, which includes a retransmission mechanism to ensure reliable transfer over the connection between servers. However, it is still possible that messages may be lost within servers, and so the end-to-end transfer from originator to recipient may not be reliable. The store-and-forward nature of SMTP is similar to that of IP, and just as TCP needs to be superimposed on IP in order to ensure reliable transfer across a network, so too something needs to be superimposed on a series of SMTP servers if there is to be end-to-end reliability between mail originator and recipient. The SMTP specification states: "When the receiver-SMTP accepts a piece of mail … it is accepting responsibility for delivering or relaying the message. It must take this responsibility seriously. It MUST NOT lose the message for frivolous reasons, such as because the host later crashes or because of a predictable resource shortage." [5, Section 6.1]. While this recognises that servers have the potential to contribute to unreliability, the caveats in the form of "frivolous reasons" and "predictable resource shortage" (italics added) suggest that servers may not be able to completely avoid contributing to unreliability. It is also important to note that it is one thing to mandate that a server "MUST NOT lose the message", but it is another to enforce this by checking whether an implementation of a server is losing any messages. Just as the end-to-end arguments [6][7] require end-systems to implement error control (e.g. in transport protocols such as TCP) because they cannot trust routers to be able to detect and recover from all errors, so too end-users of email services need to implement error control because they cannot be certain of the reliability of SMTP servers that their message will pass through. The second feature of email that can detract from reliability is that recipients can automatically forward messages to (an)other recipient(s). This prevents the originator from knowing the identity of the ultimate destination(s) of a message, and hence whether any positive feedback that it may have received (such as Delivery Service Notifications) indicate delivery to the final

destination or just delivery to some intermediary. Section 2 of this paper provides background about the main protocol used for email transfer, SMTP, with emphasis on how SMTP addresses reliability. Section 3 describes active source responses to known errors, such as automatically retransmitting a message, either later, or to an alternate email address. Section 4 describes our implementation of these responses, and Section 5 concludes. The authors apologise to readers for this version of this paper having very limited novelty. The version that was submitted for review to the conference included many substantial novel ideas which filled the page limit stated in the CFP of "20 double-spaced pages ... using 11-point type". However the paper was accepted only as a short paper of "4 pages using 10 point type". More than halving the length of the paper has left little space for substance. Readers who are interested in the more substantial ideas from the full paper should email the authors.

II. SMTP background

Most email today is sent using the Simple Mail Transfer Protocol (SMTP) [5], although in Section 3.20 we will consider an idea from an alternative protocol, X.400, that is not yet supported by SMTP. SMTP is used between Mail Transfer Agents (MTAs), also known as "mail servers". An originating MTA (e.g. in an originating mail client) uses SMTP to send a message to another MTA, usually to a local MTA charged with forwarding messages out of the originating domain. Local MTAs may be used because they may be better connected, and so can persist longer, than the originating MTA. A more recent reason for requiring transfer through local MTAs is to control the flow of messages from a domain (e.g. to curb spam or the spread of worms). SMTP servers cope with unreliable networks or unavailable peers by persisting in their attempts to contact downstream servers. The most popular server implementation (sendmail, www.sendmail.org) retries every 15 minutes to deliver messages, issues a warning message (with a status code beginning with 4) after trying for 4 hours, and keeps trying for up to 4 days, after which it issues a permanent bounce message. We return to this possibility of extensive delays (4 hours or 4 days) before the originator receives feedback when considering retransmission by the originator in Section 3.1. Of the many fields [8] that can appear in message headers, the most relevant to this paper are the From, Sender, To, CC, and BCC fields which are used for addressing messages. SMTP provides several types of feedback, which can be used to detect problems, including 3-digit numeric reply codes, enhanced dotted decimal codes, and finally Message Disposition Notifications. These can be provided synchronously (when a client is connected to the server), or asynchronously (after the SMTP session) in the form of a "bounce message". For example, one server may synchronously indicate that it accepts a message to be relayed to another server, but later find that it cannot deliver the message to that server, and so asynchronously sends a bounce message. A source cannot equate no negative feedback with no error, since the feedback may merely be slow to arrive, or may itself have been lost. Delivery Status Notifications (DSNs) [9] extend these asynchronous responses by standardising their format and providing for positive feedback. Servers that support DSN indicate so in response to an ESMTP EHLO command, and allow the client to specify (with a "NOTIFY" parameter) when DSNs should be generated (NEVER, or on SUCCESS, FAILURE or DELAY) and (with a "RET" parameter) whether the full message or just headers should be included in the DSN. While the positive feedback provided by DSNs can assist in ensuring email reliability, DSN is not yet widely deployed (e.g. at the time of writing, none of the major free mail providers, Yahoo, Hotmail and Gmail support DSN). The preceding feedback has related to events in servers. An originator may also be interested in whether a receiving mail client has processed a message. There have been some attempts to provide Message Disposition notifications from user agents (e.g. RFC 3798 [10], or by HTML messages covertly linking to message-specific images on servers that log when these images are accessed), but these can compromise the privacy of end-users who do not wish to reveal when they read specific messages.

III. Alternate recipients

If a message can't be delivered to the preferred recipient, then it may be desirable to send it instead to an alternate recipient. For example, when the destination mailbox is full, the message might be sent to an alternate mailbox. While using an alternative is superficially obvious and simple, there are some subtleties that are worth noting. First, there would often be some reason for preferring delivery to the initial address rather than the alternate address, and delivering to the alternate address may prevent immediate loss but cause other problems. For example, the alternate address may be for an email account that the recipient checks less often, in which case using the alternate may introduce delay (before the human end-user reads the message) in order to avoid immediate loss. Another example is that the alternate address may lead to receipt by people who are peripheral to the intended recipient (e.g. to one or multiple peers of the intended recipient), which may cause delay or increase load on others. Second, it is important that the sender be aware that delivery to the initial address was unsuccessful so that they consider updating the preferred address for subsequent messages. Related work: The dominance of Internet protocols over alternatives, such as OSI, has also led to SMTP being the

dominant mail transfer protocol. However, email facilities in some alternate protocol suites provide features that have not yet appeared in SMTP. X.400 [11] provided an "Alternate Recipient Assignment Service", which allowed an originator or recipient (or domain of the recipient) to specify ("assign") an alternate recipient to which messages would be sent if they could not be delivered to the "preferred recipient". Originators could also specify whether a message is permitted to be delivered to alternates. SMTP does not provide a corresponding feature, even though this could be used to improve deliverability. (RFC 2156 does define a SMTP Alternate-Recipient header field for interworking between SMTP and X.400.) Because in our implementation it is the source that uses alternate addresses, our implementation does not need to modify SMTP or introduce a new header field to indicate alternates to downstream servers.

IV. Implementation

These tools have been implemented in Java (for portability between operating systems) as a proxy. The proxy would typically reside on an end-user's computer, but could also be located on a separate machine that serves multiple users. The use of a proxy eliminates the need to create plug-ins for separate mail user agent programs – they need only be configured to use the localhost (127.0.0.1) as the mail server. The proxy needs root/administrator access rights to run so that it can bind to reserved ports 25 (SMTP) and 110 (POP3), but (when running on an end-user's computer) it only accepts connections from the localhost to those ports, and so does not open security problems from making more mail servers available to other machines. Figure 1 shows the structure of the proxy software, with the interface to the client at the top, storage for messages being processed in the central MailAccountHandler, and the interface to other mail servers on the bottom.
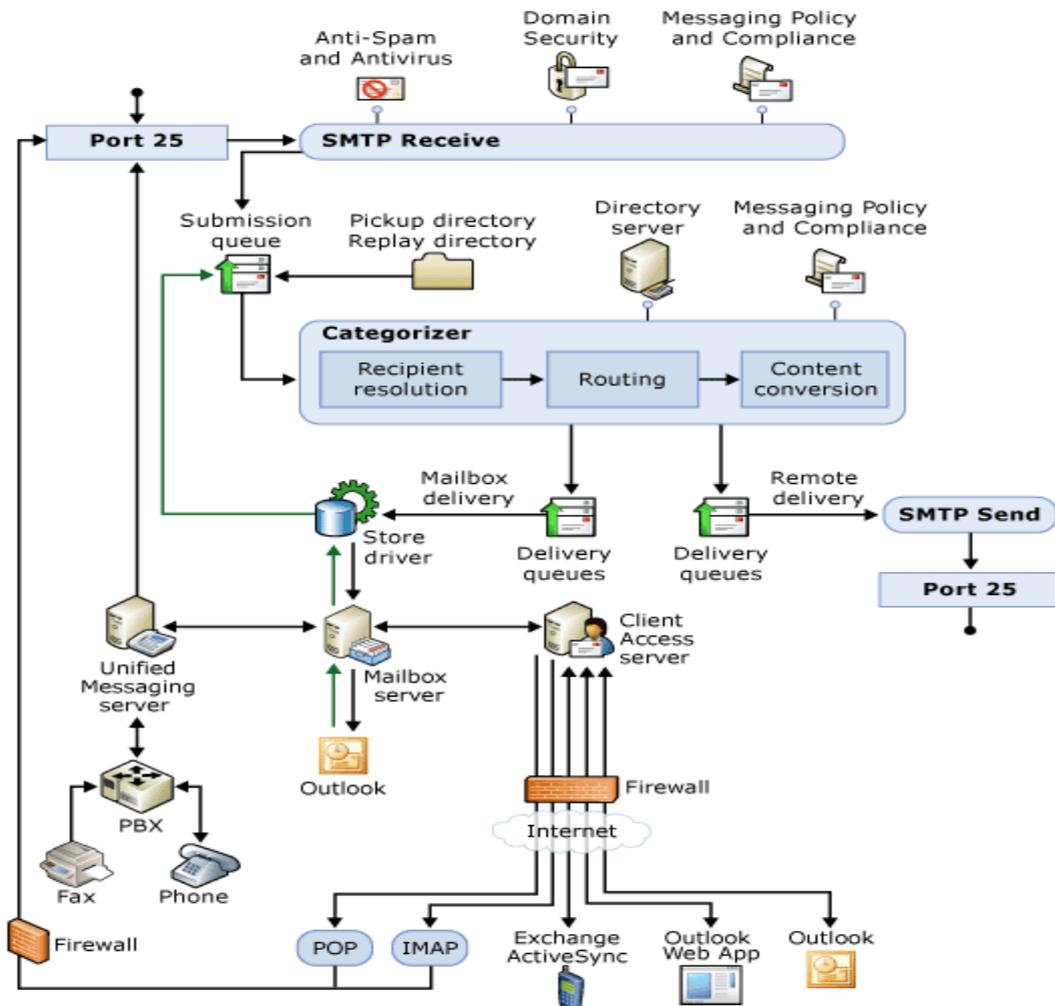

Figure 1: Structure of the proxy software

V. Conclusion

This paper has described two ways by which end- systems, sources in the case of this paper, can improve email reliability. Several more interesting techniques can be implemented at sources and receivers, and will be the subject of a future paper.

References

[1]. P. Lyman and H. Varian: "How Much Information", http://www.sims.berkeley.edu/how-much-info-2003
[2]. T. Moors: "Email reliability", http://uluru.ee.unsw.edu.au/~tim/dependable/email/
[3]. M. Afergan and R. Beverly: "The state of the email address", Computer Communication Review, 35(1):29- 35, Jan. 2005
[4]. S. Garriss, M. Kaminsky, M. Freedman, B. Karp, D. Mazihres and H. Yu: "RE: Reliable Email", Proc. NSDI, pp. 297-310, May 2006
[5]. J. Klensin: "Simple Mail Transfer Protocol," IETF, RFC 2821, Apr. 2001
[6]. J. Saltzer, D. Reed and D. Clark: "End-to-end arguments in system design", ACM Transactions on Computer Systems, 2(4):277-88, Nov. 1984
[7]. T. Moors: "A critical review of "end-to-end arguments in system design"", Proc. IEEE Int'l Conf. on Comm. (ICC), pp. 1214-9, Apr.-May 2002
[8]. G. Klyne and J. Palme: "Registration of Mail and MIME Header Fields", IETF RFC 4021, Mar. 2005
[9]. K. Moore: "Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)," IETF, RFC 3461, Jan. 2003
[10]. T. Hansen: "Message Disposition Notification," IETF, RFC 3798, May 2004
[11]. ISO/IEC: "Information technology - Message Handling Systems (MHS) - Part 1: System and Service Overview," Standard 10021-1: 1997