Design Considerations on N1-Mail Programs

3F-7 Hayashi, Tsunetoshi
Hokkaido University Computing Center

1. Introduction The electronic mail service on the inter-university N1 computer network has been started and is currently available on several host computers in the network [1]. The programs required for this service are now being developed at computing centers in Hokkaido University and University of Tokyo. The programs are the mail message receiver, the mail message sender, the user interface, and several utility programs to support miscellaneous housekeeping tasks required in the service. The mail message receiver, the mail retrieving user interface, utility programs to maintain mailbox datasets, and outer parts of mail sender are programmed at Hokkaido, while the rest at Tokyo. The works at Hokkaido has been nearly completed at the time of the writing of this manuscript, except for the mailbox maintaining utility.

The N1-mail system itself is defined based on the CCITT X400 series specifications of the Message Handling System (MHS), as shown in [1] and its related papers. As the specifications give the syntactic aspects of the messaging system only, some auxilliary definitions for functional facilities are made complementing the specifications. The programs are designed to implement the specifications and the definitions. In this report, we will present the basic design plan and the considerations made while implementing the plan.

2. Message Receiving Program As the N1-mail service is based on the MHS specifications, its messaging system is also defined in terms of the MHS messaging model. The MHS messaging system comprises three layers of message transmission functions, presiding over the lower layers of the CCITT Open Systems Interconnection (OSI) networking model. Thus the MHS messaging system is the application layer facility. The N1-mail messaging system also formally comprises three layers by definition, as it is modeled after the MHS. It need not work in the same way as the three layers model. The lower layers are also different from those of the OSI model, that is, they are the Network Virtual Terminal (NVT) protocol of N1 network. With these considerations, the messaging program may have a rather broad choice in dividing its function into sub-functional modules and selecting the ways of their mutual interaction. We employed the following implementation strategy.

- The messaging program is divided into three sub-functions; the data transmission, the envelope parser, and the heading parser along with the router and the mailbox access routines. The first two roughly correspond to the data transfer layer (DTL) and the message transfer layer (MTL), and the rest to the user agent layer (UAL) of the MHS model.
- Chiefly the data transmission takes the control of the whole program, and it pushes the control around to the succeeding programs, passing the message data through pseudo-pipelines placed between these programs.

A pseudo-pipeline is a queue data structure using heap in the main memory. A simple subroutine provides this function. Pseudo-pipelines are placed between the data transmission and the envelope parser; the envelope parser and the heading parser; and the heading parser and the mailbox access routines. This mechanism is used to separate the flow of control of mutually interacting programs in a non-concurrent, single stream environment.

- The envelope parser decomposes the nested data structure into an easy-to-handle flat database, stacking the contents message data into another pseudo-pipeline. It employs the (recursive) descent parsing method for parsing the message structure. Then it pushes the control to the succeeding program, the heading parser.
- The heading parser does the same to the heading part of the message, stacking the body part text

Design Considerations on N1-Mail Programs
Tsunetoshi HAYASHI
Hokkaido University

into a pseudo-pipeline. Formally, this must be done right after the message has been rerouted to the receiving client, however, this implementation takes slightly different approach from that of standard one. This is because of the efficiency, to avoid repeated parsing of the message. Then the heading parser passes the control to the router program.

- The router program fetches the text from the pseudo-pipeline and put it into the common mailbox dataset. The mailbox access routines are exploited in this process.

These programs could be made to run in parallel interfaced through true pipelines. We did not, however, employed this approach, we preferred to avoid the excessive complexity of multitasking programming, which is not relevant to the functionality of the mail programs.

3. Mailbox Datasets The mailbox consists of two virtual sequential access method (VSAM) datasets, one to store accompanying information of a mail, and the other to store texts of a mail. They are called the mail control dataset and the mail text dataset. The former dataset is indexed by a key constituted from a user identification number and a character string unique among every mail; while the latter is indexed by a key constituted from the same key string and a text sequence number.

The mail user interface can refer to user's mails by reading the mail control dataset using user identification number as a generic key. Then using the full key, it can read out the texts of a mail from the mail text dataset.

A small problem is that these datasets must be shared among several users and the mail receiving program running in a TSS job. The mutual exclusion and sharing of these datasets are obtained through short term open window, where datasets are opened for a duration that only a record is accessed.

4. Receiver User Interface The user interface of a mail system consists of two major functions; the mail sending and the mail delivery. The mail delivery part of the interface is described in this report. The interface is invoked as a TSS command processor, and was designed after the specifications given in [2], but not in full. The subcommands and functions provided by the interface are as follows.

- (L)`ist` : List up mails currently kept in the mailbox, with sender's name and subject as well as the reference number of a mail. A list of mails is also displayed when the user interface is started and some mails are deleted from the mailbox.
- (R)`eceive` : Display the contents of designated mails on the terminal.
- (H)`ardcopy` : Write the contents of designated mails into the spool printing dataset. Both of alphanumeric and Kanji mode printing is available.
- (W)`rite` : Write the contents of a designated mail onto a dataset.
- (D)`elete` : Delete designated mails from the mailbox. Mails could be accessed so long as they are kept in the mailbox, a user is required of telling explicitly that mails are no longer needed.
- (F)`ull` : Display full details of information accompanied with designated mails.
- Select a mail as the object of subcommands. If a mail is selected, subsequent commands designate the selected one as an implicit operand.

There are also commands to terminate the mail session, to unselect a mail, and to give some help messages. Some TSS commands can be invoked in a mail session.

5. Implementation The message receiving program and user interface program are written in PL/I. Small glue routines interfacing with the operating system are needed and written in the assembly langauge.

References

[1] M. Sakata *et al.*, "Constitution of Interuniversity Electronic Mail System", *IPSJ35*, 4V–5 (1987).

[2] T. Hayashi *et al.*, "Considerations on the User Facilities of the Inter-University Electronic Mail System", *IPSJ35*, 4V–8 (1987).