CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

7.1. Conclusions:

It is accepted that the extension of the Internet messaging system to more and more individuals and institutions is needed mainly for their benefit. It is also accepted that comprehensive message exchange with Multi-media and other features is becoming essential in that messaging system. However, at the same time, not every site has the capacity or preparedness to adapt to this without the coordinated help of the messaging system as a whole. The HAT concept proposed by this project is inevitably a service-oriented strategy to relieve such sites. An added advantage of the approach is that it implements administrative policies for controlling traffic and congestion arising from MIME mail to a mailserver site and the level of MIME services made available to users connected to that site. Further, the technique also integrates conventional and primitive messaging mechanisms, such as postal mail, courier by diskettes/tapes, facsimile and paging, into the realm of electronic messaging.

Although, the technique is very much appreciated by low-capacity mail environments, it is to be noted that it also allows a mail site using state-of-the-art e-mail infrastructure today, to adapt itself to keep up with the rapidly evolving computer and communications technology, until it is time to replace the existing infrastructure.

The reduction of transmission cost and network traffic are also advantages offered by these proposals as seen for instance by the physical delivery (courier) and keeping messages in the server database. Reduction of the receipt of junk mail and irrelevant messages will make most users happy.

It will not be difficult for a server to handle a large number of sites since the administration is de-centralized by allowing automatic updating. Incorporation of artificial

intelligence features(see Section 7.3 below), which is a future enhancement will fully automate the system and make it smarter.

7.2. Recommendations:

Discussions are carried out very frequently among the IETF(Internet Engineering Task Force) regarding new media types and headers. Some of those are just confined to informational RFCs whereas some become draft or proposed standards, and if several implementations were introduced such a proposed standard would become a standard.

One such ongoing discussion (during the time of writing this thesis) is how URLs could be referenced in MIME objects. This is not handled smartly in this project as yet, though the server may be asked to take limited actions on it. The reason is that when the specifications were finalised for this project, the discussions about URL's had just begun.

Similar discussions could be found from time to time, and some of them are application or field specific(e.g. anatomy, molecular science, etc.) while some are globally required. However, it is not feasible to pay attention to all of them and include support for everything in the technical specifications for the HAT concept, until such time there is a stable implementation. Once that is done, the modifications could be introduced to the implementation by way of proposing addendums to the existing technical documentation such as [16] and [17].

For any new addition to the MIME standard some solution could always be achieved on the requests of nodes by defining some rule in the ACTIONS file, and it is not necessary to make addendums to the existing documents, the reason being that the HAT concept was proposed with the expectation of future developments.

7.3. Enhancements: Looking into the Future:

It would have been better if we could incorporate "Artificial Intelligence"(AI), by way of implementing an inference engine so that the server could observe how nodes respond to certain types of messages. But it is better to embark such a project after this stage of implementation comes to a steady state.

There are several advantages that the incorporation of AI could bring. One would be the ability of the server to monitor how nodes receive messages and how the nodes request services for particular types of messages during an appropriate time period. Depending on such patterns, the server may suggest to the end node some changes in the ACTIONS file that the server would think best for those respective nodes to cope with the situation, without waiting until the node user requests such services.

Another feature may be pre-fetching media types referenced in a message, before the request comes from the node. To decide on what should be pre-fetched, the server may analyse the past behaviour of that node. This will enhance the download time for the user. Or else it may be able to do such actions like ftp during off-peak (or reduced-load) times to make the other resources available during resource intensive times.

It is also possible to keep track of how the gateways used for providing services behave, so that the server will be able to handle the messages to such gateways in a smart way. For example, if it is known that the fax gateway has a higher efficiency during a particular time of the day, due to congestion of communication lines caused during peak hours, it may keep a backup of such messages, in case the fax gateway abandons the sending of the fax after several re-tries, as is the case with many fax gateways.

These recommendations and enhancements are relative to the current state of the project. As it is being developed and the scope is widened there would be new suggestions that would further enrich the concept. However the concept itself will remain the same.

REFERENCES

[1] Borenstein, N., and N. Freed, "MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 1341, Bellcore, Innosoft, June 1992.

[2] Borenstein, N., and N. Freed, "MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 1521, Bellcore, Innosoft, September, 1993.

[3] Borenstein, N., and N. Freed, "MIME (Multipurpose Internet Mail Extensions): Format of Internet Message Bodies", Internet Draft - <draft-ietf-822ext-mime-imb-04.txt> , December, 1995.

[4] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, USC/Information Sciences Institute, October 1985.

[5] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, UDEL, August 1982.

[6] Borenstein, N., "A User Agent Configuration Mechanism for Multimedia Mail Format Information", RFC 1524, Bellcore, September 1993.

[7] Sollins, K.R., "TFTP Protocol (revision 2)", RFC-783, MIT, June 1981.

[8] Borenstein, N., "Implications of MIME for Internet Mail Gateways", RFC 1344, Bellcore, June 1992.

[9] Moore, K., "Representation of Non-ASCII Text in Internet Message Headers", RFC 1522, University of Tennessee, September 1993.

[10] Moore, K., "Representation of Non-Ascii Text in Internet Message Headers", RFC 1342, University of Tennessee, June 1992.

[11] Sirbu, M., "Content-Type Header Field for Internet Messages", STD 11, RFC 1049, CMU, March 1988.

[12] Rose, M., and E. Stefferud, "Proposed Standard for Message Encapsulation", RFC 934, Delaware and NMA, January 1985.

[13] Fernando MSD, Wijesoma WS and Dias Gihan V., "Multimedia Message Distribution in a Constrained Environment", Proc. Inet '95, Vol. 1, pp. 49-58, Inet '95, Honolulu, Hawaii, June 27-30 1995.

[14] Fernando MSD, Wijesoma WS and Dias Gihan V., "A Strategy for Multimedia Message Routing and Delivery for the Less Privileged", Proc. SEARCC '95(South East Asia Regional Computer Confederation), pp. 760-771, SEARCC '95, Colombo, Sri Lanka, September 5-8 1995.

[15] Bryan Costale with Eric Allman & Neil Rickert, "sendmail": Help for UNIX System Administrators, O'Reilly & Associates, Inc., September 1994.

[16] Fernando MSD., "Adaptive Multimedia Mail Server" Progress Report 2, Internal report, Dept. of Computer Science & Engineering, Univ. of Moratuwa, November 1995(revised version).

[17] Fernando MSD., "Adaptive Multimedia Mail Server: Implementation Project Report", Internal report, Dept. of Computer Science & Engineering, Univ. of Moratuwa, January 1996.

[18] Borenstein, N., and N. Freed, "MIME (Multipurpose Internet Mail Extensions): Conformance Criteria and Examples", Internet Draft - <draft-ietf-822ext-mime-conf-03.txt> , December, 1995.

[19] R. Troost and S. Dorner, "Communicating Presentation Information in Internet Messages: The Content-Disposition Header", RFC1806, QUALCOMM Incorporated, June 1995.

[20] D. Crocker, "MIME Encapsulation of EDI Objects", RFC1767, Brandenburg Consulting, March 1995.

[21] P. Faltstrom, D. Crocker, E. Fair, "MIME Encapsulation of Macintosh files - MacMIME", RFC1740, December 1994.

[22] P. Faltstrom, D. Crocker, E. Fair, "MIME Content Type for BinHex Encoded Files", RFC1741, December 1994.

[23] J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker, "SMTP Service Extensions", RFC1651, July 1994

[24] J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker, "SMTP Service Extension for 8bit-MIMEtransport", RFC1652, July 1994.

[25] J. Klensin, N. Freed, K. Moore, "SMTP Service Extension for Message Size Declaration", RFC1653, July 1994.

[26] D. Goldsmith, M. Davis, "UTF-7 - A Mail-Safe Transformation Format of Unicode", RFC1642, July 1994.

[27] N. Borenstein, "The text/enriched MIME Content-type",RFC1563,January 1994.

[28] H. Nussbacher, "Handling of Bi-directional Texts in MIME",RFC1556, December 1993.

[29] J. Myers, M. Rose, "The Content-MD5 Header Field", RFC1864, October 1995.

[30] J. Linn, "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures", RFC1421, February 1993.

[31] E. Huizer, "Multimedia E-mail (MIME) User Agent checklist", RFC1844, August 1995.

[32] E. Levinson, "SGML Media Types", RFC1874, December 1995.

[33] G. Vaudreuil, "The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages", RFC1892, January 1996.

[34] E. Levinson, "The Application/CALS-1840 Content-type", RFC1895, February 1996.

[35] M. Elkins, "MIME Security with Pretty Good Privacy (PGP)", RFC2015, The Aerospace Corporation, October 1996.

[36] N. Freed, K. Moore, "Definition of the URL MIME External-Body Access-Type", RFC2017, October 1996.

[37] H. Alvestrand, "A MIME Body Part for FAX", RFC2159, UNINETT, January 1998.

[38] H. Alvestrand, "Carrying PostScript in X.400 and MIME", RFC2160, UNINETT, January 1998.

[39] H. Alvestrand, "A MIME Body Part for ODA", RFC2161, UNINETT, January 1998.

[40] Fernando MSD, Wijesoma WS and Dias Gihan V., "Expansion of Multimedia Message Distribution to Constrained Environments: An Implementation Launch", Proc. CSSL '96(Computer Society Sri Lanka), CSSL '96, Colombo, Sri Lanka, 1996.

APPENDIX A

Approach to the Implementation:

The strategy was to define configuration commands in *sendmail.cf* (or any file specified explicitly for "sendmail") configuration file or to specify a command line option.

The above could be achieved through the following procedure:

i. To activate ACTIONS file processing [1] define a command line option "-a". If "-a" is not specified, it will run in original "sendmail" mode.

ii. To define the location of ACTIONS file, *sendmail.cf* (or any file specified explicitly for "sendmail") will be used as follows:

```
Aa<file name>  (default /etc/actions)
Aa/etc/actions
```

iii. To define the location of NODECAT file, *sendmail.cf* (or any file specified explicitly for "sendmail") will be used as follows:


```
An<file name>  (default /etc/nodecat)
An/etc/nodecat
```

iv. To define the location of unknown headers file, *sendmail.cf* (or any file specified explicitly for "sendmail") will be used as follows:

```
Au<file name>  (default /etc/headers)
Au/etc/headers
```

v. To define the location of log file, *sendmail.cf* (or any file specified explicitly for "sendmail") will be used as follows:

```
Al<file name>  (default /var/adm/action.log)
Al/var/adm/action.log
```

vi. To define UUCP command which is to be used when handling **SIM** has to be specified immediately after "AU" in *sendmail.cf* (or any file specified explicitly for "sendmail") and this is mandatory if "-a" command line option is given.

vii. To update databases used by "sendmail" to store ACTIONS file information "-bA" command line switch or `argv[0]==actionsdb` is used.

Notes: 1. Options a, n, u, l and U are considered only if (i) is true (i.e. "-a" command line switch is given).

2. All the options are to be handled only if "sendmail" is run as "root" or a trusted user. Other users do not have permission to run in this mode of operation.

3. In (vi), AU command will take some form as follows depending on the UUCP version being used.

```
AU/usr/lib/uucp/uucico -S$h (Taylor UUCP)
AU/usr/lib/uucp/myuucico -s$h (A shell script or a program
            to do the job)
```

In this case "sendmail" has to lock all the other messages queued for that site and once the connection is lost or completed it will unlock them. If the system crashes before transfer is complete, next time all the messages will be unlocked first to allow receiving by the remote site, but will lock them again if this server initiates it. (For reasoning, see explanation for **SIM**) Therefore the server will keep a status file and during the startup of "sendmail", it will be checked to get the history of events.

APPENDIX B

Pseudo Code for the Implementation:

```
IF (-a) THEN
/* Get configuration options */
    {
        IF (Aa<filename>) THEN f_actions=<filename>
            ELSE f_actions=/etc/actions
        IF (An<filename>) THEN f_nodecat=<filename>
            ELSE f_nodecat=/etc/nodecat
        IF (Au<filename>) THEN f_haeders=<filename>
            ELSE f_headers=/etc/headers
     /* Non-existance of f_actions, f_nodecat and f_headers
        is silently ignored */
        IF (Al<filename>) THEN f_actionlog=<filename>
            ELSE f_actiolog=/var/adm/action.log
     /* If log file does not exist, create it and update */
        IF (AU<command>) THEN f_uucommand=<command>
            ELSE IF (command==null) give an error and exit
        IF (-ac) THEN argv[0]=actionsdb /* see next module */
        IF found /etc/print-server THEN print-server=that-server
           ELSE
                {
                    print-server=null
                    give a warning message and ignore
                }
        IF found /etc/fax-server THEN fax-server=that-server
           ELSE
                {
                    fax-server=null
                    give a warning message and ignore
                }
        IF found /etc/cur-server THEN cur-server=that-server
           ELSE
                {
                    cur-server=null
                    give a warning message and ignore
                }
   }
```

```
Define enumerated types for OPTIONS and NODETYPES /* ref. [1] */

  {
   WHILE NOT END OF f_nodecat
    {
      Parse entry in f_nodecat
      IF correct keep correct entry temporarily till f_actions is
               processed
      IF wrong (AND NOT fatal) ignore the wrong entry but produce
               error notification
               /* fatal errors are unlikely */
    }

   WHILE NOT END OF f_actions
    {
      Parse entry for one host in f_actions
      IF correct update dabase
      ELSE IF wrong AND NOT fatal THEN produce error report
                 and proceed
      /* eg. fax-server not defined, SIM defined for LS, etc */
      ELSE IF fatal THEN exit
    }
   Flush temporary f_nodecat variables from memory
  }
  IF argv[0]==actionsdb THEN exit

  FOR each message DO
  {
  /* Look for system messages such as SIM-attempt-failures, FTP
  attempt failures, etc. and node requests such as KDB requests,
  ACTIONS file modifications etc. */

    IF for this-host's-postmaster DO
       {
           OPTION={FTP,SPM,KDB,DIS,PRN,FAX,CUR,SIM,PGR,WWW}
           get OPTION, flag
           IF (OPTION-related)
               {
                 set the flag
                 DO OPTION-module(null,flag)
               }
```

```
            /* eg. if OPTION==FTP then
            OPTION-related=FTP-related and
            OPTION-module=FTP-module */
        IF (ACTIONS-related) AND (update allowed)
            {
                update f_actions file
                fork actionsdb
            }
        ELSE IF (ACTIONS-related) AND (update-not-allowed)
            send to root
            /* root will have to manually update ACTIONS file */
    }
ELSE
{
    Get next-host
    Get node-category
    IF (HS) skip the message /* i.e return */
    ELSE
    {
      get info from actions database for next-host
      IF (first.option==SPM)
        { set global.SPM true
          get $L and $S values /* lines and size */
        }
      FTP.option=false
      FOR all other options DO /* i.e. except SPM */
      /* no need to check validity of options for this node
        since it has been already parsed during f_actions
        file parsing */
      {
        IF option==FTP
            {
             FTP.option==True
             DO FTP-module ( string-in-actions-database, flag )
            }
        IF option==SIM
             DO SIM-module ( string-in-actions-database, flag )
        IF option==SPM
             DO SPM-module ( string-in-actions-database, flag )
        IF option==KDB
             DO KDB-module ( string-in-actions-dtabase, flag )
        IF option==DIS
             DO DIS-module ( string-in-actions-database, flag )
        IF option==CUR
```

```
                    DO CUR-module ( string-in-actions-database, flag )
               IF option==PRN
                    DO PRN-module ( string-in-actions-database, flag )
               IF option==FAX
                    DO FAX-module ( string-in-actions-database, flag )
               IF option==PGR
                    DO PGR-module ( string-in-actions-database, flag )
               IF option==WWW
                    DO WWW-module ( string-in-actions-database, flag )
               IF nodecategory==(DU OR ND) AND (FTP.option==False)
                    DO FTP-module ( null, null )
               IF nodecategory==ND
                    DO CUR-module ( null, null )
             }
           } /* end ELSE */
         } /* end ELSE */

FTP-module( string, flag )
{
    IF (string==null) AND (flag==null) THEN
    {
        FOR message-header AND body-part-headers DO
        {
          search "content-type: message/external-body; access-type"
          IF type IN { ftp, tftp, anon-ftp }
                    {
                        get ftp info
                            /* "site" - the machine from which the file
                                may be obtained,
                                "name" - name of the file that contains
                                actual body data
                                "type" - ftp/tftp/anon-ftp
                                */
                        IF type==ftp {
                                        get login-id and password
                                        IF not found them
                                         {
                                          send a message asking for them
                                          return
                                         }
                                    }
                        fork ftp session
                        store "server-trying-ftp" file in the data base
                                /* ref. KDB */
```

```
                    update KDBINFO without ftp info
                    store ftp info in a file temporarily for
                                                    trapping
                    change header to "Access-type: mailserver with
                                parameters server=this-server
                } /* end IF */
    } /* end FOR */
}  /* end IF */
ELSE IF (string!=null) AND (flag==null) THEN
{
    FOR message-header AND body-part-headers DO
       FOR condition[i] in string /* condition for FTP */
       {
            IF found condition.found[i]=True
       }
       for all condition.found[i] parse according to boolean
                                        statement in the string
       IF the above returned True DO
       {
           search "content-type: message/external-body; access-type"
           IF type IN { ftp, tftp, anon-ftp }
            {
                get ftp info
                   /* "site" - the machine from which the file
                                                may be obtained,
                      "name" - name of the file that contains
                                  actual body data
                      "type" - ftp/tftp/anon-ftp
                   */
                 IF type==ftp {
                                get login-id and password
                                IF not found them
                                 {
                                   send a message asking for them
                                   return
                                 }
                            }
                 fork ftp session
                 store "server-trying-ftp" file in the data base
                        /* ref. KDB */
                 update KDBINFO without ftp info
                 store ftp info in a file temporarily for
                                                trapping
                 change header to "Access-type: mailserver with
```

```
                                        parameters server=this-server

                    } /* end IF type IN */
                } /* end IF above */
        } /* end ELSE IF */
        ELSE IF (flag==failed) DO
        {
            get ftp info from the temporary trapping file
            send an e-mail asking for verification of data
            keep "server-tried ftp and failed" file in the data base
            update KDBINFO without ftp info
            /* keep stored ftp info temporary trapping file further */
        } /* end ELSE IF */
        ELSE IF (flag==recipient.request) /* with altered parameters for
                                             login id & password */
        {
            check temporary file info and get them
            fork ftp session with given parameters /* file information
                                            should not get changed */
            store "server-trying-ftp-again-with-given-parameters" file
            update KDBINFO without ftp info
        } /* end ELSE IF */
        ELSE IF (flag==recipient.request.failed)
        {
            store "server-unable-to-get-file"
            update KDBINFO without ftp info
            remove temporary ftp info file
        }
        ELSE IF (flag==success)
        {
            IF (global.SPM==true) split message
            store file
            update KDBINFO with ftp info
            remove temporary ftp info file
        }
        ELSE IF (flag==FTP.request) /* this is from KDB links to FTP */
        {
            fork ftp session with given parameters in string
            /* file and site information should be taken from string */
            store "server-trying-ftp" file
            update KDBINFO without ftp info
            store ftp info in a file temporarily for trapping
        }
    } /* end FTP-module */
```

```
/* Note: ftp session should send messages to postmaster of this server
         with required flags.
         ftp session program should have mechanisms for re-trying in
         case of time-outs.
         updating KDBINFO should consider EXn    */

SIM-module (string, flag)  /* flag cannot have SIM.request */
{
    IF (nodecategory==DU) DO
    {
        IF (flag==null) AND (condition found) DO
        /* refer how condition evaluated in FTP-module */
        {
            lock all messages for the remote site
            lock receiving from the remote site
            fork f_uucommand with this message spooled
        }
        ELSE IF (flag==SIM.failed.first)
            fork f_uucommand with this message spooled
        ELSE IF (flag==SIM.failed.second) DO
        {
            unlock all messages
            unlock receiving
            send this message as normal e-mail
        }
        ELSE IF (flag==SIM.success)
        {
            unlock all messages
            unlock receiving
        }
    } /* end nodecategory */

ELSE IF (nodecategory==LS)
        insert "Precedence: Special-delivery" header
} /* end SIM-module */

SPM-module (string, flag)
{
    search for "Message/partial"
    IF (flag==null) AND (message/partial not present) DO
    {
        IF $L OR $S given get lower of them ELSE assume default values
        IF Content-Type: Application/* {
```

```
                                        encode base64
                                        add CTE header
                                       }
        add Mesage/Partial header
        split message
    } /* end IF */
    ELSE IF (flag==null) AND (message/partial present) DO
    {
        update partial-message-status file (temporary)
        IF all parts found DO
            {
                combine all parts
                remove message/partial header
                /* this will be handled normally in the next call to
                   this procedure and hence leave it at this stage */
            }
    } /* end ELSE IF */
    ELSE IF (flag==SPM.request) DO /* this is the only user request that
                                      could come through postmaster */
    {
        get file info from KDBINFO and verify authentication
        IF $L OR $S given get lower of them ELSE assume defaults
        IF Content-Type: Application/* {
                                        encode base64
                                        add CTE header
                                       }
        add Mesage/partial header
        split message
    } /* end ELSE IF */
} /* end SPM-module */

KDB-module (string, flag)
{
    IF (flag==null) and (condition found) DO
    /* refer how condition evaluated in FTP-module */
    {
        types.found=false  /* to handle ftp links */
        get FTP.condition for this node
        IF (condition found) OR (nodecategory==DU) DO
        {
            WHILE not end of message DO
            {
                look for "Message/external-body: access-type"
                get type[i]
```

```
                IF type[i] IN {ftp,tftp,anon-ftp}
                {
                   get info-type[i] /* site, file, etc */
                   change to "Message/external-body:
                               access-type=mailserver" with
                               server=this-server
                   types.found=true
                   IF type[i]==ftp
                   {
                        get login-id and password
                        IF not found them
                        {
                            send a message asking for them
                            store ftp info temporarily in a file
                            type[i]=ftp.wait /* just to avoid ftp'ing */
                        }
                   }
                }
                i++
            } /* end WHILE */
        } /* end IF */
        IF global.SPM==true
        {
            split the message
            store in data base as Message/partial
        }
        update KDBINFO with EXn
        IF (types.found=true) FOR all i DO
            {
                IF type[i] IN {ftp,tftp,anon-ftp}
                DO FTP-module ( info.type[i], FTP.request )
            }
    } /* end flag==null */
    ELSE (flag==KDB.request) DO
    {
        get file info from KDBINFO and verify authentication
        send file as message
    }
} /* end KDB-module */


DIS-module (string, flag)
{
    IF (flag==null) DO
    search condition in message headers OR see whether mesage stisfies
                condition /* size, length etc..*/
                /* refer how condition evaluated in FTP-module */
```

```
    IF found DO
    {
        IF SAB specified DO
        {
            search SAB condition
            IF found { prepare abstract
                       send abstract }
        } /* end IF SAB */
        send notification to sender
    } /* end IF */
    IF (flag==DIS.request) DO /* this is to discard from the database */
    {
        get file info from KDBINFO and verify authentication
        discard file
        send a message to sender
    }
} /* end DIS-module */

CUR-module (string, flag)
{
    IF (string==null) AND (flag==null) DO /* nodetype is ND */
    {
        IF (cur-server==null) spool in courier base
        ELSE send to cur-server
    }
    ELSE IF ((string!=null) AND (flag==null)) AND (condition found) DO
    /* nodetype is LS or DU */
    /* refer how condition evaluated in FTP-module */
    {
        IF (cur-server==null) spool in courier base
        ELSE send to cur-server
        IF SAB found
            { check for condition
              send abstract to recipient by e-mail
            }
        ELSE send abstract /* I choosed to send abstract by default */
    } /* end ELSE IF */
    ELSE IF (flag==CUR.request) DO
    {
        get file info from KDBINFO and verify authentication
        IF (cur-server==null) spool in courier base
        ELSE send to cur-server
        /* no need to send abstract since it was sent when saved in
           database */
    } /* end ELSE IF */
} /* end CUR-module */

PRN-module (string, flag)
{
    IF (print-server==null) print-server=this-server
```

```
        ELSE print-server=that server
IF (flag==null) AND (condition found) DO
/* refer how condition evaluated in FTP-module */
{
    IF Mime-version not present DO
    {
        insert address and blank lines [1]
        IF print-server==this-server THEN send message to prn
        ELSE send to print-server
    }
    ELSE IF Mime-version present DO
    {
        IF Message/partial present DO
        {
            update partial-message-status file (temporarily)
            IF all prts found DO
            {
                combine the parts
                remove Message/partial header
                remove entry from partial-message-status file
            }
            ELSE return
        }
        IF Multipart/Alternative present DO
        {
            select first alternative
            insert address and blank lines [1]
            IF print-server==this-server THEN send message to prn
            ELSE send message to print-server
        }
        ELSE IF audio/video/application found DO
         /* application - for the time being only. later we will
            handle it more precisely */
        {
            skip audio/video/application body parts
            form one message using the rest of the message
            insert address and blank lines [1]
            IF print-server==this-server THEN send message to prn
            ELSE send message to print-server
        }
    } /* end ELSE IF Mime-version */
} /* end (flag==null)
ELSE IF (flag==PRN.request) DO
```

```
    {
        get file info from KDBINFO and verify authentication
        insert address and blank lines [1]
        IF print-server==this-server send file to prn
        ELSE send to print-server
    }
} /* end PRN-module */


FAX-module (string, flag)
{
    IF ((condition found) AND (/etc/fax-server not found))
      OR ((flag==FAX.request) AND (/etc/fax-server not found))
      /* refer how condition evaluated in FTP-module */
    {
        give an error
        IF nodecategory in {LS,DU}
            send message as an e-mail - mention that server cannot fax
        ELSE send to postmaster of this server
    }

    ELSE IF ((condition found) AND (/etc/fax-server found))
                                    AND (flag==null) DO
    {
        IF Mime-version not present DO
        {
            insert /etc/fax-server file
            replace FAXNUMBER
            send message to fax-server
        }
        IF Mime-version present DO
        {
            IF Message/partial present DO
            {
                update partial-message-status file (temporarily)
                IF all parts found
                    { combine them
                      remove entry from partial-message-status file
                    }
                ELSE return
            }
            ELSE IF Multipart/Alternative present DO
            {
                select first alternative
```

```
                insert /etc/fax-server
                replace FAXNUMBER
                send that part to fax-server
            }
            ELSE IF audio/video/application found DO
             /* application - for the time being only. later we will
                handle it more precisely */
            {
                skip audio/video/application body parts
                form one message using the rest of the message
                insert /etc/fax-server
                replace FAXNUMBER
                send message to fax-server
            }
        } /* end Mime-version */
    } /* end ELSE IF ... flag==null */

    ELSE IF (flag==FAX.request) DO /* non existance of fax-server
                                      handled erlier */
    {
        get file info from KDBINFO and verify authentication
        insert /etc/fax-server
        replace FAXNUMBER
        send to fax-server
    }
} /* end FAX-module */



/* The following modules are not yet designed */

PGR-module ( string-in-actions-database, flag )
{
/* not implemented yet /
} /* end PGR-module */

WWW-module ( string-in-actions-database, flag )
{
/* not implemented yet /
} /* end WWW-module */
```