ISTITUTO NAZIONALE DI FISICA NUCLEARE

Sezione di Pisa

INFN/TC-88/26
9 Settembre 1988

R. Fantechi, R. Brazioli, V. Emiliani, S.M. Fisher, P. Palazzi, Z.X. Qian, W.R. Zhao:

Amsend: A Database Driven Interface to Electronic Mail Systems

Servizio Documentazione
dei Laboratori Nazionali di Frascati

INFN - Istituto Nazionale di Fisica Nucleare
Sezione di Pisa

INFN/TC-88/26
9 Settembre 1988

AMSEND: A DATABASE DRIVEN INTERFACE TO ELECTRONIC MAIL SYSTEMS (*)

R. Fantechi :INFN, Sezione di Pisa, Italy
R. Brazioli :CERN DD Division, Geneva, Switzerland
V. Emiliani :Department of Physics, University of Parma, Italy
S.M. Fisher :Rutherford Appleton Laboratory, Chilton, UK
P. Palazzi :CERN DD Division, Geneva, Switzerland
Z.X. Qian :IHEP Academia Sinica, Beijing, China
W.R. Zhao :IHEP Academia Sinica, Beijing, China

ABSTRACT

In large High Energy Physics Experiments, each of them with about 500 collaborators, there is a strong need for reliable communication. Though existing electronic mail systems and gateways are very useful, the complex address formats and the fact that people doesn't always know on which machine and with which login name his friend is working, make the use of electronic mail systems a little difficult.

AMSEND is an answer to these problems: all the knowledge of computers inside a collaboration is held in a relational database and AMSEND (using a Fortran callable package to access these data) finds automatically the address and the procedure to use starting from the family name of the recipient. AMSEND is running on VAX/VMS and on IBM/CMS.

(*) Presentato da R. Fantechi al Simposio Europeo DECUS, Roma 1987

INTRODUCTION

At CERN, the European Laboratory for Particle Physics, scientists are collaborating to build four large experiments to be placed 50 to 150 m. underground in the tunnel of the LEP accelerator, which will be ready by Spring 1989. The design of the detectors started more than 6 years ago and the expected life of the experiment is more than ten years.

Each of these collaborations involves about 500 physicists, engineers and technicians from about 30 laboratories in Europe and outside. A given part of the detector is the responsibility of more than one institute with people working at CERN and at home.

This number of people and the fact that they are working on the same topics in places hundreds of kilometers apart, create the need for an intense exchange of information. Periodic meetings are one of the ways to exchange news and latest results. Paper mail is used for less urgent communication and for the periodic delivery of collaboration notes and documentation.

For fast communication, telephone and electronic mail are used. The latter is becoming more and more used, especially by those working on software.

Problems with Electronic Mail Systems

In the Aleph collaboration, people work on about 50 computers both at CERN and at home laboratories. It has been chosen to use only VAXes (with the VMS operating system) and IBM (with the VM/CMS operating system). The machines are connected to several networks: DECNET for a part of the VAXes, JANET for all the machines in the United Kingdom and BITNET for other VAXes and IBMs (fig. 1). Moreover there are some machines connected also to the public X25 network. Several different mail systems are in use. Fortunately CERN provides a set of mail gateways between several networks. These gateways together with some others around the world provide an almost 100% connectivity between Aleph computers.

The use of the gateway is a little clumsy: physicists are often far from the networking world so the complex address formats needed to send mail to a friend through a gateway or the need to remember which is the best gateway to use or which is the mail procedure to use with that gateway are enough to discourage not only the casual user, but also those physicists who are accustomed to electronic mail.
There is another aspect of electronic mail systems which has to be solved inside a big collaboration : the knowledge, for each member, of the data (login name, computer name, network name) needed to send a mail message. All this information is not easy to guess: the login name is often different from the family name, computers have often cryptic names as nodes of networks and are often on more than one network. The user does not know what gateway to use.
These reasons led us to develop AMSEND and its database as a solution to Aleph electronic mail problems.

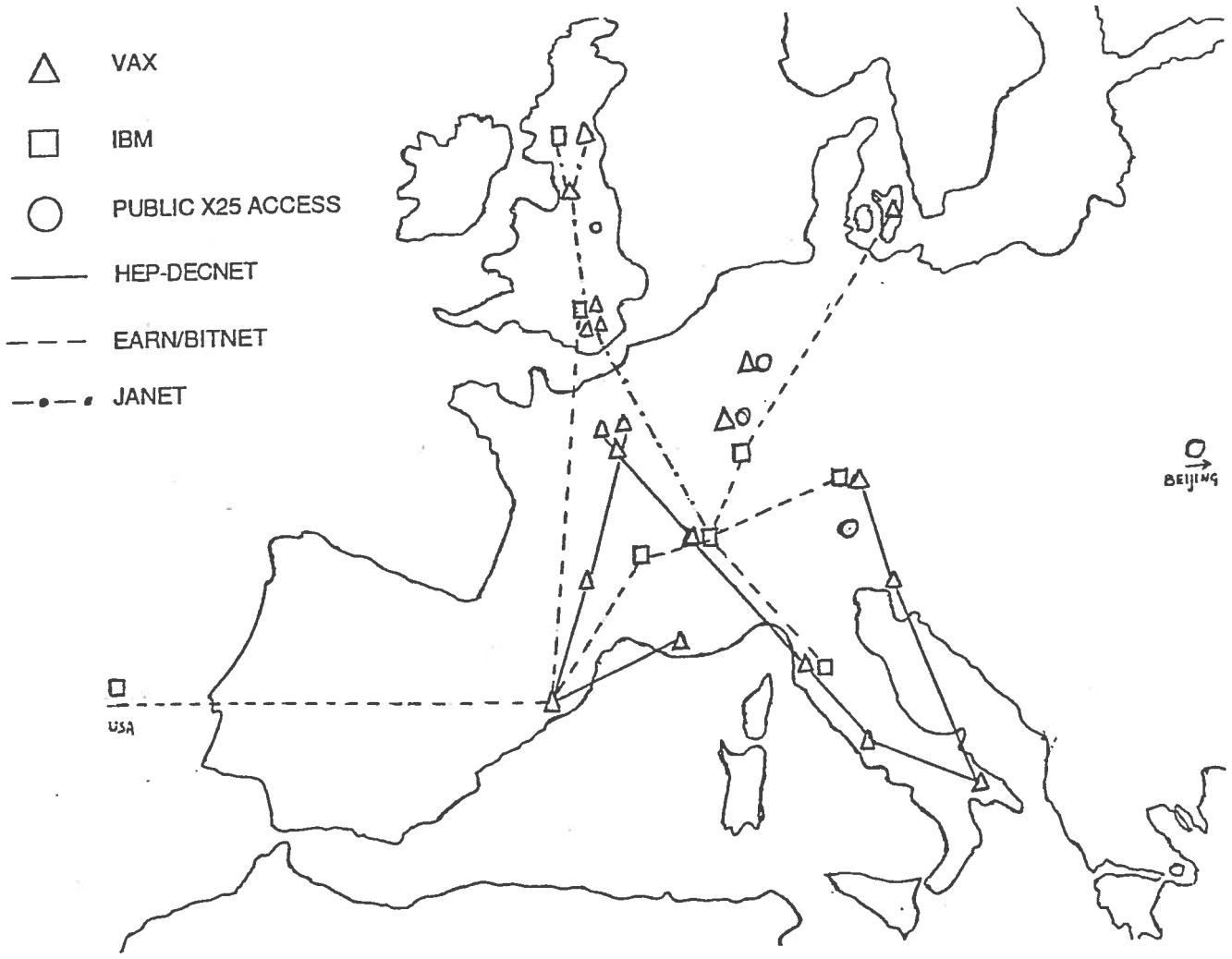


Fig. 1: ALEPH Computers and Networks

WHAT AMSEND IS

The aim of AMSEND is to have an interface to Electronic Mail Systems which is able to:

- Initiate mail procedures with simple commands in the operating system syntax.
- Identify Aleph people by the family name or from expressions like "member@institute", rather than the less obvious syntax "user@node.network".
- Interpret distribution lists containing recipient names in the form described above.
- Find out which procedure has to be followed for the reply to that message and append to the message an explanation of how the recipient may reply if he does not use AMSEND.
- Run on both VAX and IBM.

AMSEND is not designed to be another Mail system (like VMSMAIL, EAN, etc.). It uses existing mechanisms for foldering and reply. Moreover we are not creating another Electronic Mail Directory, which could hold addresses for everybody and which could be distributed. We need a well defined amount of data relative to the collaboration which can

be kept up to date, because communication with the right person in the right place is vital. Messages to the outside world are few and often they are sent by people with some knowledge of electronic mail.

HOW AMSEND IS BUILT

AMSEND is implemented as a Fortran program which takes its input from a command line and from the ADAMO Direct Access File, a database which holds the knowledge about the Aleph network. Output results are a series of mail commands and optionally a log message.

The command line

The command line contains a list of "recipients" which can usually be the surname of a person or his surname and firstname. Other possibilities have been implemented, like "member@institut" or "member@node" to address people directly where they are currently supposed to be working, rather than at their specified preferred logins. It is possible also to address someone not registered in the Aleph database with "!user@node.network" in the standard way of Electronic Mail addresses (the network must anyway be known to the database) and with "!" which acts as an escape character to recognize this special syntax. With "*" one can address all the Aleph members. Distribution lists are allowed, in the form of files with one line for each recipient in any of the described formats. A nick-name mechanism is also implemented to allow shorter definitions of complex names.

Several options can be added to the command:

- LOG to display log information of the work being done.
- NOSEND to inhibit the execution of the mail commands.
- FILE to specify the name of a file containing the message. If not specified, the standard editor on that machine is entered to compose the message.
- SUBJECT to define the subject for the message.

Structure of the program

The program can be divided into four principal parts (fig. 2a,2b):

Initialisation The ADAMO structure is initialized, the command line is interpreted and for each recipient work tables with login information are built. Moreover the message file is edited if it is the case. The database information is not read now, but each table is read as it is needed, saving wait time at program startup.

Creation of network information In this phase, the program finds for each pair user/login all the possible paths, in the sense of all possible destination networks for these users.

Definition of the way to follow After getting information about the sender (through system primitives), the program finds for each user all the possible ways and select the best one using the priority field in the gateway table. For each way, it builds also the way to be followed for the reply. This information is added at the end of the message as a "Reply

Hint". In spite of the fact that different messages are so generated, it is possible to group all the reply ways in few groups, with a message for each group. This mechanism and the distribution list capability of many mail systems minimize the number of messages sent.

Definition of the mail commands The program starts a loop on different mail procedures and on different reply ways to prepare mail messages. The following steps are followed:

- The reply hint is built according to the characteristics of the reply procedure and appended to the message.
- If the procedure to be used needs to add header lines in front of the message, they are generated together with the commands to put them in the message.
- If the procedure needs a distribution list, it is created.
- The commands to send the message are generated.

At the end of the loop, all the mail commands are executed. The details of the behaviour of this last part vary depending on the implementation (VAX or IBM).

THE NETWORK DATABASE

AMSEND relies upon a relational database which holds network information. Since the beginning of 1986, it was decided to setup a relational database under ORACLE for the information related to the work of the Aleph Secretariat. It has been natural to extend this database with a subschema Alephnet for the networking within Aleph. Information is updated using ORACLE screens and, using an ADAMO tool, one is able to extract all the information in the form of a Direct Access File which can be read by a Fortran program using the ADAMO Table Package.

The Alephnet subschema (fig. 3a) is composed of 9 tables and in addition refers to two tables (Member and Institute) in the subschema WhosWho (fig. 3b).

Computer

Holds information about a computer as a machine (type, operating system, functions, etc.) A relationship column to the member table provides the definition of the person responsible for that machine.

ComputeOn

Is a table to implement a many to many relationship between computers and institutes. A computer is used by one or more institutes and one institute can use more than one computer.

Login

Defines login information for people on computers. They are the login name and the "preferred login" flag. Two relation columns with member and computer provide the knowledge of who is the member and on which machine he is.

Network

Defines networks specifying network names. The concept of network in the context of mail commands has been extended to entities like subsets of other networks and environments where specific mail commands are to be used.

Node

Is a table to implement a many to many relationship between computers and networks. A computer can be a node of one or more networks. A network obviously contains many computers. The table contains data about node names and node numbers (as in Decnet).

Gateway

Is a table (fig.4) which defines the functionality of electronic mail transfer between two networks. It has two relationship columns to the network table to access the origin and destination network. Each row contains the mail procedure to be used (with a relationship column to the table MailProc), the address format and a priority number associated with it. In the case of selection of more than one way to send mail, that with the highest priority value is chosen. This happens usually when at least one of the two computers is on more than one network.

MailRules

It gives (fig.5) a complete description of the sequence of commands to issue for each mail procedure (referred with a relation column). It can contain both lines of commands and line of addressing information to be put inside the message. Additional information includes the sequence number of the command or the addressing line and if the line is a destination address (to make distribution lists).

MailProc

Defines Mail procedures to use, with their names, if they allow distribution lists, and if it runs on IBM or VAX. Several "dummy Mail procedures" are defined to allow for example to send mail from IBM nodes through gateways adding addressing lines in front of the message.

MailInst

Implements a many to many relationship between computer and mailproc. A computer can have one or more mail procedures installed and one procedure can be installed on more than one machine.

PORTABILITY ON VAX AND IBM

The kernel of the program is written in standard Fortran 77, avoiding all the extensions. An exception is the use of long names on VAX for table names and attributes. They are automatically converted into six character names in the IBM version (fig. 6a,6b). System dependent routines are very few (initialisation, upper and lower case conversion and the final routine) and they are maintained with the ADAMO tool SAC which can process conditional code in a precompiler step (fig.7).

VMS Implementation

The CDU utility is used to define and parse the command line.
AMSEND uses CLI routines to get the values of the various parameters and qualifiers.
The editor to compose the message is spawned using LIB$SPAWN.
The user can choose the preferred editor changing a logical name.
The routine which prepares mail write several files with the messages and reply hints, distribution lists and a file with all the commands to execute.
This single command file is spawned at the end.

VM Implementation

The program is driven by a REXX exec file which parses the input and sets REXX variables with the parameter values.
The initialization routine gets the values of these variables using a Fortran callable interface to REXX execs.
The last subroutine writes a file with all the program generated information (header lines, reply hints, commands, distribution lists)
After program exit, the REXX exec reads this file, prepare mail messages, sets distribution lists and executes commands.

HOW AMSEND LOOKS TO THE USER

The use of AMSEND is quite simple: the command are in the standard syntax of the host operating system and standard help files are provided on both machines. Fig. 8a and 8b show a part of the VMS help file for AMSEND. Fig.9 is the log of an AMSEND session, while in fig. 10 a message with the reply hint is showed.

The installation on another machine is easy: files have to be copied and then on VMS two logical names have to be defined in the system login file as well as the definition of the command. On IBM no other operation is necessary.

EXPERIENCE AND THE FUTURE

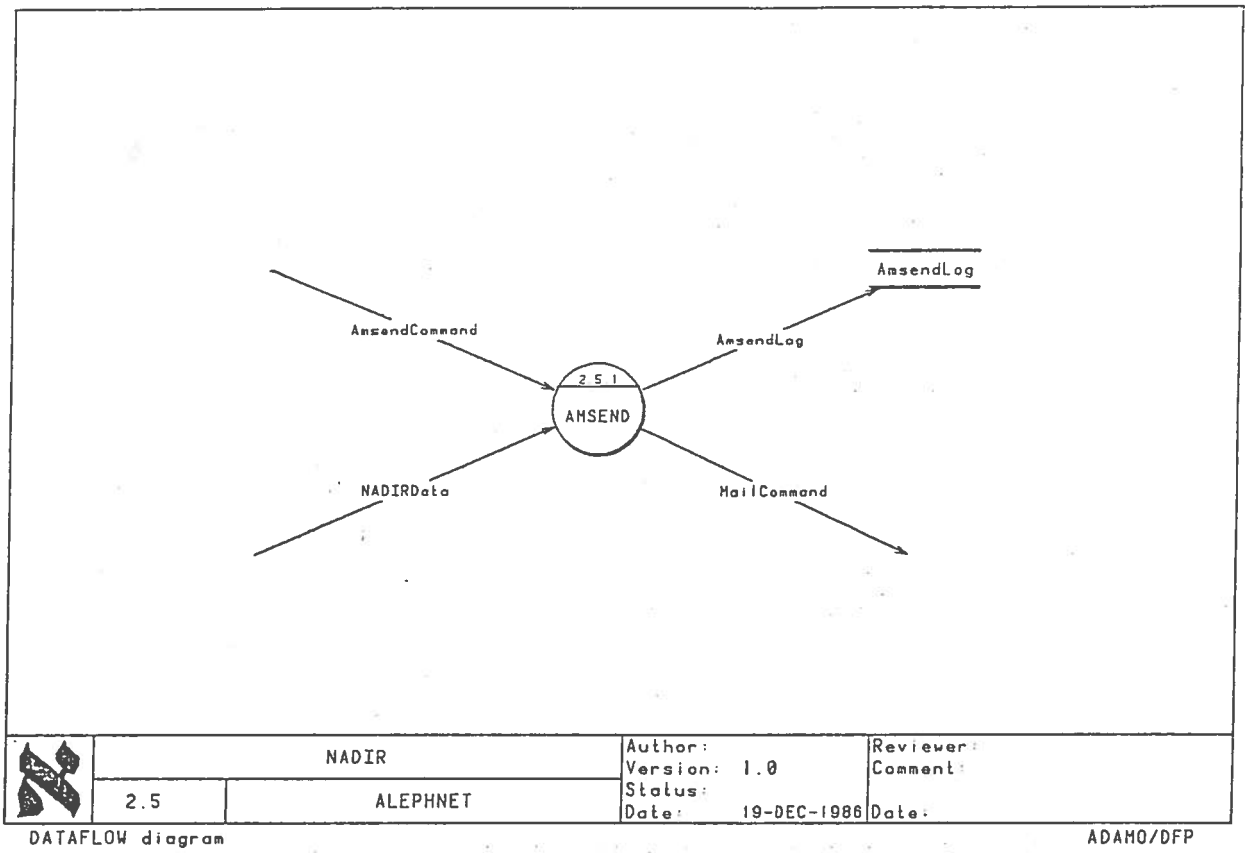
The first version of AMSEND was delivered in October 1986 on a central VAX at CERN and on some other VAXes. After some months of use, the development of a second version was started with the following goals:

- Add some feature and options (i.e. send to non Aleph members, the NOSEND option).
- Complete rewriting of the last part of the program, with the definition of the mailrules table. In this way we can have all the logic for the commands in the database.
- Minimizing the number of commands spawned.
- Moving AMSEND to IBM.

The second version was released at the end of July 1987 and installed on VAXes at CERN and outside. The IBM version, which was released at the same time and installed on the CERN IBM, is a first one and so will be subjected to tests, expecially outside CERN.

AMSEND is being used also by another LEP experiment and the collaboration with them has been fruitful: their network configuration is different from the Aleph one, but the program works with no code changes, only changing definitions in the database. Moreover they helped us in the debugging phase.

For the future we foresee little development. The installation of AMSEND on all the Aleph machines and the use of it will help us to tune both the program and the database.



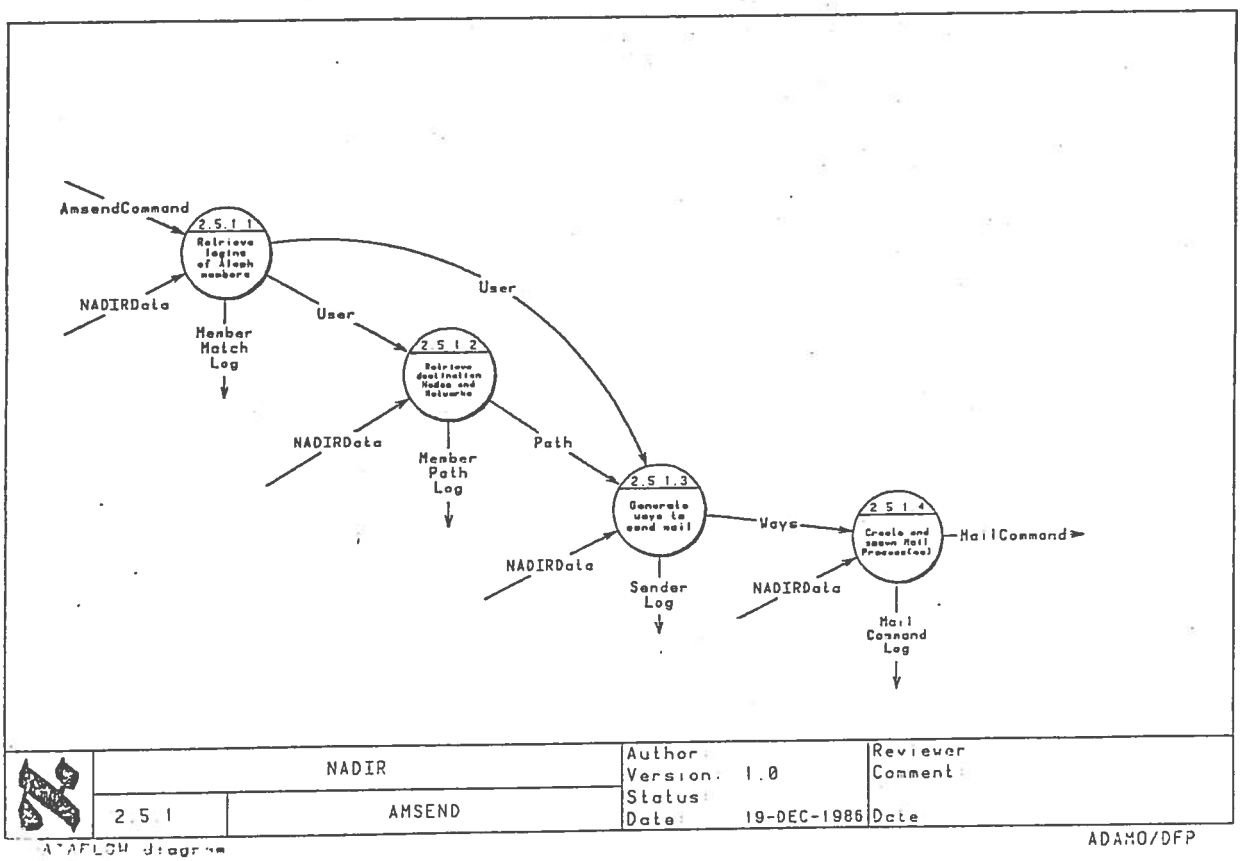


Fig 2a, 2b: AMSEND Data Flow Diagrams

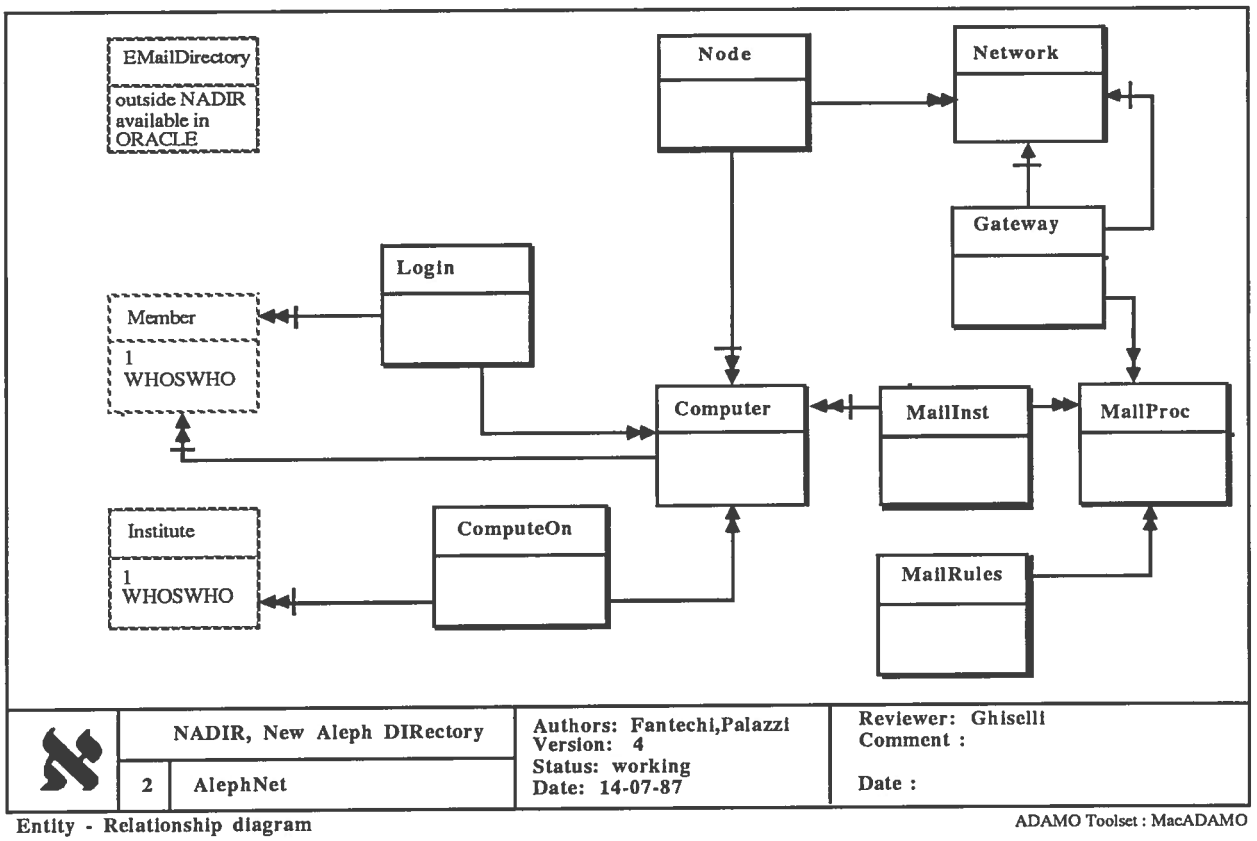


Fig. 3a: The AlephNet Subschema

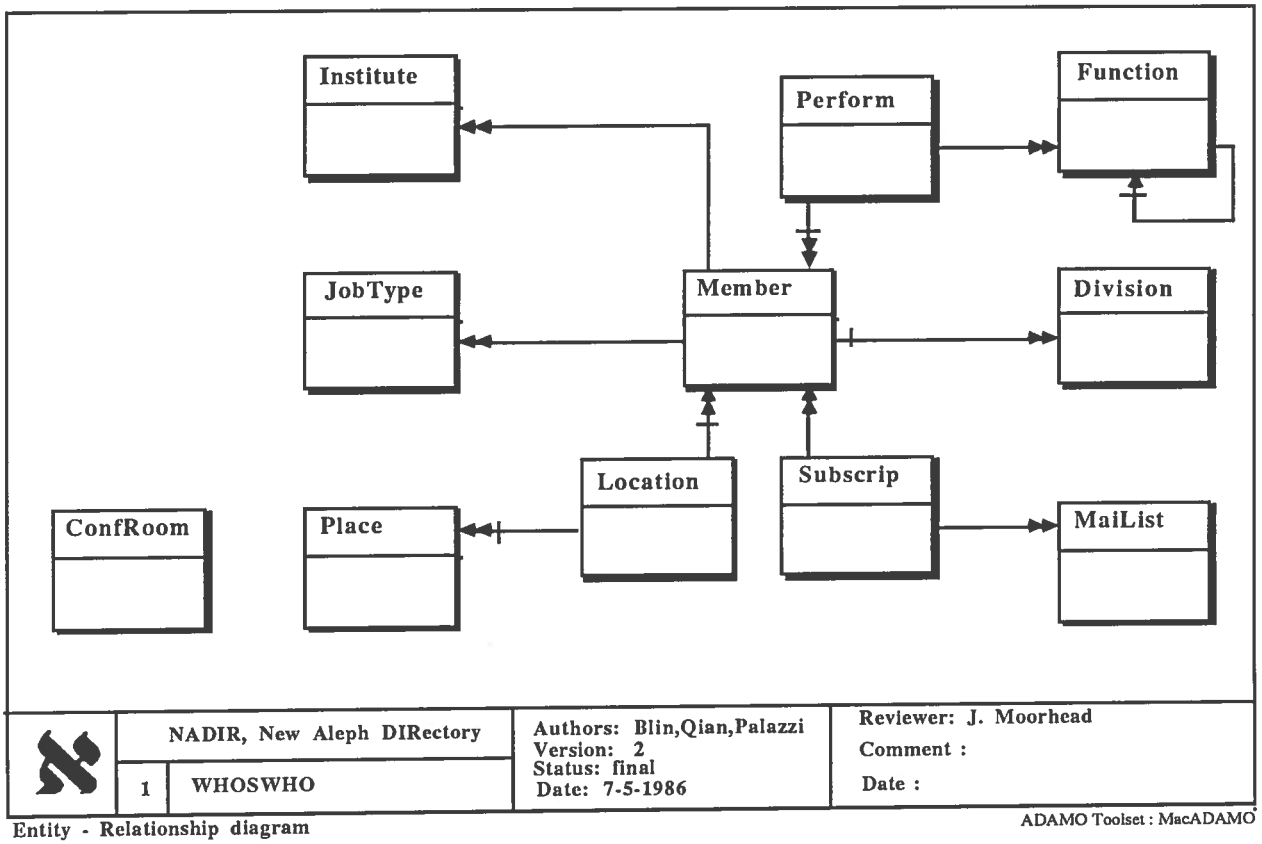


Fig. 3b: The WhosWho Subschema

NETNAME	NETNAME	Procedure	Address Format	Priority	Reply
BITNET	BITNET	VMSMAIL	jnet%"[us]@[ho]"	40	N
BITNET	BITNET	MAIL	[us]@[ho]	30	N
BITNET	BITNET	NOTE	[us] AT [ho]	31	N
BITNET	CERN	MAIL	[us]@[ho].CERN	30	Y
BITNET	CERN	PROC_1	cernvax mailer	90	Y
BITNET	CERN	PROC_2	cernvax mailer	50	Y
BITNET	DECNET	PROC_1	cernvax mailer	90	Y
BITNET	DECNET	PROC_2	cernvax mailer	90	Y
BITNET	INFNET	MAIL	infngw@ipivaxin	44	Y
BITNET	INFNET	PROC_3	ipivaxin infngw	44	Y
BITNET	INFNET	PROC_6	jnet%"infngw@ipivaxin"	44	Y
BITNET	JANET	MAIL	[us]@[re]	80	N
BITNET	JANET	PROC_2	cernvax mailer	90	N
BITNET	JANET	PROC_5	ukacrl mailer	85	N
BITNET	JANET	PROC_4	ukacrl mailer	85	N
BITNET	JANET	PROC_1	cernvax mailer	90	N
CERN	BITNET	EAN	[us]@[ho].BITNET	46	N
CERN	CERN	VMSMAIL	vxgift::mint%"[us]@[ho].CERN"	40	Y
CERN	CERN	MAIL	[us]@[ho]	41	Y
CERN	CERN	EAN	[us]@[ho]	21	Y
CERN	DECNET	VMSMAIL	vxgift::mint%"[us]@[ho].DECNET"	21	Y
CERN	DECNET	MAIL	[us]@[ho].DECNET	40	Y
CERN	DECNET	EAN	[us]@[ho].DECNET	30	Y
CERN	JANET	VMSMAIL	vxgift::mint%"[us]@[re]"	49	N
CERN	JANET	MAIL	[us]@[re]	51	N
CERN	JANET	EAN	[us]@[re]	50	N
DECNET	BITNET	VMSMAIL	vxgift::mint%"[us]@[ho].BITNET"	45	N
DECNET	BITNET	VMSMAIL	vxdev::gateway::"[us]@[ho]"	55	N
DECNET	CERN	VMSMAIL	vxgift::mint%"[us]@[ho].CERN"	40	Y
DECNET	DECNET	VMSMAIL	vxgift::mint%"[us]@[ho].DECNET"	50	Y
DECNET	DECNET	VMSMAIL	[ho]::[us]	20	Y
DECNET	JANET	VMSMAIL	vxgift::mint%"[us]@[re]"	60	N
INFNET	BITNET	PROC_6	vaxpi::infngw	44	N
JANET	BITNET	NOTE	[us] EARN.[ho]	49	N
JANET	BITNET	POST	[us]%[ho]@UK.AC.RL.EARN	49	N
JANET	CERN	NOTE	[us] CERN.[ho]	51	N
JANET	CERN	POST	[us]@CERN.[ho]	51	N
JANET	JANET	NOTE	[us] [eh]	50	N
JANET	JANET	POST	[us]@[eh]	50	N
PSI	PSI	VMSMAIL	[ho]::[us]	24	N

Fig. 4: The content of the Gateway table

Table : MailRules ADAMO/TAP
Count = 42

*Index : ID +ID
Printed along : ID [MINC,42]

ID	Rule	Com	Dest	SeqNo	#MailP
1	MAIL/SUBJ=[sj] [fn] [af]	T	F	1	# 1
					#
2	POST [af] [fn]	T	F	1	# 4
					#
3	EXEC MAIL [af] (NOEDIT NOLOG FILE [fn] SUBJECT [sj]	T	F	1	# 2
					#
4	EXEC NOTE [af] (NONOTEBOOK PROFILE AMSEND$T	T	F	1	# 3
					#
5	SEND/FILE/PUNCH/CLASS=M [fn] CERNVAX MAILER	T	F	1	# 5
6	From: <[su]@[sh].BITNET>	F	F	1	# 5
7	To: <[us]@[ho].[dn]>	F	T	2	# 5
8	Subj: [sj]	F	F	3	# 5
	.....				#
14	MAIL/SUBJ=[sj] [fn] [af]	T	F	1	# 11
15	MAIL FROM:<[su]@[sh].BITNET>	F	F	1	# 11
16	RCPT TO:<[us]@[ho].INFNET>	F	T	2	# 11
17	DATA	F	F	3	# 11
	.....				#
22	CP SPOOL PUNCH TO [rs] CLASS M	T	F	1	# 6
23	CP TAG DEV PUNCH CERNVAX MAILER	T	F	2	# 6
24	PUNCH [fn] (NOH	T	F	3	# 6
25	From: <[su]@[sh].BITNET>	F	F	1	# 6
26	To: <[us]@[ho].[dn]>	F	T	2	# 6
27	Subj: [sj]	F	F	3	# 6

Fig. 5: A summary of the Mailrules table

```
      SUBROUTINE GETSND (SOURCE)
C
C
C.  Create User Table with all SENDER Info
C
C
C
      IMPLICIT NONE
      INTEGER INDNAM,ILGNAM,INOBCO
      CHARACTER LOGNAM*16,NODNAM*32
      LOGICAL OK
      INCLUDE 'WIN(Flags)'
      INCLUDE 'WIN(Login)'
      INCLUDE 'WIN(Member)'
      INCLUDE 'WIN(Computer)'
.....

C
C     Get USERNAME and NodeName
C
      CALL SYSLOG(LOGNAM,NODNAM)
      INDNAM = GETIND(Node,'NodeName')
      Node_NodeName = NODNAM
      CALL SELTAB (Node,INDNAM,C1,C2)
.....
C
C      Insert in Path  Nodes reacheble from SENDER
C
      INOBCO = GETIND(Node,'Computer')
      Node_Computer = Computer_ID
      CALL SELTAB(Node,INOBCO,C1,C2)
      DO 100 INODE = C1,C2
      CALL FETTAB(Node,INOBCO,INODE)
      Network_ID  =  Node_Network
      CALL  GETTAB(Network)
      Path_ID = NEXT
      Path_Computer  =  Computer_ID
      Path_Node = Node_ID
      Path_NoNa = Node_NodeName
      Path_Network  =  Network_ID
      Path_NeNa = Network_NetName
      Path_User  =  User_ID
      CALL  INSTAB(Path)
100      CONTINUE
      RETURN
      END
```

Fig.6a: A part of a VAX subroutine

```
      SUBROUTINE GETSND (SOURCE)
C
C
C.        Create User Table with all SENDER Info
C
C
C
      IMPLICIT LOGICAL(A-Z)
      INTEGER INDNAM,ILGNAM,INOBCO
      CHARACTER LOGNAM*16,NODNAM*32
      LOGICAL OK
      INCLUDE (FLAGS  )
      INCLUDE (LOGIN  )
      INCLUDE (MEMBER )
      INCLUDE (COMPUTER)

C
C   Get USERNAME and NodeName
C
      CALL SYSLOG(LOGNAM,NODNAM)
      INDNAM = GETIND(NODE$$,'NodeName')
      NODENN = NODNAM
      CALL SELTAB (NODE$$,INDNAM,C1,C2)
.....
C
C    Insert in Path. Nodes reacheble from SENDER
C
      INOBCO = GETIND(NODE$$,'Computer')
      NODECO = COMPID
      CALL SELTAB(NODE$$,INOBCO,C1,C2)
      DO 100 INODE = C1,C2
      CALL FETTAB(NODE$$,INOBCO,INODE)
      NETWID = NODENE
      CALL GETTAB(NETW$$)
      PATHID = NEXT
      PATHCO = COMPID
      PATHNO = NODEID
      PATHNN = NODENN
      PATHNE = NETWID
      PATHNM = NETWNN
      PATHUS = USERID
      CALL  INSTAB(PATH$$)
100   CONTINUE
      RETURN
      END
```

Fig.6b: A part of a IBM subroutine

```
C
C    Scan RCPT(s)
C
*IF DEF,VAX
10    STATUS = CLI$GET_VALUE('P1',Flags_RCPT)
      IF (STATUS.EQ.%LOC(CLI$_COMMA)) THEN
          CALL FLLRCP(Flags_RCPT)
          GOTO 10
      ELSE IF (STATUS.EQ.SS$_NORMAL) THEN
          CALL FLLRCP(Flags_RCPT)
      ELSE
          CALL FOR$EXIT(STATUS)
      ENDIF
*EI
*IF DEF,IBM
      CALL VMREXX ('F','RECIPIENT.0',CRCPT,STATUS)
      CALL RDINT (CRCPT,NRCPT)
      DO 10 I=1,NRCPT
      REXVAR(1:10)= 'RECIPIENT.'
      CALL WTINT (I,REXVAR(11:))
      ILEN=INDEX(REXVAR,' ')-1
      CALL VMREXX ('F',REXVAR(1:ILEN),Flags_RCPT,STATUS)
      CALL FLLRCP(Flags_RCPT)
10    CONTINUE
*EI
C
C     Check User Table. If empty generate an exit.
C
*IF DEF,VAX
      IF (Flags_ValidList) OPEN (UNIT=15,FILE=Files_ValidDist,
   +  CARRIAGECONTROL='LIST',STATUS='NEW')
*EI
*IF DEF,IBM
      IF (Flags_ValidList) THEN
          CALL VMCMS ('FILEDEF 15 DISK '//Files_ValidDist)
          OPEN (UNIT=15,STATUS='OLD')
      ENDIF
*EI
      CALL CHKRCP
      IF (Flags_ValidList) CLOSE (UNIT=15)
```

Fig.7: An example of conditional code

```
AMSEND

Aleph_Mail_SEND, sends mail to any ALEPH member known to NADIR (New Aleph
DIRectory). The recipient is specified by surname and optional first name.
The destination computer, user name, the best network(s ) and mail server
are worked out by AMSEND, reading an ADAMO DAF file extracted from NADIR.
A hint about how to reply is added to the message.

AMSEND  recipient[,recipient]          : mail recipient(s)
        /SUBJECT=subject               : string specifying the mail subject
        /FILE=fileid                   : a file containing the message to be sent
        /LOG                           : logs the session on the terminal.
        /NOSEND                        : no mail is sent out, useful for tests
        /VALIDATE_LIST=fileid          : a new distribution list is created with
                                         all people registered on the Aleph machines



Additional information available:

Recipient     Command_Qualifiers
/SUBJECT     /FILE     /LOG     /NOSEND     /VALIDATE_LIST
```

Fig. 8a: The main level for AMSEND help file

```
AMSEND

    Recipient

The recipient can be specified in one of the following ways:
    surname                        : A member surname (or the beginning)
    "surname  firstname"           : A member surname and firstname
    "surname@institute"            : Mail will be sent at member's institute
    "surname@node"                 : Mail will be sent at the specified node
    nickname                       : A member nickname (see below)
    "@distribution[.DLS]"          : A distribution list ( " )
    *                              : All Aleph members (use with care!)
    "!user@node.network"           : For someone not in Aleph. The network
                                   : must be known by Amsend

If you specify more then one recipient they must be separated by commas.
When more than one person matches, you will be prompted to choose.
More than one distribution list can be specified as well, each of them
between double quotes and separated by commas. Any mixture of the
syntaxes for the recipient can be used on the command line.


Additional information available:

Nicknames   Distribution_List
```

Fig. 8b: The help level for recipient format

```
$ amsend/log/file=test.msg/subj="Test"  palazzi,finch,  "fantechi@pisa"
Recipient PALAZZI matched as Paolo PALAZZI
Recipient FINCH matched as Alex FINCH
Recipient FANTECHI matched as Riccardo FANTECHI

Member Paolo PALAZZI addressed on 1 Login(s)
Member Alex FINCH addressed on 1 Login(s)
Member Riccardo FANTECHI addressed on 2 Login(s)
```

RECIPIENT	LOGIN	NODE	NETWORK
Paolo PALAZZI	PALAZZI	VXCRNA	DECNET
	PALAZZI	VXCRNA	CERN
	PALAZZI	CERN.VXCRNA	JANET
	PALAZZI	psi%022846811405	PSI
Alex FINCH	AJF	UK.AC.LANCS.PH.V1	JANET
Riccardo FANTECHI	RF	IPIINFN	BITNET
	FANTECHI	IPIVAXIN	BITNET
	FANTECHI	VAXPI	DECNET
	FANTECHI	VAXPI	INFNET
RECIPIENT	LOGIN	NODE	NETWORK
Riccardo FANTECHI	FANTECHI	VXCRNA	DECNET
	FANTECHI	VXCRNA	CERN
	FANTECHI	CERN.VXCRNA	JANET
	FANTECHI	psi%022846811405	PSI

Fig. 9: An example of an AMSEND session

```
From: FANTECHI@CERN.VXCRNA 31-AUG-1987 15:02
To: FINCH
Subj:


Date:  31-AUG-1987  15:01:14
From: FANTECHI@CERN.VXCRNA
To: FINCH@UK.AC.LANCS.PH.V1


Hello, this is a message sent using AMSEND.

Reply Hint - Use Amsend or:

POST /TO=FANTECHI/AT=CERN.VXCRNA   [filename]
```

Fig. 10: A message sent by AMSEND