program code for transmitting, to the server, a query as to whether the second process is connected to the computer network (Etherphone, Zellweger pgs. 1-3 and Swinehart 1, pg. 4);

program code for receiving a network protocol address of the second process from the server, when the second process is connected to the computer network (Etherphone, Swinehart 1, pg. 4); *and*

program code, responsive to the network protocol address of the second process, for establishing a point-to-point communication link between the first process and the second process over the computer network (Etherphone, Swinehart 1, pg. 4 and Zellweger, pg. 2).

The Etherphone, Vin and RFC 1531 references were not previously discussed by the examiner nor applied to claims 1-7 and 10-44 in the prior examination of the patent as discussed above.

It is agreed that the consideration of Etherphone, Vin and RFC 1531 raises an SNQ as to claims 1-7 and 10-44 of the Hutton patent as pointed out above. There is a substantial likelihood that a reasonable examiner would consider these teachings important in deciding whether or not these claims are patentable.

Accordingly, Etherphone, Vin and RFC 1531 raise a substantial new question of claims 1-7 and 10-44, which question has not been decided in a previous examination of the Hutton patent nor was there a final holding of invalidity by the Federal Courts regarding the Hutton patent.

<u>VocalChat and RFC 1531</u>

7) The VocalChat reference disclose an address server with an address database for storing network protocol addresses usable by network nodes to establish point-to-point communications.

RFC 1531 discloses how TCP/IP addresses are assigned dynamically by a DHCP server.

The Request shows that the VocalChat references and RFC 1531 in combination teach *program code for transmitting to the server a network protocol address received by the first process following connection to the computer network* (VocalChat ReadMe, Page 2, Help File page 2, RFC 1531, section 2.2);

program code for transmitting, to the server, a query as to whether the second process is connected to the computer network (VocalChat Help File, pages 2, 22 and 26);

program code for receiving a network protocol address of the second process from the server, when the second process is connected to the computer network (VocalChat Help File, page 2); *and*

program code, responsive to the network protocol address of the second process, for establishing a point-to-point communication link between the first process and the second process over the computer network (VocalChat Help File, page 17, User Guide, page 2).

The VocalChat and RFC 1531 references were not previously discussed by the examiner nor applied to claims 1-7 and 10-44 in the prior examination of the patent as discussed above.

It is agreed that the consideration of VocalChat and RFC 1531 raises an SNQ as to claims 1-7 and 10-44 of the Hutton patent as pointed out above. There is a substantial likelihood that a reasonable examiner would consider these teachings important in deciding whether or not these claims are patentable.

Accordingly, VocalChat and RFC 1531 raise a substantial new question of claims 1-7 and 10-44, which question has not been decided in a previous examination of the Hutton patent nor was there a final holding of invalidity by the Federal Courts regarding the Hutton patent.

Pinard

8) Pinard is cited by Requester as supporting the primary references in alternative obviousness rejections, as well as proposed teachings for dependent claims in Hutton. Examiner agrees that many of the claims in Hutton, particularly independent claims 10 and 21, as mapped out in the Request, appear to be read on by the combination of References listed above with Pinard.

The Pinard reference was not previously discussed by the examiner nor applied to claims 1-7 and 10-44 in the prior examination of the patent as discussed above.

It is agreed that the consideration of Pinard in combination with the references above raises an SNQ as to claims 1-7 and 10-44 of the Hutton patent as pointed out above. There is a substantial likelihood that a reasonable examiner would consider these teachings important in deciding whether or not these claims are patentable.

Accordingly, Pinard raises a substantial new question of claims 1-7 and 10-44, which question has not been decided in a previous examination of the Hutton patent nor was there a final holding of invalidity by the Federal Courts regarding the Hutton patent.

Scope of Reexamination

9) Claims 1-7 and 10-44 will be reexamined as requested in the Request.

Conclusion

Extensions of time under 37 CFR 1.136(a) will not be permitted in these proceedings because the provisions of 37 CFR 1.136 apply only to "an applicant" and not to parties in a reexamination proceeding. Additionally, 35 U.S.C. 305 requires that reexamination proceedings "will be conducted with special dispatch" (37 CFR 1.550(a)). Extension of time in *ex parte* reexamination proceedings are provided for in 37 CFR 1.550(c).

The patent owner is reminded of the continuing responsibility under 37 CFR 1.565(a) to apprise the Office of any litigation activity, or other prior or concurrent proceeding, involving Patent No. 5,337,753 throughout the course of this reexamination proceeding. The third party requester is also reminded of the ability to similarly apprise the Office of any such activity or proceeding throughout the course of this reexamination proceeding. See MPEP §§ 2207, 2282 and 2286.

All correspondence relating to this ex parte reexamination proceeding should be directed as follows:

By U.S. Postal Service Mail to:

Mail Stop Ex Parte Reexam
ATTN: Central Reexamination Unit
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

By FAX to:

(571) 273-9900
Central Reexamination Unit

By hand to:

Customer Service Window
Randolph Building
401 Dulany St.
Alexandria, VA 22314

By EFS-Web:

Registered users of EFS-Web may alternatively submit such correspondence via the electronic filing system EFS-Web, at

https://sportal.uspto.gov/authenticate/authenticateuserlocalepf.html

EFS-Web offers the benefit of quick submission to the particular area of the Office that needs to act on the correspondence. Also, EFS-Web submissions are "soft scanned" (i.e., electronically uploaded) directly into the official file for the reexamination proceeding, which offers parties the opportunity to review the content of their submissions after the "soft scanning" process is complete.

Any inquiry concerning this communication or earlier communications from the Reexamination Legal Advisor or Examiner, or as to the status of this proceeding, should be directed to the Central Reexamination Unit at telephone number (571) 272-7705.

/Alexander J Kosowski/

Primary Examiner, Art Unit 3992

JA
ESK

Substitute for Form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(use as many sheets as necessary)

Complete if Known	
Application Number	Reexamination of 6,108,704
Filing Date	Herewith
First Named Inventor:	
Art Unit	
Examiner Name	
Attorney Docket Number	003801.G184

Sheet 1 of 1

U.S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number-Kind Code[2](if known)				
ABK	Exhibit F	US-	5,533,110	07-02-1996	Pinard, Deborah L., et al.	

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published	T[2]
ABK	Exhibit B	The Open Group, Technical Standard, Protocols for X/Open PC Interworking: SMB, Version 2, 1992, pages ii-xvi and pages 1-516.	
ABK	Exhibit C	ZELLWEGER, POLLE T., et al., Etherphone: Collected Papers 1987-1988, Xerox Corporation, May 1989.	
ABK	Exhibit D	VIN, HERRICK M., et al, Multimedia Conferencing in the Etherphone Environment, October 1991, pages 69-79.	
ABK	Exhibit E	DROMS, R., Dynamic Host Configuration Protocol, RFC 1531, Bucknell University, October 1993, pages 1-39.	
ABK	Exhibit G	VocalChat User's Guide Version 2.0, Vocaltec, 1994, pages 1-77.	
ABK	Exhibit H	README, VocalChat Version 2.02 & VocalChat WAN Version 2.02, Vocaltec, June 1994, pages 1-3.	
ABK	Exhibit I	VocalChat 1.01 Network Information, Vocaltec, 1994, pages 1-10.	
ABK	Exhibit J	VocalChat Information, Vocaltec, 1994, pages 1-31.	
ABK	Exhibit K	VocalChat Troubleshooting, Vocaltec, 1994, pages 1-101.	

Examiner Signature		Date Considered	3/9/09

*Examiner: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

[1]Applicant's unique citation designation number (optional). [2]Applicant is to place a check mark here if English Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SENT FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.**

Based on Form PTO/SB/08A (08-03) as modified by BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP on 09/10/03.

Search Notes	**Application/Control No.** 90/010,416	**Applicant(s)/Patent under Reexamination** 6108704
	Examiner ALEXANDER J. KOSOWSKI	**Art Unit** 3992

SEARCHED

Class	Subclass	Date	Examiner

INTERFERENCE SEARCHED

Class	Subclass	Date	Examiner

SEARCH NOTES (INCLUDING SEARCH STRATEGY)

	DATE	EXMR
Reviewed proposed submitted prior art	3/9/2009	AJK

Reexamination	**Application/Control No.** 90/010,416	**Applicant(s)/Patent Under Reexamination** 6108704
	Certificate Date	**Certificate Number**

Requester Correspondence Address: □ **Patent Owner** ☒ **Third Party**

BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
1279 OAKMEAD PARKWAY
SUNNYVALE, CA 94085-4040

LITIGATION REVIEW ☒	**AJK** (examiner initials)	**3/9/09** (date)
Case Name		Director Initials
OPEN: 2:06cv2469 Net2phone, Inc. v. Ebay, Inc et al		Eric Keasel for G-M

COPENDING OFFICE PROCEEDINGS	
TYPE OF PROCEEDING	**NUMBER**
1. no copending office proceedings	
2.	
3.	
4.	

U.S. Patent and Trademark Office DOC. CODE **RXFILJKT**

Re-Exam

STATEMENT UNDER 37 CFR 3.73(B)

Applicant / Patent Owner: Net2Phone, Inc. Docket No. 2655-0188

Control No. 90/010,416 Filed / Issued Date: 08/22/2000

Entitled: POINT-TO-POINT INTERNET PROTOCOL

Assignee: Net2Phone, Inc. A corporation
(Name of assignee) (Type of Assignee: corporation, partnership, university, government agency, etc.)

States that it is:

1. ☒ the assignee of the entire right, title, and interest; or

2. □ an assignee of less than the entire right, title and interest.
(The extent (by percentage) of its ownership interest is %)

in the patent application / patent identified above by virtue of either:

A. □ An assignment from the inventor(s) of the patent application / patent identified above. The assignment was recorded in the United States Patent and Trademark Office at Reel , Frame , or for which a copy thereof is attached.

OR

B. ☒ A chain of title from the inventor(s), of the patent application / patent identified above, to the current assignee shown below:

1. From: HUTTON, Glen W. To: Internet Telephone Company
The document was recorded in the United States Patent and Trademark Office at Reel 007981 Frame 0020, or for which a copy thereof is attached.

2. From: HUTTON, Glenn W. To: Internet Telephone Company
The document was recorded in the United States Patent and Trademark Office at Reel 008295 Frame 0167, or for which a copy thereof is attached.

3. From: Internet Telephone Company To: Netspeak Corporation
The document was recorded in the United States Patent and Trademark Office at Reel 007981 Frame 0053, or for which a copy thereof is attached.

☒ Additional documents in the chain of title are listed on a supplemental sheet.

☒ Copies of assignments or other documents in the chain of title are attached.

As required by 37 CFR 3.73(b)(1)(i), the documentary evidence of the chain of title from the original owner to the assignee was, or concurrently is being, submitted for recordation pursuant to 37 CFR 3.11.

[Note: A separate copy (*i.e.*, a true copy of the original assignment document(s)) must be submitted to Assignment Division in accordance with 37 CFR Part 3, if the assignment is to be recorded in the records of the USPTO. See MPEP 302.08]

The undersigned (whose title is supplied below) is authorized to act on behalf of the assignee.

Signature 3/13/09
Date

Michael R. Casey, Ph. D. 703-894-6400
Printed or Typed Name Telephone Number

Title: Attorney, Registration No. 40,294

STATEMENT UNDER 37 CFR 3.73(B)
Continued

4.	From: STRICKLAND, Craig B. To: Netspeak Corporation The document was recorded in the United States Patent and Trademark Office at Reel 009792 Frame 0568, or for which a copy thereof is attached.
5.	From: MATTAWAY, Shane D. To: Netspeak Corporation The document was recorded in the United States Patent and Trademark Office at Reel 010012 Frame 0953, or for which a copy thereof is attached.
6.	From: Netspeak Corporation To: VOIP Technology Holdings, LLC The document was recorded in the United States Patent and Trademark Office at Reel 016522 Frame 0205, or for which a copy thereof is attached.
7.	From: VOIP Technology Holdings, LLC To: Net2Phone, Inc. The document was recorded in the United States Patent and Trademark Office at Reel 016945 Frame 0858, or for which a copy thereof is attached.
8.	From: Netspeak Corporation To: Net2Phone, Inc. The document was recorded in the United States Patent and Trademark Office at Reel 016945 Frame 0890, or for which a copy thereof is attached.
9.	From: VOIP Technology Holdings, LLC To: Net2Phone, Inc. The document was recorded in the United States Patent and Trademark Office at Reel 017105 Frame 0240, or for which a copy thereof is attached.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

POWER OF ATTORNEY,
CORRESPONDENCE ADDRESS
AND REVOCATION OF PRIOR POWERS

Hon. Commissioner of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Revocation: I hereby revoke all previous powers of attorney given in the application identified in the attached statement under 37 CFR 3.73(b).

Power of Attorney: I hereby appoint the practitioners associated with customer number **42624**, individually and collectively, as attorney(s) or agent(s) to represent the undersigned before the United States Patent and Trademark Office (USPTO) in connection with any and all patent applications assigned only to the undersigned according to the USPTO assignment records or assignment documents attached to this form in accordance with 37 CFR 3.73(b).

I authorize Davidson Berquist Jackson & Gowdey, LLP to delete names/numbers of persons no longer with the Firm and to act and rely on instructions from and communicate directly with the entity who first sent this case to them and by whom I hereby declare that I have consented after full disclosure to be represented unless/until I instruct Davidson Berquist Jackson & Gowdey, LLP in writing to the contrary.

Correspondence Address: Please recognize or change the correspondence address for the application identified in the attached statement under 37 CFR 3.73(b) to the address associated with Customer Number **42624**.

Assignee Name and Address:

Net2Phone, Inc.
520 Broad Street, 8th Floor
Newark, New Jersey 07102

A copy of this form, together with a statement under 37 CFR 3.73(b) (Form PTO/SB/96 or equivalent) is required to be filed in each application in which this form is used. The statement under 37 CFR 3.73(b) may be completed by one of the practitioners appointed in this form if the appointed practitioner is authorized to act on behalf of the assignee, and must identify the application in which this Power of Attorney is to be filed.

SIGNATURE of Assignee of Record

The individual whose signature and title is supplied below is authorized to act on behalf of the assignee

Signature		Date	3/12/09
Name	JAMES RAANAN	Telephone	973 438 3253
Title	EVP & DIRECTOR		

CERTIFICATE OF SERVICE

The undersigned hereby certifies that a copy of this Power of Attorney and Statement Under 37 CFR 3.73 (B) is being served via First Class Mail on 03/13/09, upon the following:

Edwin H. Taylor
Blakely, Sokoloff, Taylor & Zafman, LLP
1279 Oakmead Parkway
Sunnyvale, California 94085

Daniel L. Jackson

Attorney's Docket No.: R001E PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Ex Parte Reexamination of:

U.S. Patent No. 6,108,704

Issued: August 22, 2000

Application No. 90/010,416

Filed: February 17, 2009

For: Point-to-Point Internet Protocol

Requester: Skype, Inc.

Examiner: Kosowski, Alexander J

Art Unit: 3992

Confirmation No. 1061

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Appraisal of Litigation Proceedings Pursuant to MPEP §§ 2207, 2282, and 2286

Dear Sir:

Pursuant to MPEP §§ 2207, 2282, and 2286, the Requestor hereby submits copies of a Court order setting forth a hearing date of July 1, 2009 for oral arguments on the Motion to Stay in the pending litigation (Case 2:06-cv-02469-KSH-PS NET2PHONE, INC. v. EBAY, INC. et al).

Respectfully submitted,

/Thomas C. Webster/
Thomas C. Webster
Registration No. 46,154

Dated: 6-8-09

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
1279 Oakmead Parkway
Sunnyvale, California 94085-4040
Telephone: 408/720-8300
Facsimile: 408/720-8383
Attorney Docket No.: R001E

From: njdefiling@njd.uscourts.gov
Sent: Tuesday, April 28, 2009 2:46 PM
To: ecfhelp@njd.uscourts.gov
Subject: Activity in Case 2:06-cv-02469-KSH-PS NET2PHONE, INC. v. EBAY, INC. et al Order

This is an automatic e-mail message generated by the CM/ECF system. Please DO NOT RESPOND to this e-mail because the mail box is unattended.
*****NOTE TO PUBLIC ACCESS USERS*** Judicial Conference of the United States policy permits attorneys of record and parties in a case (including pro se litigants) to receive one free electronic copy of all documents filed electronically, if receipt is required by law or directed by the filer. PACER access fees apply to all other users. To avoid later charges, download a copy of each document during this first viewing. However, if the referenced document is a transcript, the free copy and 30 page limit do not apply.**

U.S. District Court

District of New Jersey [LIVE]

Notice of Electronic Filing

The following transaction was entered on 4/28/2009 at 5:45 PM EDT and filed on 4/28/2009
Case Name: NET2PHONE, INC. v. EBAY, INC. et al
Case Number: 2:06-cv-2469
Filer:
Document Number: No document attached

Docket Text:
TEXT ORDER: Counsel are advised that the previously scheduled 5/13/09 oral argument on the pending motion to stay is rescheduled to 7/1/09 at 10:00 a.m. entered by Judge Katharine S. Hayden on 4/28/09. (rg,)

2:06-cv-2469 Notice has been electronically mailed to:

ALAN J. HEINRICH	AHeinrich@Irell.com
ALAN M FISCH	AFisch@kayescholer.com
ALLEN I. RUBENSTEIN	arubenstein@grr.com
ANDREI IANCU	aiancu@irell.com
BENJAMIN WANG	bwang@irell.com
COKE MORGAN STEWART	CStewart@kayescholer.com
GEORGE C. JONES	gjones@GrahamCurtin.com, jcostine@GrahamCurtin.com
GILLIAN T. DIFILIPPO	gdifilippo@kayescholer.com

JOSEPH M. DRAYTON jdrayton@kayescholer.com

JOSEPH P. LASALA jlasala@mdmc-law.com, sgrisez@mdmc-law.com

KATHLEEN M. FENNELLY kfennelly@grahamcurtin.com, llett@grahamcurtin.com

KEVIN JAKEL kjakel@kayescholer.com

MARIA A. SAVIO msavio@grr.com

MARKO KUO mkuo@irell.com

MORGAN CHU mchu@irell.com

PERRY GOLDBERG pgoldberg@irell.com

STEVEN STERN sstern@grr.com

STEVEN STERN sstern@grr.com

THOMAS R. CURTIN tcurtin@grahamcurtin.com, llett@grahamcurtin.com

VANDANA KOELSCH vkoelsch@kayescholer.com

VICTOR KUBLISH kubli@graysonlaw.net

WILLIAM F. O'CONNOR , JR woconnor@mdmc-law.com, wfo73@yahoo.com

2:06-cv-2469 Notice will not be electronically mailed to::

CERTIFICATE OF SERVICE FOR CONTROL NO: 90/010,416

The undersigned certifies that copies of the following:

(1) Appraisal of Litigation Proceedings; and

(2) Court Order setting forth a hearing date.

were served on

Michael R. Casey, Ph.D.
Davidson Berquist Jackson & Gowdey, LLP
4300 Wilson Blvd. – 7th Floor
Arlington, VA 22203

/Thomas C. Webster/
Thomas C. Webster
Reg. No. 46,154

Dated: /06-08-2009/

the attorney of record for the assignee of USP 6,108,704in accordance with 37 CFR § 1.915(b)(6), on the 8 day of June, 2009.

Electronic Acknowledgement Receipt

EFS ID:	5477355
Application Number:	90010416
International Application Number:	
Confirmation Number:	1061
Title of Invention:	Point-to-Point Internet Protocol
First Named Inventor/Applicant Name:	6108704
Customer Number:	42624
Filer:	Thomas Webster/Janece Shannon
Filer Authorized By:	Thomas Webster
Attorney Docket Number:	2655-0188
Receipt Date:	08-JUN-2009
Filing Date:	17-FEB-2009
Time Stamp:	21:17:29
Application Type:	Reexam (Third Party)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Reexam Miscellaneous Incoming Letter	R001E_Apprisal_6_8_09.pdf	26572 cd7550ab9f67a77bf036c31192fd0973484d20d8	no	1

Warnings:

Information:

2	Reexam Miscellaneous Incoming Letter	4_8_09_Order_6_8_09.pdf	39418 d5de53e0fac111d817dedc057816591a1adc1090	no	2
Warnings:					
Information:					
3	Reexam Certificate of Service	3801G184_CertOfService.pdf	15663 64acc0f8933bfd2a1000e0f05a12a0dff4605cf5	no	1
Warnings:					
Information:					
			Total Files Size (in bytes):	81653	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : **H04J 3/14, 3/24, H04L 12/56**	**A1**	**(11) International Publication Number:** **WO 92/19054** **(43) International Publication Date:** 29 October 1992 (29.10.92)

(21) International Application Number: PCT/US92/02995

(22) International Filing Date: 10 April 1992 (10.04.92)

(30) Priority data:
684,695 12 April 1991 (12.04.91) US

(71) Applicant: CONCORD COMMUNICATIONS, INC. [US/US]; 753 Forest Street, Marlboro, MA 01752 (US).

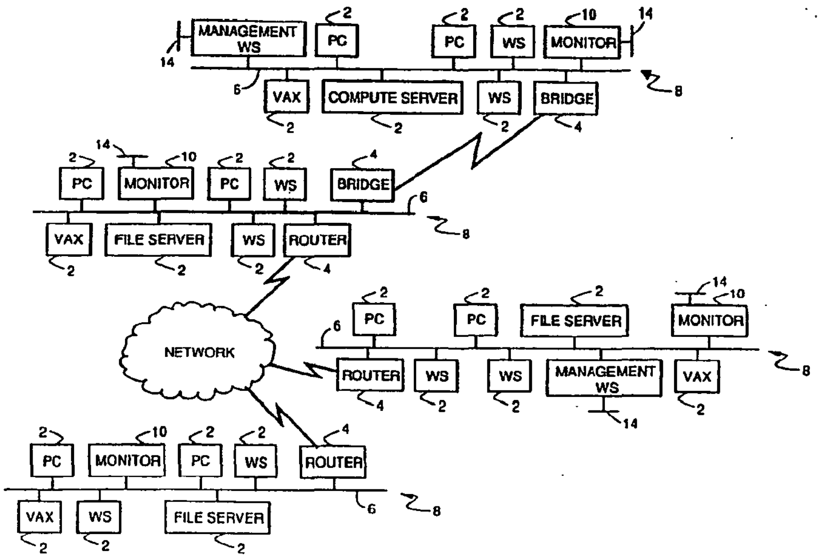
(72) Inventors: FERDINAND, Engel ; 21 Joseph Road, Northborough, MA 01532 (US). JONES, Kendall, S. ; 90 Boulder Road, Newton Center, MA 02159 (US). ROBERTSON, Kary ; 398 North Road, Bedford, MA 01739 (US). THOMPSON, David, M. ; 5127 243rd Road, Redmond, WA 98053 (US). WHITE, Gerard ; 133 Massapoag Road, Tyngsborough, MA 01879 (US).

(74) Agent: PRAHL, Eric, L.; Fish & Richardson, 225 Franklin Street, Boston, MA 02110-2804 (US).

(81) Designated States: AT (European patent), BE (European patent), CA, CH (European patent), DE (European patent), DK (European patent), ES (European patent), FR (European patent), GB (European patent), GR (European patent), IT (European patent), JP, LU (European patent), MC (European patent), NL (European patent), SE (European patent).

Published
With international search report.

(54) Title: NETWORK MONITORING



(57) Abstract

Monitoring is done of communications which occur in a network of nodes (2), each communication being effected by a transmission of one or more packets among two or more communicating nodes (2), each communication complying with a predefined communication protocol selected from among protocols available in the network. The contents of packets are detected passively and in real time, communication information (130, 152, 178) associated with multiple protocols is derived from the packet contents.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	ES	Spain	MG	Madagascar
AU	Australia	FI	Finland	ML	Mali
BB	Barbados	FR	France	MN	Mongolia
BE	Belgium	GA	Gabon	MR	Mauritania
BF	Burkina Faso	GB	United Kingdom	MW	Malawi
BG	Bulgaria	GN	Guinea	NL	Netherlands
BJ	Benin	GR	Greece	NO	Norway
BR	Brazil	HU	Hungary	PL	Poland
CA	Canada	IT	Italy	RO	Romania
CF	Central African Republic	JP	Japan	RU	Russian Federation
CG	Congo	KP	Democratic People's Republic of Korea	SD	Sudan
CH	Switzerland			SE	Sweden
CI	Côte d'Ivoire	KR	Republic of Korea	SN	Senegal
CM	Cameroon	LI	Liechtenstein	SU	Soviet Union
CS	Czechoslovakia	LK	Sri Lanka	TD	Chad
DE	Germany	LU	Luxembourg	TG	Togo
DK	Denmark	MC	Monaco	US	United States of America

NETWORK MONITORING

Background of the Invention

The invention relates to monitoring and managing communication networks for computers.

Todays computer networks are large complex systems with many components from a large variety of vendors. These networks often span large geographic areas ranging from a campus-like setting to world wide networks. While the network itself can be used by many different types of organizations, the purpose of these networks is to move information between computers. Typical applications are electronic mail, transaction processing, remote database, query, and simple file transfer. Usually, the organization that has installed and is running the network needs the network to be running properly in order to operate its business. Since these networks are complex systems, there are various controls provided by the different equipment to control and manage the network. Network management is the task of planning, engineering, securing and operating a network.

To manage the network properly, the Network Manager has some obvious needs. First, the Network Manager must trouble shoot problems. As the errors develop in a running network, the Network Manager must have some tools that notify him of the errors and allow him to diagnose and repair these errors. Second, the Network Manager needs to configure the network in such a manner that the network loading characteristics provide the best service possible for the network users. To do this the Network Manager must have tools that allow him visibility into access patterns, bottlenecks and general loading. With such data, the Network Manager can reconfigure the network components for better service.

There are many different components that need to be managed in the network. These elements can be, but are not limited to: routers, bridges, PC's, workstations, minicomputers, supercomputers, printers, file servers, switches and pbx's. Each component provides a protocol for reading and writing the management variables in the machine. These variables are usually defined by the component vendor and are usually referred to as a Management Information Base (MIB). There are some standard MIB's, such as the IETF (Internet Engineering Task Force) MIB I and MIB II standard definitions. Through the reading and writing of MIB variables, software in other computers can manage or control the component. The software in the component that provides remote access to the MIB variables is usually called an agent. Thus, an individual charged with the responsibility of managing a large network often will use various tools to manipulate the MIB's of various agents on the network.

Unfortunately, the standards for accessing MIBs are not yet uniformly provided nor are the MIB definitions complete enough to manage an entire network. The Network Manager must therefore use several different types of computers to access the agents in the network. This poses a problem, since the errors occurring on the network will tend to show up in different computers and the Network Manager must therefore monitor several different screens to determine if the network is running properly. Even when the Network Manager is able to accomplish this task, the tools available are not sufficient for the Network Manager to function properly.

Furthermore, there are many errors and loadings on the network that are not reported by agents. Flow control problems, retransmissions, on-off segment loading, network capacities and utilizations are some of the types of data that are not provided by the agents.

Simple needs like charging each user for actual network usage are impossible.

Summary of the Invention

In general, in one aspect, the invention features monitoring communications which occur in a network of nodes, each communication being effected by a transmission of one or more packets among two or more communicating nodes, each communication complying with a predefined communication protocol selected from among protocols available in the network. The contents of packets are detected passively and in real time, communication information associated with multiple protocols is derived from the packet contents.

Preferred embodiments of the invention include the following features. The communication information derived from the packet contents is associated with multiple layers of at least one of the protocols.

In general, in another aspect, the invention features monitoring communication dialogs which occur in a network of nodes, each dialog being effected by a transmission of one or more packets among two or more communicating nodes, each dialog complying with a predefined communication protocol selected from among protocols available in the network. Information about the states of dialogs occurring in the network and which comply with different selected protocols available in the network is derived from the packet contents.

Preferred embodiments of the invention include the following features. A current state is maintained for each dialog, and the current state is updated in response to the detected contents of transmitted packets. For each dialog, a history of events is maintained based on information derived from the contents of packets, and the history of events is analyzed to derive information about the dialog. The analysis of the history includes counting events and gathering statistics about events. The history is monitored for dialogs which are inactive, and dialogs which have been inactive for a predetermined period of time are purged. For example, the current state is updated to data state in response to observing the transmission of at least two data related packets from each node. Sequence numbers of data related packets stored in the history of events are analyzed and retransmissions are detected based on the sequence numbers. The the current state is updated based on each new packet associated with the dialog; if an updated current state cannot be determined, information about prior packets associated with the dialog is consulted as an aid in updating the state. The history of events may be searched to identify the initiator of a dialog.

The full set of packets associated with a dialog up to a point in time completely define a true state of the dialog at that point in time, and the step of updating the current state in response to the detected contents of transmitted packets includes generating a current state (e.g., "unknown") which may not conform to the true state. The current state may be updated to the true state based on information about prior packets transmitted in the dialog.

Each communication may involve multiple dialogs corresponding to a specific protocol. Each protocol layer of the communication may be parsed and analyzed to isolate each dialog and statistics may be kept for each dialog. The protocols may include a connectionless-type protocol in which the state of a dialog is implicit in transmitted packets, and the step of deriving information about the states of dialogs includes inferring the states of the dialogs from the packets. Keeping statistics for protocol layers may be temporarily suspended when parsing and statistics gathering is not rapid enough to match the rate of packets to be parsed.

In general, in another aspect, the invention features monitoring the operation of the network with respect to specific items of performance during normal operation, generating a model of the network based on the monitoring, and setting acceptable threshold levels for the specific items of performance based on the model. In preferred embodiments, the operation of the network is monitored with respect to the specific items of performance during periods which may include abnormal operation.

In general, in another aspect, the invention features the combination of a monitor connected to the network medium for passively, and in real time, monitoring transmitted packets and storing information about dialogs associated with the packets, and a workstation for receiving the information about dialogs from the monitor and providing an interface to a user. In preferred embodiments, the workstation includes means for enabling a user to observe events of active dialogs.

In general, in another aspect, the invention features apparatus for monitoring packet communications in a network of nodes in which communications may be in accordance with multiple protocols. The apparatus includes a monitor connected to a communication medium of the network for passively, and in real time, monitoring transmitted packets of different protocols and storing information about communications associated with the packets, the communications being in accordance with different protocols, and a workstation for receiving the information about the communciations from the monitor and providing an interface to a user. The monitor and the workstation include means for relaying the information about multiple protocols with respect to communication in the different protocols from the monitor to the workstation in accordance with a single common network management protocol.

In general, in another aspect, the invention features diagnosing communication problems between two nodes in a network of nodes interconnected by links. The operation of the network is monitored with respect to specific items of performance during normal operation. A model of normal operation of the network is generated based on the monitoring. Acceptable threshold levels are set for the specific items of performance based on the model. The operation of the network is monitored with respect to the specific items of performance during periods which may include abnormal operation. When abnormal operation of the network with respect to communication between the two nodes is detected, the problem is diagnosed by separately analyzing the performance of each of the nodes and each of the links connecting the two nodes to isolate the abnormal operation.

In general, in another aspect, the invention features a method of timing the duration of a transaction of interest occurring in the course of communication between nodes of a network, the beginning of the transaction being defined by the sending of a first packet of a particular kind from one node to the other, and the end of the transaction being defined by the sending of another packet of a particular kind between the nodes. In the method, packets transmitted in the network are monitored passively and in real time. The beginning time of the transaction is determined based on the appearance of the first packet. A determination is made of when the other packet has been transmitted. The timing of the duration of the transaction is ended upon the appearance of the other packet.

In general, in another aspect, the invention features, tracking node address to node name mappings in a network of nodes of the kind in which each node has a possibly nonunique node name and a unique node address within the network and in which node addresses can be assigned and reassigned to node names dynamically using a name binding protocol message incorporated within a packet. In the method, packets transmitted in the network are monitored, and a table linking node names to node addresses is updated based on information contained in the name binding protocol messages in the packets.

One advantage of the invention is that it enables a network manager to passively monitor multi-protocol networks at multiple layers of the communications. In addition, it organizes and presents network performance statistics in terms of dialogs which are occurring at any desired level of the communication. This technique of organizing and displaying network performance statistics provides an effective and useful view of network performance and facilitates a quick diagnosis of network problems.

Other advantages and features will become apparent from the following description of the preferred embodiment and from the claims.

Description of the Preferred Embodiments

Fig. 1 is a block diagram of a network;

Fig. 2 shows the layered structure of a network communication and a protocol tree within that layered environment;

Fig. 3 illustrates the structure of an ethernet/IP/TCP packet;

Fig. 4 illustrates the different layers of a communication between two nodes;

Fig. 5 shows the software modules within the Monitor;

Fig. 6 shows the structure of the Monitor software in terms of tasks and intertask communication mechanisms;

Figs. 7a-c show the STATS data structures which store performance statistics relating to the the data link layer;

Fig. 8 is a event/state table describing the operation of the state machine for a TCP connection;

Fig. 9a is a history data structure that is identified by a pointer found in the appropriate dialog statistics data within STATS;

Fig. 9b is a record from the history table;

Fig. 10 is a flow diagram of the Look_for_Data_State routine;

Fig. 11 is a flow diagram of the Look_for_Initiator routine that is called by the Look_for_Data_State routine;

Fig. 12 is a flow diagram of the Look_for_Retransmission routine which is called by the Look_at_History routine;

Fig. 13 is a diagram of the major steps in processing a frame through the Real Time Parser (RTP);

Fig. 14 is a diagram of the major steps in the processing a statistics threshold event;

Fig. 15 is a diagram of the major steps in the processing of a database update;

Fig. 16 is a diagram of the major steps in the processing of a monitor control request;

Fig. 17 is a logical map of the network as displayed by the Management Workstation;

Fig. 18 is a basic summary tool display screen;

Fig. 19 is a protocol selection menu that may be invoked through the summary tool display screen;

Figs. 20a-g are examples of the statistical variables which are displayed for different protocols;

Fig. 21 is an example of information that is displayed in the dialogs panel of the summary tool display screen;

Fig. 22 is a basic data screen presenting a rate values panel, a count values panel and a protocols seen panel;

Fig. 23 is a traffic matrix screen;

Fig. 24 is a flow diagram of the algorithm for adaptively establishing network thresholds based upon actual network performance;

Fig. 25 is a simple multi-segment network;

Fig. 26 is a flow diagram of the operation of the diagnostic analyzer algorithm;

Fig. 27 is a flow diagram of the source node analyzer algorithm;

Fig. 28 is a flow diagram of the sink node analyzer algorithm;

Fig. 29 is a flow diagram of the link analysis logic;

Fig. 30 is a flow diagram of the DLL problem checking routine;

Fig. 31 is a flow diagram of the IP problem checking routine;

Fig. 32 is a flow diagram of the IP link component problem checking routine;

Fig. 33 is a flow diagram of the DLL link component problem checking routine;

Fig. 34 shows the structure of the event timing database;

Fig. 35 is a flow diagram of the operation of the event timing module (ETM) in the Network Monitor;

Fig. 36 is a network which includes an Appletalk® segment;

Fig. 37 is a Name Table that is maintained by the Address Tracking Module (ATM);

Fig. 38 is a flow diagram of the operation of the ATM; and

Fig. 39 is a flow diagram of the operation of the ATM.

Also attached hereto before the claims are the following appendices:

Appendix I identifies the SNMP MIB subset that is supported by the Monitor and the Management Workstation (2 pages);

Appendix II defines the extension to the standard MIB that are supported by the Monitor and the Management Workstation (25 pages);

Appendix III is a summary of the protocol variables for which the Monitor gathers statistics and a brief description of the variables, where appropriate (17 pages);

Appendix IV is a list of the Summary Tool Values Display Fields with brief descriptions (2 pages); and

Appendix V is a description of the actual screens for the Values Tool (34 pages).

Structure and Operation

The Network:

A typical network, such as the one shown in Fig. 1, includes at least three major components, namely, network nodes 2, network elements 4 and communication lines 6. Network nodes 2 are the individual computers on the network. They are the very reason the network exists. They include but are not limited to workstations (WS), personal computers (PC), file servers (FS), compute servers (CS) and host computers (e.g., a VAX), to name but a few. The term server is often used as though it was different from a node, but it is, in fact, just a node providing special services.

In general, network elements 4 are anything that participate in the service of providing data movement in a network, i.e., providing the basic communications. They include, but are not limited to, LAN's, routers, bridges, gateways, multiplexors, switches and connectors. Bridges serve as connections between different network segments. They keep track of the nodes which are connected to each of the segments to which they are connected. When they see a packet on one segment that is addressed to a node on another of their segments, they grab the packet from the one segment and transfer it to the proper segment. Gateways generally provide connections between different network segments that are operating under different protocols and serve to convert communications from one protocol to the other. Nodes send packets to routers so that they may be directed over the appropriate segments to the intended destination node.

Finally, network or communication lines 6 are the components of the network which connect nodes 2 and elements 4 together so that communicatons between nodes 2 may take place. They can be private lines, satellite lines or Public Carrier lines. They are expensive resources and are usually managed as separate entities. Often networks are organized into segments 8 that are connected by network elements 4. A segment 8 is a section of a LAN connected at a physical level (this may include repeaters). Within a segment, no protocols at layers above the physical layer are needed to enable signals from two stations on the same segment to reach each other (i.e., there are no routers, bridges, gateways...).

<u>The Network Monitor and the Management Workstation</u>:

In the described embodiment, there are two basic elements to the monitoring system which is to be described, namely, a Network Monitor 10 and a Management

Workstation 12. Both elements interact with each other over the local area network (LAN).

Network Monitor 10 (referred to hereinafter simply as Monitor 10) is the data collection module which is attached to the LAN. It is a high performance real time front end processor which collects packets on the network and performs some degree of analysis to search for actual or potential problems and to maintain statistical information for use in later analysis. In general, it performs the following functions. It operates in a promiscuous mode to capture and analyze all packets on the segment and it extracts all items of interest from the frames. It generates alarms to notify the Management Workstation of the occurence of significant events. It receives commands from the Management Workstation, processes them appropriately and returns responses.

Management Workstation 12 is the operator interface. It collects and presents troubleshooting and performance information to the user. It is based on the SunNet Manager (SNM) product and provides a graphical network-map-based interface and sophisticated data presentation and analysis tools. It receives information from Monitor 10, stores it and displays the information in various ways. It also instructs Monitor 10 to perform certain actions. Monitor 10, in turn, sends responses and alarms to Management Workstation 12 over either the primary LAN or a backup serial link 14 using SNMP with the MIB extensions defined later.

These devices can be connected to each other over various types of networks and are not limited to connections over a local area network. As indicated in Fig. 1, there can be multiple Workstations 12 as well as multiple Monitors 10.

Before describing these components in greater detail, background information will first be reviewed regarding communication protocols which specify how communications are conducted over the network and regarding the structure of the packets.

The Protocol Tree:

As shown in Fig. 2, communication over the network is organized as a series of layers or levels, each one built upon the next lower one, and each one specified by one or more protocols (represented by the boxes). Each layer is responsible for handling a different phase of the communication between nodes on the network. The protocols for each layer are defined so that the services offered by any layer are relatively independent of the services offered by the neighbors above and below. Although the identities and number of layers may differ depending on the network (i.e., the protocol set defining communication over the network), in general, most of them share a similar structure and have features in common.

For purposes of the present description, the Open Systems Interconnection (OSI) model will be presented as representative of structured protocol architectures. The OSI model, developed by the International Organization for Standardization, includes seven layers. As indicated in Fig. 2, there is a physical layer, a data link layer (DLL), a network layer, a transport layer, a session layer, a presentation layer and an application layer, in that order. As background for what is to follow, the function of each of these layers will be briefly described.

The physical layer provides the physical medium for the data transmission. It specifies the electrical and mechanical interfaces of the network and deals with bit level detail. The data link layer is responsible for ensuring an error-free physical link between the communicating nodes. It is responsible for creating and recognizing frame boundaries (i.e., the boundaries of the packets of data that are sent over the network.) The network layer determines how packets are routed within the network. The transport layer accepts data from the layer above it (i.e., the session layer), breaks the packets up into smaller units, if required, and passes these to the network layer for transmission over the network. It may insure that the smaller pieces all arrive properly at the other end. The session layer is the user's interface into the network. The user must interface with the session layer in order to negotiate a connection with a process in another machine. The presentation layer provides code conversion and data reformatting for the user's application. Finally, the application layer selects the overall network service for the user's application.

Fig. 2 also shows the protocol tree which is implemented by the described embodiment. A protocol tree shows the protocols that apply to each layer and it identifies by the tree structure which protocols at each layer can run "on top of" the protocols of the next lower layer. Though standard abbreviations are used to identify the protocols, for the convenience of the reader, the meaning of the abbreviations are as follows:

ARP	Address Resolution Protocol
ETHERNET	Ethernet Data Link Control
FTP	File Transfer Protocol
ICMP	Internet Control Message Protocol
IP	Internet Protocol
LLC	802.2 Logical Link Control
MAC	802.3 CSMA/CD Media Access Control
NFS	Network File System
NSP	Name Server Protocol
RARP	Reverse Address Resolution Protocol
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol

TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
UDP	User Datagram Protocol

Two terms are commonly used to describe the protocol tree, namely, a protocol stack and a protocol family (or suite). A protocol stack generally refers to the underlying protocols that are used when sending a message over a network. For example, FTP/TCP/IP/LLC is a protocol stack. A protocol family is a loose association of protocols which tend to be used on the same network (or derive from a common source). Thus, for example, the TCP/IP family includes IP, TCP, UDP, ARP, TELNET and FTP. The Decnet family includes the protocols from Digital Equipment Corporation. And the SNA family includes the protocols from IBM.

The Packet:

The relevant protocol stack defines the structure of each packet that is sent over the network. Fig. 3, which shows an TCP/IP packet, illustrates the typical structure of a packet. In general, each level of the protocol stack takes the data from the next higher level and adds header information to form a protocol data unit (PDU) which it passes to the next lower level. That is, as the data from the application is passed down through the protocol layers in preparation for transmission over the network, each layer adds its own information to the data passed down from above until the complete packet is assembled. Thus, the structure of a packet ressembles that of an onion, with each PDU of a given layer wrapped within the PDU of the adjacent lower level.

At the ethernet level, the PDU includes a destination address (DEST MAC ADDR), a source address (SRC MAC ADDR), a type (TYPE) identifying the protocol which is running on top of this layer, and a DATA field for the PDU from the IP layer.

Like the ethernet packet, the PDU for the IP layer includes an IP header plus a DATA field. The IP header includes a type field (TYPE) for indicating the type of service, a length field (LGTH) for specifying the total length of the PDU, an identification field (ID), a protocol field (PROT) for identifying the protocol which is running on top of the IP layer (in this case, TCP), a source address field (SRC ADDR) for specifying the IP address of the sender, a destination address field (DEST ADDR) for specifying the IP address of the destination node, and a DATA field.

The PDU built by the TCP protocol also consists of a header and the data passed down from the next higher layer. In this case the header includes a source port field (SRC PORT) for specifying the port number of the sender, a destination port field (DEST PORT) for specifying the port number of the destination, a sequence number field (SEQ NO.) for specifying the sequence number of the data that is being sent in this packet, and an acknowledgment number field (ACK NO.) for specifying the number of the acknowledgment being returned. It also includes bits which identify the packet type, namely, an acknowledgment bit (ACK), a reset connection bit (RST), a synchronize bit (SYN), and a no more data from sender bit (FIN). There is also a window size field (WINDOW) for specifying the size of the window being used.

The Concept of a Dialog:

The concept of a dialog is used throughout the following description. As will become apparent, it is a concept which provides a useful way of conceptualizing, organizing and displaying information about the performance of a network - for any protocol and for any layer of the multi-level protocol stack.

As noted above, the basic unit of information in communication is a packet. A packet conveys meaning between the sender and the receiver and is part of a larger framework of packet exchanges. The larger exchange is called a dialog within the context of this document. That is, a dialog is a communication between a sender and a receiver, which is composed of one or more packets being transmitted between the two. There can be multiple senders and receivers which can change roles. In fact, most dialogs involve exchanges in both directions.

Stated another way, a dialog is the exchange of messages and the associated meaning and state that is inherent in any particular exchange at any layer. It refers to the exchange between the peer entities (hardware or software) in any communication. In those situations where there is a layering of protocols, any particular message exchange could be viewed as belonging to multiple dialogs. For example, in Fig. 4 Nodes A and B are exchanging packets and are engaged in multiple dialogs. Layer 1 in Node A has a dialog with Layer 1 in Node B. For this example, one could state that this is the data link layer and the nature of the dialog deals with the message length, number of messages, errors and perhaps the guarantee of the delivery. Simultaneously, Layer n of Node A is having a dialog with Layer n of node B. For the sake of the example, one could state that this is an application layer dialog which deals with virtual terminal connections and response rates. One can also assume that all of the other layers (2 through n-1) are also having simultaneous dialogs.

In some protocols there are explicit primitives that deal with the dialog and they are generally referred to as connections or virtual circuits. However, dialogs exist even in stateless and connectionless protocols. Two more examples will be described to help clarify the concept further, one dealing with a connection oriented protocol and the other dealing with a connectionless protocol.

In a typical connection oriented protocol, Node A sends a connection request (CR) message to Node B. The CR is an explicit request to form a connection. This is the start of a particular dialog, which is no different from the start of the connection. Nodes A and B could have other dialogs active simultaneously with this particular dialog. Each dialog is seen as unique. A connection is a particular type of dialog.

In a typical connectionless protocol, Node A sends Node B a message that is a datagram which has no connection paradigm, in fact, neither do the protocol(s) at higher layers. The application protocol designates this as a request to initiate some action. For example, a file server protocol such as Sun Microsystems' Network File System (NFS) could make a mount request. A dialog comes into existence once the communication between Nodes A and B has begun. It is possible to determine that communication has occurred and to determine the actions being requested. If in fact there exists more than one communication thread between Nodes A and B, then these would represent separate, different dialogs.

<u>Inside the Network Monitor</u>:

Monitor 10 includes a MIPS R3000 general purpose microprocessor (from MIPS Computer Systems, Inc.) running at 25 MHz. It is capable of providing 20 mips processing power. Monitor 10 also includes a 64Kbyte instruction cache and a 64Kbyte data cache, implemented by SRAM.

The major software modules of Monitor 10 are implemented as a mixture of tasks and subroutine libraries as shown in Fig. 5. It is organized this way so as to minimise the context switching overhead incurred during critical processing sequences. There is NO PREEMPTION of any module in the monitor subsystem. Each module is cognizant of the fact that it should return control to the kernel in order to let other tasks run. Since the monitor subsystem is a closed environment, the software is aware of real time constraints.

Among the major modules which make up Monitor 10 is a real time kernel 20, a boot/load module 22, a driver 24, a test module 26, an SNMP Agent 28, a Timer module 30, a real time parser (RTP) 32, a Message Transport Module (MTM) 34, a statistics database (STATS) 36, an Event Manager (EM) 38, an Event Timing Module (ETM) 40 and a control module 42. Each of these will now be described in greater detail.

Real Time Kernel 20 takes care of the general housekeeping activities in Monitor 10. It is responsible for scheduling, handling intertask communications via queues, managing a potentially large number of timers, manipulating linked lists, and handling simple memory management.

Boot/Load Module 22, which is FProm based, enables Monitor 10 to start itself when the power is turned on in the box. It initializes functions such as diagnostics, and environmental initialization and it initiates down loading of the Network Monitor Software including program and configuration files from the Management Workstation. Boot/load module 22 is also responsible for reloading program and/or configuration data following internal error detection or on command from the Management Workstation. To accomplish down loading, boot/load module 22 uses the Trivial File Transfer Protocol (TFTP). The protocol stack used for loading is TFTP/UDP/IP/ethernet over the LAN and TFTP/UDP/IP/SLIP over the serial line.

Device Driver 24 manages the network controller hardware so that Monitor 10 is able to read and write packets from the network and it manages the serial interface. It does so both for the purposes of monitoring traffic (promiscuous mode) and for the purposes of communicating with the Management Workstation and other devices on the network. The communication occurs through the network controller hardware of the physical network (e.g. Ethernet). The drivers for the LAN controller and serial line interface are used by the boot load module and the MTM. They provide access to the chips and isolate higher layers from the hardware specifics.

Test module 26 performs and reports results of physical layer tests (TDR, connectivity,...) under control of the Management Workstation. It provides traffic load information in response to user requests identifying the particular traffic data of interest. The load information is reported either as a percent of available bandwidth or as frame size(s) plus rate.

SNMP Agent 28 translates requests and information into the network management protocol being used to communicate with the Management Workstation, e.g., the Simple Network Management Protocol (SNMP).

Control Module 42 coordinates access to monitor control variables and performs actions necessary when these are altered. Among the monitor control variables which it handles are the following:

set reset monitor - transfer control to reset logic;

set time of day - modify monitor hardware clock and generate response to Management Workstation;

get time of day - read monitor hardware clock and generate response to Workstation;

set trap permit - send trap control ITM to EM and generate response to Workstation;

get trap permit - generate response to Workstation;

Control module 42 also updates parse control records within STATS when invoked by the RTP (to be described) or during overload conditions so that higher layers of parsing are dropped until the overload situation is resolved. When overload is over it restores full parsing.

Timer 30 is invoked periodically to perform general housekeeping functions. It pulses the watchdog timer at appropriate intervals. It also takes care of internal time stamping and kicking off routines like the EM routine which periodically recalculates certain numbers within the statistical database (i.e., STATS).

Real Time Parser (RTP) 32 sees all frames on the network and it determines which protocols are being used and interprets the frames. The RTP includes a protocol parser and a state machine. The protocol parser parses a received frame in the "classical" manner, layer-by-layer, lowest layer first. The parsing is performed such that the statistical objects in STATS (i.e., the network parameters for which performance data is kept) are maintained. Which layers are to have statistics stored for them is determined by a parse control record that is stored in STATS (to be described later). As each layer is parsed, the RTP invokes the appropriate functions in the statistics module (STATS) to update those statistical objects which must be changed.

The state machine within RTP 32 is responsible for tracking state as appropriate to protocols and connections. It is responsible for maintaining and updating the connection oriented statistical elements in

STATS. In order to track connection states and events, the RTP invokes a routine within the state machine. This routine determines the state of a connection based on past observed frames and keeps track of sequence numbers. It is the routine that determines if a connection is in data transfer state and if a retransmission has occurred. The objectives of the state machine are to keep a brief history of events, state transitions, and sequence numbers per connection; to detect data transfer state so that sequence tracking can begin; and to count inconsistencies but still maintain tracking while falling into an appropriate state (e.g. unknown).

RTP 32 also performs overload control by determining the number of frames awaiting processing and invoking control module 42 to update the parse control records so as to reduce the parsing depth when the number becomes too large.

Statistics Module (STATS) 36 is where Monitor 10 keeps information about the statistical objects it is charged with monitoring. A statistical object represents a network parameter for which performance information is gathered. This information is contained in an extended MIB (Management Information Base), which is updated by RTP 32 and EM 38.

STATS updates statistical objects in response to RTP invocation. There are at least four statistical object classes, namely, counters, timers, percentages (%), and meters. Each statistical object is implemented as appropriate to the object class to which it belongs. That is, each statistical object behaves such that when invoked by RTP 32 it updates and then generates an alarm if its value meets a preset threshold. (Meets means that for a high threshold the value is equal to or greater than the threshold and for a low threshold the value is equal to or less than the threshold. Note that a single object may have both high and low thresholds.)

STATS 36 is responsible for the maintenance and initial analysis of the database. This includes coordinating access to the database variables, ensuring appropriate interlocks are applied and generating alarms when thresholds are crossed. Only STATS 36 is aware of the internal structure of the database, the rest of the system is not.

STATS 36 is also responsible for tracking events of interest in the form of various statistical reductions. Examples are counters, rate meters, and rate of change of rate meters. It initiates events based on particular statistics reaching configured limits, i.e., thresholds. The events are passed to the EM which sends a trap (i.e., an alarm) to the Management Workstation. The statistics within STATS 36 are readable from the Management Workstation on request.

STATS performs lookup on all addressing fields. It assigns new data structures to address field values not currently present. It performs any hashing for fast access to the database. More details will be presented later in this document.

Event Manager (EM) 38 extracts statistics from STATS and formats it in ways that allow the Workstation to understand it. It also examines the various statistics to see if their behavior warrants a notification to the Management Workstation. If so, it uses the SNMP Agent software to initiate such notifications.

If the Workstation asks for data, EM 38 gets the data from STATS and sends it to the Workstation. It also performs some level of analysis for statistical, accounting and alarm filtering and decides on further action (e.g. delivery to the Management Workstation).

EM 38 is also responsible for controlling the delivery of events to the Management Workstation, e.g., it performs event filtering. The action to be taken on receipt of an event (e.g. threshold exceeded in STATS) is specified by the event action associated with the threshold. The event is used as an index to select the defined action (e.g. report to Workstation, run local routine xxxx, ignore). The action can be modified by commands from the Management Workstation (e.g., turn off an alarm) or by the control module in an overload situation. An update to the event action, however, does not affect events previously processed even if they are still waiting for transmission to the Management Workstation. Discarded events are counted as such by EM 38.

EM 38 also implements a throttle mechanism to limit the rate of delivery of alarms to the console based on configured limits. This prevents the rapid generation of multiple alarms. In essence, Monitor 10 is given a maximum frequency at which alarms may be sent to the Workstation. Although alarms in excess of the maximum frequency are discarded, a count is kept of the number of alarms that were discarded.

EM 38 invokes routines from the statistics module (STATS) to perform periodic updates such as rate calculations and threshold checks. It calculates time averages, e.g., average traffic by source stations, destination stations. EM 38 requests for access to monitor control variables are passed to the control module.

EM 38 checks whether asynchronous traps (i.e., alarms) to the Workstation are permitted before generating any.

EM 38 receives database update requests from the Management Workstation and invokes the statistics module (STATS) to process these.

Message Transport Module (MTM) 34, which is DRAM based, has two distinct but closely related functions. First, it is responsible for the conversion of Workstation commands and responses from the internal format used within Monitor 10 to the format used to communicate over the network. It isolates the rest of the system from the protocol used to communicate within Management Workstation. It translates between the internal representation of data and ASN.1 used for SNMP. It performs initial decoding of Workstation requests and directs the requests to appropriate modules for processing. It implements SNMP/UDP/IP/LLC or ETHERNET protocols for LAN and SNMP/UDP/IP/SLIP protocols for serial line. It receives network management commands from the Management Workstation and delivers these to the appropriate module for action. Alarms and responses destined for the Workstation are also directed via this module.

Second, MTM 34 is responsible for the delivery and reception of data to and from the Management Workstation using the protocol appropriate to the network. Primary and backup communication paths are provided transparently to the rest of the monitor modules (e.g. LAN and dial up link). It is capable of full duplex delivery of messages between the console and monitoring module. The messages carry event, configuration, test and statistics data.

Event Timing Module (ETM) 40 keeps track of the start time and end times of user specified transactions over the network. In essence, this module monitors the responsiveness of the network at any protocol or layer specified by the user.

Address Tracking Module 42 keeps track of the node name to node address bindings on networks which implement dynamic node addressing protocols.

Memory management for Monitor 10 is handled in accordance with following guidelines. The available memory is divided into four blocks during system initialization. One block includes receive frame buffers. They are used for receiving LAN traffic and for receiving secondary link traffic. These are organized as linked lists of fixed sized buffers. A second block includes system control message blocks. They are used for intertask messages within Monitor 10 and are organized as a linked list of free blocks and multiple linked lists of in process intertask messages. A third block includes transmit buffers. They are used for creation and transmission of workstation alarms and responses and are organized as a linked list of fixed sized buffers. A fourth block is the statistics. This is allocated as a fixed size area at system initialization and managed by the statistics module during system operation.

Task Structure of Monitor;

The structure of the Monitor in terms of tasks and intertask messages is shown in Fig. 6. The rectangular blocks represent interrupt service routines, the ovals represent tasks and the circles represent input queues.

Each task in the system has a single input queue which it uses to receive all input. All inter-process communications take place via messages placed onto the input queue of the destination task. Each task waits on a (well known) input queue and processes events or inter-task messages (i.e., ITM's) as they are received. Each task returns to the kernel within an appropriate time period defined for each task (e.g. after processing a fixed number of events).

Interrupt service routines (ISR's) run on receipt of hardware generated interrupts. They invoke task level processing by sending an ITM to the input queue of the appropriate task.

The kernel scheduler acts as the base loop of the system and calls any runnable tasks as subroutines. The determination of whether a task is runnable is made from the input queue, i.e., if this has an entry the task has work to perform. The scheduler scans the input queues for each task in a round robin fashion and invokes a task with input pending. Each task processes items from its input queue and returns to the scheduler within a defined period. The scheduler then continues the scan cycle of the input queues. This avoids any task locking out others by processing a continuously busy input queue. A task may be given an effectively higher priority by providing it with multiple entries in the scan table.

Database accesses are generally performed using access routines. This hides the internal structure of the database from other modules and also ensures that appropriate interlocks are applied to shared data.

The EM processes a single event from the input queue and then returns to the scheduler.

The MTM Xmit task processes a single event from its input queue and then returns control to the scheduler. The MTM Recv task processes events from the input queue until it is empty or a defined number (e.g. 10) events have been processed and then returns control to the scheduler.

The timer task processes a single event from the input queue and then returns control to the scheduler.

RTP continues to process frames until the input queue is empty or it has processed a defined number (e.g. 10) frames. It then returns to the scheduler.

The following sections contain a more detailed description of some of the above-identified software modules.

The Statistics Module (STATS):

The functions of the statistics module are:

* to define statistics records;

* to allocate and initialize statistics records;

* to provide routines to lookup statistics records, e.g. lookup_id_addr;

* to provide routines to manipulate the statistics within the records, e.g. stats_age, stats_incr and stats_rate;

* to provide routines to free statistics records, e.g. stats_allocate and stats_deallocate

It provides these services to the Real Time Parser (RTP) module and to the Event Manager (EM) module.

STATS defines the database and it contains subroutines for updating the statistics which it keeps.

STATS contains the type definitions for all statistics records (e.g. DLL, IP, TCP statistics). It provides an initialization routine whose major function is to allocate statistics records at startup from cacheable memory. It provides lookup routines in order to get at the statistics. Each type of statistics record has its own lookup routine (e.g. lookup_ip_address) which returns a pointer to a statistics record of the appropriate type or NULL.

As a received frame is being parsed, statistics within statistics records need to be manipulated (e.g. incremented) to record relevant information about the frame. STATS provides the routines to manipulate those statistics. For example, there is a routine to update counters. After the counter is incremented/decremented and if there is a non-zero threshold associated with the counter, the internal routine compares its value to the threshold. If the threshold has been exceeded, the Event Manager is signaled in order to send a trap to the Workstation. Besides manipulating statistics, these routines, if necessary, signal the Event Manager via an Intertask Message (ITM) to send a trap to the Management Workstation.

The following is an example of some of the statistics records that are kept in STATS.

o monitor statistics
o mac statistics for segment
o llc statisics for segment
o statistics per ethernet/lsap type for segment
o ip statistics for segment
o icmp statistics for segment
o tcp statistics for segment
o udp statistics for segment
o nfs statistics for segment
o ftp control statistics for segment
o ftp data statistics for segment
o telnet statistics for segment
o smtp statistics for segment
o arp statistics for segment

o statistics per mac address
o statistics per ethernet type/lasp per mac address
o statistics per ip address (includes icmp)
o statistics per tcp socket
o statistics per udp socket
o statistics per nfs socket
o statistics per ftp control socket
o statistics per ftp data socket
o statistics per telnet socket
o statistics per smtp socket
o arp statistics per ip address

o statistics per mac address pair
o statistics per ip pair (includes icmp)

o statistics per tcp connection
o statistics per udp pair
o statistics per nfs pair
o statistics per ftp control connection
o statistics per ftp data connection
o statistics per telnet connection
o statistics per smtp connection

o connection histories per udp and tcp socket

All statistics are organized similarly across protocol types. The details of the data structures for the DLL level are presented later.

As noted earlier, there are four statistical object classes (i.e., variables), namely, counts, rates, percentages (%), and meters. They are defined and implemented as follows.

A count is a continuously incrementing variable which rolls around to 0 on overflow. It may be reset on command from the user (or from software). A threshold may be applied to the count and will cause an alarm when the threshold count is reached. The threshold count fires each time the counter increments past the threshold value. For example, if the threshold is set to 5, alarms are generated when the count is 5, 10, 15,...

A rate is essentially a first derivative of a count variable. The rate is calculated at a period appropriate to the variable. For each rate variable, a minimum, maximum and average value is maintained. Thresholds may be set on high values of the rate. The maximums and minimums may be reset on command. The threshold event is triggered each time the rate calculated is in the threshold region.

As commonly used, the % is calculated at a period appropriate to the variable. For each % variable a minimum, maximum and average value is maintained. A threshold may be set on high values of the %. The threshold event is triggered each time the % calculated is in the threshold region.

Finally, a meter is a variable which may take any discrete value within a defined range. The current value has no correlation to past or future values. A threshold may be set on a maximum and/or minimum value for a meter.

The rate and % fields of network event variables are updated differently than counter or meter fields in that they are calculated at fixed intervals rather than on receipt of data from the network.

Structures for statistics kept on a per address or per address pair basis are allocated at initialization time. There are several sizes for these structures. Structures of the same size are linked together in a free pool. As a new structure is needed, it is obtained from a free queue, initialized, and linked into an active list. Active lists are kept on a per statistics type basis.

As an address or address pair (e.g. mac, ip, tcp...) is seen, RTP code calls an appropriate lookup routine. The lookup routine scans active statistics structures to see if a structure has already been allocated for the statistics. Hashing algorithms are used in order to provide for efficient lookup. If no structure has been allocated, the lookup routine examines the appropriate parse control records to determine whether statistics should be kept, and, if so, it allocates a structure of the appropriate size, initializes it and links it into an active list.

Either the address of a structure or a NULL is returned by these routines. If NULL is returned, the RTP does not stop parsing, but it will not be allowed to store the statistics for which the structure was requested.

The RTP updates statistics within the data base as it runs. This is done via macros defined for the RTP. The macros call on internal routines which know how to manipulate the relevant statistic. If the pointer to the statistics structure is NULL, the internal routine will not be invoked.

The EM causes rates to be calculated. The STATS module supplies routines (e.g. stats_rate) which must be called by the EM in order to perform the rate calculations. It also calls subroutines to reformat the data in the database in order to present it to the Workstation (i.e., in response to a get from the Workstation).

The calculation algorithms for the rate and % fields of network event variables are as follows.

The following rates are calculated in units per second, at the indicated (approximate) intervals:

1. 10 second intervals:
   e.g. DLL frame, byte, ethernet, 802.3, broadcast, multicast rates
2. 60 second intervals
   e.g., all DLL error, ethertype/dsap rates
   all IP rates.
   TCP packets, bytes, errors, retransmitted packets, retransmitted bytes, acks, rsts
   UDP packet, error, byte rates
   FTP file transfer, byte transfer, error rates

For these rates, the new average replaces the previous value directly. Maximum and minimum values are retained until reset by the user.

The following rates are calculated in units per hour at the indicated time intervals:

1. 15 minute interval.

e.g., TCP - connection rate
Telnet connection rate
FTP session rate

The hourly rate is calculated from a sum of the last twelve 5 minute readings, as obtained from the buckets for the pertinent parameter. Each new reading replaces the oldest of the twelve values maintained. Maximum and minimum values are retained until reset by the user.

There are a number of other internal routines in STATS. For example, all statistical data collected by the Monitor is subject to age out. Thus, if no activity is seen for an address (or address pair) in the time period defined for age out, then the data is discarded and the space reclaimed so that it may be recycled. In this manner, the Monitor is able to use the memory for active elements rather than stale data. The user can select the age out times for the different components. The EM periodically kicks off the aging mechanism to perform this recycling of resources. STATS provides the routines which the EM calls, e.g. stats_age.

There are also routines in STATS to allocate and de-allocate Statistics, e.g., stats_allocate and stats_de-allocate. The allocate routine is called when stations and dialogs are picked up by the Network Monitor. The de-allocate routine is called by the aging routines when a structure is to be recycled.

The Data Structures in STATS

The general structure of the database within STATS is illustrated by Figs. 7a-c, which shows information that is maintained for the Data Link Layer (DLL) and its organization. A set of data structures is kept for each address associated with the layer. In this case there are three relevant addresses, namely a segment address, indicating which segment the node is on, a MAC address for the node on the segment, and an address which identifies the dialog occurring over that layer. The dialog address is the combination of the MAC addresses for the two nodes which make up the dialog. Thus, the overall data structure has three identifiable components: a segment address data structure (see Fig. 7a), a MAC address data structure (see Fig. 7b) and a dialog data structure (see Fig. 7c).

The segment address structure includes a doubly linked list 102 of segment address records 104, each one for a different segment address. Each segment address record 104 contains a forward and backward link (field 106) for forward and backward pointers to neighboring records and a hash link (field 108). In other words, the segment address records are accessed by either walking down the doubly linked list or by using a hashing mechanism to generate a pointer into the doubly linked list to the first record of a smaller hash linked list. Each record also contains the address of the segment (field 110) and a set of fields for other information. Among these are a flags field 112, a type field 114, a parse_control field 116, and an EM_control field 118. Flags field 112 contains a bit which indicates whether the identified address corresponds to the address of another Network Monitor. This field only has meaning in the MAC address record and not in the segment or dialog address record. Type field 114 identifies the MIB group which applies to this address. Parse control field 116 is a bit mask which indicates what subgroups of statistics from the identified MIB group are maintained, if any. Flags field 112, type field 114 and parse control field 116 make up what is referred to as the parse control record for this MAC address. The Network Monitor uses a default value for parse control field 116 upon initialization or whenever a new node is detected.

The default value turns off all statistics gathering. The statistics gathering for any particular address may subsequently be turned on by the Workstation through a Network Monitor control command that sets the appropriate bits of the parse control field to one.

EM_control field 118 identifies the subgroups of statistics within the MIB group that have changed since the EM last serviced the database to update rates and other variables. This field is used by the EM to identify those parts of STATS which must be updated or for which recalculations must be performed when the EM next services STAT.

Each segment address record 104 also contains three fields for time related information. There is a start_time field 120 for the time that is used to perform some of the rate calculations for the underlying statistics; a first_seen field 122 for the time at which the Network Monitor first saw the communication; and a last_seen field 124 for the time at which the last communication was seen. The last_seen time is used to age out the data structure if no activity is seen on the segment after a preselected period of time elapses. The first_seen time is a statistic which may be of interest to the network manager and is thus retrievable by the Management Workstation for display.

Finally, each segment address record includes a stats_pointer field 126 for a pointer to a DLL segment statistics data structure 130 which contains all of the statistics that are maintained for the segment address. If the bits in parse_control field 116 are all set to off, indicating that no statistics are to be maintained for the address, then the pointer in stats_pointer field 126 is a null pointer.

The list of events shown in data structure 130 of Fig. 7a illustrates the type of data that is collected for this address when the parse control field bits are set to on. Some of the entries in DLL segment statistics data structure 130 are pointers to buckets for historical data. In the case where buckets are maintained, there are twelve buckets each of which represents a time period of five minutes duration and each of which generally contains two items of information, namely, a count for the corresponding five minute time period and a MAX rate for that time period. MAX rate records any spikes which have occurred during the period and which the user may not have observed because he was not viewing that particular statistic at the time.

At the end of DLL segment statistics data structure 130, there is a protocol_Q pointer 132 to a linked list 134 of protocol statistics records 136 identifying all of the protocols which have been detected running on top of the DLL layer for the segment. Each record 136 includes a link 138 to the next record in the list, the identity of the protocol (field 140), a frames count for the number of frames detected for the identified protocol (field 142); and a frame rate (field 144).

The MAC address data structure is organized in a similar manner to that of the segment data structure (see Fig. 7b). There is a doubly linked list 146 of MAC address records 148, each of which contains the same type of information as is stored in DLL segment address records 104. A pointer 150 at the end of each MAC address record 148 points to a DLL address statistics data structure 152, which like the DLL segment address data structure 130, contains fields for all of the statitics that are gathered for that DLL MAC address. Examples of the particular statistics are shown in Fig. 7b.

At the end of DLL address statistics data structure 152, there are two pointer fields 152 and 154, one for a pointer to a record 158 in a dialog link queue 160, and the other for a pointer to a linked list 162 of protocol statistics records 164. Each dialog link queue entry 158 contains a pointer to the next entry (field 168) in the queue and it contains a dialog_addr pointer 170 which points to an entry in the DLL dialog queue which involves the MAC address. (see Fig. 7c). Protocol statistics records 164 have the same structure and contain the same categories of information as their counterparts hanging off of DLL segment statistics data structure 130.

The above-described design is repeated in the DLL dialog data structures. That is, dialog record 172 includes the same categories of information as its counterpart in the DLL segment address data structure and the MAC address data structure. The address field 174 contains the addresses of both ends of the dialog concatenated together to form a single address. The first and second addresses within the single address are arbitrarily designated nodes 1 and 2, respectively. In the stats_pointer field 176 there is a pointer to a dialog statistics data structure 178 containing the relevant statistics for the dialog. The entries in the first two fields in this data structure (i.e., fields 180 and 182) are designated protocol entries and protocols. Protocol entries is the number of different protocols which have been seen between the two MAC addresses. The protocols that have been seen are enumerated in the protocols field 182.

DLL dialog statistics data structure 178, illustrated by Fig. 7c, includes several additional fields of information which only appear in these structures for dialogs for which state information can be kept (e.g. TCP connection). The additional fields identify the transport protocol (e.g., TCP) (field 184) and the application which is running on top of that protocol (field 186). They also include the identity of the initiator of the connection (field 188), the state of the connection (field 190) and the reason that the connection was closed, when it is closed (field 192). Finally, they also include a state_pointer (field 194) which points to a history data structure that will be described in greater detail later. Suffice it to say, that the history data structure contains a short history of events and states for each end of the dialog. The state machine uses the information contained in the history data structure to loosely determine what the state of each of the end nodes is throughout the course of the connection. The qualifier "loosely" is used because the state machine does not closely shadow the state of the connection and thus is capable of recovering from loss of state due to lost packets or missed communications.

The above-described structures and organization are used for all layers and all protocols within STATS.

Real Time Parser (RTP)

The RTP runs as an application task. It is scheduled by the Real Time Kernel scheduler when received frames are detected. The RTP parses the frames and causes statistics, state tracking, and tracing operations to be performed.

The functions of the RTP are:

* obtain frames from the RTP Input Queue;
* parse the frames;
* maintain statistics using routines supplied by the STATS module;
* maintain protocol state information;

- * notify the MTM via an ITM if a frame has been received with the Network Monitor's address as the destination address; and
- * notify the EM via an ITM if a frame has been received with any Network Monitor's address as the source address.

The design of the RTP is straightforward. It is a collection of routines which perform protocol parsing. The RTP interfaces to the Real Time Kernel in order to perform RTP initialization, to be scheduled in order to parse frames, to free frames, to obtain and send an ITM to another task; and to report fatal errors. The RTP is invoked by the scheduler when there is at least one frame to parse. The appropriate parse routines are executed per frame. Each parse routine invokes the next level parse routine or decides that parsing is done. Termination of the parse occurs on an error or when the frame has been completely parsed.

Each parse routine is a separately compilable module. In general, parse routines share very little data. Each knows where to begin parsing in the frame and the length of the data remaining in the frame.

The following is a list of the parse routines that are available within RTP for parsing the different protocols at the various layers.

Data Link Layer Parse - rtp_dll_parse:

This routine handles Ethernet, IEEE 802.3, IEEE 802.2, and SNAP. See RFC 1010, Assigned Numbers for a description of SNAP (Subnetwork Access Protocol).

Address Resolution Protocol Parse - rtp_arp_parse

ARP is parsed as specified in RFC 826.

Internet Protocol Parse - rtp_ip_parse

IP Version 4 is parsed as specified in RFC 791 as amended by RFC 950, RFC 919, and RFC 922.

Internet Control Message Protocol Parse - rtp_icmp_parse

ICMP is parsed as specified in RFC 792.

Unit Data Protocol Parse - rtp_udp_parse

UDP is parsed as specified in RFC 768.

Transmission Control Protocol Parse - rtp_tcp_parse

TCP is parsed as specified in RFC 793.

Simple Mail Transfer Protocol Parse - rtp_smtp_parse

SMTP is parsed as specified in RFC 821.

File Transfer Protocol Parse - rtp_ftp_parse

FTP is parsed as specified in RFC 959.

Telnet Protocol Parse - rtp_telnet_parse

The Telnet protocol is parsed as specified in RFC 854.

Network File System Protocol Parse - rpt_nfs_parse

The NFS protocol is parsed as specified in RFC 1094.

The RTP calls routines supplied by STATS to look up data structures. By calling these lookup routines, global pointers to data structures are set up. Following are examples of the pointers to statistics data structures that are set up when parse routines call Statistics module lookup routines.

mac_segment, mac_dst_segment, mac_this_segment,
mac_src, mac_dst, mac_dialog
ip_src_segment, ip_dst_segment, ip_this_segment,
ip_src, ip_dst, ip_dialog
tcp_src_segment, tcp_dst_segment,
tcp_this_segment,
tcp_src, tcp_dst, tcp_src_socket, tcp_dst_socket,
tcp_connection

The mac_src and mac_dst routines return pointers to the data structures within STATS for the source MAC address and the destination MAC address, respectively. The lookup_mac_dialog routine returns a pointer to the data structure within STATS for the dialog between the two nodes on the MAC layer. The other STATS routines supply similar pointers for data structures relevant to other protocols.

The RTP routines are aware of the names of the statistics that must be manipulated within the data base (e.g. frames, bytes) but are not aware of the structure of the data. When a statistic is to be manipulated, the RTP routine invokes a macro which manipulates the appropriate statistics in data structures. The macros use the global pointers which were set up during the lookup process described above.

After a frame has been parsed (whether the parse was successful or not), the RTP routine examines the destination mac and ip addresses. If either of the addresses is that of the Network Monitor, RTP obtains a low priority ITM, initializes it, and sends the ITM to the MTM task. One of the fields of the ITM contains the address of the buffer containing the frame.

The RTP must hand some received frames to the EM in order to accomplish the autotopology function (described later). After a frame has been parsed (whether the parse was successful or not), the RTP routine examines the source mac and ip addresses. If either of the addresses is that of another Network Monitor, RTP obtains a low priority ITM, initializes it and sends the ITM to the EM task. The address data structure (in particular, the flags field of the parse control record) within STATS for the MAC or the IP address indicates whether the source address is that of another Network Monitor. One of the fields of the ITM contains the address of the buffer containing the frame.

The RTP receives traffic frames from the network for analysis. RTP operation may be modified by sending control messages to the Monitor. RTP first parses these messages, then detects that the messages are destined for the Monitor and passes them to the MTM task. Parameters which affect RTP operation may be changed by such control messages.

The general operation of the RTP upon receipt of a traffic frame is as follows:

```
Get next frame from input queue
get address records for these stations
For each level of active parsing
{
get pointer to start of protocol header
call layer parse routine
determine protocol at next level
set pointer to start of next layer protocol

}end of frame parsing
if this is a monitor command add to MTM input queue
if this frame is from another monitor, pass to EM
check for overload -if yes tell control
```

<u>The State Machine</u>:

In the described embodiment, the state machine determines and keeps state for both addresses of all TCP connections. TCP is a connection oriented transport protocol, and TCP clearly defines the connection in terms of states of the connection. There are other protocols which do not explicitly define the communication in terms of state, e.g. connectionless protocols such as NFS. Nevertheless, even in the connectionless protocols there is implicitly the concept of state because there is an expected order to the events which will occur during the course of the communication. That is, at the very least, one can identify a beginning and an end of the communication, and usually some sequence of events which will occur during the course of the communication. Thus, even though the described embodiment involves a connection oriented protocol, the principles are applicable to many connectionless protocols or for that matter any protocol for which one can identify a beginning and an end to the communication under that protocol.

Whenever a TCP packet is detected, the RTP parses the information for that layer to identify the event associated with that packet. It then passes the identified event along with the dialog identifier to the state machine. For each address of the two parties to the communication, the state machine determines what the current state of the node is. The code within the state machine determines the state of a connection based upon a set of rules that are illustrated by the event/state table shown in Fig. 8.

The interpretation of the event/state table is as follows. The top row of the table identifies the six possible states of a TCP connection. These states are not the states defined in the TCP protocol specification. The left most column identifies the eight events which may occur during the course of a connection. Within the table is an array of boxes, each of which sits at the intersection of a particular event/state combination. Each box specifies the actions taken by the state machine if the identified event occurs while the connection is in the identified state. When the state machine receives a new event, it may perform three types of action. It may change the recorded state for the node. The state to which the node is changed is specified by the S="STATE" entry located at the top of the box. It may increment or decrement the appropriate counters to record the information relevant to that event's occurrence. (In the table, incrementing and decrementing are signified by the ++ and the -- symbols, respectively, located after the identity of the variable being updated.) Or the state machine may take other actions such as those specified in the table as start close timer, Look_for_Data_State, or Look_at_History (to be described shortly). The particular actions which the state machine takes are specified in each box. An empty box indicates that no action is taken for that particular event/state combination. Note, however, that the occurrence of an event is also likely to have caused the update of statistics within STATS, if not by the state machine, then by some other part of the RTP. Also note that it may be desirable to have the state machine record other events, in which case the state table would be modified to identify those other actions.

Two events appearing on the table deserve further explanation, namely, close timer expires and inactivity timer expires. The close timer, which is specified by TCP, is started at the end of a connection and it establishes a period during which any old packets for the connection which are received are thrown away (i.e., ignored). The inactivity timer is not specified by TCP but rather is part of the Network Monitor's resource management functions. Since keeping statistics for dialogs (especially old dialogs) consumes resources, it is desirable to recycle resources for a dialog if no activity has been seen for some period of time. The inactivity timer provides the mechanism for accomplishing this. It is restarted each time an event for the connection is received. If the inactivity timer expires (i.e., if no event is received before the timer period ends), the connection is assumed to have gone inactive and all of the resources associated with the dialog are recycled. This involves freeing them up for use by other dialogs.

The other states and events within the table differ from but are consistent with the definitions provided by TCP and should be self evident in view of that protocol specification.

The event/state table can be read as follows. Assume, for example, that node 1 is in DATA state and the RTP receives another packet from node 1 which it determines to be a TCP FIN packet. According to the entry in the table at the intersection of FIN/DATA (i.e., event/state), the state machine sets the state of the connection for node 1 to CLOSING, it decrements the active connections counter and it starts the close timer. When the close timer expires, assuming no other events over that connection have occurred, the state machine sets node 1's state to CLOSED and it starts the inactivity timer. If the RTP sends another SYN packet to reinitiate a new connection before the inactive timer expires, the state machine sets node 1's state to CONNECTING (see the SYN/CLOSED entry) and it increments an after close counter.

When a connection is first seen, the Network Monitor sets the state of both ends of the connection to UNKNOWN state. If some number of data and acknowledgment frames are seen from both connection ends, the states of the connection ends may be promoted to DATA state. The connection history is searched to make this determination as will be described shortly.

Referring to Figs. 9a-b, within STATS there is a history data structure 200 which the state machine uses to remember the current state of the connection, the state of each of the nodes participating in the connection and a short history of state related information. History data structure 200 is identified by a state_pointer found at the end of the associated dialog statistics data structure in STATS (see Fig. 7c). Within history data structure 200, the state machine records the current state of node 1 (field 202), the current state of node 2 (field 206) and other data relating to the corresponding node (fields 204 and 208). The other data includes, for example, the window size for the receive and transmit communications, the last detected sequence numbers for the data and acknowledgment frames, and other data transfer information.

History data structure 200 also includes a history table (field 212) for storing a short history of events which have occurred over the connection and it includes an index to the next entry within the history table for storing the information about the next received event (field 210). The history table is implemented as a circular buffer which includes sufficient memory to store, for example, 16 records. Each record, shown in Fig. 9b, stores the state of the node when the event was detected (field 218), the event which was detected (i.e., received) (field 220), the data field length (field 222), the sequence number (field 224), the acknowledgment sequence number (field 226) and the identity of the initiator of the event, i.e., either node 1 or node 2 or 0 if neither (field 228).

Though the Network Monitor operates in a promiscuous mode, it may occasionally fail to detect or it may, due to overload, lose a packet within a communication. If this occurs the state machine may not be able to accurately determine the state of the connection upon receipt of the next event. The problem is evidenced by the fact that the next event is not what was expected. When this occurs, the state machine tries to recover state by relying on state history information stored in the history table in field 212 to deduce what the state is. To deduce the current state from historical information, the state machine uses one of the two previously mentioned routines, namely, Look_for_Data_State and Look_at_History.

Referring to Fig. 10, Look_for_Data_State routine 230 searches back through the history one record at a time until it finds evidence that the current state is DATA state or until it reaches the end of the circular buffer (step 232). Routine 230 detects the existence of DATA state by determining whether node 1 and node 2 each have had at least two data events or two acknowledgment combinations with no intervening connect, disconnect or abort events (step 234). If such a sequence of events is found within the history, routine 230 enters both node 1 and node 2 into DATA state (step 236), it increments the active connections counter (step 238) and then it calls a Look_for_Initiator routine to look for the initiator of the connection (step 240). If such a pattern of events is not found within the history, routine 230 returns without changing the state for the node (step 242).

As shown in Fig. 11, Look_for_Initiator routine 240 also searches back through the history to detect a telltale event pattern which identifies the actual initiator of the connection (step 244). More specifically, routine 240 determines whether nodes 1 and 2 each sent connect-related packets. If they did, routine 240 identifies the initiator as the first node to send a connect-related packet (step 246). If the search is not successful, the identity of the connection initiator remains unknown (step 248).

The Look_at_History routine is called to check back through the history to determine whether data transmissions have been repeated. In the case of retransmissions, the routine calls a Look_for_Retransmission routine 250, the operation of which is shown in Fig. 12. Routine 250 searches back through the history (step 252) and checks whether the same initiator node has sent data twice (step 254). It detects this by comparing the current sequence number of the packet as provided by the RTP with the sequence numbers of data packets that were previously sent as reported in the history table. If a retransmission is spotted, the retransmission counter in the dialog statistics data structure of STATS is incremented (step 256). If the sequence number is not found within the history table, indicating that the received packet does not represent a retransmission, the retransmission counter is not incremented (step 258).

Other statistics such as Window probes and keep alives may also be detected by looking at the received frame, data transfer variables, and, if necessary, the history.

Even if frames are missed by the Network Monitor, because it is not directly "shadowing" the connection, the Network Monitor still keeps useful statistics about the connection. If inconsistencies are detected the Network Monitor counts them and, where appropriate, drops back to UNKNOWN state. Then, the Network Monitor waits for the connection to stabilize or deteriorate so that it can again determine the appropriate state based upon the history table.

Principal Transactions of Network Monitor Modules:

The transactions which represent the major portion of the processing load within the Monitor, include monitoring, actions on threshold alarms, processing database get/set requests from the Management Workstation, and processing monitor control requests from the Management Workstation. Each of these mechanisms will now be briefly described.

Monitoring involves the message sequence shown in Fig. 13. In that figure, as in the other figures involving message sequences, the numbers under the heading SEQ. identify the major steps in the sequence. The following steps occur:

1. ISR puts Received traffic frame ITM on RTP input queue
2. request address of pertinent data structure from STATS (get parse control record for this station)
3. pass pointer to RTP
4. update statistical objects by call to statistical update routine in STATS using pointer to pertinent data structure
5. parse completed - release buffers

The major steps which follow a statistics threshold event (i.e., an alarm event) are shown in Fig. 14. The steps are as follows:

1. statistical object update causes threshold alarm
2. STATS generates threshold event ITM to event manager (EM)
3. look up appropriate action for this event
4. perform local event processing
5. generate network alarm ITM to MTM Xmit (if required)
6. format network alarm trap for Workstation from event manager data
7. send alarm to Workstation

The major steps in processing of a database update request (i.e., a get/set request) from the Management Workstation are shown in Fig. 15. The steps are as follows:

1. LAN ISR receives frame from network and passes it to RTP for parsing
2. RTP parses frame as for any other traffic on segment.
3. RTP detects frame is for monitor and sends received Workstation message over LAN ITM to MTM Recv.

4. MTM Recv processes protocol stack.
5. MTM Recv sends database update request ITM to EM.
6. EM calls STATS to do database read or database write with appropriate IMPB
7. STATS performs database access and returns response to EM.
8. EM encodes response to Workstation and sends database update response ITM to MTM Xmit
9. MTM Xmit transmits.

The major steps in processing of a monitor control request from the Management Workstation are shown in Fig. 16. The steps are as follows:

1. Lan ISR receives frame from network and passes received frame ITM to RTP for parsing.
2. RTP parses frame as for any other traffic on segment.
3. RTP detects frame is for monitor and sends received workstation message over LAN ITM to MTM Recv.
4. MTM Recv processes protocol stack and decodes workstation command.
5. MTM Recv sends request ITM to EM.
6. EM calls Control with monitor control IMPB.
7. Control performs requested operation and generates response to EM.
8. EM sends database update response ITM to MTM Xmit.
9. MTM Xmit encodes response to Workstation and transmits.

<u>The Monitor/Workstation Interface</u>:

The interface between the Monitor and the Management Workstation is based on the SNMP definition (RFC 1089 SNMP; RFC 1065 SMI; RFC 1066 SNMP MIB - Note: RFC means Request for Comments). All five SNMP PDU types are supported:

get-request

get-next-request
get-response
set-request
trap

The SNMP MIB extensions are designed such that where possible a user request for data maps to a single complex MIB object. In this manner, the get-request is simple and concise to create, and the response should contain all the data necessary to build the screen. Thus, if the user requests the IP statistics for a segment this maps to an IP Segment Group.

The data in the Monitor is keyed by addresses (MAC, IP) and port numbers (telnet, FTP). The user may wish to relate his data to physical nodes entered into the network map. The mapping of addresses to physical nodes is controlled by the user (with support from the Management Workstation system where possible) and the Workstation retains this information so that when a user requests data for node 'Joe' the Workstation asks the Monitor for the data for the appropriate address(es). The node to address mapping need not be one to one.

Loading and dumping of monitors uses TFTP (Trivial File Transfer Protocol). This operates over UDP as does SNMP. The Monitor to Workstation interface follows the SNMP philosophy of operating primarily in a polled mode. The Workstation acts as the master and polls the Monitor slaves for data on a regular (configurable) basis.

The information communicated by the SNMP is represented according to that subset of ASN.1 (ISO 8824 Specification of ASN.1) defined in the Internet standard Structure of Management Information (SMI - RFC 1065). The subset of the standard Management Information Base (MIB) (RFC 1066 SNMP MIB) which is supported by the Workstation is defined in Appendix III. The added value provided by the Workstation is encoded as enterprise specific extensions to the MIB as defined in Appendix IV. The format for these extensions follows the SMI recomendations for object identifiers so that the Workstation extensions fall in the subtree 1.3.6.1.4.1.x.1. where x is an enterprise specific node identifier assigned by the IAB.

Appendix V is a summary of the network variables for which data is collected by the Monitor for the extended MIB and which can be retrieved by the Workstation. The summary includes short decriptions of the meaning and significance of the variables, where appropriate.

The Management Workstation:

The Management Workstation is a SUN Sparcstation (also referred to as a Sun) available from Sun Microsystems, Inc. It is running the Sun flavor of Unix and uses the Open Look Graphical User Interface (GUI) and the SunNet Manager as the base system. The options required are those to run SunNet Manager with some additional disk storage requirement.

The network is represented by a logical map illustrating the network components and the relationships between them, as shown in Fig. 17. A hierarchical network map is supported with navigation through the layers of the hierarchy, as provided by SNM. The Management Workstation determines the topology of the network and informs the user of the network objects and their connectivity so that he can create a network map. To assist with the map creation process, the Management Workstation attempts to determine the stations connected to each LAN segment to which a Monitor is attached. Automatic determination of segment topology by detecting stations is performed using the autotopology algorithms as described in copending U.S. Patent Application S.N. ***,*** entitled "Automatic Topology Monitor for Multi-

Segment Local Area Network" filed on January 14, 1991 (Attorney Docket No. 13283-NE.APP), incorporated herein by reference.

In normal operation, each station in the network is monitored by a single Monitor that is located on its local segment. The initial determination of the Monitor responsible for a station is based on the results of the autotopology mechanism. The user may override this initial default if required.

The user is informed of new stations appearing on any segment in the network via the alarm mechanism. As for other alarms, the user may select whether stations appearing on and disappearing from the network segment generate alarms and may modify the times used in the aging algorithms. When a new node alarm occurs, the user must add the new alarm to the map using the SNM tools. In this manner, the SNM system becomes aware of the nodes.

The sequence of events following the detection of a new node is:

1. the location of the node is determined automatically for the user.
2. the Monitor generates an alarm for the user indicating the new node and providing some or all of the following information:
   mac address of node
   ip address of node
   segment that the node is believed to be
   located on
   Monitor to be responsible for the node
3. the user must select the segment and add the node manually using the SNM editor

4. The update to the SNM database will be detected and the file reread. The Workstation database is reconstructed and the parse control records for the Monitors updated if required.
5. The Monitor responsible for the new node has its parse control record updated via SNMP set request(s).

An internal record of new nodes is required for the autotopology. When a new node is reported by a Network Monitor, the Management Workstation needs to have the previous location information in order to know which Network Monitors to involve in autotopology. For example, two nodes with the same IP address may exist in separate segments of the network. The history makes possible the correlation of the addresses and it makes possible duplicate address detection.

Before a new Monitor can communicate with the Management Workstation via SNMP it needs to be added to the SNM system files. As the SNM files are cached in the database, the file must be updated and the SNM system forced to reread it.

Thus, on the detection of a new Monitor the following events need to occur in order to add the Monitor to the Workstation:

1. The Monitor issues a trap to the Management Workstation software and requests code to be loaded from the Sun Microsystems boot/load server.
2. The code load fails as the Monitor is not known to the unix networking software at this time.
3. The Workstation confirms that the new Monitor does not exceed the configured system limits (e.g. 5 Monitors per

Workstation) and terminates the initialization sequence if limits are exceeded. An alarm is issued to the user indicating the presence of the new Monitor and whether it can be supported.

4. The user adds the Monitor to the SNMP.HOSTS file of the SNM system, to the etc/hosts file of the Unix networking system and to the SNM map.
5. When the files have been updated the user resets the Monitor using the set tool (described later).
6. The Monitor again issues a trap to the Management Workstation software and requests code to be loaded from the Sun boot/load server.
7. The code load takes place and the Monitor issues a trap requesting data from the Management Workstation.
8. The Monitor data is issued using SNMP set requests.

Note that on receiving the set request, the SNMP proxy rereads in the (updated) SNMP.HOSTS file which now includes the new Monitor. Also note that the SNMP hosts file need only contain the Monitors, not the entire list of nodes in the system.

9. On completion of the set request(s) the Monitor run command is issued by the Workstation to bring the Monitor on line.

The user is responsible for entering data into the SNM database manually. During operation, the Workstation monitors the file write date for the SNM database. When this is different from the last date read, the SNM database is reread and the Workstation database reconstructed. In this manner, user updates to the SNM database are incorporated into the Workstation database as quickly as possible without need for the user to take any action.

When the Workstation is loaded, the database is created from the data in the SNM file system (which the user has possibly updated). This data is checked for consistency and for conformance to the limits imposed by the Workstation at this time and a warning is generated to the user if any problems are seen. If the data errors are minor the system continues operation; if they are fatal the user is asked to correct them and Workstation operation terminates.

The monitoring functions of the Management Workstation are provided as an extension to the SNM system. They consist of additional display tools (i.e., summary tool, values tool, and set tool) which the user invokes to access the Monitor options and a Workstation event log in which all alarms are recorded.

As a result of the monitoring process, the Monitor makes a large number of statistics available to the operator. These are available for examination via the Workstation tools that are provided. In addition, the Monitor statistics (or a selected subset thereof) can be made visible to any SNMP manager by providing it with knowledge of the extended MIB. A description of the statistics maintained are described elswhere.

Network event statistics are maintained on a per network, per segment and per node basis. Within a node, statistics are maintained on a per address (as appropriate to the protocol layer - IP address, port number, ...) and per connection basis. Per network statistics are always derived by the Workstation from the per segment variables maintained by the Monitors. Subsets of the basic statistics are maintained on a node to node and segment to segment basis.

If the user requests displays of segment to segment traffic, the Workstation calculates this data as follows. The inter segment traffic is derived from the node to node statistics for the intersecting set of nodes. Thus, if segment A has nodes 1, 2, and 3 and segment B has nodes 20, 21, and 22, then summing the node to node traffic for

1 -> 20,21,22
2 -> 20,21,22
3 -> 20,21,22

produces the required result. On-LAN/off-LAN traffic for segments is calculated by a simply summing node to node traffic for all stations on the LAN and then subtracting this from total segment counts.

Alarms are reported to the user in the following ways:

1. Alarms received are logged in a Workstation log.
2. The node which the alarm relates to is highlighted on the map.
3. The node status change is propagated up through the (map) hierarchy to support the case where the node is not visible on the screen. This is as provided by SNM.

Summary Tool

After the user has selected an object from the map and invokes the display tools, the summary tool generates the user's initial screen at the Management Workstation. It presents a set of statistical data selected to give an overview of the operational status of the object (e.g., a selected node or segment). The Workstation polls the Monitor for the data required by the Summary Tool display screens.

The Summary Tool displays a basic summary tool screen such as is shown in Fig. 18. The summary tool screen has three panels, namely, a control panel 602, a values panel 604, and a dialogs panel 606. The control panel includes the indicated mouse activated bottons. The functions of each of the buttons is as follows. The file button invokes a traditional file menu. The view button invokes a view menu which allows the user to modify or tailor the visual protperties of the tool. The properties button invokes a properties menu containing choices for viewing and sometimes modifying the properties of objects. The tools button invokes a tools menu which provides access to the other Workstation tools, e.g. Values Tool.

The Update Interval field allows the user to specify the frequency at which the displayed statistics are updated by polling the Monitor. The Update Once button enables the user to retrieve a single screen update. When the Update Once button is invoked not only is the screen updated but the update interval is automatically set to "none".

The type field enables the user to specify the type of network objects on which to operate, i.e., segment or node.

The name button invokes a pop up menu containing an alphabetical list of all network objects of the type selected and apply and reset buttons. The required name can then be selected from the (scrolling) list and it will be entered in the name field of the summary tool when the apply button is invoked. Alternatively, the user may enter the name directly in the summary tool name field.

The protocol button invokes a pop up menu which provides an exclusive set of protocol layers which the user may select. Selection of a layer copies the layer name into the displayed field of the summary tool when the apply operation is invoked. An example of a protocol selection menu is shown in Fig. 19. It displays the available protocols in the form of a protocol tree with multiple protocol familes. The protocol selection is two dimensional. That is, the user first selects the protocol family and then the particular layer within that family.

As indicated by the protocol trees shown in Fig. 19, the capabilities of the Monitor can be readily extended to handle other protocol families. The particular ones which are implemented depend upon the needs of the particular network environment in which the Monitor will operate.

The user invokes the apply button to indicate that the selection process is complete and the type, name, protocol, etc. should be applied. This then updates the screen using the new parameter set that the user selected. The reset button is used to undo the selections and restore them to their values at the last apply operation.

The set of statistics for the selected parameter set is displayed in values panel 604. The members of the sets differ depending upon, for example, what protocol was selected. Figs. 20a-g present examples of the types of statistical variables which are displayed for the DLL, IP, UDP, TCP, ICMP, NFS, and ARP/RARP protocols, respectively. The meaning of the values display fields are described in Appendix I, attached hereto.

Dialogs panel 606 contains a display of the connection statistics for all protocols for a selected node. Within the Management Workstation, connection lists are maintained per node, per supported protocol. When connections are displayed, they are sorted on "Last Seen" with the most current displayed first. A single list returned from the Monitor contains all current connection. For TCP, however, each connection also contains a state and TCP connections are displayed as

Past and Present based upon the returned state of the connection. For certain dialogs, such as TCP and NFS over UDP, there is an associated direction to the dialog, i.e., from the initiator (source) to the receiver (sink). For these dialogs, the direction is identified in a DIR. field. A sample of information that is displayed in dialogs panel 606 is presented in Fig. 21 for current connections.

Values Tool

The values tool provides the user with the ability to look at the statistical database for a network object in detail. When the user invokes this tool, he may select a basic data screen containing a rate values panel 620, a count values panel 622 and a protocols seen panel 626, as shown in Fig. 22, or he may select a traffic matrix screen 628, as illustrated in Fig. 23.

In rate values and count values panels 620 and 622, value tools presents the monitored rate and count statistics, respectively, for a selected protocol. The parameters which are displayed for the different protocols (i.e., different groups) are listed in Appendix II. In general, a data element that is being displayed for a node shows up in three rows, namely, a total for the data element, the number into the data element, and the number out of the data element. Any exceptions to this are identified in Appendix II. Data elements that are displayed for segments, are presented as totals only, with no distinction between Rx and Tx.

When invoked the Values Tool displays a primary screen to the user. The primary screen contains what is considered to be the most significant information for the selected object. The user can view other information for the object (i.e., the statistics for the other parameters) by scrolling down.

The displayed information for the count values and rate values panels 620 and 622 includes the following. An alarm field reports whether an alarm is currently active for this item. It displays as "*" if active alarm is present. A Current Value/Rate field reports the current rate or the value of the counter used to generate threshold alarms for this item. This is reset following each threshold trigger and thus gives an idea of how close to an alarm threshold the variable is. A Typical Value field reports what this item could be expected to read in a "normal" operating situation. This field is filled in for those items where this is predictable and useful. It is maintained in the Workstation database and is modifiable by the user using the set tool. An Accumulated Count field reports the current accumulated value of the item or the current rate. A Max Value field reports the highest value recently seen for the item. This value is reset at intervals defined by a user adjustable parameter (default 30 minutes). This is not a rolling cycle but rather represents the highest value since it was reset which may be from 1 to 30 minutes ago (for a rest period of 30 minutes). It is used only for rates. A Min Value field reports the lowest value recently seen for the item. This operates in the same manner as Max Value field and is used only for rates.

A Percent (%) field reports only for the following variables:

off seg counts:

100(in count / total off seg count)
100(out count / total off seg count)
100(transit count / total off seg count)
100(local count / total off seg count)

off seg rates

100(transit rate / total off seg rate), etc.

protocols

100(frame rate this protocol / total frame rate)

On the right half of the basic display, there the following addtional fields: a High Threshold field and a Sample period for rates field.

Set Tool

The set tool provides the user with the ability to modify the parameters controling the operation of the Monitors and the Management Workstation. These parameters affect both user interface displays and the actual operation of the Monitors. The parameters which can be operated on by the set tool can be divided into the following categories: alarm thresholds, monitoring control, segment Monitor administration, and typical values.

The monitoring control variables specify the actions of the segment Monitors and each Monitor can have a distinct set of control variables (e.g., the parse control records that are described elsewhere). The user is able to define those nodes, segments, dialogs and protocols in which he is interested so as to make the best use of memory space available for data storage. This mechanism allows for load sharing, where mulitple Monitors on the same segment can divide up the total number of network objects which are to be monitored so that no duplication of effort between them takes place.

The monitor administration variables allow the user to modify the operation of the segment Monitor in a more direct manner than the monitoring control variables. Using the set tool, the user can perform those operations such as reset, time changes etc. which are normally the prerogative of a system administrator.

Note that the above descriptions of the tools available through the Management Workstation are not meant to imply that other choices may not be made regarding the particular information which is displayed and the manner in which it is displayed.

Adaptively Setting Network Monitor Thresholds:

The Workstation sets the thresholds in the Network Monitor based upon the performance of the system as observed over an extended period of time. That is, the Workstation periodically samples the output of the Network Monitors and assembles a model of a normally functioning network. Then, the Workstation sets the thresholds in the Network Monitors based upon that model. If the observation period is chosen to be long enough and since the model represents the "average" of the network performance over the observation period, temporary undesired deviations from normal behavior are smoothed out over time and model tends to accurately reflect normal network behavior.

Referring the Fig. 24, the details of the training procedure for adaptively setting the Network Monitor thresholds are as follows. To begin training, the Workstation sends a start learning command to the Network Monitors from which performance data is desired (step 302). The start learning command disables the thresholds within the Network Monitor and causes the Network Monitor to periodically send data for a predefined set of network parameters to the Management Workstation. (Disabling the thresholds, however, is not necessary. One could have the learning mode operational in parallel with monitoring using existing thresholds.) The set of parameters may be any or all of the previously mentioned parameters for which thresholds are or may be defined.

Throughout the learning period, the Network Monitor sends "snapshots" of the network's performance to the Workstation which, in turn, stores the data in a performance history database 306 (step 304). The network manager sets the length of the learning period.

Typically, it should be long enough to include the full range of load conditions that the network experiences so that a representative performance history is generated. It should also be long enough so that short periods of overload or faulty behavior do not distort the resulting averages.

After the learning period has expired, the network manager, through the Management Workstation, sends a stop learning command to the Monitor (step 308). The Monitor ceases automatically sending further performance data updates to the Workstation and the Workstation processes the data in its performance history database (step 310). The processing may involve simply computing averages for the parameters of interest or it may involve more sophisticated statistical analysis of the data, such as computing means, standard deviations, maximum and minimum values, or using curve fitting to compute rates and other pertinent parameter values.

After the Workstation has statistically analyzed the performance data, it computes a new set of thresholds for the relevant performance parameters (step 312). To do this, it uses formulas which are appropriate to the particular parameter for which a threshold is being computed. That is, if the parameter is one for which one would expect to see wide variations in its value during network monitoring, then the threshold should be set high enough so that the normal expected variations do not trigger alarms. On the other hand, if the parameter is of a type for which only small variations are expected and larger variations indicate a problem, then the threshold should be set to a value that is close to the average observed value. Examples of formulae which may be used to compute thresholds are:

* Highest value seen during learning period;

* Highest value seen during learning period + 10%;
* Highest value seen during learning period + 50%;
* Highest value seen during learning period + user-defined percent;
* Any value of the parameter other than zero;
* Average value seen during learning period + 50%; and
* Average value seen during learning period + user-defined percent.

As should be evident from these examples, there is a broad range of possibilities regarding how to compute a particular threshold. The choice, however, should reflect the parameter's importance in signaling serious network problems and its normal expected behavior (as may be evidenced from the performance history acquired for the parameter during the learning mode).

After the thresholds are computed, the Workstation loads them into the Monitor and instructs the Monitor to revert to normal monitoring using the new thresholds (step 314).

This procedure provides a mechanism enabling the network manager to adaptively reset thresholds in response to changing conditions on the network, shifting usage patterns and evolving network topology. As the network changes over time, the network manager merely invokes the adaptive threshold setting feature and updates the thresholds to reflect those changes.

The Diagnostic Analyzer Module:

The Management Workstation includes a diagnostic analyzer module which automatically detects and diagnoses the existence and cause of certain types of network problems. The functions of the diagnostic module may actually be distributed among the Workstation and the

Network Monitors which are active on the network. In principle, the diagnostic analyzer module includes the following elements for performing its fault detection and analysis functions.

The Management Workstation contains a reference model of a normally operating network. The reference model is generated by observing the performance of the network over an extended period of time and computing averages of the performance statistics that were observed during the observation period. The reference model provides a reference against which future network performance can be compared so as to diagnose and analyze potential problems. The Network Monitor (in particular, the STATS module) includes alarm thresholds on a selected set of the parameters which it monitors. Some of those thresholds are set on parameters which tend to be indicative of the onset or the presence of particular network problems.

During monitoring, when a Monitor threshold is exceeded, thereby indicating a potential problem (e.g. in a TCP connection), the Network Monitor alerts the Workstation by sending an alarm. The Workstation notifies the user and presents the user with the option of either ignoring the alarm or invoking a diagnostic algorithm to analyze the problem. If the user invokes the diagnostic algorithm, the Workstation compares the current performance statistics to its reference model to analyze the problem and report its results. (Of course, this may also be handled automatically so as to not require user intervention.) The Workstation obtains the data on current performance of the network by retrieving the relevant performance statistics from all of the segment Network Monitors that may have information useful to diagnosing the problem.

The details of a specific example involving poor TCP connection performance will now be described. This example refers to a typical network on which the diagnostic analyzer resides, such as the network illustrated in Fig. 25. It includes three segments labelled S1, S2, and S3, a router R1 connecting S1 to S2, a router R2 connecting S2 to S3, and at least two nodes, node A on S1 which communicates with node B on S3. On each segment there is also a Network Monitor 324 to observe the performance of its segment in the manner described earlier. A Management Workstation 320 is also located on S1 and it includes a diagnostic analyzer module 322. For this example, the sympton of the network problem is degraded peformance of a TCP connection between Nodes A and B.

A TCP connection problem may manifest itself in a number of ways, including, for example, excessively high numbers for any of the following:

errors
packets with bad sequence numbers
packets retransmitted
bytes retransmitted
out of order packets
out of order bytes
packets after window closed
bytes after window closed
average and maximum round trip times

or by an unusually low value for the current window size. By setting the appropriate thresholds, the Monitor is programmed to recognize any one or more of these symptons. If any one of of the thresholds is exceeded, the Monitor sends an alarm to the Workstation. The Workstation is programmed to recognize the particular alarm as related to an event which can be further analyzed by its diagnostic analyzer module 322. Thus, the Workstation presents the user with the option of invoking its diagnostic capabilities (or automatically invokes the diagnostic capabilities).

In general terms, when the diagnostic analyzer is invoked, it looks at the performance data that the segment Monitors produce for the two nodes, for the dialogs between them and for the links that interconnect them and compares that data to the reference model for the network. If a significant divergence from the reference model is identified, the diagnostic analyzer informs the Workstation (and the user) about the nature of the divergence and the likely cause of the problem. In conducting the comparison to "normal" network performance, the network circuit involved in communications between nodes A and B is decomposed into its individual components and diagnostic analysis is performed on each link individually in the effort to isolate the problem further.

The overall structure of the diagnostic algorithm 400 is shown in Fig. 26. When invoked for analyzing a possible TCP problem between nodes A and B, diagnostic analyzer 322 checks for a TCP problem at node A when it is acting as a source node (step 402). To perform this check, diagnostic algorithm 400 invokes a source node analyzer algorithm 450 shown in Fig. 27. If a problem is identified, the Workstation reports that there is a high probability that node A is causing a TCP problem when operating as a source node and it reports the results of the investigation performed by algorithm 450 (step 404).

If node A does not appear to be experiencing a TCP problem when acting as a source node, diagnostic analyzer 322 checks for evidence of a TCP problem at node B when it is acting as a sink node (step 406). To perform this check, diagnostic algorithm 400 invokes a sink node analyzer algorithm 470 shown in Fig. 28. If a problem is identified, the Workstation reports that there is a high probability that node B is causing a TCP problem when operating as a sink node and it reports the results of the investigation performed by algorithm 470 (step 408).

Note that source and sink nodes are concepts which apply to those dialogs for which a direction of the communication can be defined. For example, the source node may be the one which initiated the dialog for the purpose of sending data to the other node, i.e., the sink node.

If node B does not appear to be experiencing a TCP problem when acting as a sink node, diagnostic analyzer 322 checks for evidence of a TCP problem on the link between Node A and Node B (step 410). To perform this check, diagnostic algorithm 400 invokes a link analysis algorithm 550 shown in Fig. 29. If a problem is identified, the Workstation reports that there is a high probability that a TCP problem exists on the link and it reports the results of the investigation performed by link analysis algorithm 550 (step 412).

If the link does not appear to be experiencing a TCP problem, diagnostic analyzer 322 checks for evidence of a TCP problem at node B when it is acting as a source node (step 414). To perform this check, diagnostic algorithm 400 invokes the previously mentioned source algorithm 450 for Node B. If a problem is identified, the Workstation reports that there is a medium probability that node B is causing a TCP problem when operating as a source node and it reports the results of the investigation performed by algorithm 450 (step 416).

If node B does not appear to be experiencing a TCP problem when acting as a source node, diagnostic analyzer 322 checks for a TCP problem at node A when it is acting as a sink node (step 418). To perform this check, diagnostic algorithm 400 invokes sink node analyzer algorithm 470 for Node A. If a problem is identified, the Network Monitor reports that there is a medium probability that node A is causing a TCP problem when operating as a sink node and it reports the results of the investigation performed by algorithm 470 (step 420).

Finally, if node A does not appear to be experiencing a TCP problem when acting as a sink node, diagnostic analyzer 322 reports that it was not able to isolate the cause of a TCP problem (step 422).

The algorithms which are called from within the above-described diagnostic algorithm will now be described. Referring to Fig. 27, source node analyzer algorithm 450 checks whether a particular node is causing a TCP problem when operating as a source node. The strategy is as follows. To determine whether a TCP problem exists at this node which is the source node for the TCP connection, look at other connections for which this node is a source. If other TCP connections are okay, then there is probably not a problem with this node. This is an easy check with a high probability of being correct. If no other good connections exist, then look at the lower layers for possible reasons. Start at DLL and work up as problems at lower layers are more fundamental, i.e., they cause problems at higher layers whereas the reverse is not true.

In accordance with this approach, algorithm 450 first determines whether the node is acting as a source node in any other TCP connection and, if so, whether the other connection is okay (step 452). If the node is performing satisfactorily as a source node in another TCP connection, algorithm 450 reports that there is no problem at the source node and returns to diagnostic algorithm 400 (step 454). If algorithm 450 cannot identify any other TCP connections involving this node that are okay, it moves up through the protocol stack checking each level for a problem. In this case, it then checks for DLL problems at the node when it is acting as a source node by calling an DLL problem checking routine 510 (see Fig. 30) (step 456). If a DLL problem is found, that fact is reported (step 458). If no DLL problems are found, algorithm 450 checks for an IP problem at the node when it is acting as a source by calling an IP problem checking routine 490 (see Fig. 31) (step 460). If an IP problem is found, that fact is reported (step 462). If no IP problems are found, algorithm 450 checks whether any other TCP connection in which the node participates as a source is not okay (step 464). If another TCP connection involving the node exists and it is not okay, algorithm 450 reports a TCP problem at the node (step 466). If no other TCP connections where the node is acting as a source node can be found, algorithm 450 exits.

Referring to Fig. 28, sink node analyzer algorithm 470 checks whether a particular node is causing a TCP problem when operating as a sink node. It first determines whether the node is acting as a sink node in any other TCP connection and, if so, whether the other connection is okay (step 472). If the node is performing satisfactorily as a sink node in another TCP connection, algorithm 470 reports that there is no problem at the source node and returns to diagnostic algorithm 400 (step 474). If algorithm 470 cannot identify any other TCP connections involving this node that are okay, it then checks for DLL problems at the node when it is acting as a sink node by calling DLL problem checking routine 510 (step 476). If a DLL problem is found, that fact is reported (step 478). If no DLL problems are found, algorithm 470 checks for an IP problem at the node when it is acting as a sink by calling IP problem checking routine 490 (step 480). If an IP problem is found, that fact is reported (step 482). If no IP problems are found, algorithm 470 checks whether any other TCP connection in which the node participates as a sink is not okay (step 484). If another TCP connection involving the node as a sink exists and it is not okay, algorithm 470 reports a TCP problem at the node (step 486). If no other TCP connections where the node is acting as a sink node can be found, algorithm 470 exits.

Referring to Fig. 31, IP problem checking routine 490 checks for IP problems at a node. It does this by comparing the IP performance statistics for the node to the reference model (steps 492 and 494). If it detects any significant deviations from the reference model, it reports that there is an IP problem at the node (step 496). If no significant deviations are noted, it reports that there is no IP problem at the node (step 498).

As revealed by examining Fig. 30, DLL problem checking routine 510 operates in a similar manner to IP problem checking routine 490, with the exception that it examines a different set of parameters (i.e., DLL parameters) for significant deviations.

Referring the Fig. 29, link analysis logic 550 first determines whether any other TCP connection for the link is operating properly (step 552). If a properly operating TCP connection exists on the link, indicating that there is no link problem, link analysis logic 550 reports that the link is okay (step 554). If a properly operating TCP connection cannot be found, the link is decomposed into its constituent components and an IP link component problem checking routine 570 (see Fig. 32) is invoked for each of the link components (step 556). IP link component problem routine 570 evaluates the link component by checking the IP layer statistics for the relevant link component.

The decomposition of the link into its components arranges them in order of their distance from the source node and the analysis of the components proceeds in that order. Thus, for example, the link components which make up the link between nodes A and B include in order: segment S1, router R1, segment S2, router R2, and segment S3. The IP data for these various components are analyzed in the following order:

IP data for segment S1
IP data for address R1
IP data for source node to R1
IP data for S1 to S2
IP data for S2
IP data for address R2
IP data for S3
IP data for S2 to S3
IP data for S1 to S3

As shown in Fig. 32, IP link component problem checking routine 570 compares IP statistics for the link component to the reference model (step 572) to determine whether network performance deviates significantly from that specified by the model (step 574). If significant deviations are detected, routine 570 reports that there is an IP problem at the link component (step 576). Otherwise, it reports that it found no IP problem (step 578).

Referring back to Fig. 29, after completing the IP problem analysis for all of the link components, logic 550 then invokes a DLL link component problem checking routine 580 (see Fig. 33) for each link component to check its DLL statistics (step 558).

DLL link problem routine 580 is similar to IP link problem routine 570. As shown in Fig. 33, DLL link problem checking routine 580 compares DLL statistics for the link to the reference model (step 582) to determine whether network performance at the DLL deviates significantly from that specified by the model (step 584). If significant deviations are detected, routine 580 reports that there is a DLL problem at the link component (step 586). Otherwise, it reports that no DLL problems were found (step 588).

Referring back to Fig. 29, after completing the DLL problem analysis for all of the link components, logic 550 checks whether there is any other TCP on the link (step 560). If another TCP exists on the link (which implies that the other TCP is also not operating properly), logic 550 reports that there is a TCP problem on the link (step 562). Otherwise, logic 550 reports that there was not enough information from the existing packet traffic to determine whether there was a link problem (step 564)

If the analysis of the link components does not isolate the source of the problem and if there were components for which sufficient information was not available (due possibly to lack of traffic over through that component), the user may send test messages to those components to generate the information needed to evaluate its performance.

The reference model against which comparisons are made to detect and isolate malfunctions may be generated by examining the behavior of the network over an extended period of operation or over multiple periods of operation. During those periods of operation, average values and maximum excursions (or standard deviations) for observed statistics are computed. These values provide an initial estimate of a model of a properly functioning system. As more experience with the network is obtained and as more historical data on the various statistics is accumulated the thresholds for detecting actual malfunctions or imminent malfunctions and the reference model can be revised to reflect the new experience.

What constitutes a significant deviation from the reference model depends upon the particular parameter involved. Some parameters will not deviate from the expected norm and thus any deviation would be considered to be significant, for example, consider ICMP messages of type "destination unreachable," IP errors, TCP errors. Other parameters will normally vary within a wide range of acceptable values, and only if they move outside of that range should the deviation be considered significant. The acceptable ranges of variation can be determined by watching network performance over a sustained period of operation.

The parameters which tend to provide useful information for identifying and isolating problems at the node level for the different protocols and layers include the following.

<u>TCP</u>
error rate
header byte rate
packets retransmitted
bytes retransmitted
packets after window closed
bytes after window closed

<u>UDP</u>
error rate
header byte rate

<u>IP</u>
error rate
header byte rate
fragmentation rate
all ICMP messages of type destination

unreachable, parameter problem,
redirection

DLL
error rate
runts

For diagnosing network segment problems, the above-identified parameters are also useful with the addition of the alignment rate and the collision rate at the DLL. All or some subset of these parameters may be included among the set of parameters which are examined during the diagnostic procedure to detect and isolate network problems.

The above-described technique can be applied to a wide range of problems on the network, including among others, the following:

TCP Connection fails to establish
UDP Connection performs poorly
UDP not working at all
IP poor performance/high error rate
IP not working at all
DLL poor performance/high error rate
DLL not working at all

For each of these problems, the diagnostic approach would be similar to that described above, using, of course, different parameters to identify the potential problem and isolate its cause.

The Event Timing Module

Referring again to Fig. 5, the RTP is programmed to detect the occurrence of certain transactions for which timing information is desired. The transactions typically occur within a dialog at a particular layer of the protocol stack and they involve a first event (i.e., an initiating event) and a subsequent partner event or response. The events are protocol messages that arrive at the Network Monitor, are parsed by the RTP and then passed to Event Timing Module (ETM) for processing. A transaction of interest might be, for example, a read of a file on a server. In that case, the initiating event is the read request and the partner event is the read response. The time of interest is the time required to receive a response to the read request (i.e., the transaction time). The transaction time provides a useful measure of network performance and if measured at various times throughout the day under different load conditions gives a measure of how different loads affect network response times. The layer of the communicaton protocol at which the relevant dialog takes place will of course depend upon the nature of the event.

In general, when the RTP detects an event, it transfers control to the ETM which records an arrival time for the event. If the event is an initiating event, the ETM stores the arrival time in an event timing database 300 (see Fig. 34) for future use. If the event is a partner event, the ETM computes a difference between that arrival time and an earlier stored time for the initiating event to determine the complete transaction time.

Event timing database 300 is an array of records 302. Each record 302 includes a dialog field 304 for identifying the dialog over which the transactions of interest are occurring and it includes an entry type field 306 for identifying the event type of interest. Each record 302 also includes a start time field 308 for storing the arrival time of the initiating event and an average delay time field 310 for storing the computed average delay for the transactions. A more detailed description of the operation of the ETM follows.

Referring to Fig. 35, when the RTP detects the arrival of a packet of the type for which timing information is being kept, it passes control to the ETM along with relevant information from the packet, such as the dialog identifier and the event type (step 320). The ETM then determines whether it is to keep timing information for that particular event by checking the event timing database (step 322). Since each event type can have multiple occurrences (i.e., there can be multiple dialogs at a given layer), the dialog identifier is used to distinguish between events of the same type for different dialogs and to identify those for which information has been requested. All of the dialog/events of interest are identified in the event timing database. If the current dialog and event appear in the event timing database, indicating that the event should be timed, the ETM determines whether the event is a starting event or an ending event so that it may be processed properly (step 324). For certain events, the absence of a start time in the entry field of the appropriate record 302 in event timing database 300 is one indicator that the event represents a start time; otherwise, it is an end time event. For other events, the ETM determines if the start time is to be set by the event type as specified in the packet being parsed. For example, if the event is a file read a start time is stored. If the event is the read completion it represents an end time. In general, each protocol event will have its own intrinsic meaning for how to determine start and end times.

Note that the arrival time is only an estimate of the actual arrival time due to possible queuing and other processing delays. Nevertheless, the delays are generally so small in comparison to the transaction times being measured that they are of little consequence.

In step 324, if the event represents a start time, the ETM gets the current time from the kernal and stores it in start time field 308 of the appropriate record in event timing database 300 (step 326). If the event represents an end time event, the ETM obtains the current time from the kernel and computes a difference between that time and the corresponding start time found in event timing database 300 (step 328). This represents the total time for the transaction of interest. It is combined with the stored average transaction time to compute a new running average transaction time for that event (step 330).

Any one of many different methods can be used to compute the running average transaction time. For example, the following formula can be used:

New Avg. = [(5 * Stored Avg.) + Transaction Time]/6.

After six transaction have been timed, the computed new average becomes a running average for the transaction times. The ETM stores this computed average in the appropriate record of event timing database 300, replacing the previous average transaction time stored in that record, and it clears start time entry field 308 for that record in preparation for timing the next transaction.

After processing the event in steps 322, 326, and 330, the ETM checks the age of all of the start time entries in the event timing database 300 to determine if any of them are too "old" (step 332). If the difference between the current time and any of the start times exceeds a preselected threshold, indicating that a partner event has not occurred within a reasonable period of time, the ETM deletes the old start time entry for that dialog/event (step 334). This insures that a missed packet for a partner event does not result in an erroneously large transaction time which throws off the running average for that event.

If the average transaction time increases beyond a preselected threshold set for timing events, an alarm is sent to the Workstation.

Two examples will now be described to illustrate the operation of the ETM for specific event types. In the first example, Node A of Fig. 25 is communicating with Node B using the NFS protocol. Node A is the client while Node B is the server. The Network Monitor resides on the same segment as node A, but this is not a requirement. When Node A issues a read request to Node B, the Network Monitor sees the request and the RTP within the Network Monitor transfers control to the ETM. Since it is a read, the ETM stores a start time in the Event Timing Database. Thus, the start time is the time at which the read was initiated.

After some delay, caused by the transmission delays of getting the read message to node B, node B performs the read and sends a response back to node A. After some further transmission delays in returning the read response, the Network Monitor receives the second packet for the event. At the time, the ETM recognizes that the event is an end time event and updates the average transaction time entry in the appropriate record with a new computed running average. The ETM then compares the average transaction time with the threshold for this event and if it has been exceeded, issues an alarm to the Workstation.

In the second example, node A is communicating with Node B using the Telnet protocol. Telnet is a virtual terminal protocol. The events of interest take place long after the initial connection has been established. Node A is typing at a standard ASCII (VT100 class) terminal which is logically (through the network) connected to Node B. Node B has an application which is receiving the characters being typed on Node A and, at appropriate times, indicated by the logic of the applications, sends characters back to the terminal located on Node A. Thus, every time node A sends characters to B, the Network Monitor sees the transmission.

In this case, there are several transaction times which could provide useful network performance information. They include, for example, the amount of time it takes to echo characters typed at the keyboard through the network and back to the display screen, the delay between typing an end of line command and seeing the completion of the application event come back or the network delays incurred in sending a packet and receiving acknowledgment for when it was received.

In this example, the particular time being measured is the time it takes for the network to send a packet and receive an acknowledgement that the packet has arrived. Since Telnet runs on top of TCP, which in turn runs on top of IP, the Network Monitor monitors the TCP acknowledge end-to-end time delays.

Note that this is a design choice of the implementation and that all events visible to the Network Monitor by virtue of the fact that information is in the packet could be measured.

When Node A transmits a data packet to Node B, the Network Monitor receives the packet. The RTP recognizes the packet as being part of a timed transaction and passes control to the ETM. The ETM recognizes it as a start time event, stores the start time in the event timing database and returns control to the RTP after checking for aging.

When Node B receives the data packet from Node A, it sends back an acknowledgment packet. When the Network Monitor sees that packet, it delivers the event to the ETM, which recognizes it as an end time event. The ETM calculates the delay time for the complete transaction and uses that to update the average transaction time. The ETM then compares the new average transaction time with the threshold for this event. If it has been exceeded, the ETM issues an alarm to the Workstation.

Note that this example is measuring something very different than the previous example. The first example measures the time it takes to traverse the network, perform an action and return that result to the requesting node. It measures performance as seen by the user and it includes delay times from the network as well as delay times from the File Server.

The second example is measuring network delays without looking at the service delays. That is, the ETM is measuring the amount of time it takes to send a packet to a node and receive the acknowledgement of the receipt of the message. In this example, the ETM is measuring transmissions delays as well as processing delays associated with network traffic, but not anything having to do with non-network processing.

As can be seen from the above examples, the ETM can measure a broad range of events. Each of these events can be measured passively and without the cooperation of the nodes that are actually participating in the transmission.

The Address Tracker Module (ATM)

Address tracker module (ATM) 43, one of the software modules in the Network Monitor (see Fig. 5), operates on networks on which the node addresses for particular node to node connections are assigned dynamically. An Appletalk® Network, developed by Apple Computer Company, is an example of a network which uses dynamic node addressing. In such networks, the dynamic change in the address of a particular service causes difficulty troubleshooting the network because the network manager may not know where the various nodes are and what they are called. In addition, foreign network addresses (e.g., the IP addresses used by that node for communication over an IP network to which if is connected) can not be relied upon to point to a particular node. ATM 43 solves this problem by passively monitoring the network traffic and collecting a table showing the node address to node name mappings.

In the following description, the network on which the Monitor is located is assumed to be an Appletalk® Network. Thus, as background for the following discussion, the manner in which the dynamic node addressing mechanism operates on that network will first be described.

When a node is activated on the Appletalk® Network, it establishes its own node address in accordance with protocol referred to as the Local Link Access Protocol (LLAP). That is, the node guesses its own node address and then verifies that no other node on the network is using that address. The node verifies the uniqueness of its guess by sending an LLAP Enquiry control packet informing all other nodes on the network that it is going to assign itself a particular address unless another node responds that the address has already been assigned. If no other node claims that address as its own by sending an LLAP acknowledgment control packet, the first node uses the address which it has selected. If another node claims the address as its own, the first node tries another address. This continues until, the node finds an unused address.

When the first node wants to communicate with a second node, it must determine the dynamically assigned node address of the second node. It does this in accordance with another protocol referred to as the Name Binding Protocol (NBP). The Name Binding Protocol is used to map or bind human understandable node names with machine understandable node addresses. The NBP allows nodes to dynamically translate a string of characters (i.e., a node name) into a node address. The node needing to communicate with another node broadcasts an NBP Lookup packet containing the name for which a node address is being requested. The node having the name being requested responds with its address and returns a Lookup Reply packet containing its address to the original requesting node. The first node then uses that address its current communications with the second node.

Referring to Fig. 36, the network includes an Appletalk® Network segment 702 and a TCP/IP segment 704, each of which are connected to a larger network 706 through their respective gateways 708. A Monitor 710, including a Real Time Parser (RTP) 712 and an Address Tracking Module (ATM) 714, is located on Appletalk network segment 702 along with other nodes 711. A Management Workstation 716 is located on segment 704. It is assumed that Monitor 710 has the features and capabilities previously described; therefore, those features not specifically related to the dynamic node addressing capability will not be repeated here but rather the reader is referred to the earlier discussion. Suffice it to say that Monitor 710 is, of course, adapted to operate on Appletalk Network segment 702, to parse and analyze the packets which are transmitted over that segment according to the Appletalk® family of protocols and to communicate the information which it extracts from the network to Management Workstation 716 located on segment 704.

Within Monitor 710, ATM 714 maintains a name table data structure 730 such as is shown in Fig. 37. Name Table 720 includes records 722, each of which has a node name field 724, a node address field 726, an IP address field 728, and a time field 729. ATM 714 uses Name Table 720 to keep track of the mappings of node names to node address and to IP address. The relevance of each of the fields of records 722 in Name Table 720 are explained in the following description of how ATM 714 operates.

In general, Monitor 710 operates as previously described. That is, it passively monitors all packet traffic over segment 702 and sends all packets to RTP 712 for parsing. When RTP 712 recognizes an Appletalk packet, it transfers control to ATM 714 which analyzes the packet for the presence of address mapping information.

The operation of ATM 714 is shown in greater detail in the flow diagram of Fig. 38. When ATM 714 receives control from RTP 712, it takes the packet (step 730 and strips off the lower layers of the protocol until it determines whether there is a Name Binding Protocol message inside the packet (step 732). If it is a NBP message, ATM 714 then determines whether it is new name Lookup message (step 734). If it is a new name Lookup message, ATM 714 extracts the name from the message (i.e., the name for which a node address is being requested) and adds the name to the node name field 724 of a record 722 in Name Table 720 (step 736).

If the message is an NBP message but it is not a Lookup message, ATM 714 determines whether it is a Lookup Reply (step 738). If it is a Lookup Reply, signifying that it contains a node name/node address binding, ATM 714 extracts the name and the assigned node address from the message and adds this information to Name Table 720. ATM 714 does this by searching the name fields of records 722 in Name Table 720 until it locates the name. Then, it updates the node address field of the identified record to contain the node address which was extracted from the received NBP packet. ATM 714 also updates time field 729 to record the time at which the message was processed.

After ATM 714 has updated the address field of the appropriate record, it determines whether any records 722 in Name Table 720 should be aged out (step 742). ATM 714 compares the current time to the times recorded in the time fields. If the elapsed time is greater than a preselected time period (e.g. 48 hours), ATM 714 clears the record of all information (step 744). After that, it awaits the next packet from RTP 712.

As ATM 714 is processing each a packet and it determines either that it does not contain an NBP message (step 732) or it does not contain a Lookup Reply message (step 738), ATM 714 branches to step 742 to perform the age out check before going on to the next packet from RTP 712.

The Appletalk to IP gateways provide services that allow an Appletalk Node to dynamically connect to an IP address for communicating with IP nodes. This service extends the dynamic node address mechanism to the IP world for all Appletalk nodes. While the flexibility provided is helpful to the users, the network manager is faced with the problem of not knowing which Appletalk Nodes are currently using a particular IP address and thus, they can not easily track down problems created by the particular node.

ATM 714 can use passive monitoring of the IP address assignment mechanisms to provide the network manager a Name-to-IP address mapping.

If ATM 714 is also keeping IP address information, it implements the additional steps shown in Fig. 39 after completing the node name to node address mapping steps. ATM 714 again checks whether it is an NBP message (step 748). If it is an NBP message, ATM 714 checks whether it is a response to an IP address request (step 750). IP address requests are typically implied by an NBP Lookup request for an IP gateway. The gateway responds by supplying the gateway address as well as an IP address that is assigned to the requesting node. If the NBP message is an IP address response, ATM 714 looks up the requesting node in Name Table 720 (step 752) and stores the IP address assignment in the IP address field of the appropriate record 722 (step 754).

After storing the IP address assignment information, ATM 714 locates all other records 722 in Name Table 720 which contain that IP address. Since the IP address has been assigned to a new node name, those old entries are no longer valid and must be eliminated. Therefore, ATM 714 purges the IP address fields of those records (step 756). After doing this cleanup step, ATM 714 returns control to RTP 712.

Other embodiments are within the following claims. For example, the Network Monitor can be adapted to identify node types by analyzing the type of packet traffic to or from the node. If the node being monitored is receiving mount requests, the Monitor would report that the node is behaving like node a file server. If the node is issuing routing requests, the Monitor would report that the node is behaving like a router. In either case, the network manager can check a table of what nodes are permitted to provide what functions to determine whether the node is authorized to function as either a file server or a router, and if not, can take appropriate action to correct the problem.

APPENDIX I

SNMP MIB Subset Supported

This is the subset of the standard MIB which can be obtained by monitoring.

Refer to RFC 1066 Management Information Base for an explanation on the items which follow.

System group:
none

Interfaces group
ifType
ifPhysAddress
ifOperStatus
ifInOctets
ifInUcastPkts
ifInNUcastPkts
ifOutOctets
ifOutUcastPkts
ifOutNUcastPkts

Address Translation group
none

IP group
ipForwarding
ipDefaultTTL
ipInReceives
ipInHdrErrors
ipInAddrErrors
ipForwDatagrams
ipReasmReqds
ipFragCreates

IP Address Table
ipAddress
ipAdEntBcastAddr

IP Routing Table
none

ICMP group
icmpInMsgs
icmplnErrors
icmpInDestUnreachs
icmpInTimeExcds
icmpInParmProbs
icmpInSrcQuenchs
icmpInRedirects
icmpInEchoes

icmpInEchoReps
icmpInTimestamps
icmpInTimestampReps
icmpInAddrMasks
icmpInAddrmaskReps
icmpOutMsgs
imcpOutDestrUnreachs
icmpOutTimeExcds
icmpOutParmProbs
icmpOutSrcQuenchs
icmpOutRedirects
icmpOutEchoes
icmpOutEchoReps
icmpOutTimestamps
icmpOutTimestampReps
icmpOutAddrMasks
icmpOutAddrmaskReps

TCP group
tcpActiveOpens
tcpPassiveOpens
tcpAttempFails
tcpEstabResets
tcpCurrEstab
tcpInSegs
tcpOutSegs
tcpRetransSegs
tcpConnTable

UDP group
udpInDatagrams
udpInErrors
udpOutDatagrams
udpOutErrors

EGP group
egpInMsgs
egpInErrors
egpOutMsgs
egpOutErrors

APPENDIX II

MIB Definitions for Network Monitor

1. Common MIB Definitions

Definitions

MIB_BUCKETS_PER_RATE	12
MIB_PROTOCOLS_PER_DIALOG	10
MibBucketsPerRate	12
MibProtocolsPerDialog	10
MIB_MAX_PROTOCOL	10
MIB_MAX_MOST_ACTIVE	5
MIB_MAX_DIALOG	3

Structures Used

```
typedef struct {
    Byte                year
    Byte                month
    Byte                date
    Byte                day
    Byte                hour
    Byte                minute
    Byte                second
    Byte                unused
} MibTimeOfDay


typedef struct mib_count32_type {
Uint32        accum         ( Long term accum. count)
Uint32        current       ( Present running count)
Uint32        highThld
} MibCount32


typedef struct mib_count64_type {
Uint64        accum         ( Long term accum. count)
Uint64        current       ( Present running count)
Uint64        highThld
} MibCount64

typedef struct mib_meter_type {
Uint32                   current
Uint32                   high
Uint32                   low
Uint32                   highThld
} MibMeter
typedef struct mib_average_meter_type {
Uint32                   current
Uint32                    high
Uint32                    low
Uint32                    highThld
} MibAverageMeter


typedef struct mib_percent_type {
     Uint32                    current
     Uint32                    high
     Uint32                    low
     Uint32                    highThld
} MibPercent


typedef struct mib_rolling_rate_type {
     Uint32                    current
     Uint32                    high
     Uint32                    low
     Uint32                    highThld
} MibRollingRate


typedef MibRollingRate MibRatePerS
typedef MibRollingRate MibRatePerH

typedef Uint32 MibShortRatePerS
typedef Uint32 MibShortRatePerM


typedef struct mib_short_count32_type {
Uint32          current          ( Present running count)
Uint32          accum            ( Long term accum. count)
} MibShortCount32


typedef struct mib_bucket_rate_type {
Uint32          current          ( Present rate)
Uint32          rates[MIB_BUCKETS_PER_RATE]( 12 5 minute
count buckets )
Uint32          maxRates[MIB_BUCKETS_PER_RATE]( 12 5-min.
max
rate buckets )
} MibBucketRate
```

Most Active Table Definitions

```
typedef struct mib_most_active_entry_type {
     MibAddress                address
     MibCount32                    packetCount
     MibRatePerS                   packetRate
} MibMostActiveEntry


typedef struct mib_most_active_table_type {
Uint32                         numEntries
Uint32                         nextEntry
MibMostActiveEntry  mostActiveEntry[MIB_MAX_MOST_ACTIVE]
     } MibMostActiveTable
```

Protocol Table Definitions

```
typedef struct mib_protocol_entry_type {
     Uint32                        protocol
     MibCount32                    packetCount
     MibRatePerS                   packetRate
} MibProtocolEntry


typedef struct mib_protocol_table_type {
     Uint32                        numEntries
     Uint32                        nextEntry
     MibProtocolEntry    protocolEntry[MIB_MAX_PROTOCOL]
     } MibProtocolTable
```

Dialog Table Definitions

```
typedef struct mib_transport_type {
     Uint32              transportProtocol
     Uint32              applicationProtocol
     Uint32              initiator
     Uint32              connectionRetries
     Uint32              addr1_window
     Uint32              addr2_window
     Uint32              state
     Uint32              closeReason
     } MibTransportType


typedef struct mib_dialog_entry_type {
MibAddress                     addresses
Uint32                         protocolEntries
Uint32
protocols[MIB_PROTOCOLS_PER_DIALOG]
MibTimeOfDay                   gmt
Uint32                         startTime
Uint32                         lastTime
Uint32                         alarmsSent
MibCount32                     packets
MibRatePerS                    packetRate
MibCount32                  bytes
MibRatePerS                 byteRate
MibCount32                  errors
MibRatePerS                 errorRate
MibCount32                  fragments
MibRatePerS                 fragmentRate
MibCount32                  rexmits
MibRatePerS                 rexmitRate
MibCount32                  flowCtrls
MibRatePerS                 flowCtrlRate
MibTransportType            transport
} MibDialogEntry
```

Values for the initiator field

```
 ConnectionInitiatorUnknown  0
 ConnectionInitiatorAddr1     1
 ConnectionInitiatorAddr2     2
```

Values for the connectionCloseReason field

```
 ConnectionCloseUnknown       0
 ConnectionCloseFin           1
 ConnectionCloseRst           2
```

Values for the connectionState field

```
 ConnectionStateUnknown       0
 ConnectionStateConnecting    1
 ConnectionStateData         2
 ConnectionStateClosing       3
 ConnectionStateClosed        4
```

```
typedef struct mib_dialog_table_type {
     Uint32                          numEntries
     Uint32                          nextEntry
     MibDialogEntry             dialogEntry[MIB_MAX_DIALOG]

     } MibDialogTable
```

2. Data link layer mib definitions for Network Monitor mib.

2.1 dll Segment -Summary Tool

```
typedef struct {
     MibShortCount32          frames
     MibBucketRate       frameRate
     MibShortCount32          bytes
     MibBucketRate       byteRate
     MibShortCount32          errors
     MibBucketRate       errorRate
     Uint32                   protocolCount
     Uint32                   mostActiveCount
     Uint32                   pairCount
     MibShortCount32          rcvOffSegs
     MibBucketRate       rcvOffSegRate
     MibShortCount32          xmtOffSegs
     MibBucketRate       xmtOffSegRate
     MibShortCount32          transits
     MibBucketRate       transitRate
     MibShortCount32          bcasts
     MibBucketRate       bcastRate
     MibShortCount32          mcasts
     MibBucketRate       mcastRate
     MibShortCount32          collisions
     MibShortRatePerS    collisionRate
     MibShortCount32          alignmtErrors
     MibShortRatePerS    alignmtErrorRate
} MibDllSegSumStats
```

2.2 dll Segment -Values Tool

```
typedef struct {
     MibCount32               frames
     MibRatePerS              frameRate
     MibCount32               bytes
     MibRatePerS              byteRate
     MibCount32               errors
     MibRatePerS              errorRate
     MibCount32               rcvOffSegs
     MibRatePerS              rcvOffSegRate
     MibCount32               xmtOffSegs
     MibRatePerS              xmtOffSegRate
     MibCount32               transits
     MibRatePerS              transitRate
     MibCount32               bcasts
     MibRatePerS              bcastRate
     MibCount32               mcasts
     MibRatePerS              mcastRate
     MibCount32               collisions
     MibRatePerS              collisionRate
     MibCount32               alignmtErrors
     MibRatePerS              alignmtErrorRate
     MibCount32               enetFrames
     MibRatePerS              enetFrameRate
     MibCount32               llcFrames
     MibRatePerS              llcFrameRate
     MibCount32               runtFrames
     MibRatePerS              runtFrameRate
} MibDllSegValStats
```

2.3 dll Address - Summary Tool

```
typedef struct {
     MibShortCount32          frames
     MibBucketRate       frameRate
     MibShortCount32          bytes
     MibBucketRate       byteRate
     MibShortCount32          errors
     MibBucketRate       errorRate
     Uint32                   protocolCount
     Uint32                   mostActiveCount
     Uint32                   pairCount
     MibShortCount32          rcvOffSegs
     MibBucketRate       rcvOffSegRate
     MibShortCount32          xmtOffSegs
     MibBucketRate       xmtOffSegRate
     MibShortCount32          xmtBcasts
     MibBucketRate       xmtBcastRate
     MibShortCount32          xmtMcasts
     MibBucketRate       xmtMcastRate
} MibDllAddrSumStats
```

2.4 dll Address- Values Tool

```
typedef struct {
     MibCount32               rcvFrames
     MibRatePerS              rcvFrameRate
     MibCount32               rcvBytes
     MibRatePerS              rcvByteRate
     MibCount32               rcvErrors
     MibRatePerS              rcvErrorRate
     MibCount32               xmtFrames
     MibRatePerS              xmtFrameRate
     MibCount32               xmtBytes
     MibRatePerS              xmtByteRate
     MibCount32               xmtErrors
     MibRatePerS              xmtErrorRate
     MibCount32               xmtBcasts
     MibRatePerS              xmtBcastRate
     MibCount32               xmtMcasts
     MibRatePerS              xmtMcastRate
     MibCount32               rcvOffSegs
     MibRatePerS              rcvOffSegRate
     MibCount32               xmtOffSegs
     MibRatePerS              xmtOffSegRate
     MibCount32               enetFrames
     MibRatePerS              enetFrameRate
     MibCount32               llcFrames
     MibRatePerS              llcFrameRate
     MibCount32                runtFrames
     MibRatePerS               runtFrameRate
} MibDllAddrValStats
```

3. IP layer mib definitions for Network Monitor mib.

3.1 ip Segment - Summary Tool

```
typedef struct {
     MibShortCount32                pkts
     MibBucketRate             pktRate
     MibShortCount32                bytes
     MibBucketRate             byteRate
     MibShortCount32                errors
     MibBucketRate             errorRate
     Uint32                         protocolCount
     Uint32                         mostActiveCount
     Uint32                         pairCount
     MibShortCount32                rcvOffSegs
     MibBucketRate             rcvOffSegRate
     MibShortCount32                xmtOffSegs
     MibBucketRate             xmtOffSegRate


     MibShortCount32                transits
     MibBucketRate             transitRate
     MibShortCount32                flowCtrls
     MibBucketRate             flowCtrlRate
     MibShortCount32                bcasts
     MibBucketRate             bcastRate
     MibShortCount32                mcasts
     MibBucketRate             mcastRate
     MibShortCount32                frgmts
     MibBucketRate             frgmtRate
} MibIpSegSumStats
```

3.2 ip Segment - Values Tool

```
typedef struct {
     MibCount32                pkts
     MibRatePerS               pktRate
     MibCount32                bytes
     MibRatePerS               byteRate
     MibCount32                errors
     MibRatePerS               errorRate
     MibCount32                rcvOffSegs
     MibRatePerS               rcvOffSegRate
     MibCount32                xmtOffSegs
     MibRatePerS               xmtOffSegRate
     MibCount32                transits
     MibRatePerS               transitRate
     MibCount32               bcasts
     MibRatePerS              bcastRate
     MibCount32               mcasts
     MibRatePerS              mcastRate
     MibCount32               hdrBytes
     MibRatePerS              hdrByteRate
     MibCount32               frgmts
     MibRatePerS              frgmtRate
} MibIpSegValStats
```

3.3 ip Address - Summary Tool

```
typedef struct {
     MibShortCount32          pkts
     MibBucketRate       pktRate
     MibShortCount32          bytes
     MibBucketRate       byteRate
     MibShortCount32          errors
     MibBucketRate       errorRate
     Uint32                   protocolCount
     Uint32                   mostActiveCount
     Uint32                   pairCount
     MibShortCount32          rcvOffSegs
     MibBucketRate       rcvOffSegRate
     MibShortCount32          xmtOffSegs
     MibBucketRate       xmtOffSegRate
     MibShortCount32          flowCtrls
     MibBucketRate       flowCtrlRate
     MibShortCount32          frgmts
     MibBucketRate       frgmtRate
     MibShortCount32          xmtBcasts
     MibBucketRate       xmtBcastRate
     MibShortCount32          xmtMcasts
     MibBucketRate       xmtMcastRate
} MibIpAddrSumStats
```

3.4 ip Address - Values Tool

```
typedef struct {
     MibCount32               rcvPkts
     MibRatePerS              rcvPktRate
     MibCount32               rcvBytes
     MibRatePerS              rcvByteRate
     MibCount32               rcvErrors
     MibRatePerS              rcvErrorRate
     MibCount32               xmtPkts
     MibRatePerS              xmtPktRate
     MibCount32               xmtBytes
     MibRatePerS              xmtByteRate
     MibCount32               xmtErrors
     MibRatePerS              xmtErrorRate
     MibCount32               rcvHdrBytes
     MibRatePerS              rcvHdrByteRate
     MibCount32                    xmtHdrBytes
     MibRatePerS                   xmtHdrByteRate
     MibCount32                    rcvFrgmts
     MibRatePerS                   rcvFrgmtRate
     MibCount32                    xmtFrgmts
     MibRatePerS                   xmtFrgmtRate
     MibCount32                    xmtBcasts
     MibRatePerS                   xmtBcastRate
     MibCount32                    xmtMcasts
     MibRatePerS                   xmtMcastRate
     MibCount32                    rcvOffSegs
     MibRatePerS                   rcvOffSegRate
     MibCount32                    xmtOffSegs
     MibRatePerS                   xmtOffSegRate
} MibIpAddrValStats
```

4. ICMP layer mib definitions for Network Monitor mib.

4.1 icmp Segment - Summary Tool

```
typedef struct {
     MibShortCount32               pkts
     MibBucketRate            pktRate

     MibShortCount32               bytes
     MibBucketRate            byteRate

     MibShortCount32               errors
     MibBucketRate            errorRate

     Uint32                        mostActiveCount
     Uint32                        pairCount

     MibShortCount32               rcvOffSegs
     MibBucketRate            rcvOffSegRate
     MibShortCount32               xmtOffSegs
     MibBucketRate            xmtOffSegRate
     MibShortCount32               transits
     MibBucketRate            transitRate

     MibShortCount32               echoReq
     MibShortCount32               echoReply
     MibShortCount32               destUnr
     MibShortCount32               srcQuench
     MibShortCount32               redir
     MibShortCount32               timeExceeded
     MibShortCount32               paramProblem
     MibShortCount32               timestampReq
     MibShortCount32               timestampReply
     MibShortCount32               addrMaskReq
     MibShortCount32               addrMaskReply
} MibIcmpSegSumStats
```

4.2 icmp Segment - Values Tool

```
typedef struct {
     MibCount32               pkts
     MibRatePerS              pktRate

     MibCount32               bytes
     MibRatePerS              byteRate

     MibCount32               errors
     MibRatePerS              errorRate

     MibCount32               rcvOffSegs
     MibRatePerS              rcvOffSegRate
     MibCount32               xmtOffSegs
     MibRatePerS              xmtOffSegRate
     MibCount32               transits
     MibRatePerS              transitRate

     MibCount32               echoReq
     MibRatePerS              echoReqRate
     MibCount32               echoReply
     MibRatePerS              echoReplyRate

     MibCount32               destUnrNet
     MibRatePerS              destUnrNetRate
     MibCount32               destUnrHost
     MibRatePerS              destUnrHostRate
     MibCount32               destUnrProtocol
     MibRatePerS              destUnrProtocolRate
     MibCount32               destUnrPort
     MibRatePerS              destUnrPortRate
     MibCount32               destUnrFrgmt
     MibRatePerS              destUnrFrgmtRate
     MibCount32               destUnrSrcRoute
     MibRatePerS              destUnrSrcRouteRate
     MibCount32               destUnrNetUnknown
     MibRatePerS              destUnrNetUnknownRate
     MibCount32               destUnrHostUnknown
     MibRatePerS              destUnrHostUnknownRate
     MibCount32               destUnrSrcHostIsolated

     MibRatePerS              destUnrSrcHostIsolatedRate
     MibCount32               destUnrNetProhibited

     MibRatePerS              destUnrNetProhibitedRate
     MibCount32               destUnrHostProhibited
     MibRatePerS              destUnrHostProhibitedRate
     MibCount32               destUnrNetTos
     MibRatePerS              destUnrNetTosRate
     MibCount32               destUnrHostTos
```

```
        MibRatePerS                 destUnrHostTosRate

        MibCount32                  srcQuench
        MibRatePerS                 srcQuenchRate

        MibCount32                  redirNet
        MibRatePerS                 redirNetRate
        MibCount32                  redirHost
        MibRatePerS                 redirHostRate
        MibCount32                  redirNetTos
        MibRatePerS                 redirNetTosRate
        MibCount32                  redirHostTos
        MibRatePerS                 redirHostTosRate

        MibCount32                  timeExceededInTransit
        MibRatePerS                 timeExceededInTransitRate
        MibCount32                  timeExceededInReass
        MibRatePerS                 timeExceededInReassRate

        MibCount32                  paramProblem
        MibRatePerS                 paramProblemRate
        MibCount32                  paramProblemOption
        MibRatePerS                 paramProblemOptionRate

        MibCount32                  timestampReq
        MibRatePerS                 timestampReqRate
        MibCount32                  timestampReply
        MibRatePerS                 timestampReplyRate

        MibCount32                  addrMaskReq
        MibRatePerS                 addrMaskReqRate
        MibCount32                  addrMaskReply
        MibRatePerS                 addrMaskReplyRate

} MibIcmpSegValStats
```

4.3 icmp Address - Summary Tool

```
typedef struct {
     MibShortCount32          pkts
     MibBucketRate       pktRate

     MibShortCount32          bytes
     MibBucketRate       byteRate

     MibShortCount32          errors
     MibBucketRate       errorRate
     Uint32                   mostActiveCount
     Uint32                   pairCount

     MibShortCount32          rcvOffSegs
     MibBucketRate       rcvOffSegRate
     MibShortCount32          xmtOffSegs
     MibBucketRate        xmtOffSegRate

     MibShortCount32          echoReq
     MibShortCount32          echoReply
     MibShortCount32          destUnr
     MibShortCount32          srcQuench
     MibShortCount32          redir
     MibShortCount32          paramProblem
     MibShortCount32          timeExceeded
     MibShortCount32          timestampReq
     MibShortCount32          timestampReply
     MibShortCount32          addrMaskReq
     MibShortCount32          addrMaskReply
} MibIcmpAddrSumStats
```

4.4 icmp Address- Values Tool

```
typedef struct {

     MibCount32           rcvPkts
     MibRatePerS          rcvPktRate
     MibCount32           rcvBytes
     MibRatePerS          rcvByteRate
     MibCount32           rcvErrors
     MibRatePerS          rcvErrorRate

     MibCount32           xmtPkts
     MibRatePerS          xmtPktRate
     MibCount32           xmtBytes
     MibRatePerS          xmtByteRate
     MibCount32           xmtErrors
     MibRatePerS          xmtErrorRate

     MibCount32           rcvOffSegs
     MibRatePerS          rcvOffSegRate
     MibCount32           xmtOffSegs
     MibRatePerS          xmtOffSegRate

     MibCount32           rcvDestUnrNet
     MibRatePerS          rcvDestUnrNetRate
     MibCount32           rcvDestUnrHost
     MibRatePerS          rcvDestUnrHostRate
     MibCount32           rcvDestUnrProtocol
     MibRatePerS          rcvDestUnrProtocolRate
     MibCount32           rcvDestUnrPort
     MibRatePerS          rcvDestUnrPortRate
     MibCount32           rcvDestUnrFrgmt
     MibRatePerS          rcvDestUnrFrgmtRate
     MibCount32           rcvDestUnrSrcRoute
     MibRatePerS          rcvDestUnrSrcRouteRate
     MibCount32           rcvDestUnrNetUnknown
```

MibRatePerS	rcvDestUnrNetUnknownRate
MibCount32	rcvDestUnrHostUnknown
MibRatePerS	rcvDestUnrHostUnknownRate
MibCount32	rcvDestUnrSrcHostIsolated
MibRatePerS	rcvDestUnrSrcHostIsolatedRate
MibCount32	rcvDestUnrNetProhibited
MibRatePerS	rcvDestUnrNetProhibitedRate
MibCount32	rcvDestUnrHostProhibited
MibRatePerS	rcvDestUnrHostProhibitedRate
MibCount32	rcvDestUnrNetTos
MibRatePerS	rcvDestUnrNetTosRate
MibCount32	rcvDestUnrHostTos
MibRatePerS	rcvDestUnrHostTosRate
MibCount32	rcvTimeExceededInTransit
MibRatePerS	rcvTimeExceededInTransitRate
MibCount32	rcvTimeExceededInReass
MibRatePerS	rcvTimeExceededInReassRate
MibCount32	rcvParamProblem
MibRatePerS	rcvParamProblemRate
MibCount32	rcvParamProblemOption
MibRatePerS	rcvParamProblemOptionRate
MibCount32	rcvSrcQuench
MibRatePerS	rcvSrcQuenchRate
MibCount32	rcvRedirNet
MibRatePerS	rcvRedirNetRate
MibCount32	rcvRedirHost
MibRatePerS	rcvRedirHostRate
MibCount32	rcvRedirNetTos
MibRatePerS	rcvRedirNetTosRate
MibCount32	rcvRedirHostTos
MibRatePerS	rcvRedirHostTosRate
MibCount32	rcvEchoReq
MibRatePerS	rcvEchoReqRate
MibCount32	rcvEchoReply
MibRatePerS	rcvEchoReplyRate
MibCount32	rcvTimestampReq
MibRatePerS	rcvTimestampReqRate
MibCount32	rcvTimestampReply
MibRatePerS	rcvTimestampReplyRate
MibCount32	rcvAddrMaskReq
MibRatePerS	rcvAddrMaskReqRate
MibCount32	rcvAddrMaskReply
MibRatePerS	rcvAddrMaskReplyRate

```
MibCount32          xmtDestUnrNet
MibRatePerS         xmtDestUnrNetRate
MibCount32          xmtDestUnrHost
MibRatePerS         xmtDestUnrHostRate
MibCount32          xmtDestUnrProtocol
MibRatePerS         xmtDestUnrProtocolRate
MibCount32          xmtDestUnrPort
MibRatePerS         xmtDestUnrPortRate
MibCount32          xmtDestUnrFrgmt
MibRatePerS         xmtDestUnrFrgmtRate
MibCount32          xmtDestUnrSrcRoute
MibRatePerS         xmtDestUnrSrcRouteRate
MibCount32          xmtDestUnrNetUnknown
MibRatePerS         xmtDestUnrNetUnknownRate
MibCount32          xmtDestUnrHostUnknown
MibRatePerS         xmtDestUnrHostUnknownRate
MibCount32          xmtDestUnrSrcHostIsolated

MibRatePerS         xmtDestUnrSrcHostIsolatedRate
MibCount32          xmtDestUnrNetProhibited
MibRatePerS         xmtDestUnrNetProhibitedRate
MibCount32          xmtDestUnrHostProhibited
MibRatePerS         xmtDestUnrHostProhibitedRate
MibCount32          xmtDestUnrNetTos
MibRatePerS         xmtDestUnrNetTosRate
MibCount32          xmtDestUnrHostTos
MibRatePerS         xmtDestUnrHostTosRate

MibCount32          xmtTimeExceededInTransit
MibRatePerS         xmtTimeExceededInTransitRate
MibCount32          xmtTimeExceededInReass
MibRatePerS         xmtTimeExceededInReassRate

MibCount32          xmtParamProblem
MibRatePerS         xmtParamProblemRate
MibCount32          xmtParamProblemOption
MibRatePerS         xmtParamProblemOptionRate

MibCount32          xmtSrcQuench
MibRatePerS         xmtSrcQuenchRate

MibCount32          xmtRedirNet
MibRatePerS         xmtRedirNetRate
MibCount32          xmtRedirHost
MibRatePerS         xmtRedirHostRate
MibCount32          xmtRedirNetTos
MibRatePerS         xmtRedirNetTosRate
MibCount32          xmtRedirHostTos
MibRatePerS         xmtRedirHostTosRate

MibCount32          xmtEchoReq
MibRatePerS         xmtEchoReqRate
MibCount32          xmtEchoReply
```

```
      MibRatePerS          xmtEchoReplyRate

      MibCount32           xmtTimestampReq
      MibRatePerS          xmtTimestampReqRate
      MibCount32           xmtTimestampReply
      MibRatePerS          xmtTimestampReplyRate

      MibCount32           xmtAddrMaskReq
      MibRatePerS          xmtAddrMaskReqRate
      MibCount32           xmtAddrMaskReply
      MibRatePerS          xmtAddrMaskReplyRate
}
```

5. TCP layer mib definitions for Network Monitor mib.

5.1 tcp Segment - Summary Tool

```
typedef struct {

     MibShortCount32          pkts
     MibBucketRate       pktRate
     MibShortCount32          bytes
     MibBucketRate       byteRate

     MibShortCount32          errors
     MibBucketRate       errorRate

     Uint32                   protocolCount
     Uint32                   mostActiveCount
     Uint32                   pairCount

     MibShortCount32          rcvOffSegs
     MibBucketRate       rcvOffSegRate
     MibShortCount32          xmtOffSegs
     MibBucketRate       xmtOffSegRate
     MibShortCount32          transits
     MibBucketRate       transitRate

     MibShortCount32          flowCtrls
     MibBucketRate       flowCtrlRate

     MibShortCount32          frgmts
     MibBucketRate       frgmtRate

     MibShortCount32          rexmts
     MibBucketRate       rexmtRate

} MibTcpSegSumStats
```

5.2 tcp Segment - Values Tool

```
typedef struct {

     MibCount32                pkts
     MibRatePerS               pktRate

     MibCount32                bytes
     MibRatePerS               byteRate

     MibCount32                errors
     MibRatePerS               errorRate

     MibCount32                rcvOffSegs
     MibRatePerS               rcvOffSegRate
     MibCount32                xmtOffSegs
     MibRatePerS               xmtOffSegRate
     MibCount32                transits
     MibRatePerS               transitRate

     MibCount32                hdrBytes
     MibRatePerS               hdrByteRate
     MibCount32                frgmts
     MibRatePerS               frgmtRate

     MibCount32                flowCtrls
     MibRatePerS               flowCtrlRate

     MibCount32                rexmts
     MibRatePerS               rexmtRate

     MibCount32                rexmtBytes
     MibRatePerS               rexmtByteRate

     MibCount32                keepAlives
     MibRatePerS               keepAliveRate

     MibCount32                windowProbes
     MibRatePerS               windowProbeRate

     MibCount32                outOfOrder
     MibRatePerS               outOfOrderRate

     MibCount32                afterWindow
     MibRatePerS               afterWindowRate

     MibCount32                afterClose
     MibRatePerS               afterCloseRate

     MibCount32                urgs
     MibRatePerS               urgRate

     MibCount32                rsts
     MibRatePerS               rstRate
     MibCount32          successfulConnections
      MibRatePerH         successfulConnectionRate
      MibCount32          connectionRetries
      MibRatePerH         connectionRetryRate
      MibCount32          failedConnections
      MibRatePerH         failedConnectionRate
      MibCount32          activeConnections
} MibTcpSegValStats
```

5.3 tcp Address - Summary Tool

```
typedef struct {

     MibShortCount32          pkts
     MibBucketRate       pktRate


     MibShortCount32          bytes
     MibBucketRate       byteRate

     MibShortCount32          errors
     MibBucketRate       errorRate

     Uint32                   protocolCount
     Uint32                   mostActiveCount
     Uint32                   pairCount

     MibShortCount32          rcvOffSegs
     MibBucketRate       rcvOffSegRate
     MibShortCount32          xmtOffSegs
     MibBucketRate       xmtOffSegRate

     MibShortCount32          flowCtrls
     MibBucketRate       flowCtrlRate

     MibShortCount32          frgmts
     MibBucketRate       frgmtRate

     MibShortCount32          rexmts
     MibBucketRate       rexmtRate

} MibTcpAddrSumStats
```

5.4 tcp Address- Values Tool

```
typedef struct {

     MibCount32               rcvPkts
     MibRatePerS              rcvPktRate
     MibCount32               xmtPkts
     MibRatePerS              xmtPktRate
```

```
    MibCount32                rcvBytes
    MibRatePerS               rcvByteRate
    MibCount32                xmtBytes
    MibRatePerS               xmtByteRate

    MibCount32                rcvErrors
    MibRatePerS               rcvErrorRate
    MibCount32                xmtErrors
    MibRatePerS               xmtErrorRate

    MibCount32                rcvOffSegs
    MibRatePerS               rcvOffSegRate
    MibCount32                xmtOffSegs
    MibRatePerS               xmtOffSegRate

    MibCount32                rcvHdrBytes
    MibRatePerS               rcvHdrByteRate
    MibCount32                xmtHdrBytes
    MibRatePerS               xmtHdrByteRate

   MibCount32           rcvFrgmts
    MibRatePerS          rcvFrgmtRate
    MibCount32           xmtFrgmts
    MibRatePerS          xmtFrgmtRate

    MibCount32                rcvRexmts
    MibRatePerS               rcvRexmtRate
    MibCount32                xmtRexmts
    MibRatePerS               xmtRexmtRate

    MibCount32                rcvRexmtBytes
    MibRatePerS               rcvRexmtByteRate
    MibCount32                xmtRexmtBytes
    MibRatePerS               xmtRexmtByteRate

    MibCount32                rcvKeepAlives
    MibRatePerS               rcvKeepAliveRate
    MibCount32                xmtKeepAlives
    MibRatePerS               xmtKeepAliveRate

    MibCount32                rcvWindowProbes
    MibRatePerS               rcvWindowProbeRate
    MibCount32                xmtWindowProbes
    MibRatePerS               xmtWindowProbeRate

    MibCount32                rcvOutOfOrder
    MibRatePerS               rcvOutOfOrderRate
    MibCount32                xmtOutOfOrder
    MibRatePerS               xmtOutOfOrderRate

    MibCount32                rcvAfterWindow
    MibRatePerS               rcvAfterWindowRate
```

```
     MibCount32                xmtAfterWindow
     MibRatePerS               xmtAfterWindowRate

     MibCount32                rcvAfterClose
     MibRatePerS               rcvAfterCloseRate
     MibCount32                xmtAfterClose
     MibRatePerS               xmtAfterCloseRate

     MibCount32                rcvUrgs
     MibRatePerS               rcvUrgRate
     MibCount32                xmtUrgs
     MibRatePerS               xmtUrgRate


     MibCount32                rcvRsts
     MibRatePerS               rcvRstRate
     MibCount32                xmtRsts
     MibRatePerS               xmtRstRate

     MibCount32                successfulConnections
     MibRatePerH               successfulConnectionRate
     MibCount32                connectionRetries
     MibRatePerH               connectionRetryRate
     MibCount32                failedConnections
     MibRatePerH               failedConnectionRate
     MibCount32                activeConnections
```

6. UDP layer mib definitions for Network Monitor mib.

6.1 udp Segment -Summary Tool

```
typedef struct {
     MibShortCount32          pkts
     MibBucketRate        pktRate
     MibShortCount32          bytes
     MibBucketRate        byteRate
     MibShortCount32          errors
     MibBucketRate        errorRate
     MibShortCount32          protocolCount
     MibShortCount32          mostActiveCount
     MibShortCount32          pairCount
     MibShortCount32          rcvOffSegs
     MibBucketRate        rcvOffSegRate
     MibShortCount32          xmtOffSegs
     MibBucketRate        xmtOffSegRate
     MibShortCount32          transits
     MibBucketRate        transitRate
     MibShortCount32          flowCtrls
     MibBucketRate        flowCtrlRate
} MibUdpSegSumStats
```

6.2 udp Segment - Values Tool

```
typedef struct {
     MibCount32                 pkts
     MibRatePerS                pktRate
     MibCount32                 bytes
     MibRatePerS                byteRate
     MibCount32                 errors
     MibRatePerS                errorRate
     MibShortCount32            protocolCount
     MibShortCount32            mostActiveCount
     MibShortCount32            pairCount
     MibCount32                 rcvOffSegs
     MibRatePerS                rcvOffSegRate
     MibCount32                 xmtOffSegs
     MibRatePerS                xmtOffSegRate
     MibCount32                 transits
     MibRatePerS                transitRate
     MibCount32                 flowCtrls
     MibRatePerS                flowCtrlRate
     MibCount32                 hdrBytes
     MibRatePerS                hdrByteRate
} MibUdpSegValStats
```

6.3 udp Address - Summary Tool

```
typedef struct {
     MibShortCount32            pkts
     MibBucketRate         pktRate
     MibShortCount32            bytes
     MibBucketRate         byteRate
     MibShortCount32            errors
     MibBucketRate         errorRate
     MibShortCount32            protocolCount
     MibShortCount32            mostActiveCount
     MibShortCount32            pairCount
     MibShortCount32            rcvOffSegs
     MibBucketRate         rcvOffSegRate
     MibShortCount32            xmtOffSegs
     MibBucketRate         xmtOffSegRate
     MibShortCount32            flowCtrls
     MibBucketRate         flowCtrlRate
} MibUdpAddrSumStats
```

6.4 udp Address- Values Tool

```
typedef struct {
     MibCount32            rcvPkts
     MibRatePerS           rcvPktRate
     MibCount32            rcvBytes
     MibRatePerS          rcvByteRate
     MibCount32           rcvErrors
     MibRatePerS          rcvErrorRate
     MibCount32           xmtPkts
     MibRatePerS          xmtPktRate
     MibCount32           xmtBytes
     MibRatePerS          xmtByteRate
     MibCount32           xmtErrors
     MibRatePerS          xmtErrorRate
     MibCount32           rcvHdrBytes
     MibRatePerS          rcvHdrByteRate
     MibCount32           xmtHdrBytes
```

7. Monitor mib definitions for Network Monitor mib.

```
typedef struct {
     int                       length
     char                 no[80]
} MibPhoneNumber

typedef struct {
     MacAddress                lanMacAddr
     IpAddress            lanIpAddr
     Uint32                    lanTftpTimeout
     Uint32                    lanTftpRetryLimit
     Uint32                    lanSnmpTimeout
     Uint32                    lanSnmpRetryLimit
     MibPhoneNumber       serialPhoneNo
     IpAddress            serialIpAddr
     Uint32                    serialTftpTimeout
     Uint32                    serialTftpRetryLimit
     Uint32                    serialSnmpTimeout
     Uint32                    serialSnmpRetryLimit
} MibWsParameters

typedef struct {
     MibAddress                address
     Uint32                    flags
     MibDeviceType        type
     Uint32                    parseControl
} MibParseControl

typedef struct {
     Uint32          numEntries
     Uint32          nextEntry
     MibParseControl mibParseControl[MIB_MAX_PCR]
     } MibParseControlOpaque


typedef struct {
     MacAddress                macAddr
     Byte                 data[256]
     Uint32                        length
derived
} MibAutoTopology
```

7.1 Monitor Control Group

```
typedef struct {
     Uint32                        monReset
     MibTimeOfDay             monTOD
     Uint32                        trapPermit
     Uint32                        dupAddrTrapPermit
     Uint32                        newNodeTrapPermit
     Uint32                        shakeTime
     Uint32                        wsMonLink
     Uint32                        minTrapInterval
     Uint32                        runMonitor
     MibWsParameters               primaryWsParams
     MibWsParameters               secondaryWsParams
     Uint32                        debugLevel
     Uint32                        parseCtrl
     Uint32                        monitorSegment
     MibAutoTopology               autoTopology
     } MibMonitorControl
```

7.2 Monitor Statistics Group

```
 typedef struct {
     MibCount32                    dllDropped
     MibRatePerS                   dllDroppedRate
     MibCount32                    ipDropped
     MibRatePerS                   ipDroppedRate
     MibCount32                    icmpDropped
     MibRatePerS                   icmpDroppedRate
     MibCount32                    tcpDropped
     MibRatePerS                   tcpDroppedRate
     MibCount32                    udpDropped
     MibRatePerS                   udpDroppedRate
     MibCount32                    arpDropped
     MibRatePerS                   arpDroppedRate
     MibCount32                    nfsDropped
     MibRatePerS                   nfsDroppedRate
     MibCount32                    dbProblem
     MibShortCount32               cpuUtilization
     MibShortCount32               memoryUtilization
```

8. Alarm Mib Definitions

8.1 Counter alarm structure

```
typedef struct {
     Uint32              alarm_class
     MibTimeOfDay        gmt
     Uint32                   time_ticks
     MibAddress               mon_address
     MibAddress               address
     Uint32                   type
     Uint32                   number
     MibCount32               value
     Uint32                   user_data_length

      OPTIONAL
     Byte                user_data[MAX_ALARM_DATA]

OPTIONAL
} MibAlarmCounter
```

8.2 Rate alarm structure

```
typedef struct {
     Uint32              alarm_class
     MibTimeOfDay        gmt
     Uint32                   time_ticks
     MibAddress               mon_address
     MibAddress               address
     Uint32                   type
     Uint32                   number
     MibRollingRate      value
     Uint32                   rate_type
     Uint32                   user_data_length

      OPTIONAL
     Byte                user_data[MAX_ALARM_DATA]

OPTIONAL
} MibAlarmRate
```

8.3 Power-up alarm structure

```
typedef struct {
     Uint32              alarm_class
     MibTimeOfDay        gmt
     Uint32                   time_ticks
     MibAddress               mon_address
     Uint32                   alarm_reason
     Uint32                   load_type
     Uint32                   cpu_hw_rev
     Uint32                   mon_link_hw_rev
```

```
    Uint32                   mgmt_link_hw_rev
    MibPhoneNumber       mon_phone_no
    Uint32                   error_type
    Uint32                   error_code
    Uint32                   error_param_1
    Uint32                   error_param_2
    Uint32                   error_param_3
} MibAlarmPowerUp
```

8.4 Link-up alarm structure

```
typedef struct {
    Uint32               alarm_class
    MibTimeOfDay         gmt
    Uint32                   time_ticks
    MibAddress               mon_address
    Uint32                   alarm_reason
    Uint32                   load_type
    Uint32                   cpu_hw_rev
    Uint32                   mon_link_hw_rev
    Uint32                   mgmt_link_hw_rev
    MibPhoneNumber       mon_phone_no
    Uint32                   error_type
    Uint32                   error_code
    Uint32                   error_param_1
    Uint32                   error_param_2
    Uint32                   error_param_3
} MibAlarmLinkUp
```

8.5 New node alarm structure

```
typedef struct {
    Uint32               alarm_class
    MibTimeOfDay         gmt
    Uint32                   time_ticks
    MibAddress               mon_address
    MibAddress               node_address
} MibAlarmNewNode
```

APPENDIX III

PROTOCOL VARIABLES

The following is a list of some of the network variables for which data is gathered by the Monitor and a brief explanation of the variable, where appropriate.

DLL Variables

Frames

A frame is a series of bytes with predefined bit sequences that mark the frame's beginning and ending points. A DLL (data link layer) entity sends a message by putting it in a frame and transmitting it on the physical network. It's called a frame because the beginning and ending bit sequences "frame" the message.

Enclosed within the frame are the messages built by higher level protocols, such as IP and UDP. For example, an IP datagram must be placed in a frame before it can be transmitted.

Ethernet frames range from 64 to 1518 bytes in length.

Bytes

Monitor maintains a count and rate for bytes transmitted and received by all monitored objects. For example, for any node, you can monitor the number of bytes in or out to measure the traffic load with respect to that node. For a segment, you can monitor the number of bytes in and out of all nodes on the segment.

Error Frames

A DLL Error Frame is logged in the following cases:

* If the frame is Ethernet, none are logged.
* If the frame is IEEE 802.3:
  - Value of length parameter in header less than 3.

Alignment Errors

The number of frames observed for the selected segment with alignment errors. An alignment error is a frame with a length that is not an exact multiple of 8 bits. The following variables are available only for segments.

Collisions

The number of collisions observed on the selected segment. A collision occurs when two stations attempt to transmit simultaneously. A certain number of collisions are normal. The following variables are available only for segments.

A higher than typical value can mean that the physical interface for a single station has malfunctioned and in not following the protocol.

Broadcast frame

A broadcast frame is a special frame that is received by all stations on the network. Common uses for broadcast frames include ARP (Address Resolution Protocol) and network testing.

Multicast Frame

A multicast frame is a special frame that is received by a predetermined set of stations. Multicasting is used to send a message to a set of stations using a single frame, thus reducing network loading.

Off-segment

Off-segment frames are frames that the Monitor observes on the local segment, but are destined for or originated by nodes not on the local segment. All off-segment frames then are either routed to, from, or across the local segment.

Off-segment variables

Off-segment variables are a measure of the amount of routing or bridging that is occurring. Excessive off-segment traffic may mean that certain nodes on one segment are communicating primarily with nodes on other segments. If you identify these nodes and move them to the segments where their primary communications partners are, you may lessen the overall loading on your network.

Off-segment Transit Frames

The number of frames observed on the selected segment not into or out of a node on the selected segment. For these frames, the selected segment is an intermediate hop in a route between the originating and destination segments. (This variable applies only to segments, not to nodes.)

IP Variables

IP Packets

An IP packet or datagram is a string of bytes that is transferred as a unit across the IP network. It has two parts: the IP header, which contains control information such as the source and destination IP addresses; and the data to be transferred to the destination user.

Bytes

The Monitor maintains a count and rate for bytes into and out of all monitored objects. For example, you can monitor the number of bytes into or out of a chosen node to measure the traffic load with respect to that node. You can monitor the number of bytes into and out of all nodes on the segment.

IP Error Packets

An IP error packet is logged when the monitor observes a packet with an error in its IP header. Possible errors are as follows:

* IP header length is less than 20 bytes
* IP header length is greater than the length of the IP packet
* Packet length is less than the IP header length.
* If offset is set for fragmentation, but the frame should not be fragmented.

IP Fragments

If an IP datagram is too large to pass through a subnetwork or router, the IP router that is transmitting the original datagram divides it into fragment datagrams. The destination station reassembles the original datagram once it has received all the fragments.

Fragmentation usually occurs because packets are being routed through a network segment that has physical technology or configuration that restricts the IP datagram size to one smaller that the IP datagram size used on the originating segment.

For example, the maximum frame size in an IEEE 802.5 physical network is 16000 octets, whereas the maximum frame size on an Ethernet physical network is about 1500 octets. In this case, a large frame originating on the IEEE 802.5 network would have to be divided into many fragments before it could be transmitted onto the Ethernet network.

Note that a fragment is a complete and correct IP datagram. Do not confuse IP fragments with the Ethernet fragment errors.

Higher than typical values for these parameters may mean that one or more commonly-used communications routes are forcing fragmentation to occur.

Example: new nodes have been added that access a server across a fragmenting route. The number of additional packets causes delays on the server's segment. The solution is to reconnect the new nodes to a different segment that has a non-fragmenting route to the server.

IP Header Bytes

The header is the portion of the IP packet that contains control information used by the protocol, such as source and destination IP addresses.

Broadcast and Multicast packets

A broadcast packet is special packet that is received by all stations on the network.

A multicast packet is a packet that is received by a predefined set of stations. Multicasting is used to send a message to a set of stations using a single packet.

IP Off-segment Packets

Off-segment packets are packets that the Monitor observes on the local segment, but are destined for, or originated by, stations not on the local segment. All off-segment packets, then, are either routed to, from, or across the local segment.

Off-segment values are a measure of the amount of routing or bridging that is occurring. Excessive off-segment traffic may mean that certain stations on one segment are communicating primarily with stations on other segments. If you identify these stations and move then to the segments where their primary communications partners are, you may lessen the overall loading on your network.

Off-segment Transit Packets

This parameter applies only to segment, not to nodes. The number of IP packets observed on the selected segment not destined for or originated by an object on the selected segment. For these packets, the selected segment is an intermediate hop in a route between the originating and destination segments.

Off-segment Transit Packets Rate

This parameter applies only to segments, not to nodes. The number of off-segment IP packets observed per second on the selected segment, not into or out of an object on the selected segment. For these packets, the selected segment is an intermediate hop in a route between the originating and destination segments.

ICMP Variables

ICMP Packets

ICMP (Internet Control Message Protocol) packets are used to control, test, and report problems with, the network. Reading through the ICMP variable descriptions should give you a good idea of how ICMP is used. A high number of ICMP packets from any source wastes traffic capacity that could otherwise be used for data packets.

Bytes

The Monitor maintains a count and rate for the number of ICMP bytes in and out of all monitored objects. A high number of ICMP bytes from any source wastes traffic capacity that could otherwise be used for data.

ICMP Errors

An ICMP error is logged when the Monitor observes an ICMP packet with an error in its ICMP header. For example, a packet may have a length field with an illegal value in it. A node that generates ICMP errors may be having software problems.

Off-segment

Off-segment packets are packets that the Monitor observes on the local segment that are destined for or sent by nodes not on the local segment. All off-segment packets are either routed to, from, or across the local segment.

A high number of ICMP packets from any source wastes traffic capacity that could otherwise be used for data packets. If there are a high number of in or transit off-segment ICMP packets, the source is on a different segment.

Destination Unreachable Packets

If for some reason a gateway cannot deliver an IP packet, it sends and ICMP Destination Unreachable packet to the sender. This packet informs the sender that the packet could not be delivered, and gives a reason. The Monitor keeps count of ICMP Destination Unreachable packets into and out of all objects, by reason. These are listed below.

Net unreachable

The network is having routing problems. Possible routing problems include: a non-operational link a node or router has an incorrect routing table

Host unreachable

See net unreachable.

Protocol unreachable

Port unreachable

Frag needed / DF set

This means fragmentation is needed but Don't Fragment flag was set. This message is sent when a router cannot forward a packet because it is too large for the next subnetwork in the route. Find out why fragmentation is being disallowed by the sending node - it may not be necessary. If it is necessary, then you must find or create an alternate route.

Source route failed

Destination net unknown

The destination network is not in the router's current routing table. This may be because the source node entered the address incorrectly (a software problem) or because the router's routing table is corrupt or incomplete.

Destination host unknown

See destination net unknown

Source host isolated

Destination net prohibited (communication with destination network administratively prohibited)

Net unreachable / TOS

This means network is unreachable for this Type of Service. This message is sent when a router cannot forward a packet because the specified Type of Service is not available for this route. Find out why this Type of Service is being specified. It may be unnecessary.

Host unreachable / TOS

This means host is unreachable for this Type of Service.

Time to Live Exceeded Packets

An IP packet is allowed to remain in transit for a fixed time. This time is called "time to live" and is specified in the IP packet by the sender. If this time expires before the packet is delivered, the packet is discarded. This mechanism prevents packets that get "stuck" in circular routes from congesting the network forever.

This mechanism is enforced by the gateways that route the packet through the network. Each gateway reduces the packet's timer value by an appropriate amount, and then checks to make sure that it has not reached zero. If the timer has reached zero, the gateway discards the packet and transmits an ICMP Time to Live Count Exceeded packet back to the sender.

Packets may get stuck in loops (circular routes) because a gateway or router has incorrect information in its routing table (example).

Reassembly Time Exceeded Packets

In routing an IP packet across a network, it is sometimes necessary to fragment it into smaller packets. This must be done to get it across a segment that cannot handle the packet at its original size.

Once a packet has been fragmented, it is not reassembled until the fragments reach the final destination. Since it is possible that one or more fragments will be lost before reaching the destination, the destination node waits only a fixed period of time to receive all the fragments. This is the reassembly time.

If the destination node has not received all of the fragments when the reassembly time expires, it sends an ICMP Fragment Reassembly Time Exceeded packet to the sender.

This problem typically occurs because one or more of the fragments has been lost.

Parameter Problem Packets

Part of each IP packet (the header) contains control information. A parameter is a unit of control information. For example, one parameter specifies the length of the packet, and another specifies whether or not fragmentation of this packet is allowed.

If a gateway detects a serious problem with a parameter, and it is not reportable through one of the other ICMP messages (such as Destination Unreachable), it sends an ICMP Parameter Problem packet back to the sender.

There is currently one specific reason tracked for the ICMP Parameter Problem packet:

Param option missing (missing option parameter)

Source Quench Packets

Gateways use the source quench mechanism to slow the rate of incoming packets. If a gateway is receiving packets too fast for it to keep up with, it will send an ICMP Source Quench Packet to one or more nodes to tell them to slow down.

Redirect Packets

The redirect mechanism allows gateways to send information about routes to hosts. This works as follows:

Each node maintains a table that contains, for each of the nodes with which it communicates, the physical address of a gateway. This gateway is the first step in the route to the destination node. When a node sends a datagram to a node that is not on its segment, it send it to the gateway indicating in its routing table for the destination node.

Gateways maintain more or less complete routing information. They check all datagrams to be routed off a segment to make sure that the optimum route is being used. For example, if there are two gateways available to Node a, and Node A attempts to send a datagram to Node B across Gateway 1 when Gateway 2 would be better, Gateway 1 will detect the problem.

When this occurs, the detecting gateway issues an ICMP Redirect packet to the sending node. This packet tells the node how it should change its routing table.

Nodes use this mechanism to learn routes from gateways. All a node really needs on startup is to know the address of a gateway. It attempts to route all of its off-segment messages through this gateway, and builds its routing table from the ICMP Redirect packets it receives back.

An ICMP Redirect packet contains a diagnostic code that specifies additional information. The Monitor counts the occurrences of each of these:

Redirect for net

This packet means that datagrams to nodes on this network should be routed differently.

Redirect for host

This packet means that a datagram to this host should be routed differently.

Redirect to TOS net

This is a redirect for the network and type of service. This packet means that datagrams to hosts on this network should be routed differently in order to obtain this type of service.

Redirect TOS host

This is a redirect for the host and type of service. This packet means that a datagram to this host should be routed differently in order to obtain this type of service.

Echo Packets

The echo mechanism is used to verify that a destination is currently reachable, or to test the delay time between nodes. Echo is often referred to as "ping." The echo mechanism involves two ICMP packets: Echo Request and Echo Reply. The Monitor maintains counts for both of these.

Note that some diagnostic tools issue a series of ICMP Echo Request packets and then monitor and analyze the ICMP Echo Response packets.

A high number of these packets wastes traffic capacity.

Echo Request

This is a request that the addressed node send back an Echo Response packet.

Echo Response

This is a response packet sent by a node when it has received an Echo Request packet.

Timestamp Packets

The timestamp mechanism is used by nodes to synchronize their clocks. Node A sends an ICMP Timestamp Request packet to Node B, requesting that Node B return the current time of its system clock. Node B sends an ICMP Timestamp Response packet with the requested time to Node A. Node A can roughly synchronize its clock with Node B based on the response timestamp.

Timestamp Request

This is a request that the addressed node send back a Timestamp Response packet.

Timestamp Response

This is a response packet sent by a node when it has received a Timestamp Request packet.

Address Mask Packets

The IP protocol's addressing scheme allows sites to group multiple physical networks (segments) into a single addressable subnet. The subnet addressing scheme allows a site to determine, to an extent, which IP address bits identify the network (including subnet) and which identify nodes in the local subnet. For example, a site may determine that the first three octets in the IP address specify the network, and the last octet specifies the node in the network.

The division of address bits between network and node is represented by an address mask. The address mask is a string of 32 bits, where each bit used to specify network is set to 1, and bits that identify node are set to 0.

A node learns the address mask for its local subnet by requesting the information from a gateway. To do so it sends an ICMP Address Mask Request message to the gateway. If it does not know the address of the gateway, it may broadcast the request. The gateway replies with an ICMP Address Mask Response.

Address Mask Request

This is a request that the addressed node send back an Address Mask Response packet.

Address Mask Response

This is a response packet sent by a node when it has received an Address Mask Request packet.

TCP Variables

TCP Packets

A TCP packet (sometimes referred to as a segment) is a string of bytes that is transferred as a unit across the IP network. It has two parts: the TCP header, which contains control information such as the source and destination TCP ports; and the data to be transferred to the destination user.

Bytes

The Monitor maintains a count and rate for bytes into and out of all monitored objects. For example, you can monitor the number of bytes into or out of a chosen node to measure the traffic load with respect to that node. You can monitor the number of bytes into and out of all nodes on the segment. The byte count includes header and data bytes.

Header Bytes

The header is the portion of the TCP packet that contains control information used by the protocol, such as source and destination TCP ports. Comparing the number of TCP header bytes to the total number of TCP bytes gives an idea of the amount of TCP overhead on a connection.

Error Packets

A TCP error is logged for each packet observed with one of the following problems:

* length of TCP packet is less than 20 bytes
* TCP Header length is less than 20 bytes
* TCP header length is greater than the length of the TCP packet
* TCP header length is greater than 20 bytes but the length of the TCP packet is less than the TCP header length.

Retransmissions

A Retransmission is a TCP packet that contains some data that has already been sent at least once. A Retransmission may or may not be an exact duplicate of the packet already transmitted.

Note that if the underlying packet delivery system (DLL) creates a duplicate, it is counted as a retransmission.

When a TCP entity sends a data packet to its remote partner, it waits a predetermined period of time (tracked by a retransmission timer) for an acknowledgement (ACK) from the remote partner. If this time expires without the ACK being received, it retransmits the data contained in the presumably lost packet. It may retransmit a packet identical to the one lost, or it may combine data from multiple lost packets into a new packet, or it may combine lost data with new data into a new packet.

Excessive retransmissions can mean that a gateway is overloaded or down, that a system is overloaded, or that network parameters are misconfigured. In general, small dedicated networks should see few retransmissions. Larger, more diverse networks with routers, bridges and gateways with different capabilities and capacities are likely to have more retransmissions.

Bytes Retransmitted

Byte Retransmitted are TCP data bytes that have already been sent at least once.

See Retransmissions.

Out of Order Packets

Out of Order Packets are packets containing bytes with lower sequence numbers than bytes in previously seen packets.

Packets do not necessarily arrive in the order they were sent in. The receiving node puts the data in the correct order once it has received all packets. A high value may mean that some packets are being sent by way of a slower route, or that there is an overloaded or down bridge or router.

Out of Order Bytes

Out of Order Bytes are bytes with lower sequence numbers than bytes seen in previous packets.

Data out of Window Packets

Data out of Window Packets are packets that contains data that is not within the boundaries of the receiving partner's currently advertised window. The data is either acknowledged data or data that the partner is not ready to receive.

Bytes out of Window

Bytes out of Window are bytes that are not within the boundaries of the receiving partner's currently advertised window. The data is either acknowledged data or data that the partner is not ready to receive.

Packets after Close

Packets after Close are packets observed after a connection has been closed. These may be packets that had been "lost" on the network, or it may indicate a malfunction in the sending station.

RST Packets

A packet in which the RST (reset) bit is set.

SYN Control Packets

A packet in which the SYN bit is set.

FIN Control Packets

A packet in which the FIN bit is set.

URG Control Packets

An URG Control Packet is a packet in which the Urgent pointer is set.

The packet contains data that the receiving application should process as soon as possible. For example, the control-key sequences used by some applications are often sent as Urgent data.

Keepalives

A Keepalive is a TCP packet that a user sends to check to see if a connection is still active. The Keepalive packet contains either not data or one garbage byte of data that is outside the remote partner's last advertised window. The remote partner responds with either an ACK, confirming that the connection is alive, or a RST, indicating that the connection had been dropped.

Although widely implemented, the keepalive mechanism is not part of the TCP protocol, so you will not necessarily see keepalive activity.

Keepalives mean that a connection has been up for a long time without and activity. Resources may be unnecessarily tied up.

Window Probes

A Window Probe is a TCP packet that is sent to check the size of the remote partner's window when the last advertised window size was zero. The Window Probe packet contains one byte of data. The remote partner responds with an ACK packet, which contains the size of the remote partner's current window size.

Non-data packets, which may include window update information, may be lost and are not be retransmitted. It may therefore become necessary to check the remote partner's window size if that information has not been received for some period of time. This can mean that a node is runnind a faulty TCP implementation, that timers are misconfigured, or packets are being lost.

Window Update Only Packets

A Window Update Only packet is a packet that contains no data, but in which the advertised window size has been updated.

APPENDIX IV

Summary Tool - Values Display Fields

Packet Rate
total packets per second at this protocol layer received and transmitted at segment or node

Byte Rate
total bytes per second at this protocol layer received and transmitted at segment or node

Errors
total errors at this protocol layer received and transmitted at segment or node

Broadcast Pkt Rate
total number packets per second at this protocol layer addressed to broadcast address

Multicast Pkt Rate
total number packets per second at this protocol layer addressed to multicast address

Source Quenches
total number of ICMP source quench packets received and transmitted from this segment or node.

Fragments
total number of IP fragmented packets received and transmitted from this segment or node.

Flow Controls

UDP
total number of ICMP source quench packets received and transmitted on this UDP port.

TCP
total number of ICMP source quench packets received and transmitted on this TCP port.

NFS
total number of ICMP source quench packets received and transmitted on this NFS port.

Retransmissions
total number of TCP packets retransmitted on this TCP port.

Off Segment Packets

in
%traffic at this protocol layer received by nodes on this segment originating from other segments

in = 100(packet rate / packet rate rcv from off seg)

out
% traffic at this protocol layer transmitted by nodes on this segment to nodes on other segments

out = 100(packet rate / packet rate xmt to off seg)

Transit
% traffic at this protocol layer originating from other segments which are addressed to nodes not on this segment

transit = 100(packet rate / packet rate transit)

Local
% Traffic at this protocol layer which originates and terminates on this segment

local = 100 -(in + out + transit)

Most Active Protocols
The five most active protocols running above this layer (ie the users of this layer). The protocols are displayed as % and ranked in decreasing order.

protocol % = 100(protocol packet rate/packet rate)

APP IV - 1

Most Active Nodes The five most active nodes at this protocol layer . The nodes are displayed as % and ranked in decreasing order.

node % = 100(node packet rate/packet rate)

ICMP Types Seen The total number of these specific ICMP packet types transmitted and received on this segment or node.

Total Segment Bandwidth The % of the available bandwidth used by this protocol. If the screen is a segment display it is % used by all nodes on the segment, if it is a node display it is the % used by that node.

% = 100(8 * frame rate / 10000000)

Total Active Dialogs The number of dialogs detected for the node or segment at this protocol layer.

APP.IV - 2

5. Actual Screens for Values Tool APPENDIX V

5.1 Data Link Group

5.1.1 Definition

This screen summarizes the data link parameters.

5.1.2 Defaults

1 This is a "complete values" screen. It shows all of the values for the DLL protocol layer.

2 The user comes from a context of a specific segment or node and this screen must preserve that context.

5.1.3 Primary Screen Layout

Standard Column Headings

- Frames
  - Rcv
  - Xmt
  - Total
- Frm rate
  - Rcv
  - Xmt
  - Total
- Bytes
  - Rcv
  - Xmt
  - Total
- Byte rate
  - Rcv
  - Xmt
  - Total
- Errors
  - Rcv
  - Xmt
  - Total
- Error rate
  - Rcv
  - Xmt
  - Total
- 802.3 frames
  - Rcv
  - Xmt
  - Total
- ethernet frames
  - Rcv
  - Xmt
  - Total
- 802.3 frame rate
  - Rcv
  - Xmt
  - Total
- ethernet frame rate
  - Rcv
  - Xmt
  - Total
- Bcast Xmt
- Bcast rate
- Mcast Xmt
- Mcast rate
- Off seg
  - Rcv
  - Xmt
  - [Transit]

[local]
Total
Off seg rate
Rcv
Xmt
[Transit]
[local]
Total
Runts Xmt
[Allignment]
[Collisions]

Protocol	Pkt Count	Pkt Rate	%
Protocol 1			
Protocol 2			
.			
.			
Protocol n			

5.1.4 Secondary Screen Layout

Extended Column Headings
rows as for primary screen

5.2 IP Group

5.2.1 *Definition*

This screen provides information for the IP network layer running on the segment or node.

5.2.2 Defaults

1 This is a "complete values' screen. It shows all of the values for the IP protocol type

2 The user comes from a context of a specific segment or node and this screen must preserve that context

5.2.3 Primary Screen Layout

Standard Column Headings
Pkts
Pkt rate
Bytes
Byte rate
Errors
Error rate
Frags
Frag rate
Header bytes
Header rate
Bcast Xmt
Bcast rate
Mcast Xmt
Mcast rate
Off seg
Off seg rate

Protocol	Pkt Count	Pkt Rate	%
Protocol 1			
Protocol 2			
.			
.			
Protocol n			

5.2.4 Secondary Screen Layout

Extended Column Headings
rows as for primary screen

5.3 ICMP Group

5.3.1 Definition

This screen provides information for the ICMP protocol s/w running on the segment or node.

5.3.2 Defaults

1 This is a "complete values" screen. It shows all of the values for the ICMP protocol type

2 The user comes from a context of a specific segment or node and this screen must preserve that context.

5.3.3 Primary Screen Layout

Standard Column Headings
Pkts
Pkt rate
Bytes
Byte rate
Errors
Error rate
Off seg
Off seg rate
D.U. net
D.U. host
D.U. Prot
D.U. port
D.U. frag
D.U. Src route
D.U. Net Unk.
D.U. Host Unk.
D.U. Src Host isol.
D.U. Dnet Ad Prob
D.U. Dhost Ad Prob
D.U. Net Unr.
D.U. Time Xd Trans
D.U. Time Xd Reass
Param prob
Param opt miss.
src quench
redir net
redir host
redir tos net
redir tos host
Echo req
Echo Resp
Ts req
Ts resp
Addr mask req
Addr mask resp

5.3.4 Secondary Screen Layout

Extended Column Headings

rows as for primary screen

5.4 UDP Group

5.4.1 Definition

This screen provides information for the UDP protocol s/w running on the segment or node.

5.4.2 Defaults

1 This is a "complete values" screen. It shows all of the values for the UDP protocol type

2 The user comes from a context of a specific segment or node and this screen must preserve that context.

5.4.3 Primary Screen Layout

	Standard Column Headings
Pkts	
Pkt rate	
Bytes	
Byte rate	
Errors	
Error rate	
Header bytes	
Header rate	
off seg	
off seg rate	

Protocol	Pkt Count	Pkt Rate	%
Protocol 1			
Protocol 2			
.			
.			
Protocol n			

5.4.4 Secondary Screen Layout

Extended Column Headings

rows as for primary screen

5.5 TCP Group

5.5.1 Definition

This screen provides information for the TCP protocol s/w running on the segment or node.

5.5.2 Defaults

1 This is a "complete values" screen. It shows all of the values for the TCP protocol type

2 The user comes from a context of a specific segment or node and this screen must preserve that context

5.5.3 Primary Screen Layout

Standard Column Headings
number connections
Pkts
Pkt rate
bytes
Byte rate
header bytes
Hdr byt rt
errors
Error rate
persists
keep alives
rexmits
bytes rexmit
ack only pkt
window probes
pkts urg only
window update only
control pkts
dup only pkts
part dup pkts
dup bytes
out order pkts
out order bytes
data pkts after window
bytes after window
pkts after close
dup acks
ack pkts
off seg
off seg rate

Protocol	Pkt Count	Pkt Rate	%
Protocol 1			
Protocol 2			
.			
.			
Protocol n			

5.5.4 Secondary Screen Layout

Extended Column Headings
rows as for primary screens

5.6 NFS Group

5.6.1 Definition

These screens provide information for the NFS protocol s/w running on the segment or node. The screens show the breakdown of activity by servers and clients for filesystems. directories and files.

5.6.2 Defaults -client/server

1 This is a "complete values" screen. It shows all of the values for the NFS protocol type

2 The user comes from a context of either a segment or a node and this screen must preserve that context.

5.6.3 Primary Screen Layout -client/server

Standard Column Headings
total nfs ops
nfs ops rate
read opss
read rate
write ops
bytes read
bte read rate
bytes written
bytes written rate
write rate
write cache
create file
remove file
rename file
create dir
remove dir
null ops
get file attr
set file attr
look ups
read link
create link
create sym lnk
get fsys attr
mount
unmount
readmount
unmountall
readexport

File Systems on Server
file system 1
file system 2
.
.
file system n

5.6.4 Secondary Screen Layout

Extended Column Headings
rows as for primary screens

5.6.5 Navigation

Double clicking on a file system invokes the file system screen for the selected file system.

5.6.6 Defaults -file system

1 This is a "complete values" screen. It shows all of the values for the NFS protocol type for this file system.

2 The user comes from a context of either an nfs client or server and this screen must preserve that context.

5.6.7 Primary Screen Layout -file system

Standard Column Headings
total nfs ops
nfs ops rate
read ops
read op rate
write ops
write op rate
bytes read
bte read rate
bytes written
bytes written rate
write cache
create file
remove file
rename file
create dir
remove dir
null ops
get file attr
set file attr
look ups
read link
create link
create sym lnk
get fsys attr
mount
unmount

Directories in File System
directory 1
directory 2
.
.
directory n

5.6.8 Secondary Screen Layout

Extended Column Headings
rows as for primary screens

5.6.9 Navigation

Double clicking on a directory invokes the directory screen for the selected directory.

5.6.10 Defaults -directory

1 This is a "complete values" screen. It shows all of the values for the NFS protocol type for this directory.

2 The user comes from a context of an nfs file system and this screen must preserve that context.

5.6.11 Primary Screen Layout -directory

Standard Column Headings
total nfs ops
nfs ops rate
read ops
read ops rate
write ops
write ops rate
bytes read
bte read rate
bytes written
bytes written rate
write cache
create file
remove file
rename file
null ops
get file attr
set file attr
look ups
read link
create link
create sym lnk
create sym lnk
Attributes
type
mode
nlinks
uid
gid
size
blocksize
rdev
blocks
fileid
atime
mtime
ctime
Files in Directory
file 1
file 2
.
.
file n

5.6.12 Secondary Screen Layout

Extended Column Headings
rows as for primary screens

5.6.13 Navigation

Double clicking on a file invokes the file screen for the selected file.

5.6.14 Defaults -file

1 This is a "complete values" screen. It shows all of the values for the NFS protocol type for this file.

2 The user comes from a context of an nfs file directory and this screen must preserve that context.

5.6.15 Primary Screen Layout -file

Standard Column Headings
total nfs ops
nfs ops rate
read ops
read ops rate
write ops
write ops rate
bytes read
bte read rate
bytes written
bytes written rate
write cache
null ops
get file attr
set file attr
look ups
read link
create link
create sym lnk

Attributes
type
mode
nlinks
uid
gid
size
blocksize
rdev
blocks
fileid
atime
mtime
ctime

5.6.16 Secondary Screen Layout

Extended Column Headings
rows as for primary screens

5.7 ARP Group

5.7.1 Definition

This screen provides information for the ARP protocol s/w running on the segment or node.

5.7.2 Defaults

1 This is a "complete values" screen. It shows all of the values for the ARP protocol type

2 The user comes from a context of either a segment or a node and this screen must preserve that context.

5.7.3 Primary Screen Layout

Standard Column Headings

TBD

5.7.4 Secondary Screen Layout

Extended Column Headings

rows as for primary screens

5.8 RARP Group

5.8.1 Definition

This screen provides information for the RARP protocol s/w running on the segment or node.

5.8.2 Defaults

1 This is a "complete values" screen. It shows all of the values for the RARP protocol type

2 The user comes from a context of either a segment or a node and this screen must preserve that context.

5.8.3 Primary Screen Layout

Standard Column Headings
TBD

5.8.4 Secondary Screen Layout

Extended Column Headings
rows as for primary screens

5.9 Telnet Group

5.9.1 Definition

This screen provides information for the Telnet protocol s/w running on the segment or node.

5.9.2 Defaults

1 This is a "complete values" screen. It shows all of the values for the Telnet protocol type

2 The user comes from a context of either a segment or a node and this screen must preserve that context.

5.9.3 Primary Screen Layout

Standard Column Headings
TBD

5.9.4 Secondary Screen Layout

Extended Column Headings
rows as for primary screens

5.10 FTP Group

5.10.1 Definition

This screen provides information for the FTP protocol s/w running on the segment or node.

5.10.2 Defaults

1 This is a "complete values" screen. It shows all of the values for the FTP protocol type

2 The user comes from a context of either a segment or a node and this screen must preserve that context.

5.10.3 Primary Screen Layout

Standard Column Headings

TBD

5.10.4 Secondary Screen Layout

Extended Column Headings

rows as for primary screens

5.11 Dialogue Data Group

5.11.1 Definition

This screen displays all of the Data available for a particular dialogue. This screen is shown when the user clicks on an entry in the Summary Tool dialogue information.

Each dialog screen represents a single dialog. Thus at the UDP or TCP level two nodes may have multiple dialogs (each with a unique port pair) and each of these will be represented as a seperate entity.

Because the user cannot uniquely identify the dialog he requires from the menus (he does not know the port numbers involved) the only mechanism to invoke these screens is by selection of a dialog from the approriate summary screen. This problem also prevents the user from 'clicking' through all the dialogs on ports between a node pair (may be addressed in later phase).

5.11.2 Defaults

1 This is a "complete values" screen. It shows all of the values available for the selected connection.

2 There are several different contexts for this screen. The user may select this option from the summary tools for all protocols. This screen must reflect the node, layer and specific connection context from which the user entered

The content of this screen is essentially the same as the corresponding row entry from the Traffic matrix screen for the DLL and IP layers. Their inclusion is to provide the user with a consistent navigaion paradigm accross the layers (and to provide this functionality in release 1 which does ot include the Traffic matrix support).

The data set displayed in this screen will be appropriate to the protocols used between the nodes. The variables shown are those selected for TCP/IP protocols. Where nodes converse using multiple protocols this will be expanded to select data from each protocl set.

5.11.3 Primary Screen -DLL

node name	node name
mac address	mac address
ip address	ip address
Network Protocols:	
start time	last seen time

Standard Column Headings

frames
bytes
errors
flow ctl
ip frags
tcp retransmissions

5.11.4 Secondary Screen Layout -DLL

Extended Column Headings

rows as for primary screens

5.11.5 Primary Screen -IP

node name	node name
mac address	mac address
ip address	ip address
Transport Protocols:	
start time	last seen time

Standard Column Headings

Pkts
bytes
header bytes
errors
fragments
TCP retransmissions
ICMP

5.11.6 Secondary Screen Layout -IP

Extended Column Headings

rows as for primary screens

5.11.7 Primary Screen -ICMP

This is invoked by selection of the ICMP entry from the IP screen.

node name	node name
mac address	mac address
ip address	ip address

Standard Column Headings

Pkts
Bytes
Errors
Off seg
D.U. net
D.U. host
D.U. Prot
D.U. port
D.U. frag
D.U. Src route
D.U. Net Unk.
D.U. Host Unk.
D.U. Src Host isol.
D.U. Dnet Ad Prob
D.U. Dhost Ad Prob
D.U. Net Unr.
D.U. Time Xd Trans
D.U. Time Xd Reass
Param prob
Param opt miss.
src quench
redir net
redir host
redir tos net
redir tos host
Echo req
Echo Resp
Ts req
Ts resp
Addr mask req
Addr mask resp

5.11.8 Secondary Screen Layout

Extended Column Headings

rows as for primary screens

5.11.9 Primary Screen -UDP

node name	node name
mac address	mac address
ip address	ip address
port number	port number
Application Protocol:	
start time	last seen time

Standard Column Headings

Pkts
bytes
errors
ip frags
flow ctl

5.11.10 Secondary Screen Layout -UDP

Extended Column Headings

rows as for primary screens

5.11.11 Primary Screen -TCP

node name node name
mac address mac address
ip address ip address
port number port number
Application Protocol:
Connection Status: [active, closed-ok, closed reset, unknown]
start time last seen time

Standard Column Headings
Pkts
bytes
header bytes
errors
pkts bad seq #
bytes not acked
persists
keep alives
pkts rexmit
bytes rexmit
ack only pkt
window probes
pkts urg only
window update only
control pkts
dup only pkts
part dup pkts
dup bytes
out order pkts
out order bytes
data pkts after window
bytes after window
pkts after close
dup acks
acks unsent data
ack pkts
bytes acked by acks
current window

5.11.12 Secondary Screen Layout -TCP

Extended Column Headings

rows as for primary screens

5.11.13 Primary Screen -NFS

node name	node name
mac address	mac address
ip address	ip address
port number	port number
start time	last seen time

Standard Column Headings

variables as for NFS Group

5.11.14 Secondary Screen Layout -NFS

Extended Column Headings

rows as for primary screens

5.11.15 Navigation

As for NFS group a hieararchy of screens is available:

1 client to server

2 client to file system

3 client to directory

4 client to file

5.12 Traffic Matrix Group (Not in release 1)

5.12.1 Definition

This screen shows traffic distribution between a selected node (or segment) and other nodes (or segments) in the network.

For the DLL and IP layers it is essentially a repeat of the dialogue screens. For the UDP and TCP layers however it represents a summation over multiple connections between the two nodes.

5.12.2 Defaults

1 The user comes from a context of a specific segment or node plus a protocol level and this screen must preserve this context.

2 If the selection propagated from the Summary Tool is a segment then the distribution is segment to segment, if the selection is a node then the distribution is node to node.

3 Values are shown in order of heaviest traffic to lightest.

4 The initial screen has the heaviest pairs of nodes or segments. Scrolled screens contain progressively lighter traffic loads.

5 The user can select the column by which the nodes are to be ordered and request reordering. This allows the user to use this screen look at flow control for example.

6 Double clicking on a node or segment in the display area allows the user to move to this object as the focus of the traffic matrix ie if the user is looking at a matrix for node A and selects node B (which is one of the nodes in the matrix) they will get the traffic matrix for B.

7 Double clicking on the node which is the focus of the matrix (eg A in the above example) selects the next segment or node, consistent with the current view. Node views click to other nodes on the segment, Segment views click to other segments. The segment (or) node selection will be ordered alphabetically.

8 The data maintained between two nodes (or segments) will be aged out if no communication between them occurs for a defined period (settable by the user -eventually).

5.12.3 Primary Screen DLL

Node(Segment) Name

	frm	frm	byte	byte	err	err	flow	flow	tffc
		rate		rate		rate	ctl	ct rt	%
node(segment)1									
node(segment)2									
.									
.									
.									
node(segment)n									

This scrolls down to accomodate all nodes (or segments) required.

5.12.4 Secondary Screen

frag	frag	tcp	tcp
	rate	rexmit	rexm rt

rows as primary screen

5.12.5 Primary Screen IP

Node(Segment) Name

	pkt	pkt	err	err	frag	frag	icmp	flw	flw	tffc
		rate		rate		rate		ctl	ct rt	%
node(segment)1										
node(segment)2										
.										
.										
.										
node(segment)n										

This scrolls down to accomodate all nodes (or segments) required.

5.12.6 Primary Screen ICMP

This is invoked by selection of the ICMP entry for a node (segment) pair. The user is vectored to the IP traffic matrix screen in this case.

5.12.7 Primary Screen TCP

Node(Segment) Name

	pkt	pkt rate	err	err rate	act conn	rxmt	rxmt rate	flw ctl	flw ct rt	tffc %	# conns
node(segment)1											
node(segment)2											
.											
.											
.											
node(segment)n											

This scrolls down to accomodate all nodes (or segments) required.

5.12.8 Primary Screen UDP

Node(Segment) Name

	pkt	pkt	err	err	actv	flow	flow		tffc
		rate		rate	conn	ctl	ctl rt	%	
node(segment)1									
node(segment)2									
.									
.									
.									
node(segment)n									

This scrolls down to accomodate all nodes (or segments) required.

5.12.9 Primary Screens NFS

5.12.9.1 Client to Server

Node(Segment) Name

	pkt	pkt	err	err	actv	flow	flow	tffc
		rate		rate	conn	ctl	ctl rt	%
node(segment)1								
node(segment)2								
.								
.								
.								
node(segment)n								

File systems on this node

file system 1
file system 2
.
.
file system n

This scrolls down as required.

5.12.9.1.1 Navigation

Double clicking on a file system invokes the file system screen for the selected file system.

5.12.9.2 Client to File System

Node(Segment) Name
File System name

	pkt	pkt	err	err	actv	flow	flow	tffc
		rate		rate	conn	ctl	ctl rt %	
node(segment)1								
node(segment)2								
.								
.								
.								
node(segment)n								

Directories on this file system

directory 1
directory 2
.
.
directory n

This scrolls down as required.

5.12.9.2.1 Navigation

Double clicking on a directory invokes the directory screen for the selected directory.

5.12.9.3 Client to Directory

Node(Segment) Name
File System name
directory name

	pkt	pkt	err	err	actv	flow	flow	tffc
		rate		rate	conn	ctl	ctl rt %	
node(segment)1								
node(segment)2								
.								
.								
.								
node(segment)n								

files in this directory

file 1
file 2
.
.
file n

This scrolls down as required.

5.12.9.3.1 Navigation

Double clicking on a file invokes the file screen for the selected file.

4.12.9.7 Client to File

Node(Segment) Name
File System name
directory name
file name

	pkt	pkt rate	err	err rate	actv conn	flow ctl	flow ctl rt %	tffc
node(segment)1								
node(segment)2								
. . .								
node(segment)n								

This scrolls down as required.

5.13 Summary Screen for Traffic Matrix

	Seg1	Seg2	Seg3	....	Segn
Seg1		frame byte error	frame byte error		frame byte error
Seg2	frame byte error		frame byte error		frame byte error
Seg3	frame byte error	frame byte error			frame byte error
. . .					
Segn	frame byte error	frame byte error	frame byte error	....	

Claims

1. A method for monitoring communications which occur in a network of nodes, each communication being effected by a transmission of one or more packets among two or more communicating nodes, each communication complying with a predefined communication protocol selected from among protocols available in said network, said method comprising

detecting passively and in real time the contents of packets, and

deriving, from said detected contents of said packets, communication information associated with multiple said protocols.

2. The method of claim 1 wherein said step of deriving communication information includes deriving communication information from associated with multiple layers of at least one of said protocols.

3. A method for monitoring communication dialogs which occur in a network of nodes, each dialog being effected by a transmission of one or more packets among two or more communicating nodes, each dialog complying with a predefined communication protocol selected from among protocols available in said network, said method comprising

detecting the contents of packets, and

deriving from said detected contents of said packets, information about the states of dialogs occurring in said network and which comply with different selected protocols available in said network.

4. The method of claim 3 wherein said step of deriving information about the states of dialogs comprises

maintaining a current state for each dialog, and

updating the current state in response to the detected contents of transmitted packets.

5. The method of claim 3 wherein said step of deriving information about the states of dialogs comprises

maintaining, for each dialog, a history of events based on information derived from the contents of packets, and

analyzing the history of events to derive information about the dialog.

6. The method of claim 5 wherein said step of analyzing the history includes counting events.

7. The method of claim 5 wherein said step of analyzing the history includes gathering statistics about events.

8. The method of claim 5 further comprising

monitoring the history of events for dialogs which are inactive, and

purging from the history of events dialogs which have been inactive for a predetermined period of time.

9. The method of claim 4 wherein said step of deriving information about the states of dialogs comprises

updating said current state in response to observing the transmission of at least two data related packets between nodes.

10. The method of claim 5 wherein said step of analyzing the history of events comprises

analyzing sequence numbers of data related packets stored in said history of events, and

detecting retransmissions based on said sequence numbers.

11. The method of claim 4 further comprising

updating the current state based on each new packet associated with said dialog, and

if an updated current state cannot be determined, consulting information about prior packets associated with said dialog as an aid in updating said state.

12. The method of claim 5 further comprising

searching said history of events to identify the initiator of a dialog.

13. The method of claim 5 further comprising

searching the history of events for packets which have been retransmitted.

14. The method of claim 4 wherein

the full set of packets associated with a dialog up to a point in time completely define a true state of the dialog at that point in time,

said step of updating the current state in response to the detected contents of transmitted packets comprises generating a current state which may not conform to the true state.

15. The method of claim 5 wherein the step of updating the current state comprises updating the current state to "unknown".

16. The method of claim 14 further comprising updating the current state to the true state based on

information about prior packets transmitted in the dialog.

17. The method of claim 15 further comprising updating the current state to the true state based on information about prior packets transmitted in the dialog.

18. The method of claim 3 wherein said step of deriving information about the states of dialogs occurring in said network comprises parsing said packets in accordance with more than one but fewer than all layers of a protocol.

19. The method of claim 3 wherein each said communication protocol includes multiple layers, and each dialog complies with one of said layers.

20. The method of claim 3 wherein said protocols include a connectionless-type protocol in which the state of a dialog is implicit in transmitted packets, and said step of deriving information about the states of dialogs includes inferring the states of said dialogs from said packets.

21. The method of claim 4 further comprising

parsing said packets in accordance a protocol and

temporarily suspending parsing of some layers of said protocol when parsing is not rapid enough to match the rate of packets to be parsed.

22. A method of analyzing the performance of a network of nodes which communicate via dialogs, each dialog being effected by a transmission of one or more packets among two or more communicating nodes, each

dialog complying with a predefined communication protocol selected from among protocols available in said network, said method comprising

monitoring the operation of the network with respect to specific items of performance during normal operation,

generating a model of said network based on said monitoring, and

setting acceptable threshold levels for said specific items of performance based on said model.

23. The method of claim 22 further comprising

monitoring the operation of the network with respect to the specific items of performance during periods which may include abnormal operation.

24. Apparatus for monitoring communication dialogs which occur in a network of nodes, each dialog being effected by a transmission of one or more packets among two or more communicating nodes, each dialog complying with a predefined communication protocol selected from among protocols available in said network, said apparatus comprising

a monitor connected to the network medium for passively, and in real time, monitoring transmitted packets and storing information about dialogs associated with said packets, and

a workstation for receiving said information about dialogs from said monitor and providing an interface to a user.

25. The apparatus of claim 24 wherein said workstation further comprises

means for enabling a user to observe events of acitve dialogs.

26. Apparatus for monitoring packet communications in a network of nodes in which communications may be in accordance with multiple protocols, said apparatus comprising

a monitor connected to a communication medium of the network for passively, and in real time, monitoring transmitted packets of different protocols and storing information about communications associated with said packtes, said communications being in accordance with different protocols, and

a workstation for receiving said information about said communciations from said monitor and providing an interface to a user,

said monitor and said workstation including means for relaying said information about multiple protocols with respect to communication in said different protocols from said monitor to said workstation in accordance with a single common network management protocol.

27. A method of diagnosing communication problems between two nodes in a network of nodes interconnected by links, comprising

monitoring the operation of the network with respect to specific items of performance during normal operation,

generating a model of normal operation of said network based on said monitoring, and

setting acceptable threshold levels for said specific items of performance based on said model.

28. The method of claim 27 further comprising the steps of

monitoring the operation of the network with respect to the specific items of performance during periods which may include abnormal operation, and when abnormal operation of the network with respect to communication between the two nodes is detected, diagnosing the problem by separately analyzing the performance of each of the nodes and each of the links connecting the two nodes to isolate the abnormal operation.

29. A method of timing the duration of a transaction of interest occurring in the course of communication between nodes of a network, the beginning of said transaction being defined by the sending of a first packet of a particular kind from one node to the other, and the end of said transaction being defined by the sending of another packet of a particular kind between the nodes, comprising

passively and in real time monitoring packets transmitted in the network,

beginning to time said transaction upon the appearance of said first packet,

determining when the other packet has been transmitted, and

ending the timing of the duration of the transaction upon the appearance of the other packet.

30. A method for tracking node address to node name mappings in a network of nodes of the kind in which each node has a possibly nonunique node name and a unique node address within the network and in which node addresses can be assigned and reassigned to node names dynamically using a name binding protocol message incorporated within a packet, said method comprising

monitoring packets transmitted in said network, and updating a table linking node names to node addresses based on information contained in said name binding protocol messages in said packets.

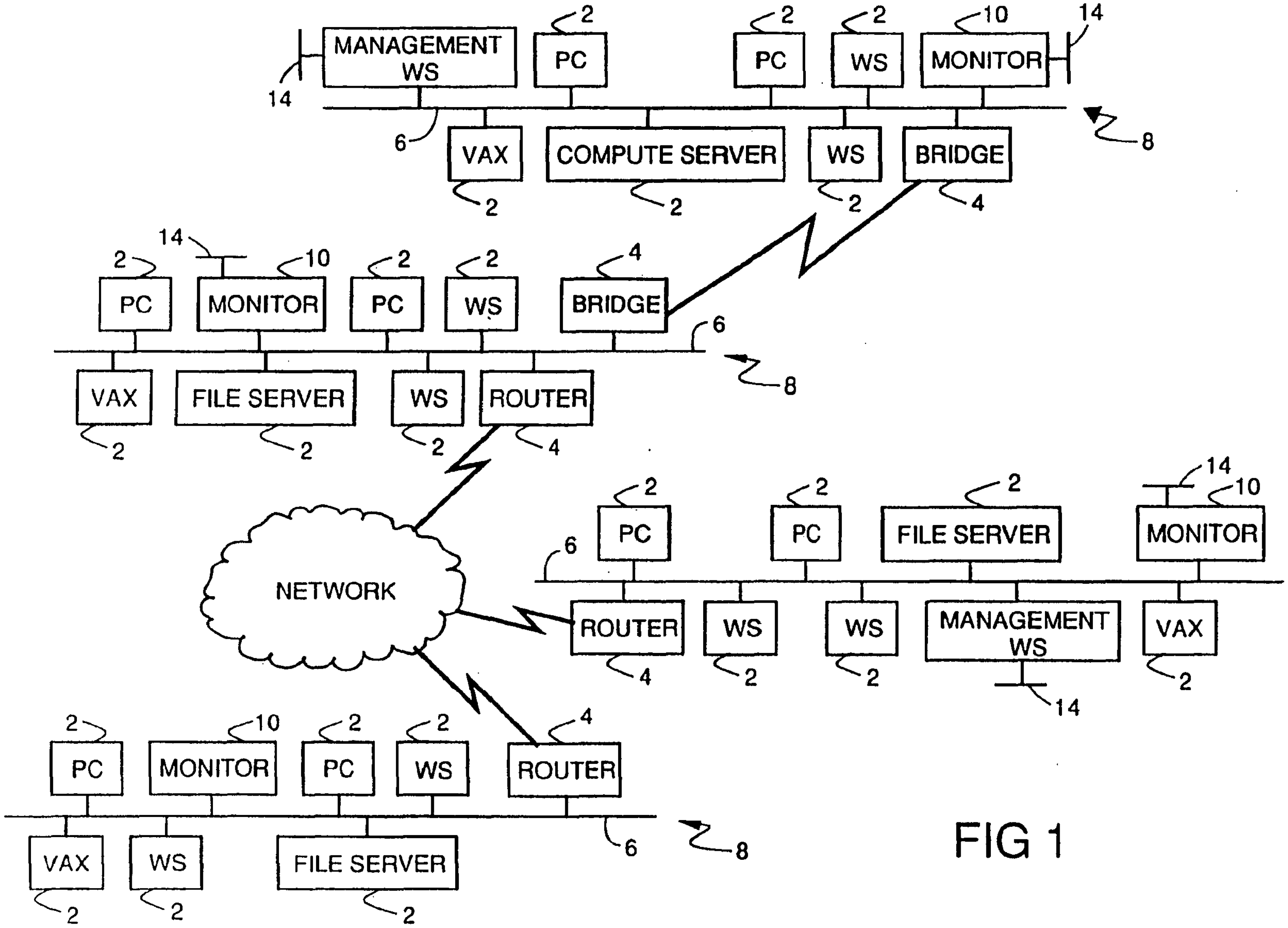

MANAGEMENT WS
PC
PC
WS
MONITOR
VAX
COMPUTE SERVER
WS
BRIDGE
PC
MONITOR
PC
WS
BRIDGE
VAX
FILE SERVER
WS
ROUTER
NETWORK
PC
PC
FILE SERVER
MONITOR
ROUTER
WS
WS
MANAGEMENT WS
VAX
PC
MONITOR
PC
WS
ROUTER
VAX
WS
FILE SERVER
2
4
6
8
10
14

FIG 1

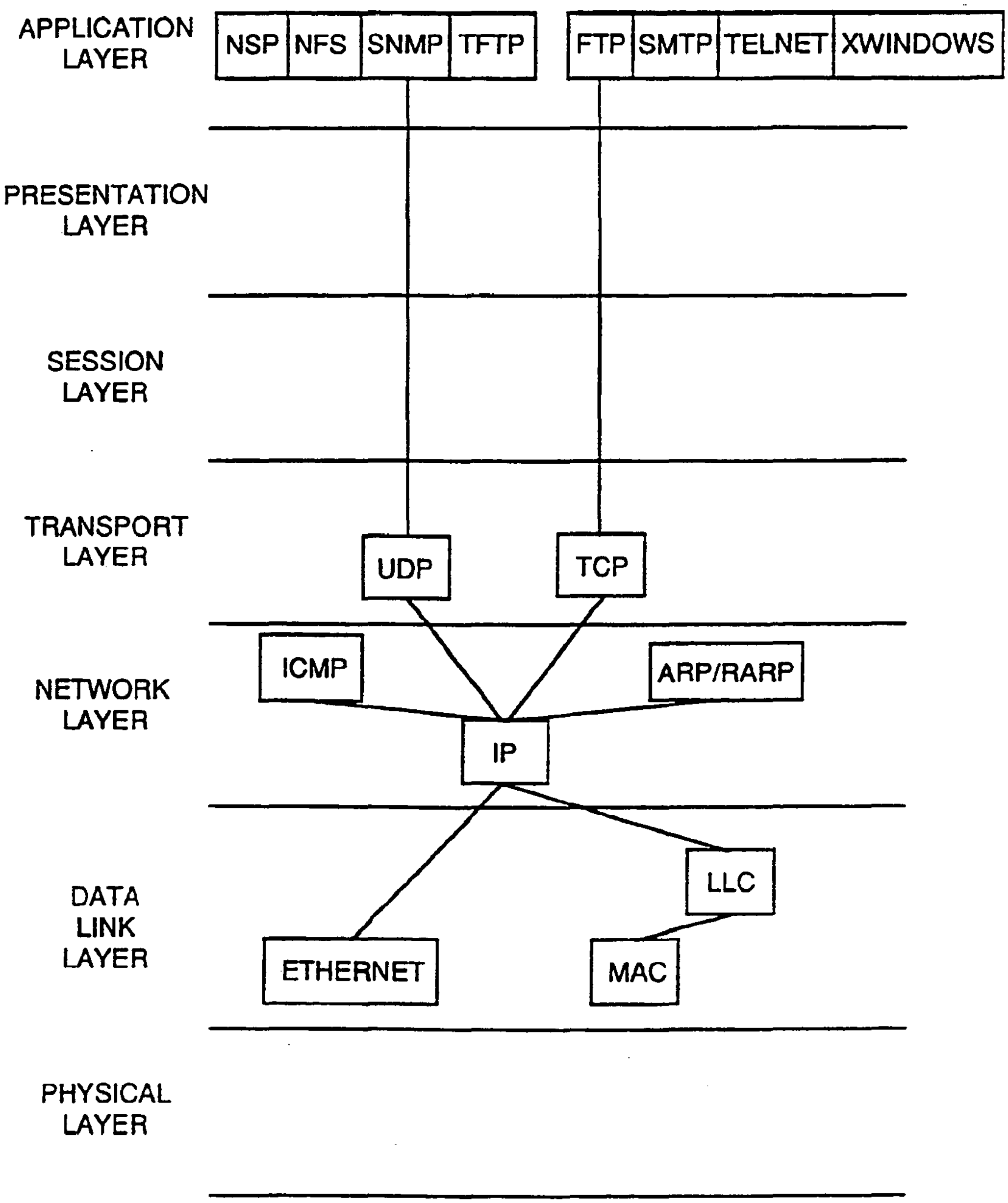

APPLICATION LAYER
NSP
NFS
SNMP
TFTP
FTP
SMTP
TELNET
XWINDOWS
PRESENTATION LAYER
SESSION LAYER
TRANSPORT LAYER
UDP
TCP
NETWORK LAYER
ICMP
ARP/RARP
IP
DATA LINK LAYER
LLC
ETHERNET
MAC
PHYSICAL LAYER

FIG 2

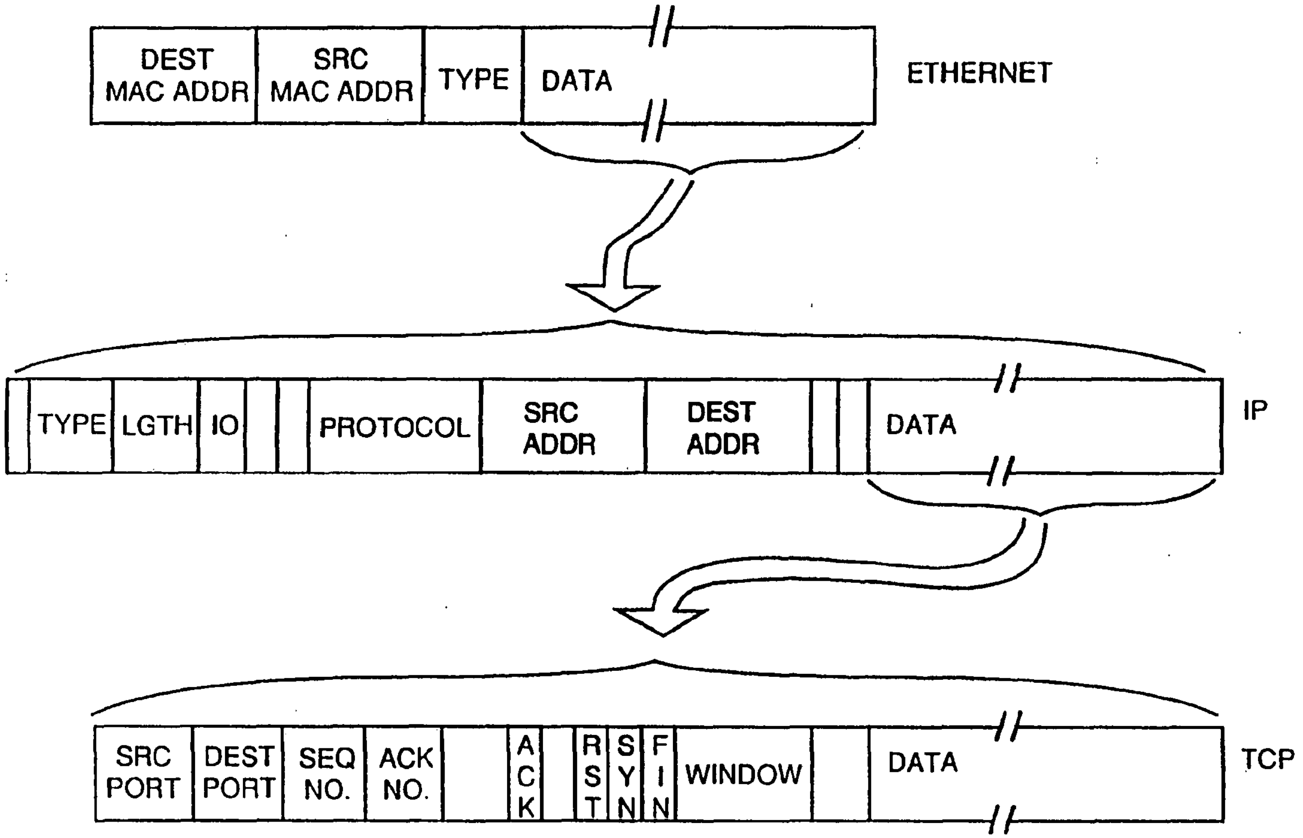

DEST MAC ADDR
SRC MAC ADDR
TYPE
DATA
ETHERNET
TYPE
LGTH
IO
PROTOCOL
SRC ADDR
DEST ADDR
DATA
IP
SRC PORT
DEST PORT
SEQ NO.
ACK NO.
ACK
RST
SYN
FIN
WINDOW
DATA
TCP

FIG 3

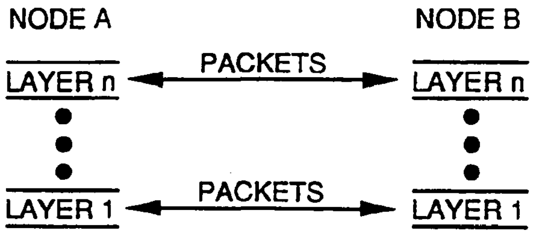


FIG 4

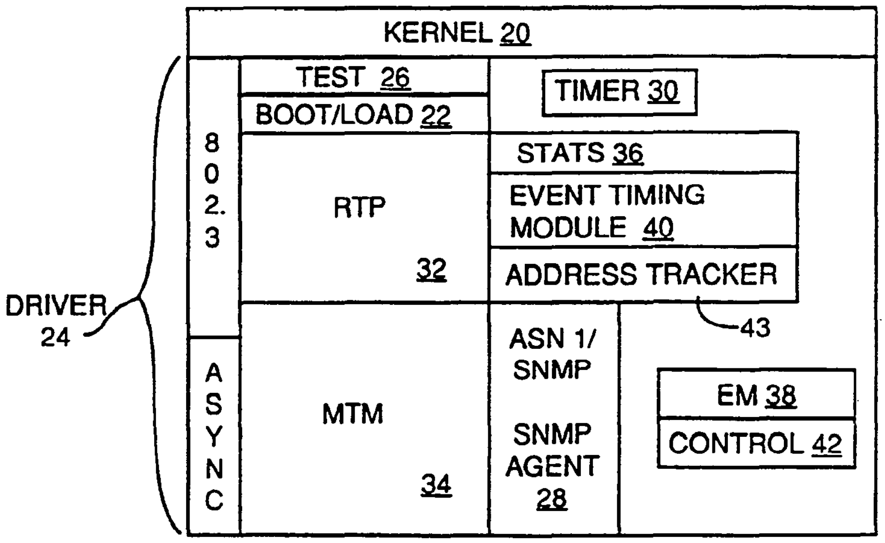


FIG 5

SUBSTITUTE SHEET

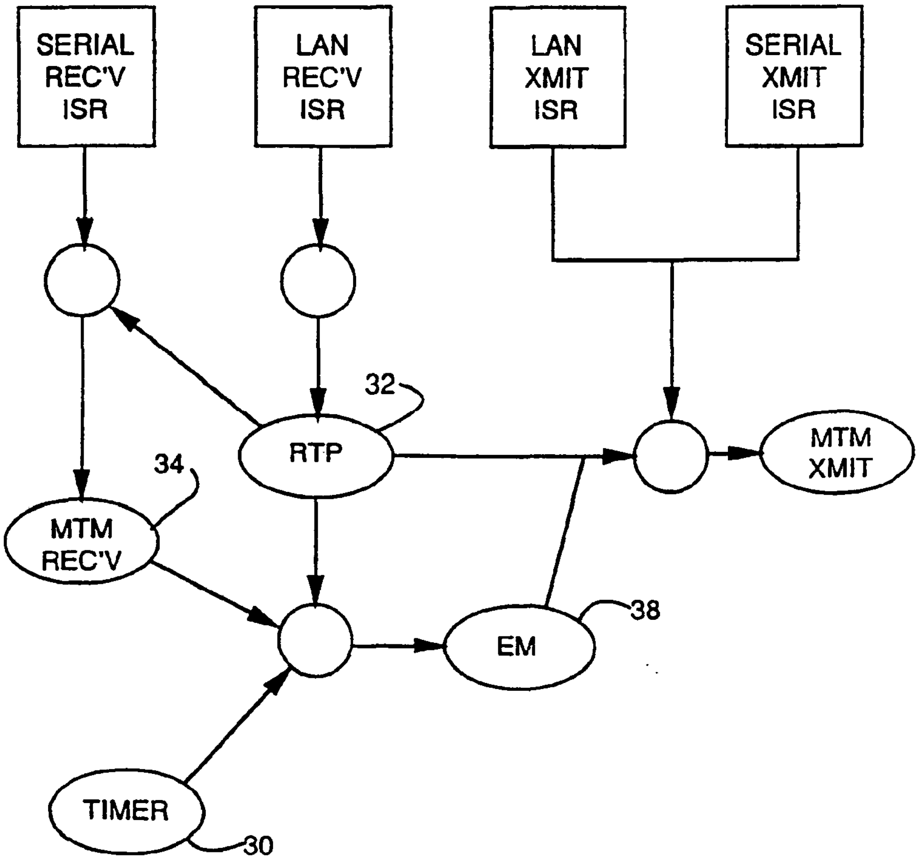

SERIAL REC'V ISR
LAN REC'V ISR
LAN XMIT ISR
SERIAL XMIT ISR
32
RTP
MTM XMIT
34
MTM REC'V
38
EM
TIMER
30

FIG 6

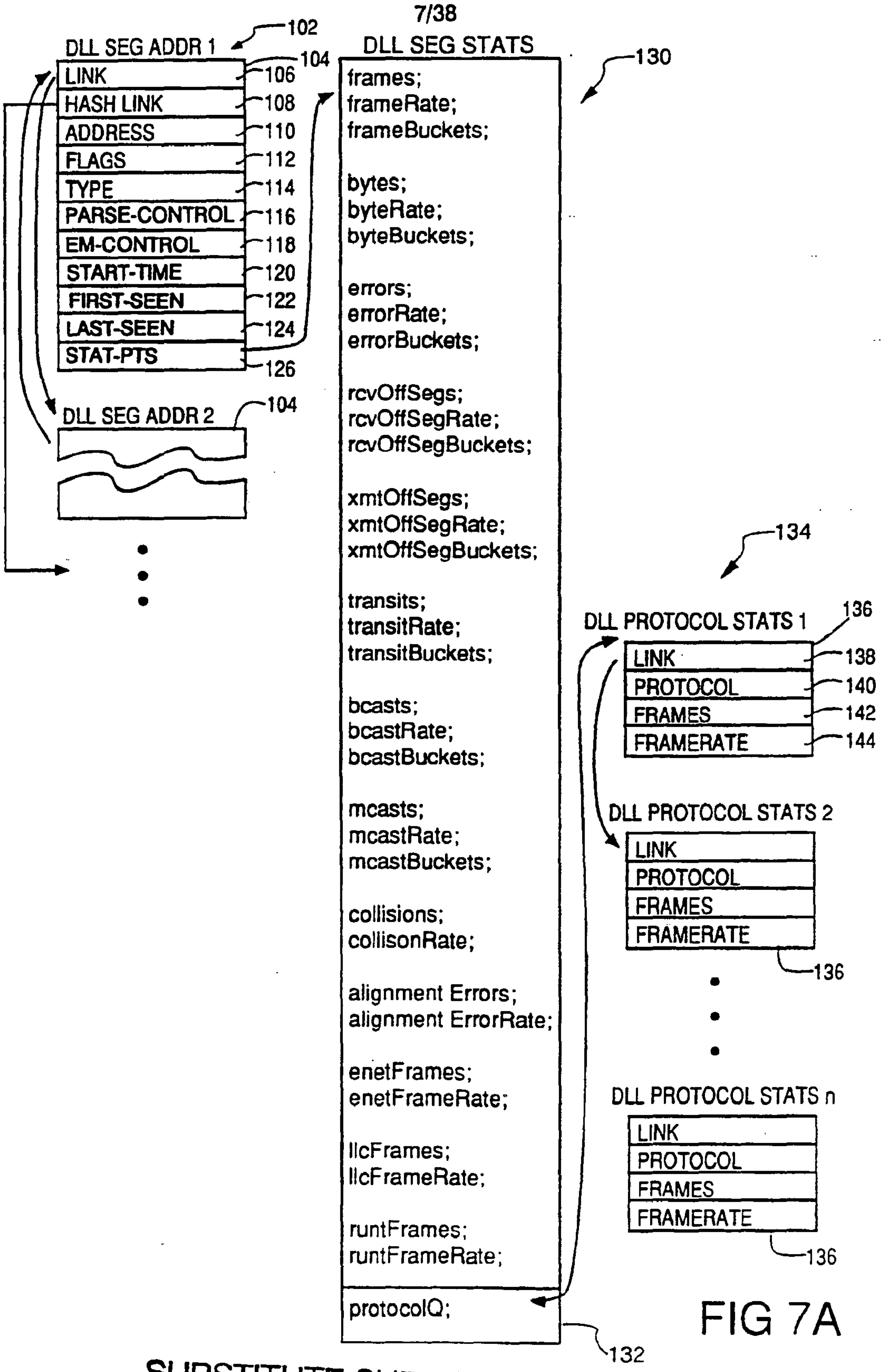


FIG 7A

SUBSTITUTE SHEET

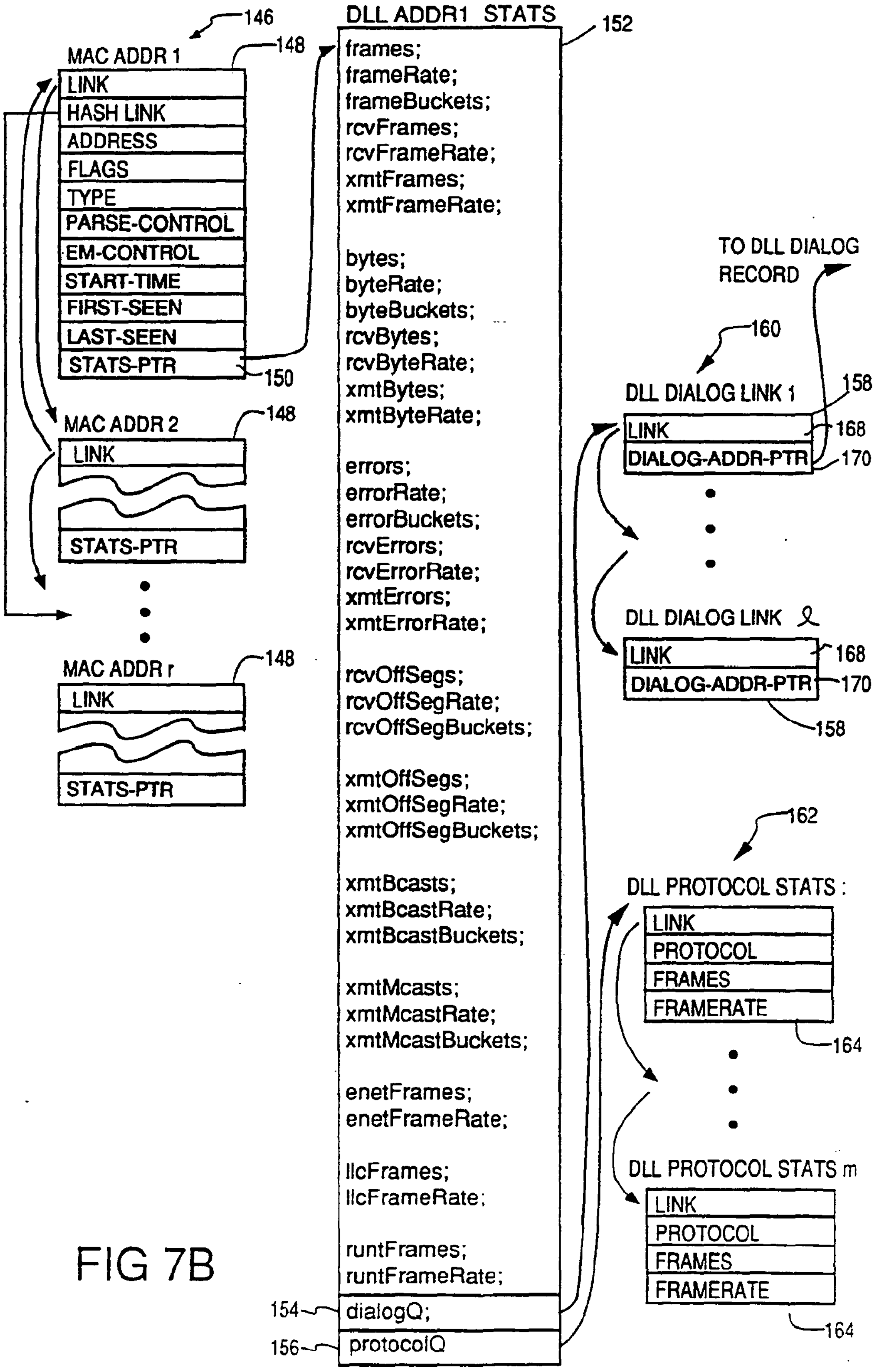

DLL ADDR1 STATS
152
146
MAC ADDR 1
148
LINK
HASH LINK
ADDRESS
FLAGS
TYPE
PARSE-CONTROL
EM-CONTROL
START-TIME
FIRST-SEEN
LAST-SEEN
STATS-PTR
150
MAC ADDR 2
148
LINK
STATS-PTR
MAC ADDR r
148
LINK
STATS-PTR
frames;
frameRate;
frameBuckets;
rcvFrames;
rcvFrameRate;
xmtFrames;
xmtFrameRate;
bytes;
byteRate;
byteBuckets;
rcvBytes;
rcvByteRate;
xmtBytes;
xmtByteRate;
errors;
errorRate;
errorBuckets;
rcvErrors;
rcvErrorRate;
xmtErrors;
xmtErrorRate;
rcvOffSegs;
rcvOffSegRate;
rcvOffSegBuckets;
xmtOffSegs;
xmtOffSegRate;
xmtOffSegBuckets;
xmtBcasts;
xmtBcastRate;
xmtBcastBuckets;
xmtMcasts;
xmtMcastRate;
xmtMcastBuckets;
enetFrames;
enetFrameRate;
llcFrames;
llcFrameRate;
runtFrames;
runtFrameRate;
154
dialogQ;
156
protocolQ
TO DLL DIALOG RECORD
160
DLL DIALOG LINK 1
158
LINK
168
DIALOG-ADDR-PTR
170
DLL DIALOG LINK ℓ
LINK
168
DIALOG-ADDR-PTR
170
158
162
DLL PROTOCOL STATS :
LINK
PROTOCOL
FRAMES
FRAMERATE
164
DLL PROTOCOL STATS m
LINK
PROTOCOL
FRAMES
FRAMERATE
164

FIG 7B

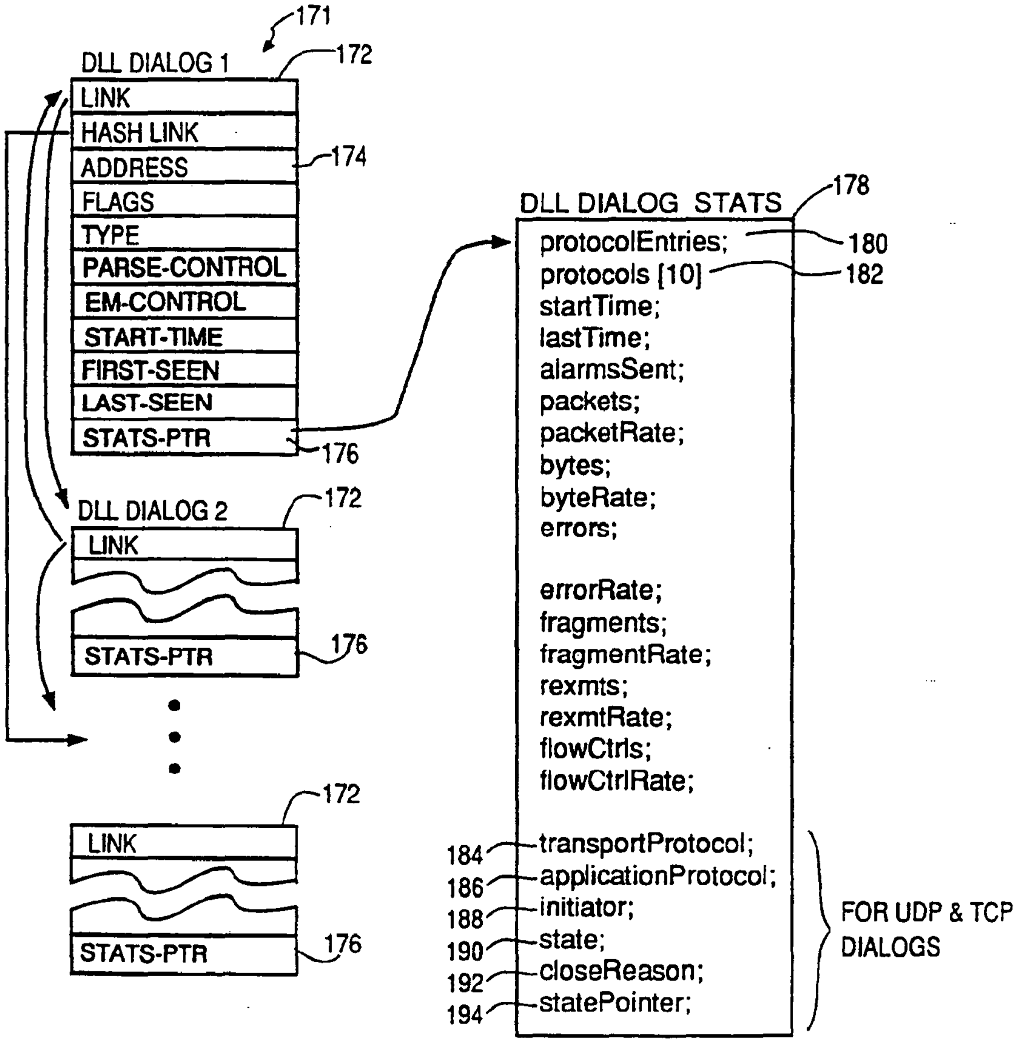


FIG 7C

SUBSTITUTE SHEET

STATE / EVENT	UNKNOWN	CONNECTING	DATA	CLOSING	CLOSED	INACTIVE
UNKNOWN					S= UNKNOWN AFTER CLOSE ++	S= UNKNOWN
CONNECT REQ OR CONNECT CNF (E.G. TCP SYN)	S= CONNECTING	CONNECTION RETRY ++	S= UNKNOWN OUT OF ORDER++ ACTIVE CONN++	S= UNKNOWN OUT OF ORDER++	S=CONNECTING AFTER CLOSE ++	S=CONNECTING
ABORT (E.G. TCP RST)	S= CLOSED START CLOSE TIMER	S= CLOSED FAILED CONN ++ START CLOSE TIMER	S= CLOSED ACTIVE CONN - - START CLOSE TIMER	S= CLOSED ACTIVE CONN - - START CLOSE TIMER	AFTER CLOSE ++	S= CLOSED START CLOSE TIMER
DATA ACK (E.G. TCP ACK)	LOOK FOR DATA STATE	LOOK FOR DATA STATE	LOOK AT HISTORY	LOOK AT HISTORY	S= UNKNOWN AFTER CLOSE ++	S= UNKNOWN LOOK FOR DATA STATE
RELEASE REQ OR CELEASE CNF (E.G. TCP FIN)	S= CLOSING START CLOSE TIMER	S= CLOSING START CLOSE TIMER	S= CLOSING ACTIVE CONN - - START CLOSE TIMER		S= UNKNOWN AFTER CLOSE ++	S= CLOSING
DATA	LOOK FOR DATA STATE	LOOK FOR DATA STATE	LOOK AT HISTORY	OUT OF ORDER++	S= UNKNOWN AFTER CLOSE ++	S= UNKNOWN LOOK FOR DATA STATE
CLOSE TIMER EXPIRES				S= CLOSED		
INACTIVE TIMER EXPIRES	RECYCLE RESOURCES	RECYCLE RESOURCES FAILED CONN++	RECYCLE RESOURCES ACTIVE CONN --	RECYCLE RESOURCES	RECYCLE RESOURCES	RECYCLE RESOURCES

FIG 8

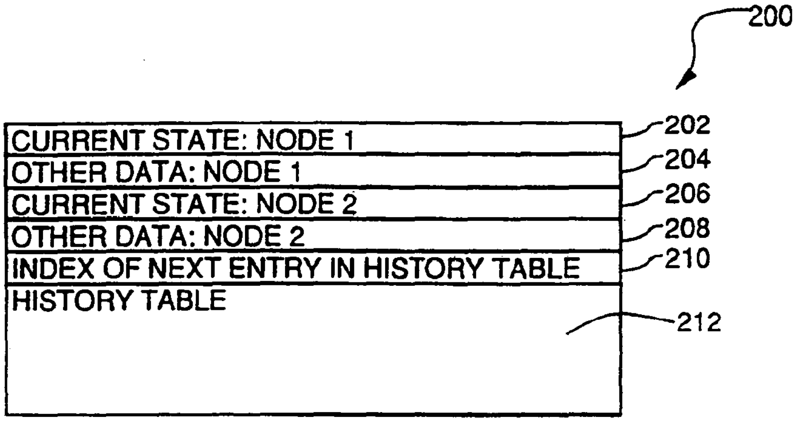

200
CURRENT STATE: NODE 1
202
OTHER DATA: NODE 1
204
CURRENT STATE: NODE 2
206
OTHER DATA: NODE 2
208
INDEX OF NEXT ENTRY IN HISTORY TABLE
210
HISTORY TABLE
212

FIG 9A

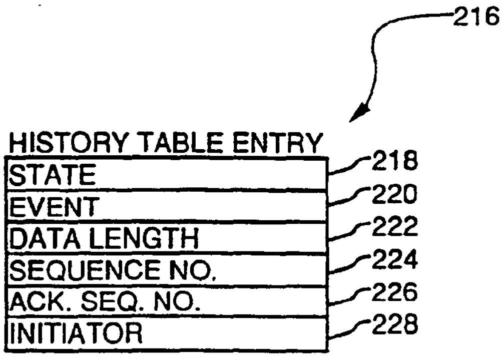

216
HISTORY TABLE ENTRY
STATE
218
EVENT
220
DATA LENGTH
222
SEQUENCE NO.
224
ACK. SEQ. NO.
226
INITIATOR
228

FIG 9B

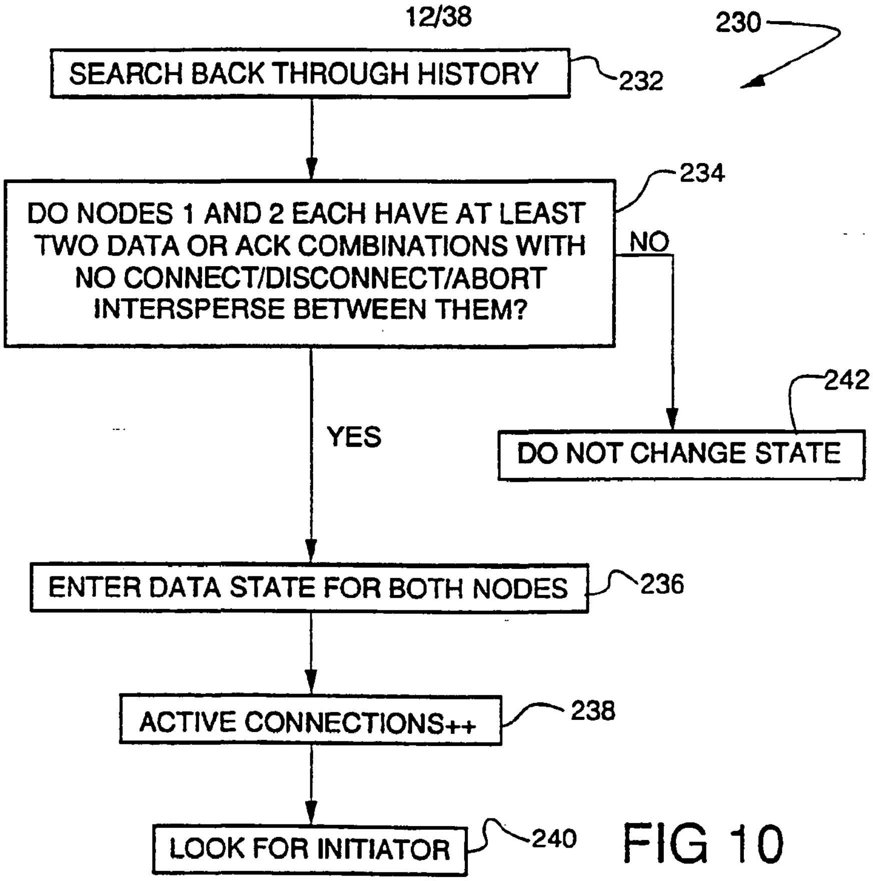

230
SEARCH BACK THROUGH HISTORY
232
DO NODES 1 AND 2 EACH HAVE AT LEAST TWO DATA OR ACK COMBINATIONS WITH NO CONNECT/DISCONNECT/ABORT INTERSPERSE BETWEEN THEM?
234
NO
242
DO NOT CHANGE STATE
YES
ENTER DATA STATE FOR BOTH NODES
236
ACTIVE CONNECTIONS++
238
LOOK FOR INITIATOR
240

FIG 10

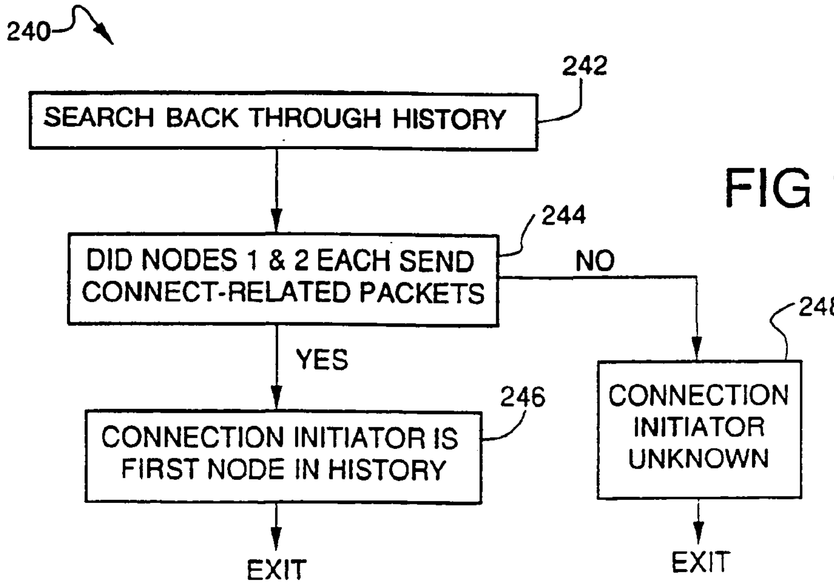

240
242
SEARCH BACK THROUGH HISTORY
244
DID NODES 1 & 2 EACH SEND CONNECT-RELATED PACKETS
NO
248
YES
246
CONNECTION INITIATOR IS FIRST NODE IN HISTORY
CONNECTION INITIATOR UNKNOWN
EXIT
EXIT

FIG 11

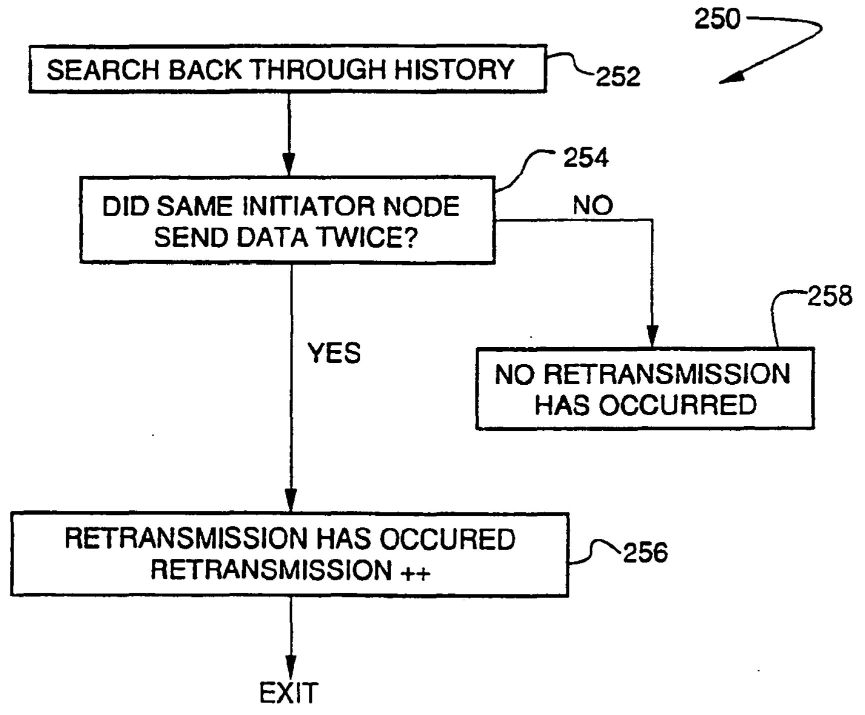

250
SEARCH BACK THROUGH HISTORY
252
DID SAME INITIATOR NODE SEND DATA TWICE?
254
NO
YES
NO RETRANSMISSION HAS OCCURRED
258
RETRANSMISSION HAS OCCURED RETRANSMISSION ++
256
EXIT

FIG 12

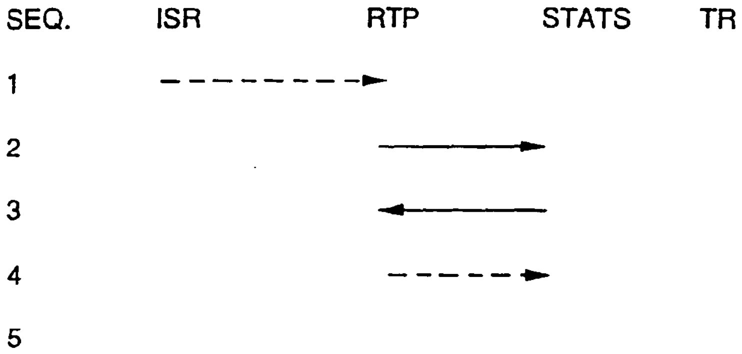


FIG 13

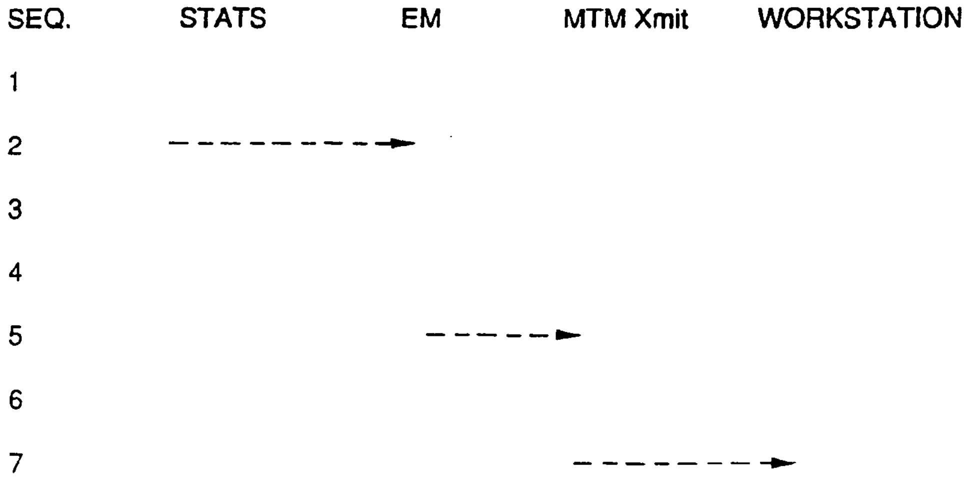


FIG 14

SUBSTITUTE SHEET

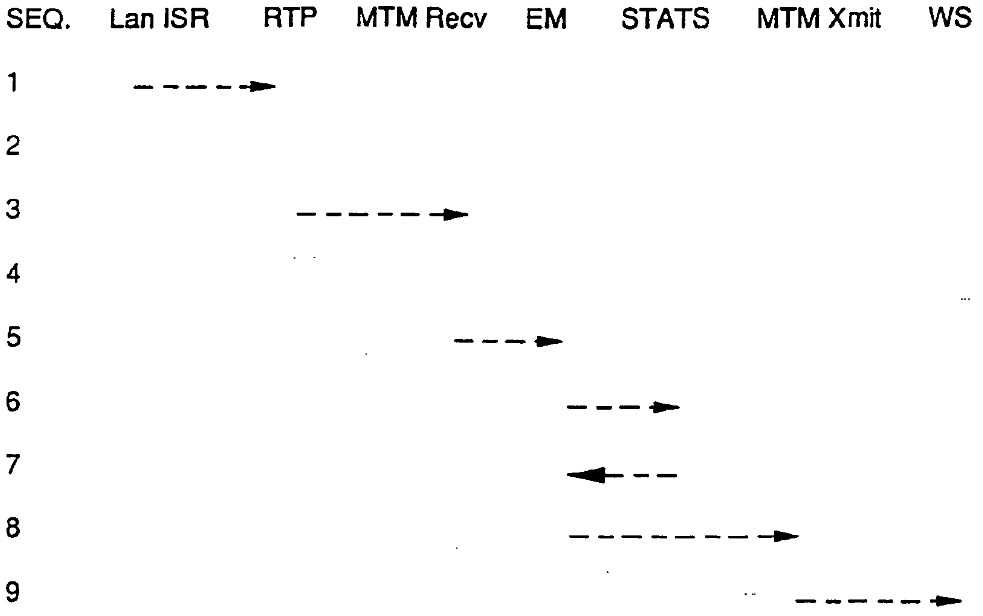

SEQ.
Lan ISR
RTP
MTM Recv
EM
STATS
MTM Xmit
WS
1
2
3
4
5
6
7
8
9

FIG 15

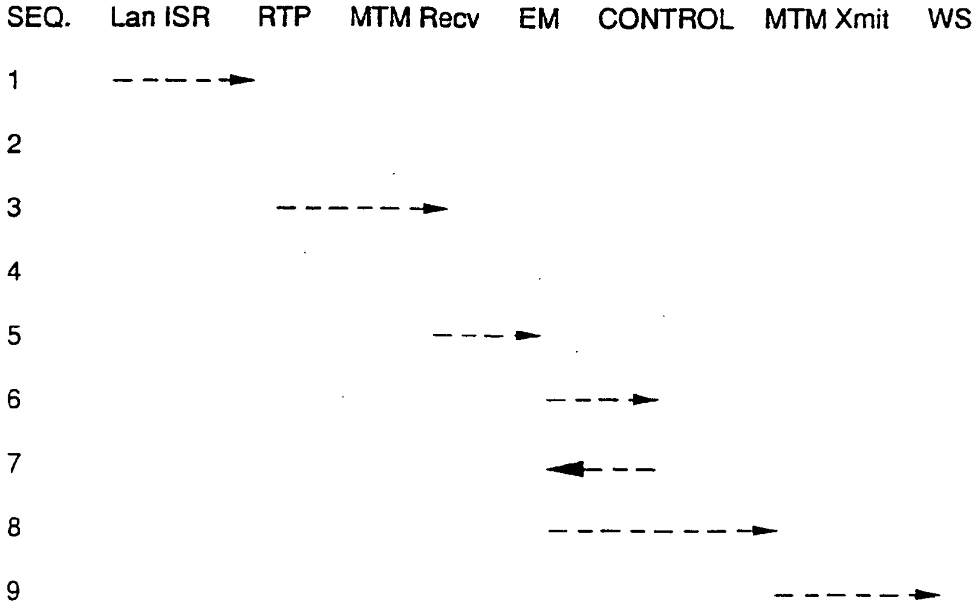

SEQ.
Lan ISR
RTP
MTM Recv
EM
CONTROL
MTM Xmit
WS
1
2
3
4
5
6
7
8
9

FIG 16

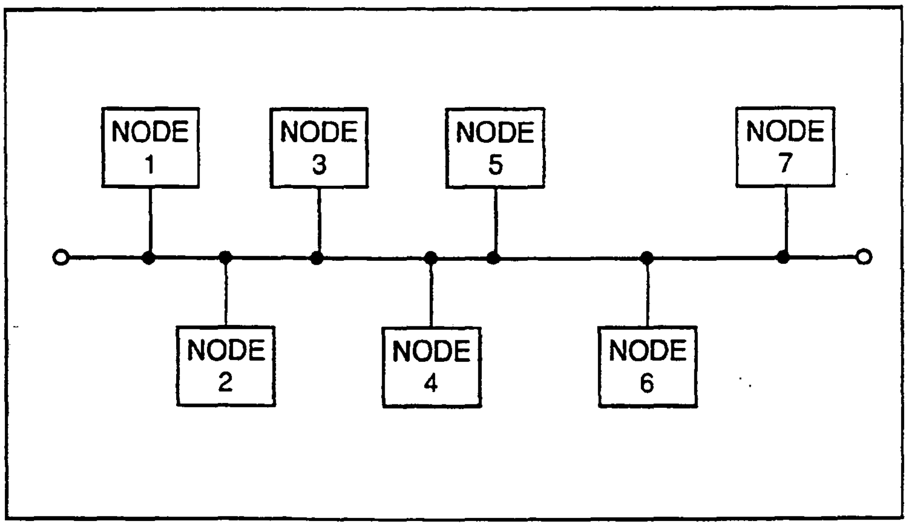
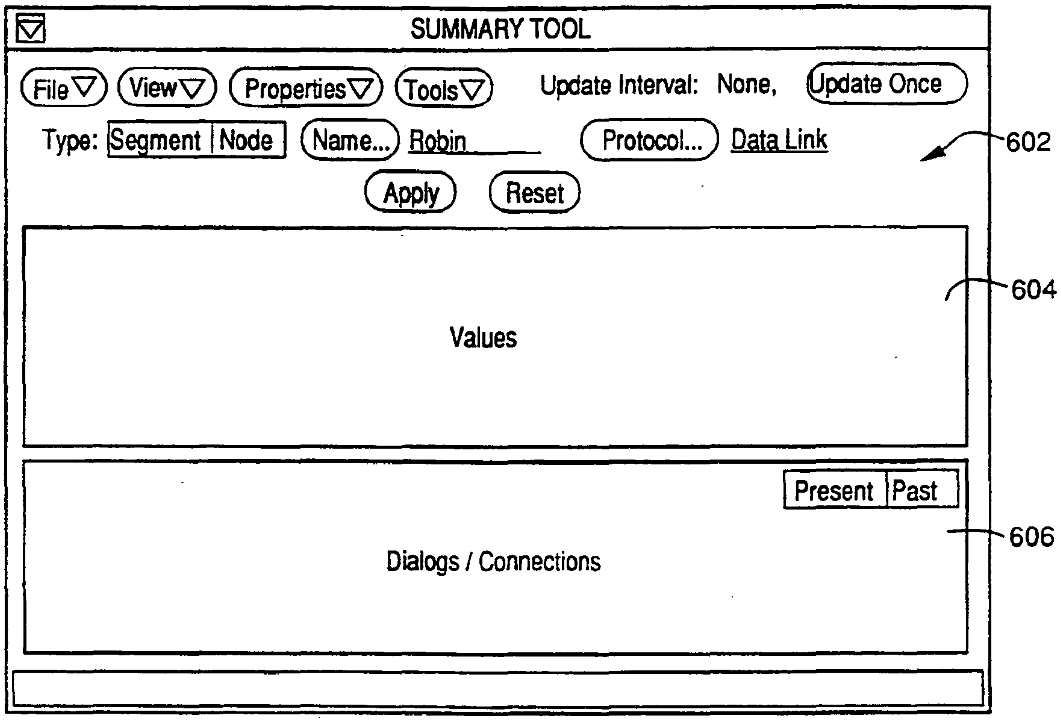


FIG 17



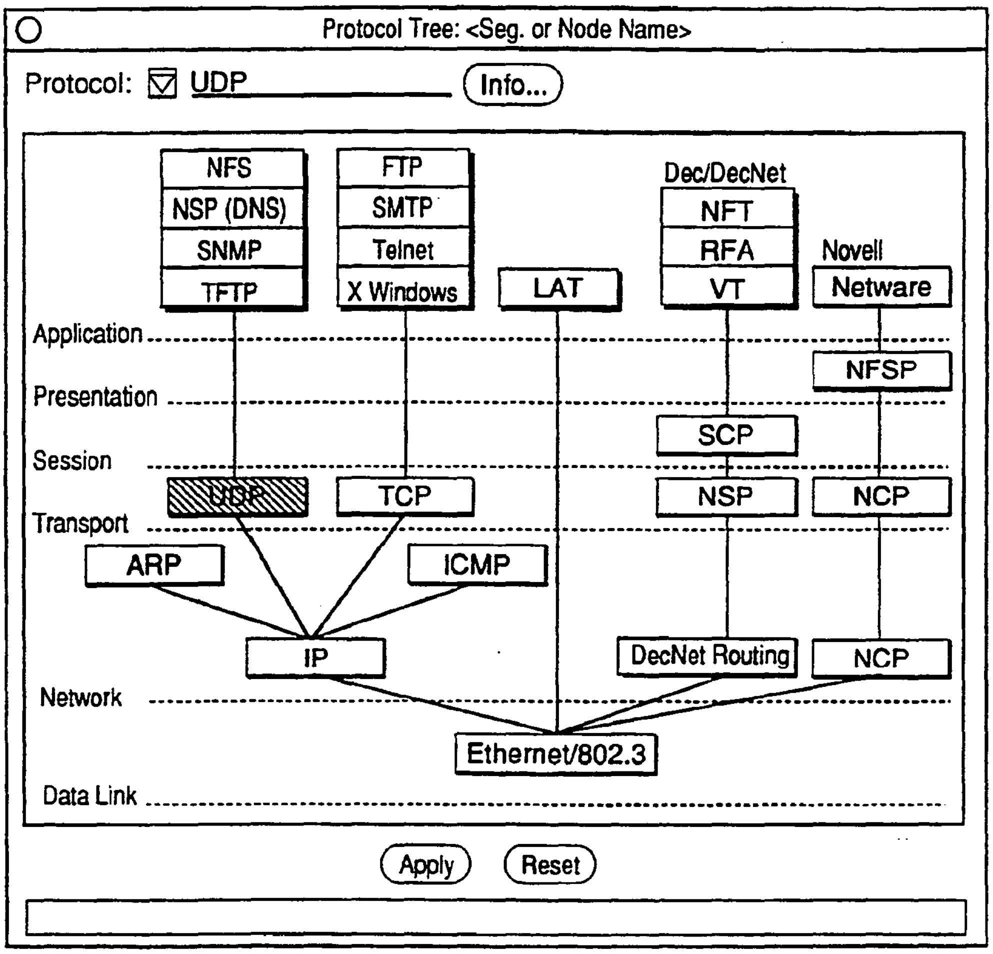
Update Interval: None,

Type: Segment | Node

Name... Robin

Protocol... Data Link

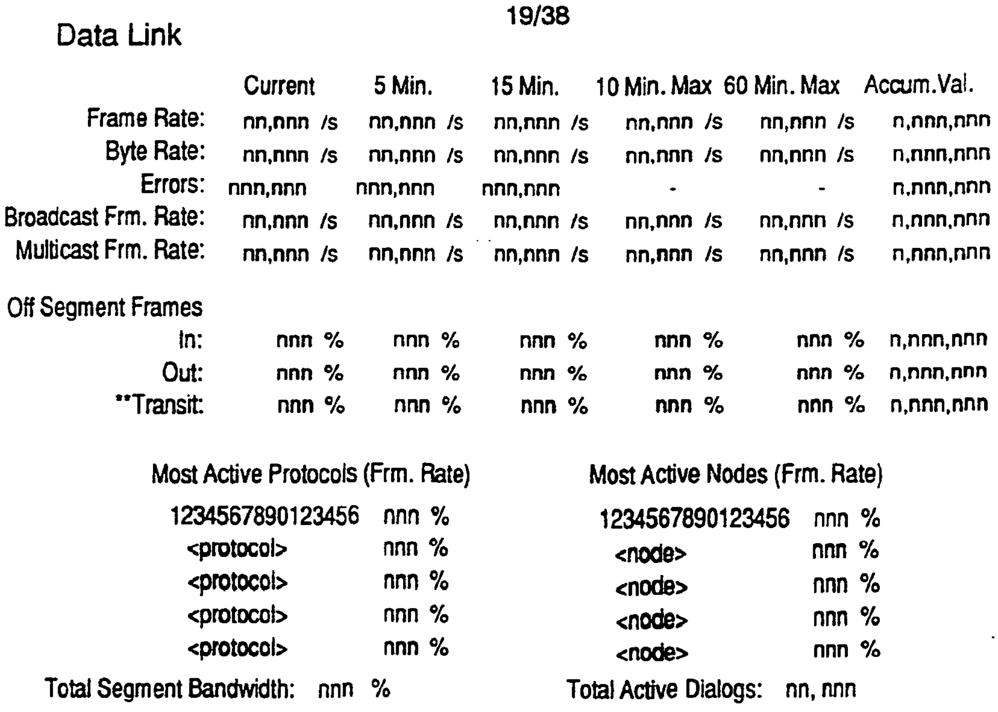
FIG 18



Protocol Tree: <Seg. or Node Name>

Protocol: ☑ UDP

FIG 19



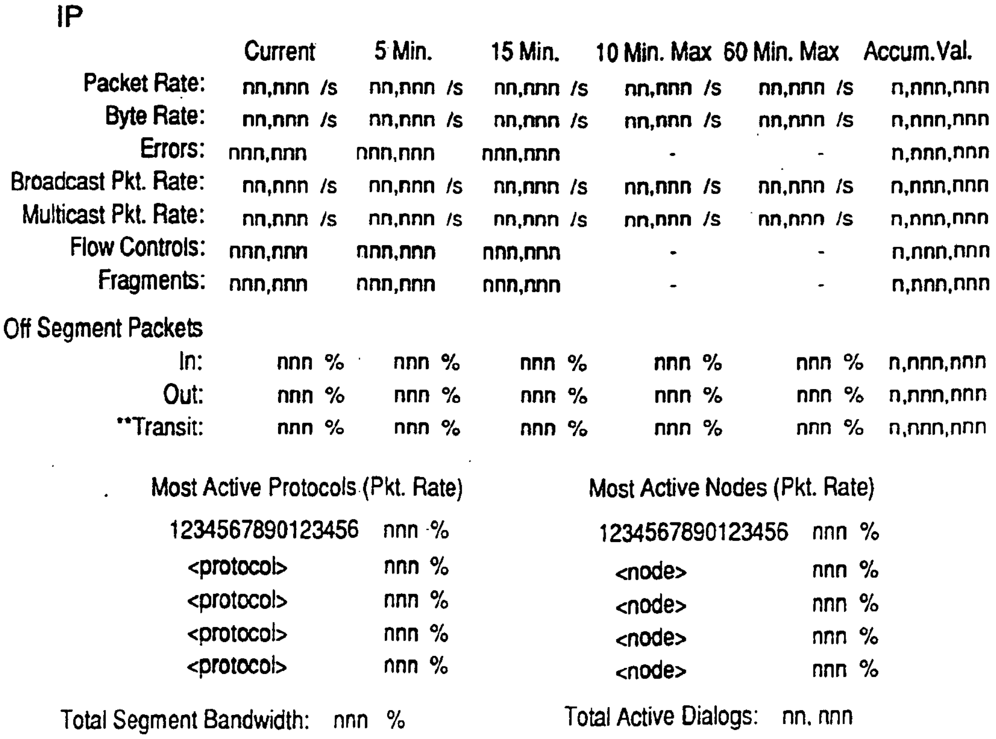
Data Link

	Current	5 Min.	15 Min.	10 Min. Max	60 Min. Max	Accum.Val.
Frame Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Byte Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Errors:	nnn,nnn	nnn,nnn	nnn,nnn	-	-	n,nnn,nnn
Broadcast Frm. Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Multicast Frm. Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Off Segment Frames						
In:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn
Out:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn
**Transit:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn

Most Active Protocols (Frm. Rate)		Most Active Nodes (Frm. Rate)	
1234567890123456	nnn %	1234567890123456	nnn %
<protocol>	nnn %	<node>	nnn %
<protocol>	nnn %	<node>	nnn %
<protocol>	nnn %	<node>	nnn %
<protocol>	nnn %	<node>	nnn %

Total Segment Bandwidth: nnn % Total Active Dialogs: nn, nnn

FIG 20A



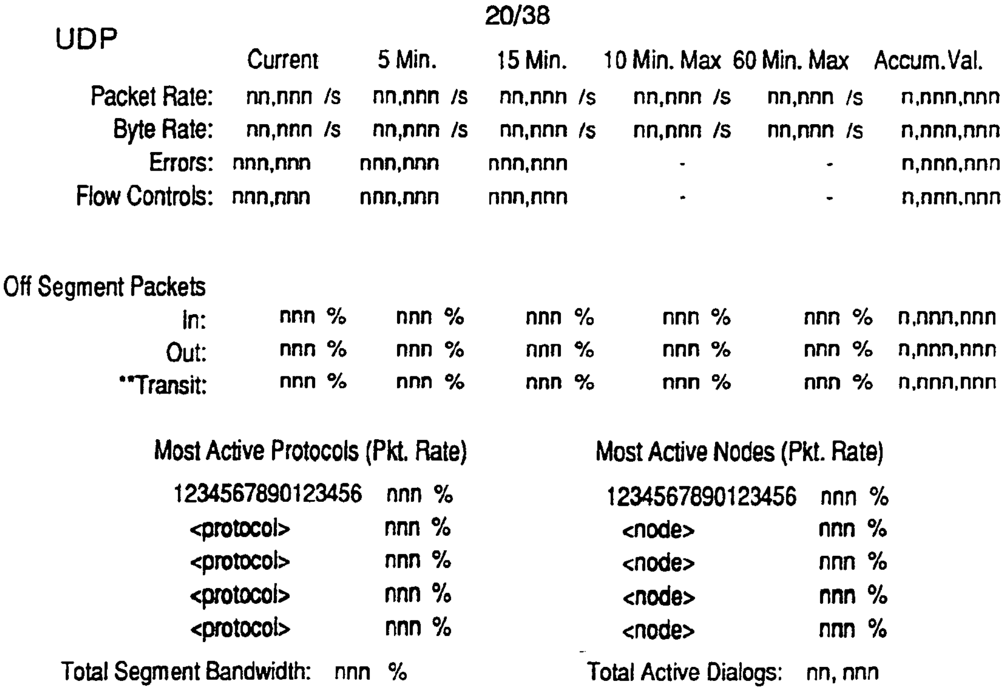
IP

	Current	5 Min.	15 Min.	10 Min. Max	60 Min. Max	Accum.Val.
Packet Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Byte Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Errors:	nnn,nnn	nnn,nnn	nnn,nnn	-	-	n,nnn,nnn
Broadcast Pkt. Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Multicast Pkt. Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Flow Controls:	nnn,nnn	nnn,nnn	nnn,nnn	-	-	n,nnn,nnn
Fragments:	nnn,nnn	nnn,nnn	nnn,nnn	-	-	n,nnn,nnn
Off Segment Packets						
In:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn
Out:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn
**Transit:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn

Most Active Protocols (Pkt. Rate)		Most Active Nodes (Pkt. Rate)	
1234567890123456	nnn %	1234567890123456	nnn %
<protocol>	nnn %	<node>	nnn %
<protocol>	nnn %	<node>	nnn %
<protocol>	nnn %	<node>	nnn %
<protocol>	nnn %	<node>	nnn %

Total Segment Bandwidth: nnn % Total Active Dialogs: nn, nnn

FIG 20B



UDP

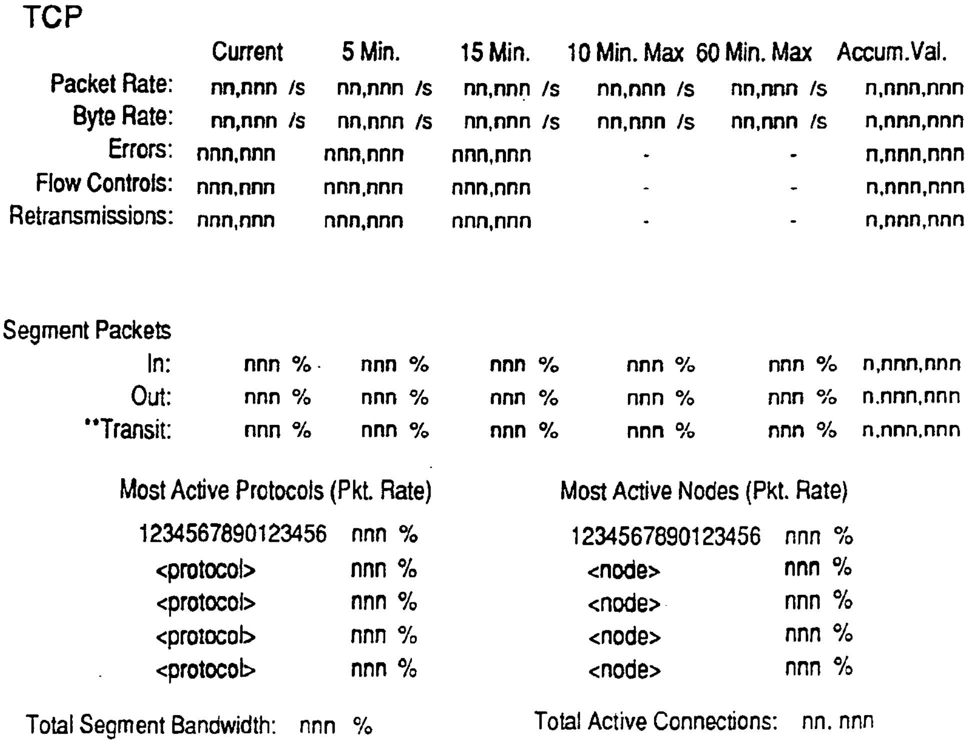
	Current	5 Min.	15 Min.	10 Min. Max	60 Min. Max	Accum.Val.
Packet Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Byte Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Errors:	nnn,nnn	nnn,nnn	nnn,nnn	-	-	n,nnn,nnn
Flow Controls:	nnn,nnn	nnn,nnn	nnn,nnn	-	-	n,nnn,nnn

Off Segment Packets						
In:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn
Out:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn
**Transit:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn

Most Active Protocols (Pkt. Rate)		Most Active Nodes (Pkt. Rate)	
1234567890123456	nnn %	1234567890123456	nnn %
<protocol>	nnn %	<node>	nnn %
<protocol>	nnn %	<node>	nnn %
<protocol>	nnn %	<node>	nnn %
<protocol>	nnn %	<node>	nnn %

Total Segment Bandwidth: nnn % Total Active Dialogs: nn, nnn

FIG 20C



TCP

	Current	5 Min.	15 Min.	10 Min. Max	60 Min. Max	Accum.Val.
Packet Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Byte Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Errors:	nnn,nnn	nnn,nnn	nnn,nnn	-	-	n,nnn,nnn
Flow Controls:	nnn,nnn	nnn,nnn	nnn,nnn	-	-	n,nnn,nnn
Retransmissions:	nnn,nnn	nnn,nnn	nnn,nnn	-	-	n,nnn,nnn

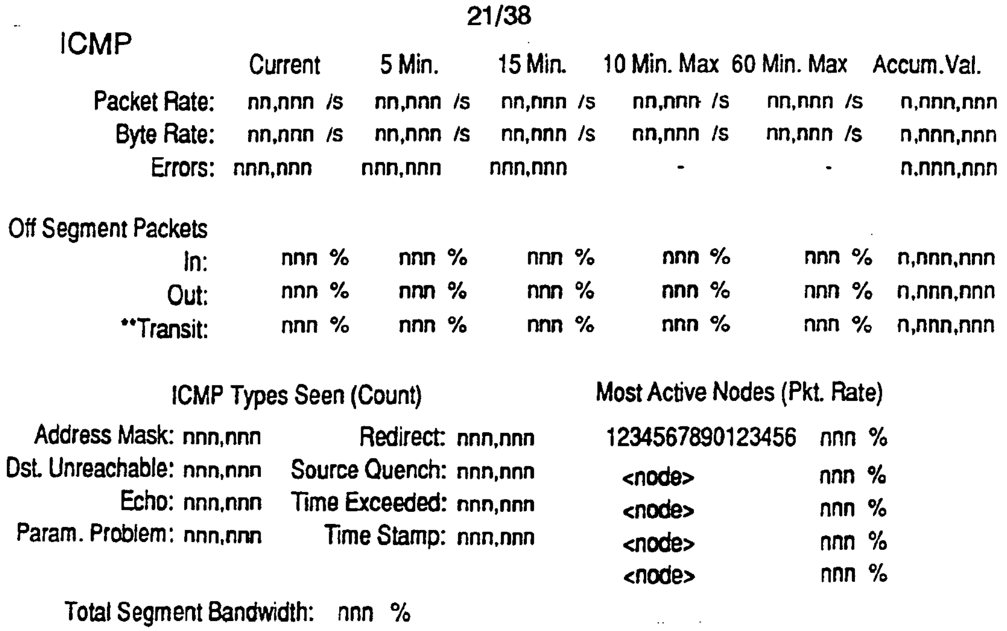
Off Segment Packets						
In:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn
Out:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn
**Transit:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn

Most Active Protocols (Pkt. Rate)		Most Active Nodes (Pkt. Rate)	
1234567890123456	nnn %	1234567890123456	nnn %
<protocol>	nnn %	<node>	nnn %
<protocol>	nnn %	<node>	nnn %
<protocol>	nnn %	<node>	nnn %
<protocol>	nnn %	<node>	nnn %

Total Segment Bandwidth: nnn % Total Active Connections: nn. nnn

FIG 20D

SUBSTITUTE SHEET



ICMP

	Current	5 Min.	15 Min.	10 Min. Max	60 Min. Max	Accum.Val.
Packet Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Byte Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Errors:	nnn,nnn	nnn,nnn	nnn,nnn	-	-	n,nnn,nnn
Off Segment Packets						
In:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn
Out:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn
**Transit:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn

ICMP Types Seen (Count)

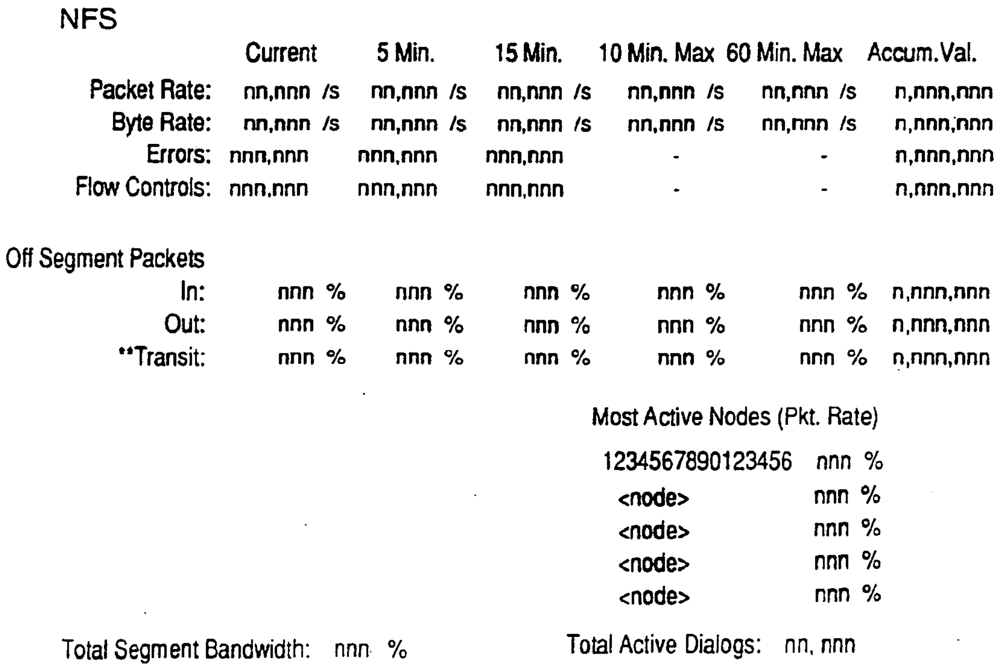
Address Mask:	nnn,nnn	Redirect:	nnn,nnn
Dst. Unreachable:	nnn,nnn	Source Quench:	nnn,nnn
Echo:	nnn,nnn	Time Exceeded:	nnn,nnn
Param. Problem:	nnn,nnn	Time Stamp:	nnn,nnn

Most Active Nodes (Pkt. Rate)

1234567890123456	nnn %
<node>	nnn %
<node>	nnn %
<node>	nnn %
<node>	nnn %

Total Segment Bandwidth: nnn %

FIG 20E



NFS

	Current	5 Min.	15 Min.	10 Min. Max	60 Min. Max	Accum.Val.
Packet Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Byte Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn;nnn
Errors:	nnn,nnn	nnn,nnn	nnn,nnn	-	-	n,nnn,nnn
Flow Controls:	nnn,nnn	nnn,nnn	nnn,nnn	-	-	n,nnn,nnn
Off Segment Packets						
In:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn
Out:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn
**Transit:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn

Most Active Nodes (Pkt. Rate)

1234567890123456	nnn %
<node>	nnn %
<node>	nnn %
<node>	nnn %
<node>	nnn %

Total Segment Bandwidth: nnn %

Total Active Dialogs: nn, nnn

FIG 20F

SUBSTITUTE SHEET

Arp/Rarp	Current	5 Min.	15 Min.	10 Min. Max	60 Min. Max	Accum.Val.
Packet Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Byte Rate:	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	nn,nnn /s	n,nnn,nnn
Errors:	nnn,nnn	nnn,nnn	nnn,nnn	-	-	n,nnn,nnn
Off Segment Packets						
In:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn
Out:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn
**Transit:	nnn %	nnn %	nnn %	nnn %	nnn %	n,nnn,nnn

Most Active Nodes (Pkt. Rate)	
1234567890123456	nnn %
<node>	nnn %
<node>	nnn %
<node>	nnn %
<node>	nnn %

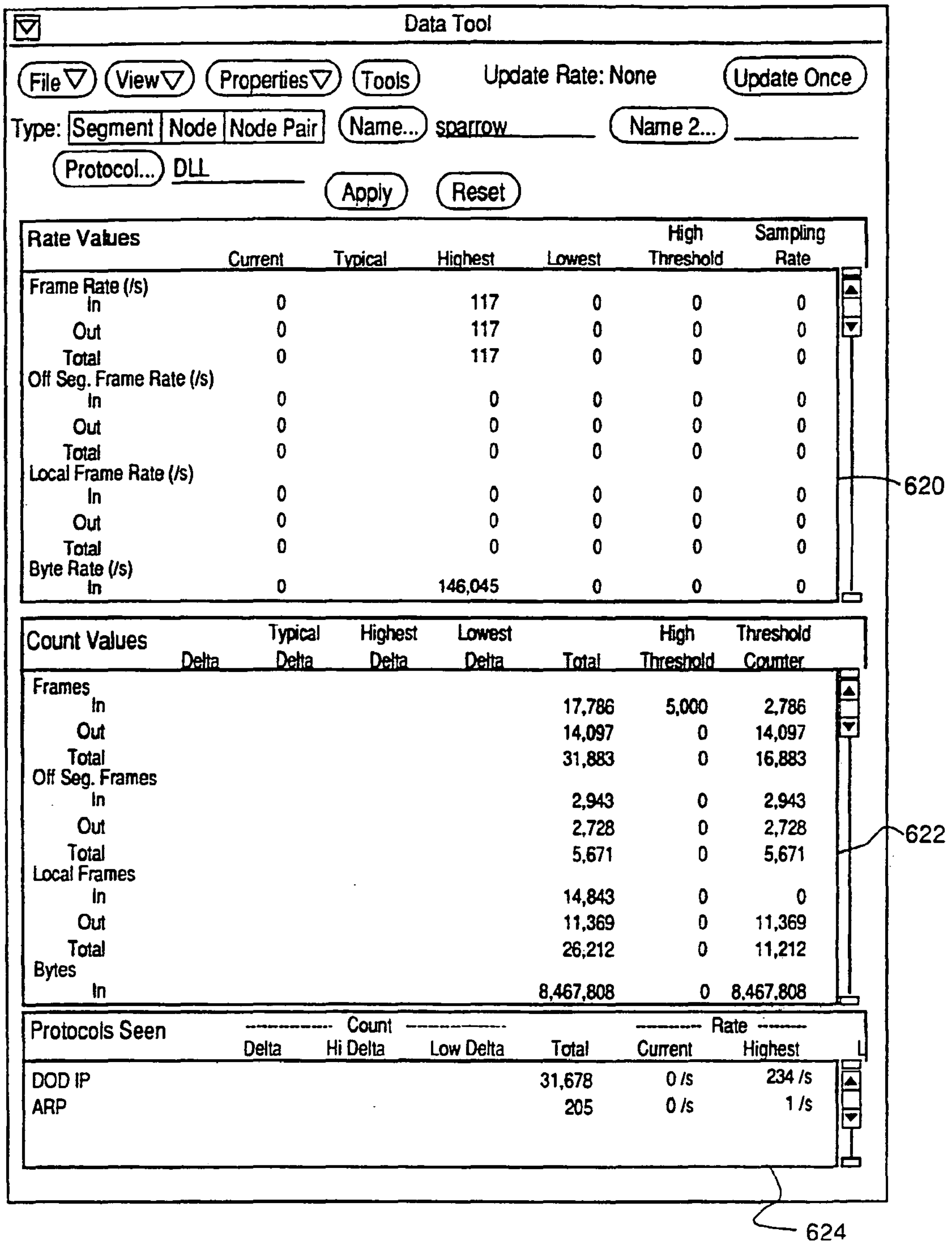
Total Segment Bandwidth: nnn %

FIG 20G

					Packets			
Start	Last				Summary			
Time	Seen	Dir.	Partner Node	Protocols	Rate	%	Count	Errors
hh:mm:ss	hh:mm:ss	1234	1234567890123456	1234567890123456	nn,nnn /s	nnn %	n,nnn,nnn	nnn,nnn
10:23:04	15:31:47	To	robin	XNS,XEROX-PUP	325 /s	6%	2,641	0
07:21:38	13:25:51	From	hawk	DOD-IP, X25 BBN-SIMNET	87 /s	3%	127	1
10/31/90	08:22:30	?	hawk	APPLETALK	13 /s	1%	24,192	4

FIG 21

SUBSTITUTE SHEET



Update Rate: None

Type: Segment | Node | Node Pair

Name... sparrow

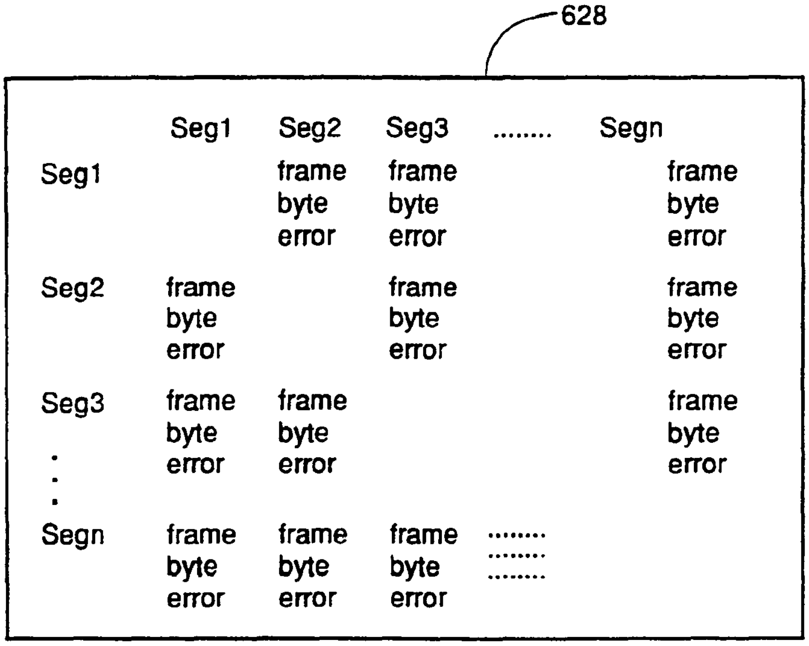
Protocol... DLL

Rate Values	Current	Typical	Highest	Lowest	High Threshold	Sampling Rate
Frame Rate (/s)						
In	0		117	0	0	0
Out	0		117	0	0	0
Total	0		117	0	0	0
Off Seg. Frame Rate (/s)						
In	0		0	0	0	0
Out	0		0	0	0	0
Total	0		0	0	0	0
Local Frame Rate (/s)						
In	0		0	0	0	0
Out	0		0	0	0	0
Total	0		0	0	0	0
Byte Rate (/s)						
In	0		146,045	0	0	0

Count Values	Delta	Typical Delta	Highest Delta	Lowest Delta	Total	High Threshold	Threshold Counter
Frames							
In					17,786	5,000	2,786
Out					14,097	0	14,097
Total					31,883	0	16,883
Off Seg. Frames							
In					2,943	0	2,943
Out					2,728	0	2,728
Total					5,671	0	5,671
Local Frames							
In					14,843	0	0
Out					11,369	0	11,369
Total					26,212	0	11,212
Bytes							
In					8,467,808	0	8,467,808

Protocols Seen	Count				Rate	
	Delta	Hi Delta	Low Delta	Total	Current	Highest
DOD IP				31,678	0 /s	234 /s
ARP				205	0 /s	1 /s

FIG 22



	Seg1	Seg2	Seg3	....	Segn
Seg1		frame byte error	frame byte error		frame byte error
Seg2	frame byte error		frame byte error		frame byte error
Seg3	frame byte error	frame byte error			frame byte error
⋮					
Segn	frame byte error	frame byte error	frame byte error	....	

FIG 23

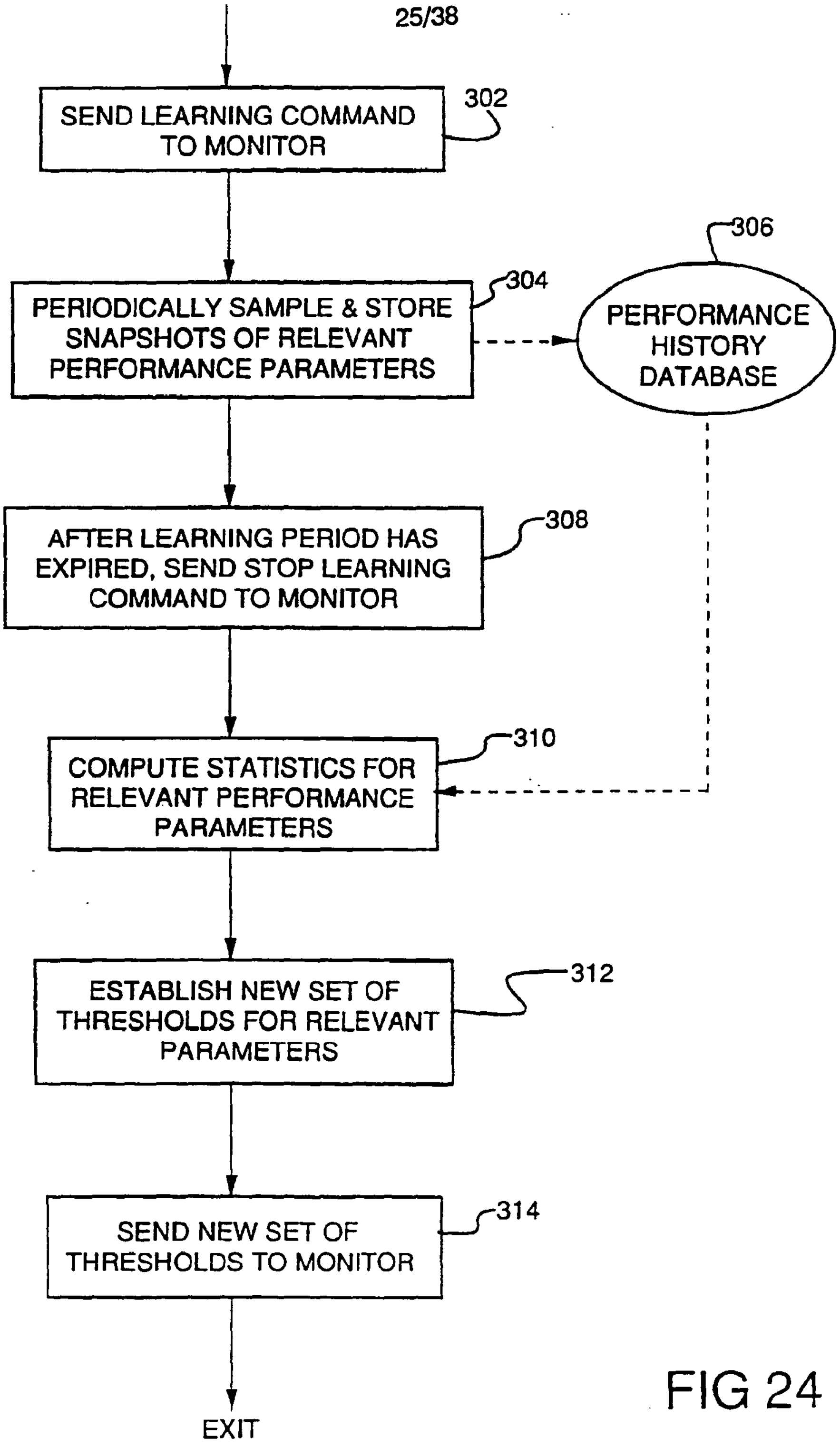

SEND LEARNING COMMAND TO MONITOR
302
PERIODICALLY SAMPLE & STORE SNAPSHOTS OF RELEVANT PERFORMANCE PARAMETERS
304
PERFORMANCE HISTORY DATABASE
306
AFTER LEARNING PERIOD HAS EXPIRED, SEND STOP LEARNING COMMAND TO MONITOR
308
COMPUTE STATISTICS FOR RELEVANT PERFORMANCE PARAMETERS
310
ESTABLISH NEW SET OF THRESHOLDS FOR RELEVANT PARAMETERS
312
SEND NEW SET OF THRESHOLDS TO MONITOR
314
EXIT

FIG 24

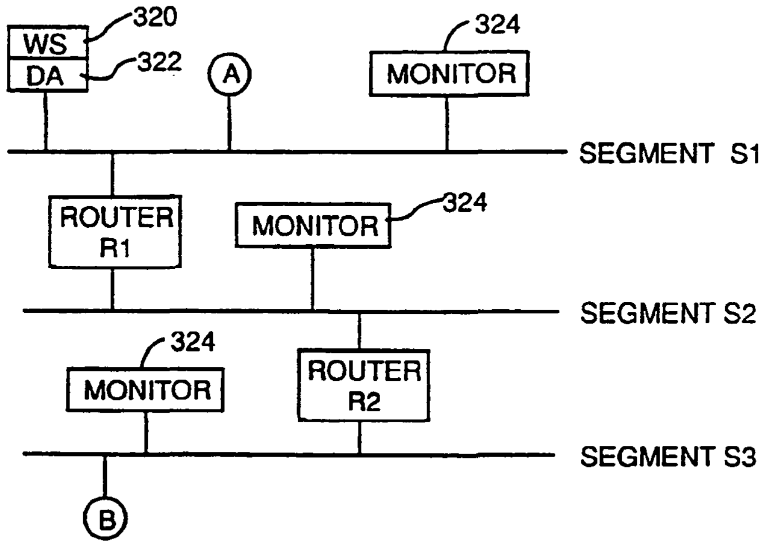

320
WS
DA
322
A
324
MONITOR
SEGMENT S1
ROUTER R1
MONITOR
324
SEGMENT S2
324
MONITOR
ROUTER R2
SEGMENT S3
B

FIG 25

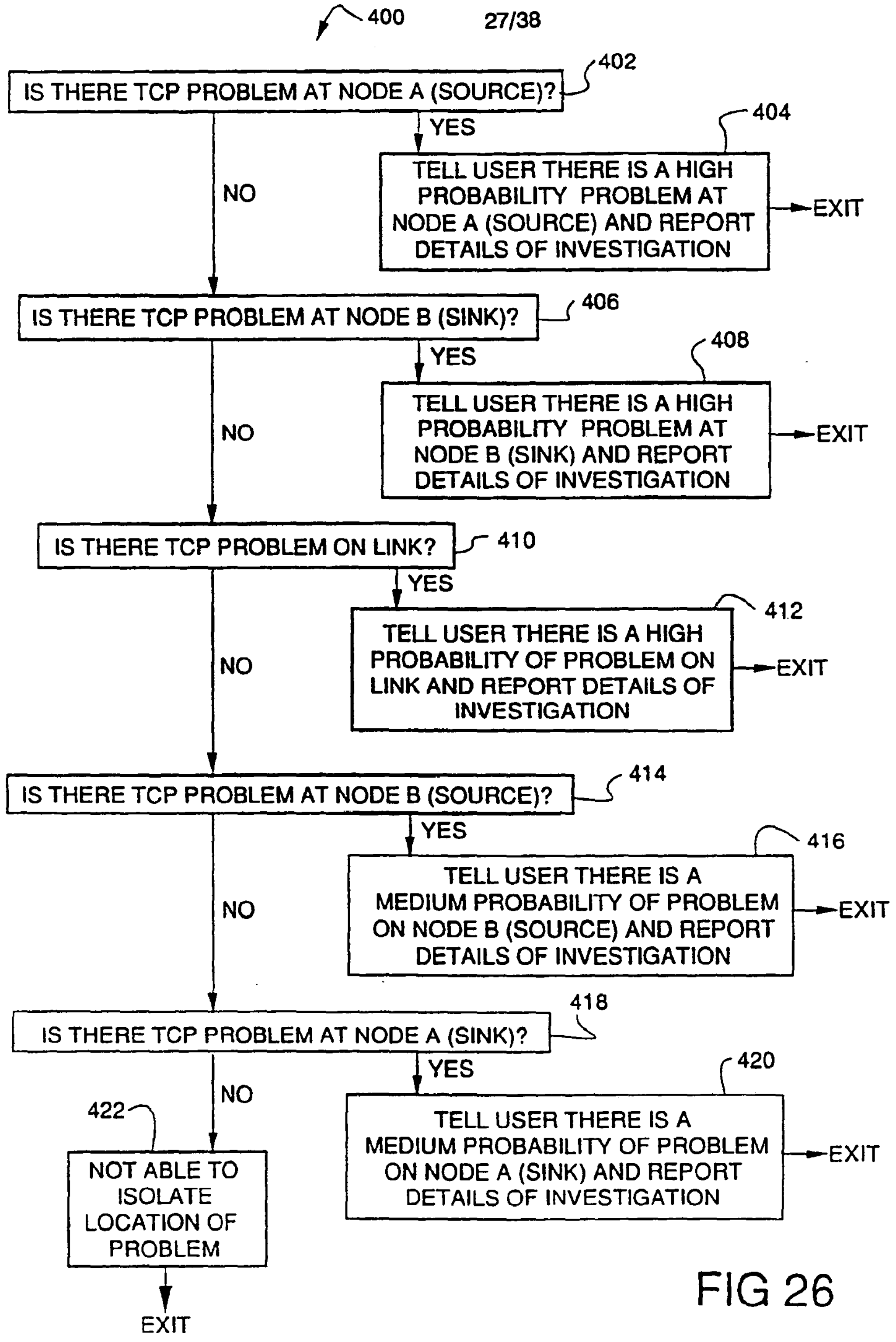

400
402
IS THERE TCP PROBLEM AT NODE A (SOURCE)?
YES
NO
404
TELL USER THERE IS A HIGH PROBABILITY PROBLEM AT NODE A (SOURCE) AND REPORT DETAILS OF INVESTIGATION
EXIT
406
IS THERE TCP PROBLEM AT NODE B (SINK)?
YES
NO
408
TELL USER THERE IS A HIGH PROBABILITY PROBLEM AT NODE B (SINK) AND REPORT DETAILS OF INVESTIGATION
EXIT
410
IS THERE TCP PROBLEM ON LINK?
YES
NO
412
TELL USER THERE IS A HIGH PROBABILITY OF PROBLEM ON LINK AND REPORT DETAILS OF INVESTIGATION
EXIT
414
IS THERE TCP PROBLEM AT NODE B (SOURCE)?
YES
NO
416
TELL USER THERE IS A MEDIUM PROBABILITY OF PROBLEM ON NODE B (SOURCE) AND REPORT DETAILS OF INVESTIGATION
EXIT
418
IS THERE TCP PROBLEM AT NODE A (SINK)?
YES
NO
420
TELL USER THERE IS A MEDIUM PROBABILITY OF PROBLEM ON NODE A (SINK) AND REPORT DETAILS OF INVESTIGATION
EXIT
422
NOT ABLE TO ISOLATE LOCATION OF PROBLEM
EXIT

FIG 26

SUBSTITUTE SHEET

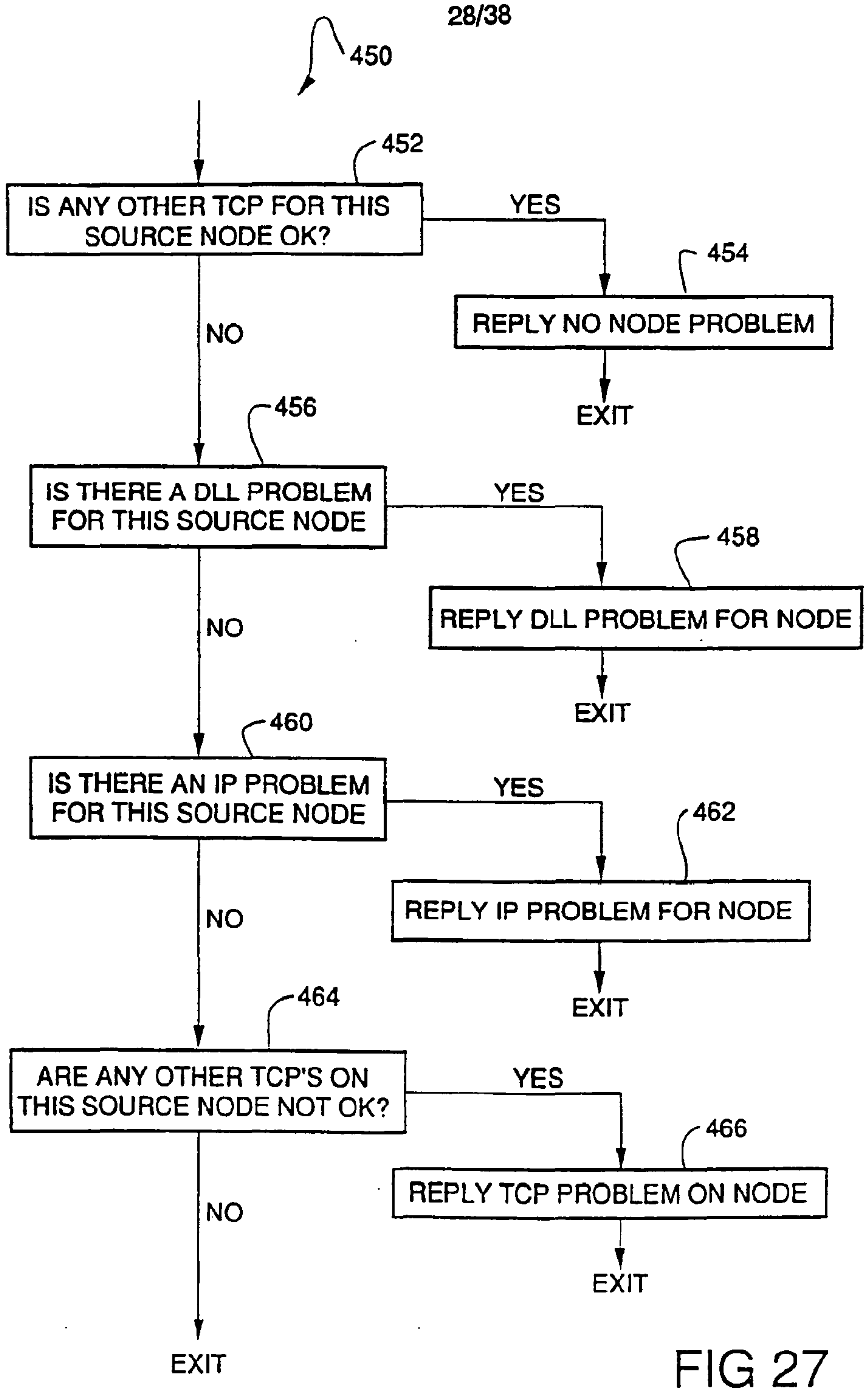

450
452
IS ANY OTHER TCP FOR THIS SOURCE NODE OK?
YES
454
REPLY NO NODE PROBLEM
EXIT
NO
456
IS THERE A DLL PROBLEM FOR THIS SOURCE NODE
YES
458
REPLY DLL PROBLEM FOR NODE
EXIT
NO
460
IS THERE AN IP PROBLEM FOR THIS SOURCE NODE
YES
462
REPLY IP PROBLEM FOR NODE
EXIT
NO
464
ARE ANY OTHER TCP'S ON THIS SOURCE NODE NOT OK?
YES
466
REPLY TCP PROBLEM ON NODE
EXIT
NO
EXIT

FIG 27

SUBSTITUTE SHEET

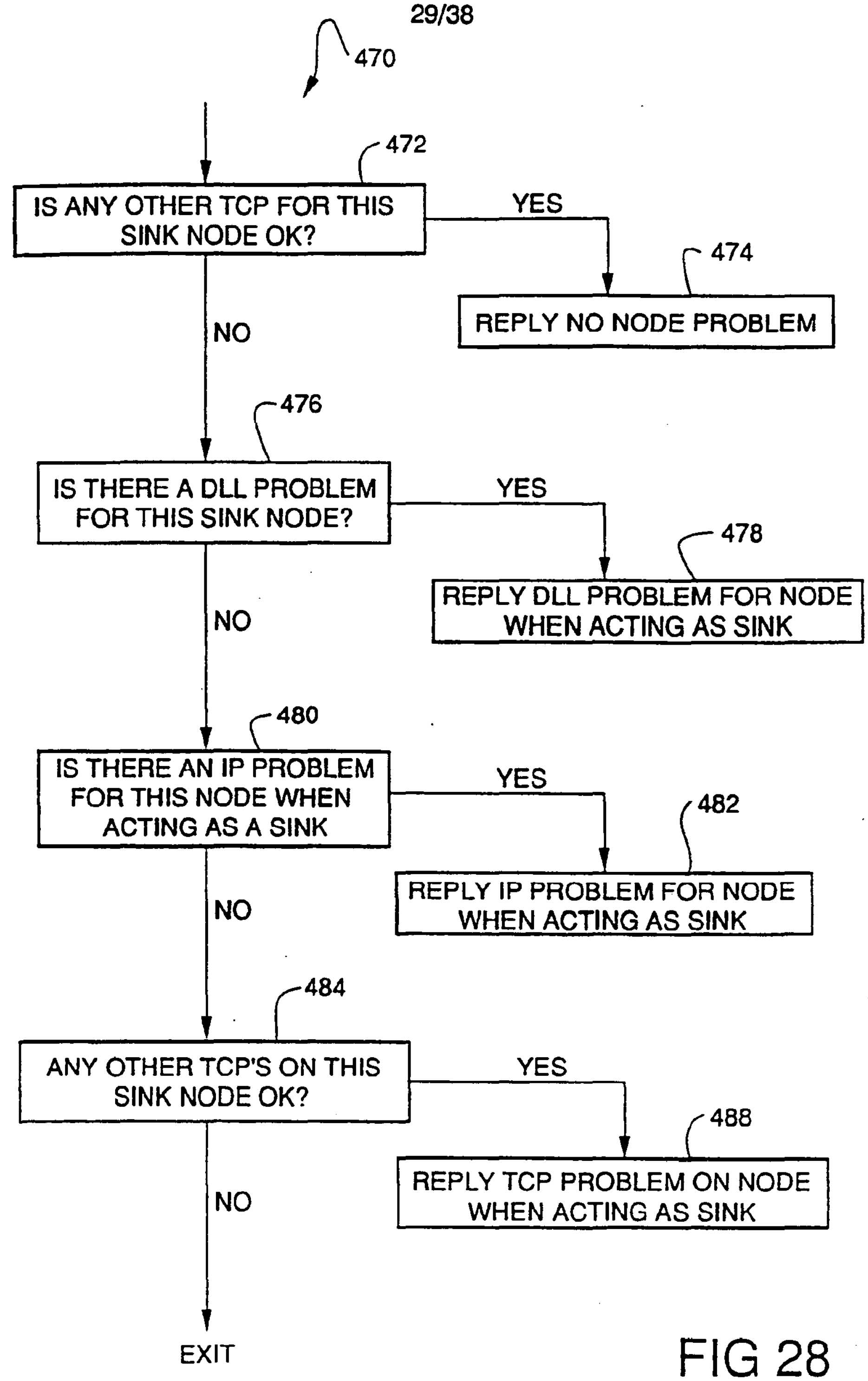

470
472
IS ANY OTHER TCP FOR THIS SINK NODE OK?
YES
474
REPLY NO NODE PROBLEM
NO
476
IS THERE A DLL PROBLEM FOR THIS SINK NODE?
YES
478
REPLY DLL PROBLEM FOR NODE WHEN ACTING AS SINK
NO
480
IS THERE AN IP PROBLEM FOR THIS NODE WHEN ACTING AS A SINK
YES
482
REPLY IP PROBLEM FOR NODE WHEN ACTING AS SINK
NO
484
ANY OTHER TCP'S ON THIS SINK NODE OK?
YES
488
REPLY TCP PROBLEM ON NODE WHEN ACTING AS SINK
NO
EXIT

FIG 28

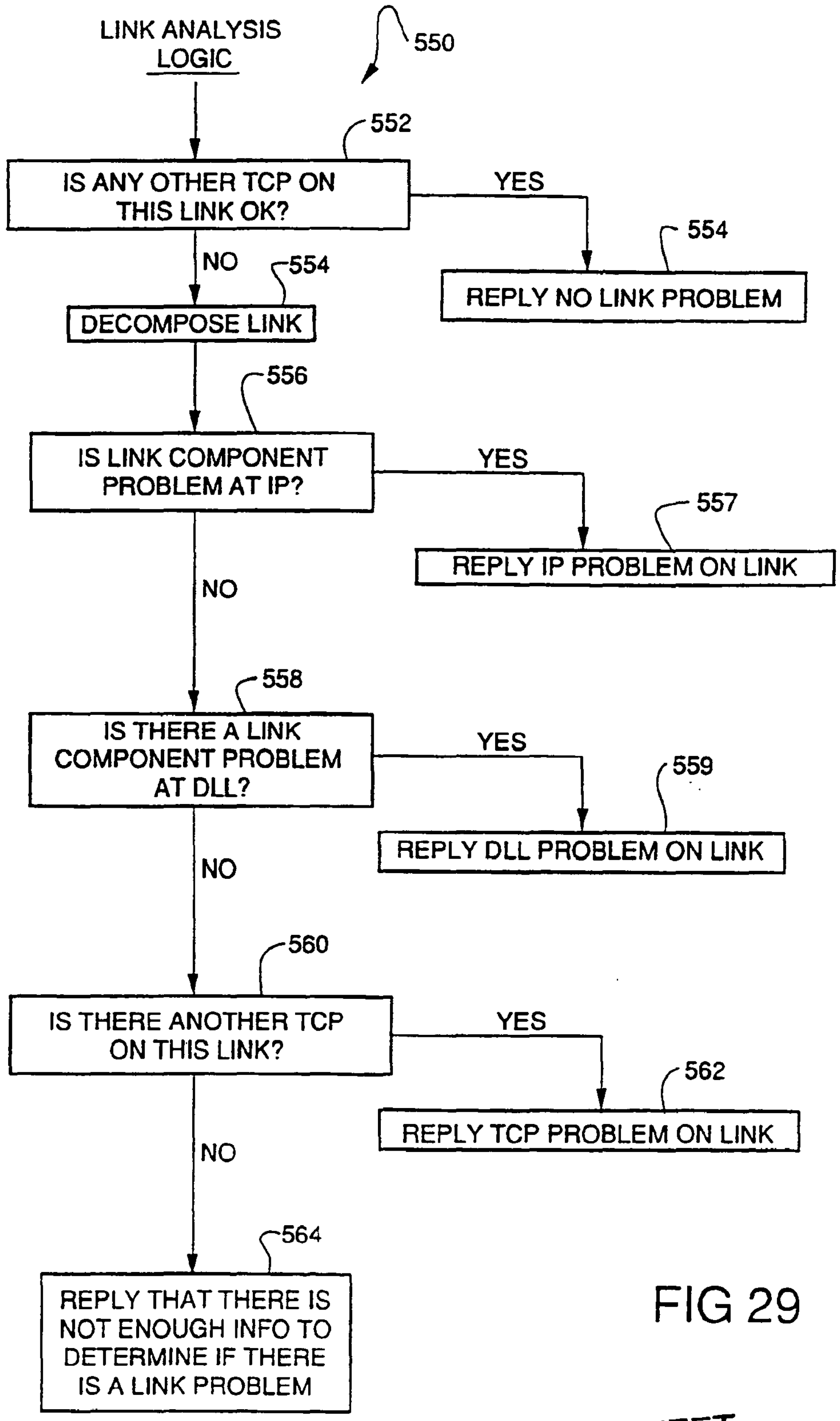

LINK ANALYSIS LOGIC
550
552
IS ANY OTHER TCP ON THIS LINK OK?
YES
554
REPLY NO LINK PROBLEM
NO
554
DECOMPOSE LINK
556
IS LINK COMPONENT PROBLEM AT IP?
YES
557
REPLY IP PROBLEM ON LINK
NO
558
IS THERE A LINK COMPONENT PROBLEM AT DLL?
YES
559
REPLY DLL PROBLEM ON LINK
NO
560
IS THERE ANOTHER TCP ON THIS LINK?
YES
562
REPLY TCP PROBLEM ON LINK
NO
564
REPLY THAT THERE IS NOT ENOUGH INFO TO DETERMINE IF THERE IS A LINK PROBLEM

FIG 29

SUBSTITUTE SHEET

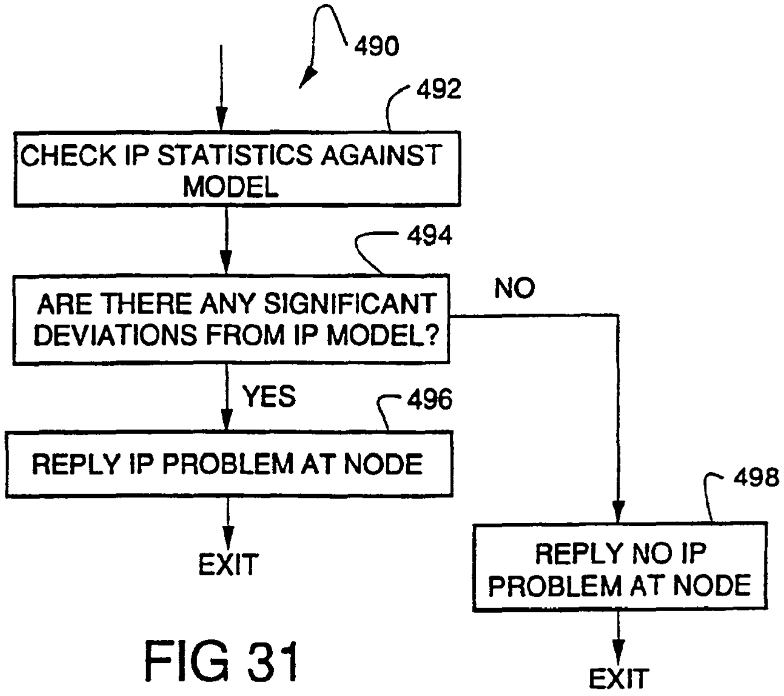

490
492
CHECK IP STATISTICS AGAINST MODEL
494
ARE THERE ANY SIGNIFICANT DEVIATIONS FROM IP MODEL?
NO
YES
496
REPLY IP PROBLEM AT NODE
EXIT
498
REPLY NO IP PROBLEM AT NODE
EXIT

FIG 31

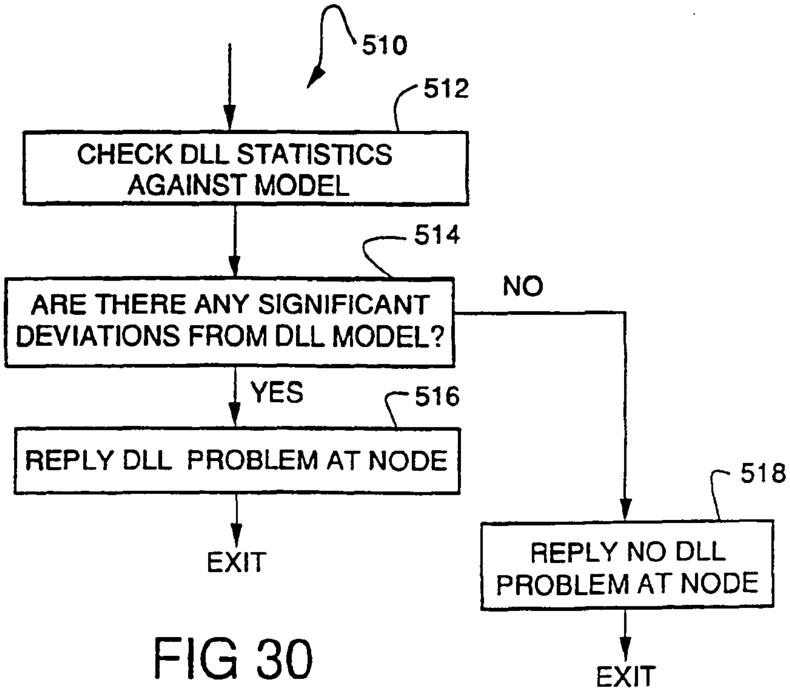

510
512
CHECK DLL STATISTICS AGAINST MODEL
514
ARE THERE ANY SIGNIFICANT DEVIATIONS FROM DLL MODEL?
NO
YES
516
REPLY DLL PROBLEM AT NODE
EXIT
518
REPLY NO DLL PROBLEM AT NODE
EXIT

FIG 30

SUBSTITUTE SHEET

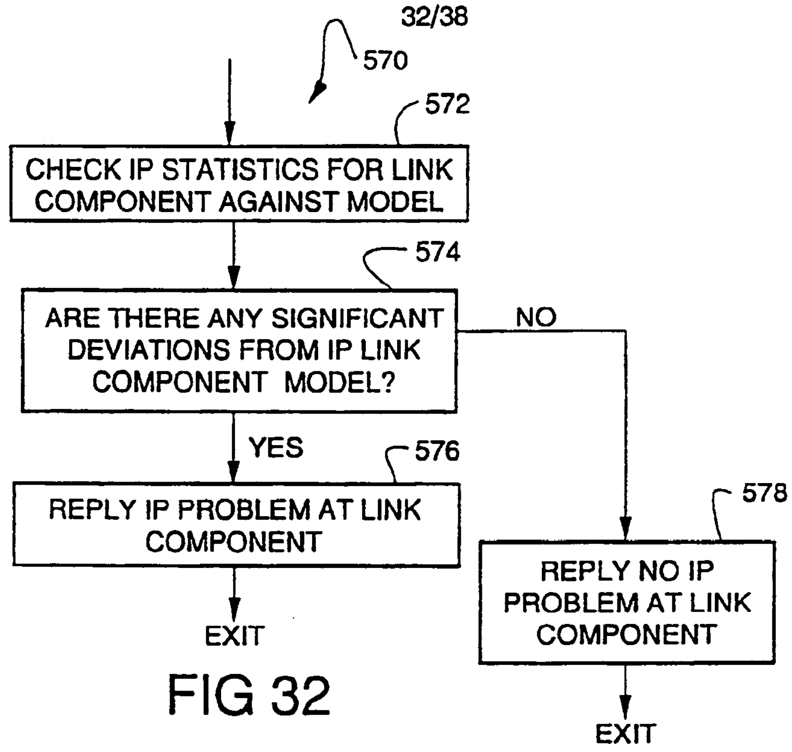

570
572
CHECK IP STATISTICS FOR LINK COMPONENT AGAINST MODEL
574
ARE THERE ANY SIGNIFICANT DEVIATIONS FROM IP LINK COMPONENT MODEL?
NO
YES
576
REPLY IP PROBLEM AT LINK COMPONENT
578
REPLY NO IP PROBLEM AT LINK COMPONENT
EXIT
EXIT

FIG 32

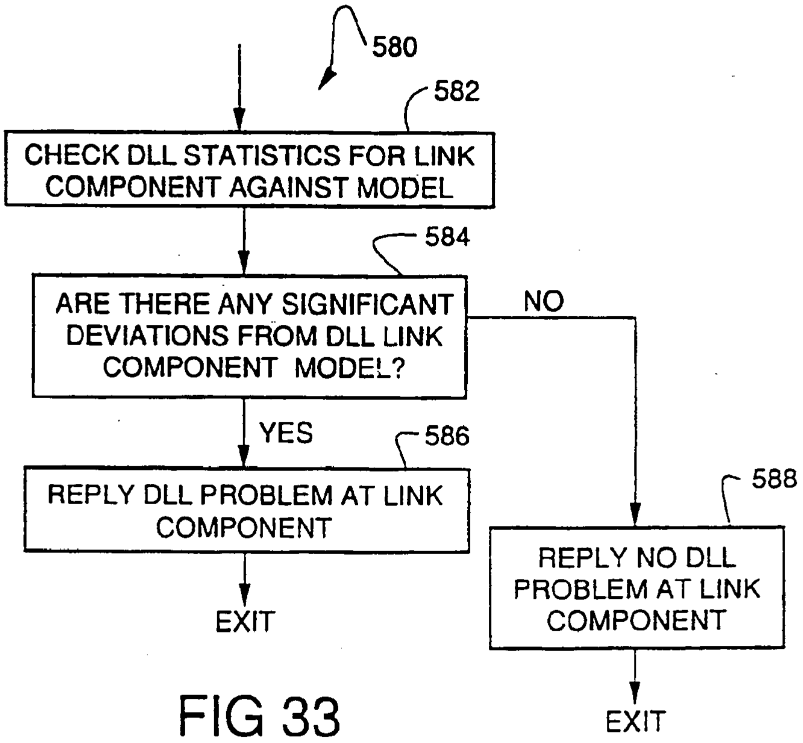

580
582
CHECK DLL STATISTICS FOR LINK COMPONENT AGAINST MODEL
584
ARE THERE ANY SIGNIFICANT DEVIATIONS FROM DLL LINK COMPONENT MODEL?
NO
YES
586
REPLY DLL PROBLEM AT LINK COMPONENT
588
REPLY NO DLL PROBLEM AT LINK COMPONENT
EXIT
EXIT

FIG 33

SUBSTITUTE SHEET.

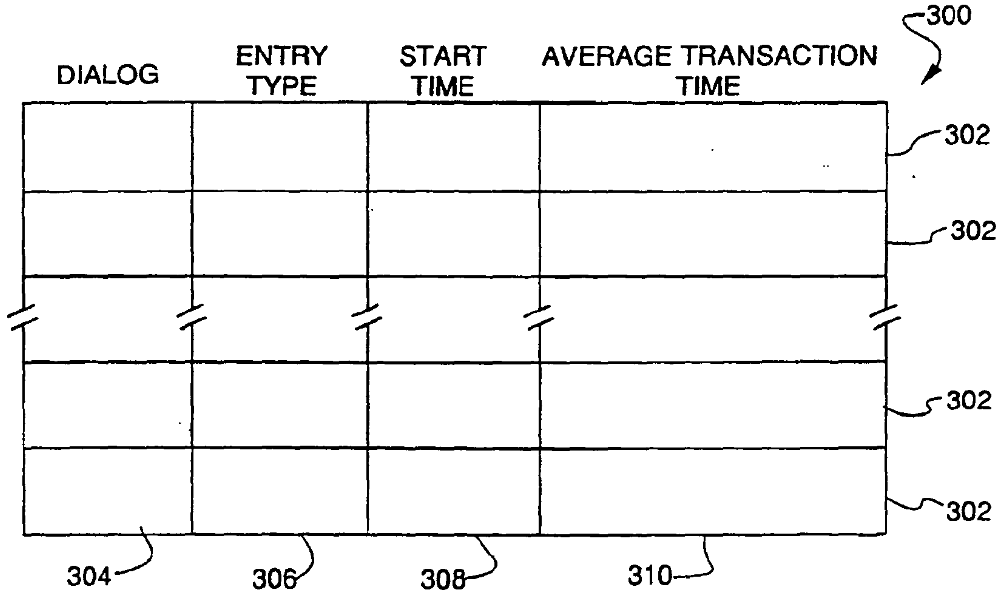

300
DIALOG
ENTRY TYPE
START TIME
AVERAGE TRANSACTION TIME
302
302
302
302
304
306
308
310

FIG 34

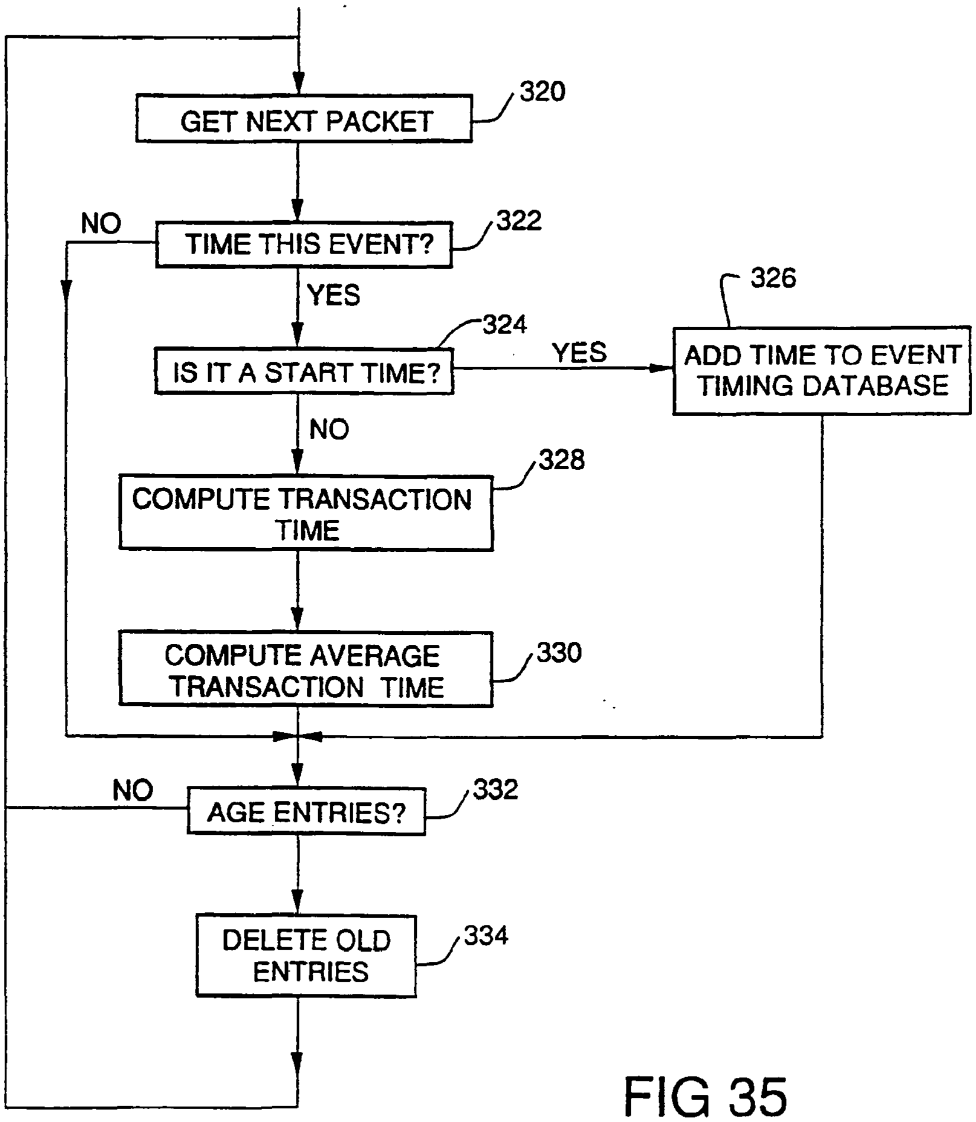

GET NEXT PACKET
320
NO
TIME THIS EVENT?
322
YES
326
324
IS IT A START TIME?
YES
ADD TIME TO EVENT TIMING DATABASE
NO
COMPUTE TRANSACTION TIME
328
COMPUTE AVERAGE TRANSACTION TIME
330
NO
AGE ENTRIES?
332
DELETE OLD ENTRIES
334

FIG 35

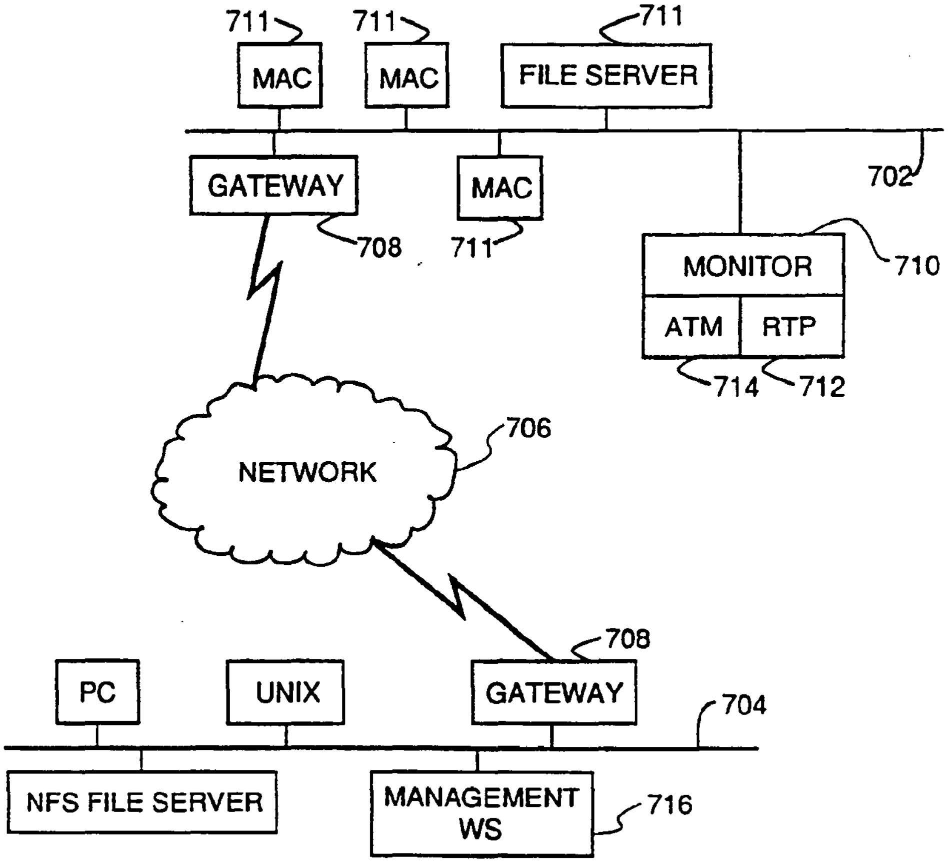

711
MAC
711
MAC
711
FILE SERVER
GATEWAY
MAC
702
708
711
MONITOR
710
ATM
RTP
714
712
NETWORK
706
708
PC
UNIX
GATEWAY
704
NFS FILE SERVER
MANAGEMENT WS
716

FIG 36

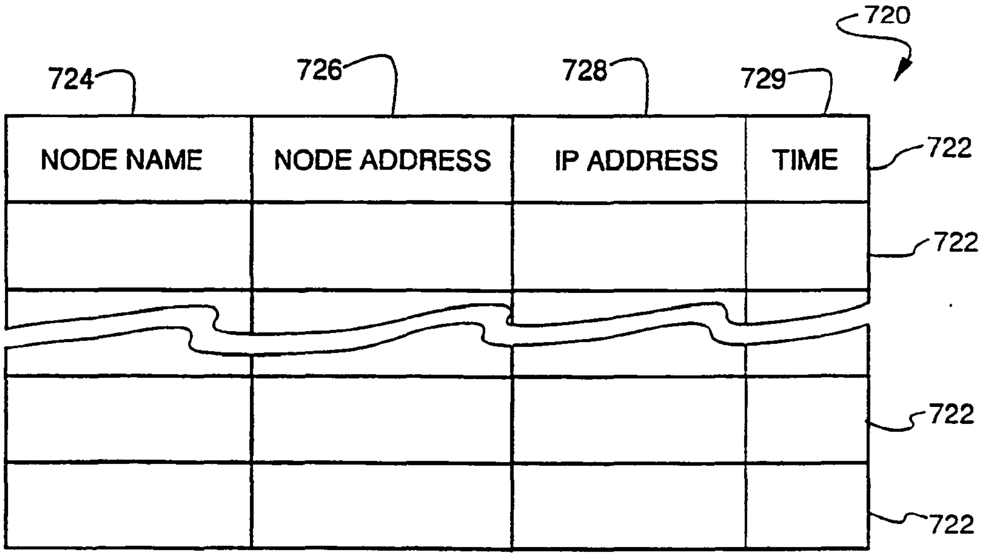

720
724
726
728
729
NODE NAME
NODE ADDRESS
IP ADDRESS
TIME
722
722
722
722

FIG 37

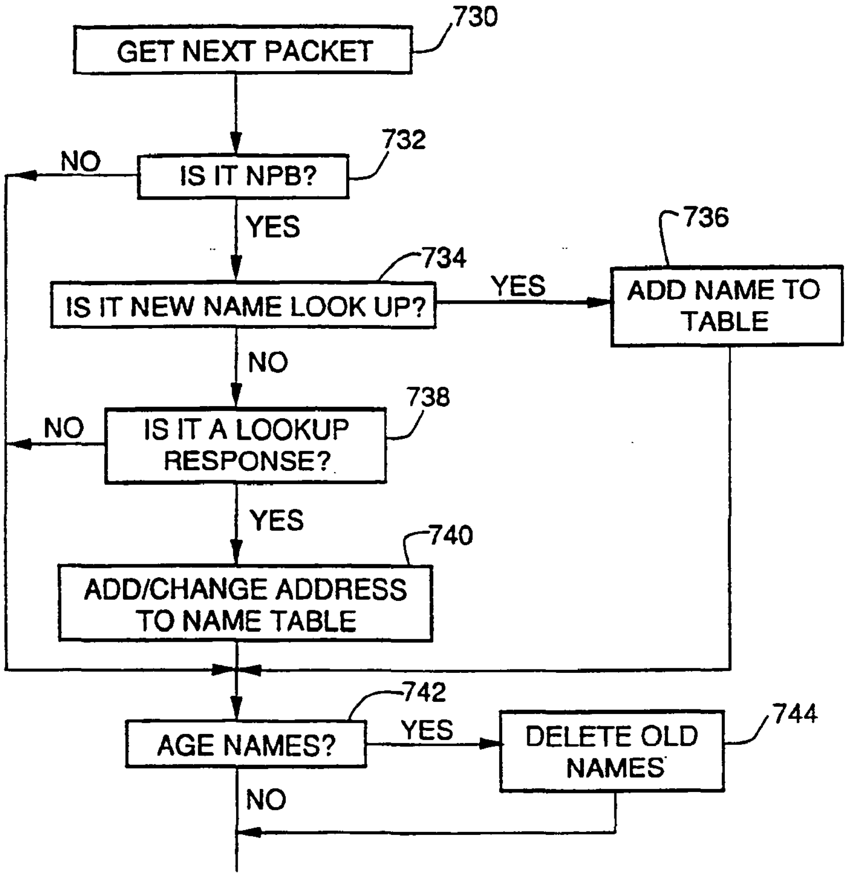

GET NEXT PACKET
730
IS IT NPB?
732
NO
YES
IS IT NEW NAME LOOK UP?
734
YES
ADD NAME TO TABLE
736
NO
IS IT A LOOKUP RESPONSE?
738
NO
YES
ADD/CHANGE ADDRESS TO NAME TABLE
740
AGE NAMES?
742
YES
DELETE OLD NAMES
744
NO

FIG 38

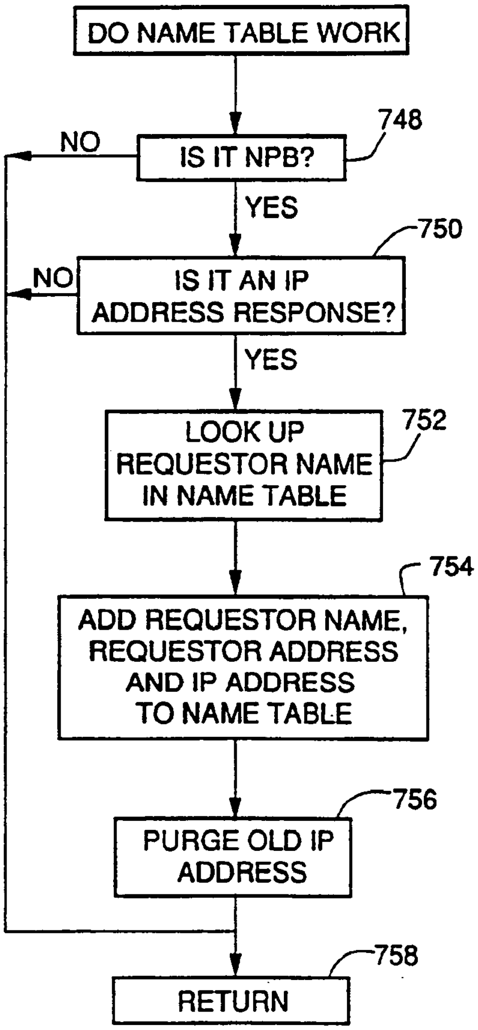

DO NAME TABLE WORK
748
NO
IS IT NPB?
YES
750
NO
IS IT AN IP ADDRESS RESPONSE?
YES
LOOK UP REQUESTOR NAME IN NAME TABLE
752
754
ADD REQUESTOR NAME, REQUESTOR ADDRESS AND IP ADDRESS TO NAME TABLE
756
PURGE OLD IP ADDRESS
758
RETURN

FIG 39

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US92/02995

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :H04J 3/14; H04J 3/24; H04L 12/56
US CL :370/13, 17, 94.1; 340/825.52
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 370/60; 371/20.1; 340/825.06, 825.07, 825.53; 364/514, 550

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

USPTO APS (Network Monitor);
(Protocol analyzer)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X,P Y	US, A, 5,101,402 (Chiu et al) 31 March 1992 (31.03.92). Column 6, line 32 to column 8, line 10. Figs. 15 and 16.	1-7 24-26
X Y	US, A, 4,887,260 (Carden et al) 12 December 1989 (12.12.89). Column 3, lines 21-51; Column 5, lines 50-68; Column 6, line 48 to column 7, line 38; Fig. 6.	1,3-7,9 24-26,29
A,P	US, A, 5,025,491 (Tsuchiya et al) 18 June 1991 (18.06.91)	30
X	US, A, 4,817,080 (Soha) 28 March 1989 (28.03.89) column 4, lines 23.31; column 5, lines 19-37; claim 1; Fig.1 and 3.	1,24-26

□ Further documents are listed in the continuation of Box C. □ See patent family annex.

*	Special categories of cited documents:
"A"	document defining the general state of the art which is not considered to be part of particular relevance
"E"	earlier document published on or after the international filing date
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O"	document referring to an oral disclosure, use, exhibition or other means
"P"	document published prior to the international filing date but later than the priority date claimed

Date of the actual completion of the international search 08 JULY 1992	Date of mailing of the international search report 31 JUL 1992
Name and mailing address of the ISA/ Commissioner of Patents and Trademarks Box PCT	Authorized officer Ngoc Ho Nguyen fn. H. KIZOU NGUYEN NGOC-HO INTERNATIONAL DIVISION

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 497 022 A1

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 91300772.0

(22) Date of filing: 31.01.91

(51) Int. Cl.5: G06F 15/40, G06F 9/44,
H04M 3/56, H04N 7/14

(43) Date of publication of application:
05.08.92 Bulletin 92/32

(84) Designated Contracting States:
DE FR GB

(71) Applicant: Hewlett-Packard Company
Mail Stop 20 B-O, 3000 Hanover Street
Palo Alto, California 94304(US)

(72) Inventor: Wink, Martin
c/o Hewlett-Packard Limited, Nine Mile Ride
Wokingham, Berkshire, RG11 3LL(GB)
Inventor: Jennings, Richard
c/o Hewlett-Packard Limited, Nine Mile Ride
Wokingham, Berkshire, RG11 3LL(GB)
Inventor: Baker, Colin
c/o Hewlett-Packard Limited, HP Labs, Filton Road
Stoke Gifford, Bristol, BS12 6QZ(GB)

(74) Representative: Smith, Denise Mary et al
Hewlett-Packard Limited Cain Road
Bracknell, Berkshire RG12 1HN(GB)

(54) Conference system.

(57) The present invention relates to a distributed object-based computer system in which sharable objects are split into client and server components (see Figure 7). Each client object contains a reference to the associated server object component. By copying client object components to other users, these other users obtain access to the relevant server-object component. This feature is described in the context of a distributed conferencing system.

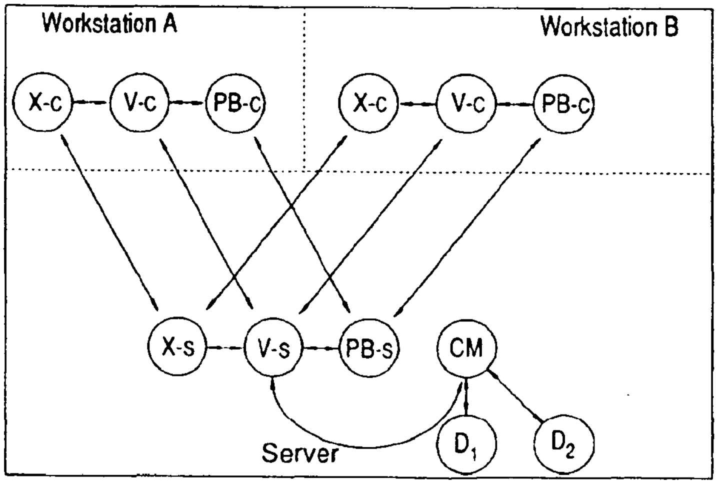


FIG 7

Rank Xerox (UK) Business Services

The present invention relates to a distributed computer system and relates particularly, but not exclusively, to a multimedia distributed object-based conference system.

The object-based approach to system development is becoming well-established. The basic idea is to program the system in terms of software objects. each having its own data and methods for operating on the data. Objects intercommunicate by means of messages. An advantage in encapsulating data and methods in this way is that the resulting system is relatively easy to maintain and develop. An example is NewWave Mail (produced and sold by Hewlett-Packard) which is an object-based electronic mail applications program in which messages and message components, such as text, distribution lists, etc, are treated as objects.

An object can be regarded as a discrete entity which can individually be moved, copied, destroyed, etc. An object is initially some data stored on disc or other medium. If object management software wishes to pass a message to it, one or more processes will be initiated which read the data as part of initialization. If an object is fully defined by its data and has no processes associated with it, it is said to be "inactive". If an object has one or more processes associated with it and is defined by the state of that process or processes and data then it is said to be "active".

A distributed object based system is one in which several workstations are interconnected over a network and messages between objects of the system can be sent over the network. Objects themselves may also be transmissible over the network. A network may comprise several interconnected intelligent workstations or a central computer connected to several terminals (workstations) or several interconnected server machines with intelligent workstations connected to each server, or a mixture of these possibilities. The term "workstation" is intended to be applicable to all of these possibilities.

In a distributed object based system there are benefits in splitting sharable semantic and presentation parts so as to enable more than one user to access the semantic part of a shared object. For example. in the context of a distributed conferencing system a whiteboard object would have a semantic part defining the state of the object and a presentation part for defining the appearance of the object to be displayed to a user and for enabling the user to make input. Several users may have access to a presentation part for viewing the whiteboard object so that they can each make contributions in a manner similar to a group of people clustered around a real whiteboard.

The workstations may be arranged in a client-server arrangement with semantic object parts stored on server machines and presentation object parts stored on client machines. Alternatively, semantic object parts may be distributed around user machines on a network of intelligent workstations.

According to the present invention we provide an object based distributed computer system comprising a network of workstations and means for transmitting objects between workstations characterized by objects including a first object type for storing data and a second object type for presenting data to a user, wherein objects of the second type reference an associated object of the first type to enable a plurality of users of workstations to access data of the object of the first type, comprising means for transmitting an object of the second type between workstations thereby to create a reference to the associated object of the first type for each workstation receiving an object of the second type.

The present invention provides an effective way of enabling further users to have access to a semantic object part, either for the purpose of autonomous working or for the purpose of participating in a joint activity.

In the embodiment to be described, the system comprises means for copying an object of the second type between workstations. In that embodiment transmitted objects of the second type include an identifier for the associated object of the first type.

The system according to the present invention may be in the form of a conferencing system comprising means enabling users of the workstations to participate in a meeting over the network wherein objects of the first type store meeting data and objects of the second type are for presenting meeting data. The invention also provides a method of convening a meeting using such a system comprising transmitting an object of the second type between workstations thereby to create a reference to the associated object of the first type for each workstation receiving an object of the second type.

It is believed that poor communications are a major cause of the poor performance of distributed teams of people working on a given project. The present invention advantageously provides an improved conference system for facilitating distributed meetings.

A particular embodiment of the present invention will now be described, by way of example. with reference to the accompanying drawings in which:

Figure 1 is a diagram of a distributed system according to the present invention;

Figure 2 shows the major components of a server and workstation of the system;

Figure 3 shows a voice and data network structure;

Figure 4 shows video facilities for a client workstation;
Figure 5 shows a video network structure;
Figure 6 illustrates the main objects in the system;
Figure 7 illustrates the functionally split nature of the objects in the system;
Figure 8 shows the major components of the system infrastructure;
Figure 9 shows a typical Venue;
Figure 10 shows a CoMedian directory;
Figures 11 - 14 illustrate message sequences for system operations;
Figures 15 - 27 show screens during a typical user session.

The main components of a multi-media distributed object-based conferencing system according to the invention will first be described.

Referring to Figure 1, a multimedia distributed object-based conference system according to the present invention is indicated at 10. The system 10 comprises servers S connected over a network 12. The network 12 may be a wide area network (WAN) or a local area network (LAN) or a metropolitan area network (MAN). Client workstations C are connected to each of the servers S. Each site requires a server S.

Servers S communicate with each other by opening virtual circuits between pairs of servers. Although in principle, client workstations C could communicate directly with each other, this creates practical problems and therefore each client workstation C has only one virtual channel open to its local server S to enable client workstations to communicate with each other via servers S.

Referring to Figure 2, each server S comprises:

hardware 14, such as an HP9000 300 HP-UX computer (HP is a trade mark of Hewlett Packard Company);

operating system software 16, such as HP-UX software;

Remote Object Access Manager (ROAM) software 18 for managing communications with client workstations C connected to the server S and other servers on the network;

COM software 20 providing object management facilities:

server objects 21 which are objects to be shared between users and which correspond to the semantic object parts mentioned in the introduction.

Each client workstation C comprises:

hardware 22, such as an IBM-AT compatible PC;

operating system software 24, such as DOS software;

windowing software 26, such as MS Windows applications software;

an object management facility (OMF) 28, such as a Standard NewWave OMF. (Newwave is a trade mark of Hewlett-Packard Company used for a family of applications software);

objects software 30, such as NewWave objects and specialized client objects 32 and a ROAM object 34 for handling communication with objects on other computers. The client objects 32 correspond to the presentation object parts mentioned in the introduction.

The user of a client workstation C therefore has a windowed user interface within which to manipulate objects of the system and can cause objects to be transmitted over the network 12 via the associated server S.

The system 10 provides multimedia facilities to users. For example, each client workstation C may have voice and/or video communication facilities as well as data communication facilities.

A possible voice and data network structure 40 is shown in Figure 3. In each of two sites designated A and B, a networked PC server 42 is connected to the local PABX. The PC server 42 contains one or more multi-port telephone interface cards (such as the VBX-300 card made by Natural Microsystems Inc.). The PABX is controlled by the PC server 42 and users can use their existing standard desk telephones 44 which are connected to the local PABX and conveniently located near their client workstations C.

Each of the sites A and B comprises a LAN and a LAN/WAN bridge interconnecting the LAN with a WAN.

In use, the PC server 42 receives commands from servers S to set up, maintain and close down telephone conference calls. To the PABX, the PC server 42 appears as a normal telephone user and can therefore dial other users adding them in to conference calls using DTMF.

In order to conduct conferences over a wider area, PC servers 42a and 42b on respective sites A and B connect to each other over the public switched telephone network (PSTN) and add in their own local users to the conference.

Referring to Figure 4, each client workstation C with video facilities has a video camera 46, two or more VHF TV receivers 48, a microphone 50, a preamplifier 51 and a VHF modulator 52.

Furthermore, the client workstations C may be fitted with video cards to enable a user to view video in windows.

A possible video network is shown in Figure 5. The video network is based on a central video switch 54 connected using a star topology to client workstations C. Video signals are modulated on to VHF carriers and transmitted over standard analogue cabling 56. The video switch 54 is a conventional cable television switch. Several such switches can be cascaded in a bar arrangement for large systems.

For long distance video communications. a device 58 for compressing and decompressing video signals (a "codec") may be used and the signals are transmitted using ISDN telephone lines.

The architecture of the object-based system 10 will now be described.

With reference to Figure 6, the structure of one user's portion of the system is represented. The functions of the objects are as follows:

a Venue object (V) is an electronic meeting place allowing control over media channels and providing a location for storing shared objects. A user may have several Venue objects;

a Phone Booth object (PB) controls the creation of Venue objects and oversees the setting up, maintenance and closing down of conferences. The PB comprises a processor for handling incoming and outgoing calls;

a Connection Manager object (CM) controls driver components (D_1 ... D_n) which handle media connections for the system 10:

a Directory object (D) which provides a list of potential meeting participants.

Object X represents another system object for performing a specific meeting-related function. eg. a whiteboard function.

Figure 6 is a conceptual representation of the system 10 and the arrows represent inter-object communication. In the embodiment being described. the system comprises client workstations C and servers S and most of the objects referred to in Figure 6 are functionally-split into a server component and one or more client components as indicated in Figure 7.

The server objects handle the centralized and distribution - oriented aspects whereas the client objects handle the presentation aspects. Hence shared applications can be written with one server object connected to a plurality of client objects on different client workstations.

In Figure 7, PB-s means a Phone Booth server object and PB-c means a Phone Booth client object, and so on.

In this embodiment. the client objects are implemented as NewWave objects ie. several new classes of NewWave objects have been added: Venue objects, ROAM objects, Whiteboard objects, Phone Booth objects. Thus the semantic part of these functionally split objects runs on an HP-UX server and the user interface runs on MS-DOS NewWave client workstations.

The client workstations are each running an object-based system of the type described in European Patent Application No.339220A, the description of which is incorporated herein as Appendix A. Appendices A-D mentioned in attached Appendix A are not attached as part of this application but are incorporated herein by reference. Appendix A describes how objects are linked together by parent-child links and how objects can be copied. During a copy operation, the container of the object to be copied sends a message to the OMF28 asking the OMF28 to copy the relevant object and identifying the container object which is to receive the copy object.

The OMF28 performs the copy function and then sends a message to the target container object instructing it to insert the copy object as one of its child objects.

Mailing an object involves serialising the object. transmitting it to its destination and deserialising it. Serialising an object involves converting it to files, say DOS files, containing the data of the object and information about its properties and its child objects.

Server objects are not linked by parent-child links in the manner in which client objects are so linked. All client objects contain a reference to their associated server object. Figure 8 shows the form of data item 60 used to name objects. The data item 60 is an eight-byte array following the convention used for Internet Protocol (IP) addresses. The first 64 bits is a machine identifier M I/D comprising a 32 bit server IP address and a 32 bit machine IP address. For a server object the server IP address and the machine IP address will be the same whereas for a client object these will be different. If there is only one domain per machine, the domain identifier D I/D is zero. The object identifier O I.D comprises a 32 bit generation count and a 16 bit tag. The 16 bit tag uniquely identifies the object within the relevant storage domain. Since tags are reusable when an object is deleted a generation count is used to ensure that each object is uniquely-named in time. The generation count is simply the time in seconds.

When a client object is closed (inactive) it appears as an icon on a user's screen. The user opens the object by clicking on the icon. Opening a client object causes it to send a message to its associated server object informing the server object that the client object is now active i.e. a Here Am I message. Until then, the server object is unaware of the existence of the client object. In other words, links between client and server objects are non-persistent and 'weak' i.e. the existence of a server object does not guarantee the existence of a corresponding client object and vice-versa. Server objects only store the identities of corresponding client objects which are currently active. Opening a client object means that a user can view the state of the object and can make input to it. The client object regularly updates, and is updated by, the server object.

Figure 9 depicts the components involved in a typical active server object which is associated with client objects on two different client workstations C_1 and C_2. Each object is given a unique object identifier comprising components identifying the relevant client/server machine, the relevant storage domain and a number for the particular object. On the client side, the system has an object management facility (OMF) 60 for keeping a record of what objects are presently on the particular client workstation and which is involved in object creation and deletion, object naming, object activation and deactivation and inter-object message routing. This is a standard NewWave OMF. There is a client object manager library (COMLIB-C) 61 statically linked to each client object CO providing access to the functionality of a ROAM client object 62. In other words, the COMLIB-C 61 has been added to standard NewWave objects to form the client objects for functionally split objects. Communication through the COMLIB-C 61 is network transparent, ie. objects only need to know the object identifiers of other objects, not their locations.

On the server side there is a primitive object management facility (COM-S) 63 providing file management and object naming and message sending facilities in conjunction with the operating system software 64. A server object manager library (COMLIB-S) 65 is statically linked to each server object SO enabling access to the functionality of the object management facility 63 and a ROAM server object 66.

When client object CO_1 wishes to send a message to the corresponding server object SO, the ROAM client object 62 passes the message to the ROAM server object 66 which passes the message on to the server object SO. Messages from the server object SO to client objects are sent in the reverse manner. If a message is to be sent between objects on the same server the COMLIB-S 65 sends it directly without involving the ROAM server object 66. Messages are also sent between servers via the ROAM server object 66 and, in this way, communication between client workstations connected to different server machines is possible.

The functionality of certain objects in the system will now be described. The term "click" will be used in this specification to denote a selection made by the user of a workstation using an input device, such as a mouse. The term "drag" will denote moving the input device whilst such a selection is made so as to "drag" an item across the screen.

The Venue provides an electronic meeting room, inside of which person-to-person calls, group meetings and presentations to large groups can be held.

Venues provide a binding between the people involved in a meeting, the data which they are sharing, and the media channels connecting them. They are scalable from just two people up to many people, the exact number is subject to technical constraints. This allows a meeting to start off as a simple phone call between two people, build up as experts are brought in, to become a full group discussion without having to decide to move to a different object because the nature of the meeting has changed.

The Venue is a shared object and therefore exists on a server machine. The client workstations have Venue client objects which provide an interface to the Venue server objects running on the corresponding server. There may be many Venue client objects on different client workstations for a particular Venue server object.

Figure 10 shows the appearance of a Venue to a user. The Venue is being viewed in a window 70 having a title bar 72 and a menu bar 74. At the top is a participants' area 76 where the people in the Venue can be seen and where their media channels can be controlled. Beneath that is a shared area 78 where objects for use in the meeting are stored.

The participants in a Venue are displayed side by side, with each participant being represented by a still bitmap 80, a name 82 accompanied by an indication of whether that user is present in the meeting or absent and status banner 84 indicating that an absent user has been invited to the meeting, and a row of media control buttons 86. The bitmap 80 may be replaced by a motion video window when video in windows is available and the video channel is in use.

Beneath the participants' area are three media buttons 86 for telephone, video and data and each one can be in one of four states. The states are:

Button Appearance	Meaning
No button	This person does not have this media channel available.
White, unhighlighted	The media channel is available, but not chosen for use.
Black	The media channel has been selected, but is inactive because the person is not present in the Venue or the connection has not been completed yet.
Red	The media channel is being used.

The lower portion of the Venue is taken up by the shared object area 78. This acts as a shared folder, storing objects on the server and making them accessible to all users of the Venue. Inactive objects are represented by an icon such as icon 88 in Figure 10. Objects in the shared object area 78 may be client objects e.g. Whiteboard client objects. or may be standard NewWave objects. It is possible to move objects into and out of the shared object area 78 of the Venue-client object. Moving a functionally-split object such as a Whiteboard object into the shared object area 78 does not entail moving the Whiteboard-server object but just the Whiteboard-client object. The OMF28 instructs the Venue client object to insert the Whiteboard-client object as one of its children. The Whiteboard-client object is then serialised by the Venue-client object and sent to the Venue-server object. The Venue-server object updates its other active Venue-client object with the news that a new Whiteboard object is available in the Venue and these Venue-client objects display the Whiteboard-client object icon in their shared object areas 78 accordingly. The Whiteboard-server object remains on whatever server it was initially stored. Subsequent opening of the Whiteboard object by any of the users of the Venue cause a copy of the Whiteboard-client object to be serialised by the Venue-server and sent to the relevant client-workstation where it is deserialised providing access to the contents of the Wniteboard object for that user. When that user subsequently closes (deactivates) the Whiteboard object, the copy of the Whiteboard-client object remains on that machine for subsequent use.

In contrast, if a NewWave object icon is moved into the shared object area 78 of a Venue-client object, this causes the NewWave object to be serialised and sent from the client workstation to the server machine which stores the relevant Venue-server object. The Venue-server object then instructs its other active Venue-client objects to display the relevant NewWave object icon. Subsequent opening of the Newwave object by a user of such an active Venue-client object causes a copy of the NewWave object to be made and sent to the relevant client workstation. Each such user thus obtains a separate copy of the NewWave object and changes which a user makes are not reflected in the copies held on the other users' machines. This is a consequence of the non-functionally split nature of NewWave objects and is an implementational feature rather than one which is important to the present invention.

There is one Phone Booth server object on every server machine and one Phone Booth client object on every client workstation. The Phone Booth client object arranges for the creation and activation of Venue client objects on client workstations and the Phone Booth server object manages the creation of Venue server objects and the convening of Venues. On opening a Phone Booth client object the user is presented with a directory 90 of possible meeting participants as shown in Figure 11. The directory 90 comprises a list 92 of potential participants, an area 94 for displaying a picture of a participant, a media selection area 96 and an options area 98 displaying three options: Convene, Select and Cancel. Unavailable media options are greyed out in the area 96.

When a name is selected by choosing the Select option and then selecting a name from the directory 90, a picture of that participant appears in the area 94 as shown. The media connections are selectable by checking the relevant boxes in the media selection area 96. Checking the box beside the name of the person in the area 94 adds that person to the list of Venue participants. In addition, the initials of the media options chosen (Phone, Video, Data) appear against the participant's name in the list 92. A previously selected participant can be de-selected by de-checking the box beside the name of that person in the area 94. Taking the Cancel option means that none of the changes made since the window for the directory 90 was brought up will be implemented. The **Convene** option will be described later.

There is also a Connection Manager object on each server machine providing the facility to interconnect users using different media. The Connection Manager object handles the generic operations involved in establishing non-data interconnections. Drivers for each medium available, eg. video, telephone, handle the specific operations involved in carrying out switching requests during use. The Connection Manager object performs the following services:

maintains a list of media resources available in the system:

- detects when resources fail
- monitors resource/channel availability (ie, monitors, microphones, speakers, cameras);

sets up connections between people using different media:

- point-to-point
- multi-point: all that are available

maintains list of established connections and ensures synchronization with other networks. ie, maintains a model of the state of other networks;

optimizes switching to prevent unnecessary disconnect -connect transactions;

provides an interface for monitoring and auditing;

provides interface to media drivers.

Another functionally split object which is provided in this system is the Whiteboard. A Whiteboard object provides users with a shared computer whiteboard facility so that a user can draw, write and type on his/her Whiteboard or acquire an image from another source and the input will be visible to other users viewing the same Whiteboard on different client workstations. Thus the Whiteboard object is an information sharing medium which allows users to look at a picture of what they are discussing.

Figure 12 shows an example of the appearance of a Whiteboard client object. The Whiteboard is being viewed in a window 100 having a title bar 102 and a menu bar 104. A drawing area 106 of the window 100 is devoted to displaying the contents of the Whiteboard, in this case a map showing the location of a Hewlett-Packard office. At the bottom of the window 100 is an area 108 indicating the range of tools which are available to the user of the Whiteboard. These tools comprise:

a scroller	110
a pointer	112
a selection of different coloured pens	114
an eraser	116
a text selector	118

Apart from the pointer 112, the tools are personal to a user ie each of the users viewing the same Whiteboard could be using the same tool eg. a red pen, without having to wait until another of the users had finished using that tool.

The scroller 110 can be used to scroll the entire window 100 around the Whiteboard. Selecting this tool turns the cursor into a compass enabling the view of the Whiteboard to be click-dragged around by the user.

Only one user can move the pointer 112 at a time. A user takes control of the pointer by clicking on the pointer logo - this turns the cursor into a pointer. At this time, the other users viewing the Whiteboard cannot see the pointer 112. To show the pointer 112, the user needs to click it down at a chosen point in the drawing area 106. The pointer 112 then becomes visible to all of the Whiteboard users at that chosen position. The cursor of the user who has just moved the pointer 112 reverts to the default arrow.

Likewise the seven coloured pens are selectable and deselectable by clicking on the appropriate pen logo, enabling different users to make input in different colours.

The eraser 116 is selectable to remove marks on the Whiteboard. Also, direct typing of text onto the Wniteboard can be done by selecting the text selector 118.

In the area 108 there is also room for a status message 120. As users open or close the Whiteboard other users are notified by a status message.

Modes of operation of a system according to the present invention will now be described, concentrating first on utilization of the Venue.

Once a user selects participants and media as described with reference to Figure 11 and selects the Convene option a process of events is initiated to create a new Venue object. Figure 13 shows the objects and the numbered sequence of messages. Figure 13 depicts a server machine S and two client workstations A and B connected to the server machine S. On each client workstation there is initially a Phone Booth client object PB-c. On the server machine S there is initially a Phone Booth server object PB-s and a Connection Manager object CM.

On selecting the **Convene** option using client workstation A, which causes an input (dotted line referenced 1) to the Phone Booth client object PB-c, a message (referenced 2) is sent from the Phone Booth client object PB-c to the Phone Booth server object PB-s on the server machine S causing the Phone Booth server object to create a new Venue server object V-s using a **Venue Start** message (referenced 3). The Phone Booth server object PB-s then sends a **Ring** message (referenced 4) to the Phone Booth client object PB-c on client workstation B causing a dialogue box to appear on the screen of client workstation B inviting the user to take part in the proposed meeting. That user accepts or declines the invitation causing a corresponding message (referenced 5) to be sent from the Phone Booth client object PB-c' on client workstation B to the Phone Booth server object PB-s. If the invitation is accepted a **Create Venue** message (referenced 6) is sent from the Phone Booth server object PB-s to the Phone Booth client object PC-c^1 which causes it to create a new Venue client object V-c^1 on client workstation B involving sending a **Here Is Parent** message (referenced 7) to the new Venue - client object V-c^1 to notify it of the identity of the Venue server object V-s. The new Venue client object V-c^1 then sends a message (referenced 8) to the Venue server object V-s requesting information about the contents of the Venue. The reply from the Venue server object V-s is referenced 9 in Figure 13.

Messages corresponding to those referenced 6-9 are sent between the server S and client workstation A so as also to create a new Venue-client object V-c on that workstation and these messages are referenced 10-13 in Figure 13.

Finally, the Venue server object V-s sends a request (referenced 14) to the Connection Manager object CM to set up the chosen media connections and the Connection Manager object instructs the relevant media drivers accordingly (dotted line referenced 15).

The users of client workstations A and B can then communicate using the newly created Venue.

It is also possible to convene an existing Venue by selecting the **Convene** option within the Venue. This initiates a sequence of events which will be described with reference to Figure 14. Again, a server machine S and two client workstations A and B are represented.

The user selection of the **Convene** option is referenced 1 in Figure 14. This causes the Venue client object V-c to send a **Convene Request** message (referenced 2) to the Venue server object V-s which notifies the Phone Booth server object PB-s of the convene request in a message referenced 3 which identifies the intended meeting participants. The Phone Booth server object PB-s sends a **Ring** message (referenced 4) to the Phone Booth client objects PB-c on the workstations of the intended meeting participants causing a dialogue box to be displayed on these workstations inviting the users to partake in a meeting. When these users accept or decline the invitation this causes a reply message (referenced 5) to be sent from each Phone booth client object PB-c^1 to the Phone Booth server object PB-s.

The next step is for the Phone Booth server object PB-s to instruct (message referenced 6) the Phone Booth client objects PB-c^1 to create new Venue client objects V-c^1 on machines where a Venue client object linked to the Venue server object V-s is not already stored. Such new Venue client objects V-c^1 then send a message (referenced 8) to the Venue server object V-s requesting information about the contents of the Venue so that the appropriate icons can be displayed in the shared area 78 of Figure 10 on the respective client workstations. The reply message containing information about the contents of the Venue from the Venue server object V-s is referenced 9 in Figure 13.

The Venue server object V-s then sends a request (referenced 10) to the Connection Manager object CM to set up the chosen media connections and the Connection Manager object instructs the relevant media drivers (not shown) accordingly (dotted line referenced 11). The distributed meeting can then proceed.

A user can also set up a new Venue by selecting a **Create a New** menu option in NewWave Office (Figures 14-17 of Appendix A). On opening the new Venue-client object a Venue-server object also needs to be created. Figure 15 depicts the process. A server machine is indicated by S and a client workstation by C.

The act of opening the new Venue-client object V-c causes it to send a message (referenced 1) to the Phone Booth client cbject PB-c which triggers a message (referenced 2) to be sent from the Phone Booth client object PB-c to the Phone Booth server object PB-s requesting creation of a new Venue server object V-s. The Phone Booth server object PB-s creates a new Venue server object V-s using a **Venue Start** message (referenced 3). Next the new Venue-server object V-s sends a **Here Is Parent** message (referenced 4) to the Venue-client object V-c containing the ID of the Venue-server object. The new Venue client object V-c then sends a message (referenced 5) to the Venue server object V-s requesting information about the contents of the Venue and there is a corresponding reply (referenced 6) from the Venue server object.

It is possible to add new meeting participants to an active Venue by selecting an **Add New Member** menu option. This causes a directory of potential participants to be displayed as shown in Figure 11 to enable the selection of one or more further participants and associated media connections. Information on these choices is conveyed from the Venue client object to the Venue server object which updates the control panels of the relevant Venue client objects. Chosen new meeting participants are not aware of any change until someone convenes a meeting.

When a user elects to close a Venue by selecting a **CLOSE** option this causes a message to be sent from the relevant Venue-client object to its Venue-server object informing the Venue-server object that the Venue-client object is deactivating. The Venue-server object then messages the Connection Manager object to disconnect the media connections for the Venue-client object which is deactivating. The Venue-server object sends messages to all of its other Venue-client objects informing them of the deactivation of the particular Venue-client object so that these other Venue-client objects alter their appearance to indicate that the relevant meeting member is now absent.

Another way of setting up a distributed meeting is for a user to copy an existing Venue-client object to the desired meeting participants. A Venue-client object is a reference to a Venue-server object. Copying a Venue-client object to other workstations creates a reference to the relevant Venue-server object on those other workstations because in the copying process the Venue-client object's reference to its Venue-server object is preserved.

There are different ways in which a Venue-client object can be copied to other workstations. One way is to include the Venue-client object in an electronic mail message. For this option, an electronic mail message is created in the normal manner e.g. using Hewlett-Packard's NewWave Mail and a Venue-client object is included in the message using a standard copy operation. When the or each addressee receives the message, they place the Venue-client in their collection of objects in preparation for the forthcoming meeting. At the relevant time, the meeting participants open their Venue-client objects to commence the meeting. On opening the Venue-client objects, their 32 bit machine IP address is automatically updated and the Venue-client objects send a Here Am I message to the associated Venue-server object.

Another option is for the user wishing to set up a distributed meeting to copy the relevant Venue-client object and to serialise the copy of the Venue-client object to a file on floppy disc (or other shared medium such as a network drive). This file may then be transported to the workstations of the intended meeting participants and deserialised thereby providing each of these participants with a copy of the Venue-client object and thereby means for accessing the associated Venue-server objects in order to take part in the distributed meeting.

A new Whiteboard-client object can also be created using the "Create A New" option in NewWave Office. On opening the Whiteboard-client object a new Whiteboard server object needs to be created. The process is analogous to that described with reference to Figure 15 replacing references to Venue objects with references to Whiteboard objects.

A new Whiteboard object can also be created inside a Venue by selecting the "Create a New" option inside the Venue. In this case, the Venue-client object automatically activates the new Whiteboard-client object in order to initiate creation of a new Whiteboard server object (again using a process analogous to that shown in Figure 15).

In the same manner as a Venue-client object can be copied and transmitted in an electronic mail message or via floppy disc, a Whiteboard-client object can be so utilised. Again the advantage of creating a reference to the relevant Whiteboard server object for the recipients of the copied Whiteboard-client objects is obtained since each copy of the Whiteboard-client object contains a reference to the Whiteboard server object (as described with reference to Figure 8).

Also as previously described, a Whiteboard-client object can be moved into the shared items area of a Venue object by a user causing copies of the Whiteboard-client object to be made available to the other users of the Venue object thereby giving access to the asscciated Whiteboard server object to these users.

An exemplary user session will now be described with reference to Figures 16 to 33 involving hypothetical users Martin, Richi and Ed.

Figure 16 shows a screen of a client workstation (Martin's) running Hewlett Packard NewWave Software. A window 126 has:

a title bar 128 carrying the title "NewWave Office";

a menu bar 130 offering the following options:
Action, Edit, Objects, View, Settings, Task and Help;

a system menu box 132;

size boxes 134 and 136;

a vertical scroll bar 138 with scroll arrows 140 and 142 and a scroll box 144;

a horizontal scroll bar 146 with scroll arrows 148 and 150 and a scroll box 152;

The window 126 displays icons for some standard tools at the top: Waste Basket 154, Agent 156, Printer 160, In Tray 162, Out Tray 164, File Drawer 166. The icons 168, 170 and 172 respectively on the left hand side represent work-related items:

"Project Meeting" a Venue-client object representing a reference to a Venue server object on the local server machine;

"Design Notes" a Whiteboard-client object representing a reference to a Whiteboard server object on the local server machine;

"Design Principles" a NewWave document object fully contained on the client workstation.

To "open" an object, the user double clicks on the relevant icon. Referring to Figure 17, Martin has opened the Project Meeting Venue which is shown in a window 174. The window 174 has a menu bar 176 which has similar options to the menu bar 130 of the window 126 except a Meeting option instead of the Setting option. The window 174 displays a participants area 178, showing only Martin, and a shared items area 180 which is empty. Underneath a bit map 182 of Martin is a name bar 184 which includes a notification of presence and three media control buttons 185-7 for Phone, Video and Data respectively. Only the Data button 187 is highlighted in this example, ie. blacked out in Figure 17.

On selecting the Meeting option from the menu bar 176 of the window 174, a CoMedian directory window 190 appears, Figure 18. The reference numerals for the CoMedian directory which were used in Figure 11 will be retained here. Martin selects the name Richard Jennings from the list 92 of potential participants causing a picture of Richard to appear in the area 94 together with crosses in the video and data boxes in the area 96 to indicate Richard's media selections. This means that Richard will be contacted through the system for data sharing with both video and audio travelling over video connections. Martin then clicks on the Convene button in the options area 98 to add Richard to the Venue which causes Richard's image to join Martin's image in the Venue as shown at 192 in Figure 19. Richard is marked as absent at 194 and a banner 196 is displayed indicating that he has been invited. Martin has selected both video and data connections for himself in order to match what was selected for Richard. This causes the video and data buttons 186 and 187 to be highlighted in a first colour to show that they are currently in use albeit only locally to Martin's own workstation. Richard's video and data media buttons 186a and 187a are highlighted in a second colour to indicate that they have been requested but are not yet in use.

While waiting for Richard to join the Venue, Martin is moving the Design Notes and Design Principles objects 170 and 172 into the shared items area 180 of the Venue by clicking on each object and dragging it to the area 180.

Moving now to Richard's workstation, shown in Figure 20, the invitation to join the Venue has reached his machine and has caused a bell 200 to appear at the bottom of his screen. The bell 200 is flashing and making a ringing sound to attract his attention. Richard clicks on the bell 200 and the result is shown in Figure 21. An invitation message box 202 is brought up telling Richard that he has been invited to a meeting and giving the name of the meeting and the name of the person who convened the meeting. The invitation message box 202 comprises two options: Accept and Decline. Richard clicks on the Accept option to accept the invitation to join the meeting.

Referring to Figure 22 accepting the invitation causes a Venue client object automatically to be created and a window 204 to be opened for Richard. The chosen media connections have been set up so that Richard can now see and hear Martin and the objects that Martin has placed into the shared items area 180 are available to him. Figure 23 shows that Martin can see the same Venue having the same contents on his workstation. Referring to Figure 24, during the meeting, Martin has opened a window 206 on the Design Notes whiteboard object. Martin informs Richard of this so that Richard can also view the whiteboard object and then both Martin and Richard can scribble on the whiteboard and view each others input. When their meeting is finished both Martin and Richard close and save the Venue.

Figure 25 shows the Venue object 168 saved in Richard's NewWave office. In Figure 26, Richard has just opened his NewWave office and is viewing the Venue 168 in a window 208. Martin is not present (although he would be if, coincidently, he had his Venue open at the same time as Richard. In that situation, the relevant media connections would automatically be set up). Referring to Figure 27, Richard has selected the Meeting menu item using the cursor 210 so as to bring up the CoMedian directory 212 and he has selected Ed Davies in the manner previously described. Ed Davies does not have video capability, instead he is selected by telephone. Clicking on the Select button will cause Ed to be added to the Venue without beginning a Convene operation.

Referring to Figure 28, Richard is about to initiate a Convene operation by selecting the Action item from the menu bar 214 of the window 208, and selecting the Convene option from the corresponding menu 216. Since Ed does not have video capabilities, the audio from his telephone would be mixed into the video feed into Martin and Richard and their audio signals would be sent to Ed's telephone during their distributed meeting.

Turning now to Figure 29, a new session is beginning on Richard Jenning's workstation. A window 220 contains Richard's NewWave Office. Richard has created an outgoing message represented by the icon 222 called "Meeting Request" (using the "Create a New" option from the Action Menu - see Figures 14 to 17 of Appendix A). In Figure 30, on opening the outgoing message 222 it is displayed in a window 224. Richard has completed the distribution list 226 and written a cover note 228.

Referring to Figure 31, a new Venue-client object represented by the icon 230 is created (again using the "Create a New" option). The Venue-client object 230 is copied and dragged into the window 224 displaying the message. This is achieved by clicking on the icon 230 and pressing the control key whilst dragging the icon into the message. (This is an alternative method from the user perspective to the copy procedure described with reference to Figures 18-20 of Appendix A.) The bar 232 labelled "Part 3" in Figure 32 shows that the message now contains a copy of the Venue-client object. The message window 224 is then closed (Figure 33). To send the message 222 it can be dragged onto the Out Tray icon 234. This causes a copy of the message, including the Venue-client object which it contains, to be sent to the people on the distribution list. The Out Tray object 234 initiates the serialisation of the message components to enable these to be transmitted over the network. On receipt at the respective destinations, the In Tray object represented by icon 236 deserialises the message components so that these can be viewed and manipulated by the recipients. The recipients can drag the Venue-client object out of the message and into their main NewWave Office window (220). At the appointed time, the three participants open their Venue-client objects to begin a distributed meeting. During the meeting, the users can open shared objects e.g. a Whiteboard object, and modify these interactively as well as interacting through their telephone and video interconnections. For example, input made by each user to a Whiteboard-client object is relayed to the Whiteboard server-object which updates all of the other corresponding active Whiteboard-client objects of the changes.

Although only Venue shared objects and Whiteboard shared objects are available to a user in this embodiment. it is envisaged that further possibilities for shared objects are a fax object, a discourse structurer object and tools to control the external media such as a virtual monitor manager and a video cassette recorder controller.

It is envisaged that a system according to the present invention may not entail the use of dedicated server machines but that server objects could run on user workstations given a suitable inter-object messaging infrastructure.

APPENDIX A

Brief Description of the Drawings

Figure 1 is a block diagram of a computer in accordance with the preferred embodiment of the present invention.

Figures 2 and 2A show block diagrams which illustrate the relationship between objects, applications and data files in accordance with the preferred embodiment of the present invention.

Figure 3 shows a plurality of objects linked in accordance with a preferred embodiment of the present invention.

Figure 4 shows a series of objects serving as folders, as parents of objects containing data, in accordance with a preferred embodiment of the present invention.

Figure 5 illustrates the screen display which results from linking of various objects in accordance with a preferred embodiment of the present invention.

Figure 6 shows the linking of objects in order to create the screen display shown in Figure 5.

Figure 7 shows how three objects may be linked together in accordance with a preferred embodiment of the present invention.

Figure 8 and Figure 9 illustrate how an object may be copied in accordance with a preferred embodiment of the present invention.

Figure 10 and Figure 11 illustrate the copying of a public object in accordance to a preferred embodiment of the present invention.

Figures 12 through Figure 71 show the appearance on a screen of a session in which a user manipulates objects in accordance with a preferred embodiment of the present invention. Also shown are block diagrams, of how objects appearing to the user are linked in accordance to the preferred embodiment of the present invention.

Figure 72 is a block diagram of an Object Management Facility (OMF) in accordance with the preferred embodiment of the present invention.

Figure 73 shows a block diagram of the organization of HPOMF.CAT, a system file included in the OMF shown in Figure 72.

Figure 74 shows the relation between a global parent and global objects in accordance with the preferred embodiment of the present invention.

Figure 75 is a block diagram which shows how system files within the OMF shown in Figure 72 accesses data files and applications from a memory shown in Figure 1.

Figure 76 is a block diagram of the organization of the memory shown in Figure 75.

Figure 77 and Figure 78 show objects and links in accordance with the preferred embodiment of the present invention.

Figure 79 is a block diagram of the organization of HPOMF.XRF, a system file included in the OMF shown in Figure 72.

Figure 80 shows a view specification record in accordance with the preferred embodiment of the present invention.

Figure 81 shows the use of a snapshot in accordance with a preferred embodiment of the present invention.

Figure 82 shows the data path of a view when there is no snapshot, in accordance with a preferred embodiment of the present invention.

Figure 83 shows the data path of a view when there is a snapshot, in accordance with a preferred embodiment of the present invention.

Description of the Preferred Embodiment

Figure 1 shows a computer 18 having a monitor 14, a keyboard 19 and a mouse 20. A portion of computer main memory 17 is shown by an arrow 9 to be within computer 18. Within computer memory main 17 is shown an object management facility (OMF)100, an application 101, an application 102, an application 103, an application 104, an application 105 and an application 106.

Each of applications 101 to 106 store data using objects. For instance, in Figure 2, application 101 is shown to have stored data using an object 202, an object 203, an object 204 and an object 205. Similarly, application 106 is shown to have stored data in an object 207, an object 208, an object 209 and an object 210. OMF 100 stores information indicating which objects go with which application. Objects which are associated with a single application are considered to be objects of the same type, or the same class. For instance, object 202, 203, 204 and 205 are of the same class because each is associated with application 101. Similarly objects 207, 208, 209 and 210 are of the same class because each is associated with application 106. All objects of the same class use the same application. When an application is being run by computer

18, OMF 100 informs the application which object the application should access for data. That object is then considered to be active. An object is inactive when the application the object is associated with is not being run by computer 18, or when the application the object is associated with is being run, but is not being run with the data of that object.

Active objects can communicate with each other using messages. For example if two instances of application 101 are being run by computer 18, one with the data of object 202 and the other with the data of object 203, object 202 and object 203 are both active. Therefore object 202 may send a message 211 to object 203. Similarly, if computer 18 is running application 101 with the data of object 202, and is running application 106 with the data of object 207, object 202 and object 207 are both active. Therefore, object 202 may send a message 212 to object 207.

Messages, such as message 211 and 212 may be formatted to be sent and received by all types of objects. This allows for free communication between all active objects. This also allows new object types to be defined and added to the system without requiring that the existing object types be updated to use the new type.

Each object has associated with it a set of data files. For instance, object 210 is shown to have associated with it a data file 221, a data file 222 and a data file 223. Data in data files 221, 222 and 223 are in a format which can be interpreted by application 106.

Each object has associated with it a list of properties. Each property has a name and a value which may be accessed by specifying the name. In addition, each class of objects has associated with it a list of properties that are common to all objects of that class. For instance, in Figure 2A, object 205 and application 101 are shown. Object 205 has associated with it a property 231, a property 232, and a property 233. Application 101 has associated with it a property 131, a property 132 and a property 133.

Property lists can contain any number of properties. Each property value can be from zero to 3,2762 bytes in length. Properties are used to store descriptive information about objects and classes, such as names, comments and so on.

Objects may have references to other objects. These references are called links. Links are directional: one object is called the parent, the other the child. Each link has a reference name which is a number that is assigned by the parent object to identify each of its children. All of an object's children, its children's children, and so on are collectively called that object's descendents. Similarly, an object's parents, its parents' parents, and so on, are collectively called that object's ancestors. In the preferred embodiment of the present invention, an object which may be manipulated by a user, can have zero or more children and one or more parents. An object is not allowed to become its own descendent.

In Figure 3 is shown an object 301, an object 302, an object 303, an object 304, an object 305, an object 306, an object 307, an object 308 and an object 309. Objects 301-309 have links with reference names which are numbers shown in parenthesis by each link. Object 301 has a link 310, with reference name "1", to object 302. Object 301 has a link 311, with reference name "2", to object 303. Object 302 has a link 312, with reference name "7", to object 304. Object 302 has a link 313, with reference name "8", to object 305. Object 303 has a link 314, with reference name "1", to object 306. Object 303 has a link 315, with reference name "4", to object 307. Object 304 has a link 316, with reference name "1", to object 308. Object 305 has a link 317, with reference name "7", to object 308. Object 306 has a link 318, with reference name "8", to object 309. Object 307 has a link 319, with reference name "9", to object 306. Object 307 has a link 320, with reference name "13", to object 309. Object 308 has a link 321, with reference name "1", to object 309. Object 308 has a link 322, with reference name "3", to object 303.

Object 301 is a parent of 302 and 303. Object 303 is a child of object 301 and of object 308. Each of objects 302-309 are descendents of object 301. Descendents of object 303 are objects 306, 307 and 309. Object 309 has for ancestors all of objects 301-308. Object 303 has for ancestors objects 301, 302, 304, 305 and 308. And so on.

Active objects can dynamically make and delete links to other objects. When a link to an object is deleted, OMF 100 checks if the object has any other parents. If not, OMF 100 destroys the object by deleting the data files of the object and reclaiming other storage space associated with the object.

Object links may be used for various purposes. For example, folders may be in the form of objects. The children of objects used as folders may be objects containing data for use with various applications, or the objects may be other folders. Figure 4 shows an example of the use of objects as folders. An object 401 (also called folder 401), an object 402 (also called folder 402), an object 403 (also called folder 403) and an object 404 (also called folder 404) are used as folders. Folder 401 contains an object 405, used to contain data, an object 406, used to contain data, an object 407, used to contain data, and folder 402. Folder 402 contains an object 408, used to contain data, folder 403 and folder 404. Folder 403 contains an object 409, used to contain data, and an object 410, used to contain data. Folder 404 contains an object 411, used to contain data, an object 412, used to contain data and an object 413, used to contain data.

A more sophisticated use of links is to construct compound objects. For instance in Figure 5, a document 510 contains lines of text 511, lines of text 512, a graphics figure 513, a graphics figure 514 and spreadsheet data 515. As shown in Figure 6, text and formatting data is stored in an object 611, graphics data for graphics figure 513 is stored in an object 612, graphics data for graphics figure 514 is stored in an object 613 and spreadsheet data 515 is stored in object 614. Links that are used to build compound objects always have some kind of data transfer associated with the link and hence are called data links. In Figure 6 is shown a data link 615, a data link 616 and a data link 617. In document 510, data from object 612, object 613 and object 614 are merely displayed, therefore data link 614, data link 615 and data link 616 are visual data links. In a visual data link, the parent will send requests to its child to display data within the parent's window.

In Figure 7, an object 701, which contains data for a first spreadsheet, is linked through data link 704 to an object 702, which contains data for a second spreadsheet, and is linked through data link 705 to an object 703, which contains data for a third spreadsheet. The first spreadsheet uses data from the second spreadsheet and from the third spreadsheet. Since the first spreadsheet does more than merely display data from the second and the third spreadsheets, data link 704 and data link 705 are called data-passing data links.

OMF 100 does the "bookkeeping" when objects are copied or mailed. When an object is copied, OMF 100 makes copies of data files associated with the object. If the object being copied has children, OMF 100 also makes copies of the object's descendents, and builds links between the new objects to give the new compound object the same structure as the original.

For instance, Figure 8 shows object 308, from Figure 3, and the descendents of object 308. When OMF makes a copy of object 308, OMF copies each of object 308's descendents and the links shown in Figure 8. Figure 9 shows a copy of object 308. Object 308a is a copy of object 308. Object 303a is a copy of object 303. Object 306a is a copy of object 306. Object 307a is a copy of object 307. Object 309a is a copy of object 309. Link 321a is a copy of link 321. Link 322a is a copy of link 322. Link 314a is a copy of link 314. Link 315a is a copy of link 315. Link 318a is a copy of link 318. Link 319a is a copy of link 319. Link 320a is a copy of link 320.

In the preferred embodiment, the default behavior results in the copy of a parent's children when the parent is copied. However, when a child is designated as "public" it is not copied. Rather, a copy of the parent includes a link to the child. For instance, in Figure 10, a parent object 161 is to be copied. Parent object 161 is linked to a child object 162 through a link 163. Child object 162 is a public object. As shown in Figure 11, copying of parent object 161 results in new object 161a being linked to object

162 through a new link 163a. Object 161a is a copy of object 161. Link 163a is a copy of link 163.

In Figure 12 through Figure 71, it is shown how objects are displayed to a user on monitor 14. In Figure 12 a "NewWave Office" desktop is shown to include icons labelled as "File Drawer", "Waste Basket", "Diagnostic", "Printers", "Star" and "My Folder". A user (not shown) has manipulated a cursor 781, using keyboard 19 or mouse 20, to select "My Folder".

Figure 13 shows how the objects displayed on monitor 14 are linked. NewWave Office (shown as an object 700) is the parent of "File Drawer" (shown as an object 701) through a link 711, of "Waste Basket" (shown as an object 702) through a link 712, of "Diagnostic" (shown as an object 703) through a link 713, of "Printers" (shown as an object 704) through a link 714, of "My Folder" (shown as an object 705) through a link 715 and of "Star" (shown as an object 706) through a link 716.

In Figure 14, the user, using cursor 781, has selected "Create a New..." in a pull down menu 782. As a result of this selection a dialog box 779 appears as shown in Figure 15. Using cursor 781, the user has highlighted the icon "Layout" and using keyboard 19 has typed in the name "Paste Up" as a name for a new object to be created. Cursor 781 now points to a region labelled "OK". Once this region is selected, a new object titled "Paste Up" is created, as is shown in Figure 16.

In Figure 17, "Paste Up" is shown as an object 707 linked as a child of NewWave Office through a link 717.

The basic clipboard operations are Cut, Copy, and Paste. The user must select the data that is to be moved or copied, and then give either the Cut command or the Copy command. Cut moves the selected data to the clipboard (deleting it from its original location). Copy makes a copy of the selected data on the clipboard. The user must then select the location where he wants the data to be moved or copied to, and give the Paste command. This command copies the contents of the clipboard to the selected location.

In Figure 18 a user is shown to have selected "Paste Up". The selection is represented by the icon for "Paste Up" being displayed using inverse video. With cursor 781, the user selects "Copy" from a pull down menu 783. In Figure 18A a Clipboard object 720 is shown to be a parent of an object 708 through a link 721. Object 708, is a copy of object 707 ("Paste Up").

As shown in Figure 19, next the user selects "Paste" from pull down men~~l~~u 783. The result, shown in Figure 20, is the addition of an object 708, pointed to by cursor 781, which is a copy of the original "Paste Up" object 707.

In Figure 21, the new object is shown as object 708 linked as a child of NewWave Office through a link 718.

In Figure 22, "My Folder", has been opened by double clicking the icon for "My Folder" using cursor 781. The result is a new window 785 representing "My Folder".

In Figure 23, using cursor 781, "Paste Up" (object 708) is shown being dragged to window 785. In Figure 24, the process is complete and "Paste Up" (object 708) is now in window "My Folder". In Figure 25, "Paste Up", shown as object 708, is now a child of "My Folder" through link 728.

The user sets up multiple links by using the Share command. This command is an extension of the clipboard metaphor common in software packages today for moving and copying data around the system. The clipboard is a special buffer that the system uses to hold data that is in transit.

In one way, the Share command operates similarly to the Cut or Copy command described above. That is, using Share, Cut, or Copy, the user selects some data first and then gives the Share command, which results in something being put on the clipboard. In the case of the Share command, however, what is put on the clipboard is neither the actual data nor a copy of the actual data. Instead, it is a link to the selected data. When this link is pasted, a permanent connection is made between the original data and the location of the Paste. Through use of OMF 100, this link is used by the involved applications to provide easy access to the original data (in its full application) and automatic updating when the original data is modified.

In Figure 26, the NewWave Office window has been activated. "Paste Up" (object 707) has been selected, as evidenced by "Paste Up" (object 707) being in inverse video. Using cursor 781, "Share" from menu 783 is selected. In

Figure ~~720~~ 26A, Clipboard object 720 is shown to be a parent of "Paste Up" object 707 through a link 722.

In Figure 27, window 785 has been activated. From a menu 787, "Paste" is selected. The result, shown in Figure 28, is an icon 707a appearing in window 785, which indicates that "Paste Up" (object 707) is shared by window 785 and the NewWave Office window. In Figure 28A, as a result of the paste, "Paste Up" is now shown to be both a child of Clipboard 720 through link 722 and a child of "My Folder" 705 through a link 727. In Figure 29, showing just the interconnection of objects visible to the user, "Paste Up" (object 707) is shown to be a child of "My Folder" 705 through link 727. Since "Paste Up" (object 707) is shared, not copied, "Paste Up" (object 707) remains a child of NewWave Office through link 717.

One key feature of data links is automated data transfer. When a child object is open and the user changes a part of it which is "shared out", then it makes a call to OMF 100. OMF 100 checks if any of the object's parents "care" about this particular change. If they care and if they are also open, OMF 100 sends to the parents a message informing them that new data is available. The parent can then send messages to the child to produce or display the data. This feature allows the user to establish compound objects with complex data dependencies, and then have changes made to any sub-part be automatically reflected in other parts. For example, changing a number in a spreadsheet could cause a graph to be re-drawn, and updated as a figure in a document. And since an object can have many parents, a single object can be used as "boiler plate" for any number of other objects. A change in the boiler plate will be reflected in all the objects which have links to it. Automated data transfer is illustrated in the following discussion.

In Figure 30, window 785 for "My Folder" has been closed. In Figure 31, cursor 781 is used to select "Create a New..." from pull down menu 782. As a result of this selection dialog box 779 appears as shown in Figure 32. Using cursor 781, the icon HPText has been highlighted and using keyboard 19 the name "Sample Text" has been typed in as the name for a new object to be created. Cursor 781 now points to a region labelled "OK". Once this region is selected, a new object titled "Sample Text" is created, as is shown in Figure 33.

In Figure 34, "Sample Text" (object 709) is shown to be a child of NewWave Office through a link 719. In Figure 34, since "My Folder" has been closed, "Paste Up" (object 708), link 728 and link 727 are not shown. However, these still exist, but are not currently visible to a user.

In Figure 35, placing cursor 781 on the icon "Sample Text" and double clicking a button on mouse 20 results in "Sample Text" being opened. In Figure 36, an open window 789 for "Sample Text" is shown.

In Figure 37 a window 791 for "Paste Up" (object 707) has been opened by double clicking on the icon for "Paste Up". In Figure 38, using Cursor 781, controlled by mouse 20, a portion 790 of the text of "Sample Text" has been selected. The portion in inverse video stating "New Wave Office environment" is portion 790.

In Figure 39, cursor 781 is used to select the selection "Share" in a pull down menu 792. In Figure 40, an area 793 in window 791 is selected using cursor 781. In Figure 41, a selection "Paste" is selected from a pull down menu 794 using cursor 781. In Figure 42, "Sample Text" is linked to "Paste Up" (object 707) and displayed text 790 is displayed in "Paste Up" window 791. In Figure 43 "Sample Text" (object 709) is shown to be a child of "Paste Up" (object 707) through a link 729. In Figure 42, displayed text 790 is shown in gray because "Star" window 789 is open. In Figure 44, "Star" window 789 is closed so displayed text 790 is clearly displayed.

In Figure 45, a region 795 of window 791 is selected using cursor 781. Figure 46 shows cursor 781 dragging the icon "Star" into region 795 of window 791.

In Figure 47, data from "Star" (object 706) is now displayed in region 795 of window 791. As may be seen in Figure 48, "Star" (object 706) is now a child of "Paste Up" (object 707) through a link 726.

In Figure 49, a user has placed cursor 781 over region 795 of window 791 and double clicked a button on mouse 20.

The result is the opening and display of "Star" (object 706) in a window 796. Figure ~~40~~ 50 shows the use of cursor 781 to select selection "Ellipse" in a menu window 797 which results in the data within "Star" (object 706) being changed from a star to an ellipse. As shown in Figure 51, the result is a change both in data displayed in window 796 and data displayed in region 795 of window 791.

In Figure 52, cursor 781 is used to define a region 797 in window 791. In Figure 53, cursor 781 is used to select a selection "Create a New..." in pull down menu 798. As a result of this selection dialog box 799 appears in Figure 54. Dialog box 799 contains icons for the two classes of objects available which are able to display data in region 797 of window 791. Using cursor 781, the icon "HP Shape" has been highlighted. Using keyboard 19 the name "New Shape" has been typed in as the name for a new object to be created. Cursor 781 now points to a regions labelled "OK". Once this region is selected, a new object titled "New Shape" is created. Data for "New Shape" is displayed in region 797 of window 791 as is shown in Figure 55. In Figure 56, "New Shape", (object 750) is shown to be a child of "Paste Up" (object 707) through a link 760.

In Figure 57 a window 800 for "New Shape" was opened by placing cursor 781 over region 797 of window 791 and clicking twice on a button on mouse 20. In Figure 58, cursor 781 is used to select the selection "Triangle" from a pull down menu 801. The result, as shown in Figure 59, is that a triangle is now displayed both in window 800 and in region 797 of window 791.

In Figure 60, window 800 has been closed. In Figure 61, "New Shape" is selected by placing cursor 781 over region 797 of window 796, and clicking a button on mouse 20. In Figure 62, cursor 781 is used to select selection "Share" from pull down menu 794. In Figure 63, cursor 781 is used to select a region 802 of window 791. In Figure 64, cursor 781 is used to select selection "Paste" from pull down menu 794. The result, as shown in Figure 65, is the sharing of "New Shape" with data from "New Shape" being displayed in region 797 and in region 802 of window 791. In Figure 66, "New Shape" (object 750) is shown to have an additional link 770, from its parent "Paste Up" (object 707).

In Figure 67, region 797 has been selected using cursor 781. Cursor 781 is then used to select selection "Cut" from pull down menu 794. The result, as seen in Figure 68, is that region 781 has been removed from window 791. In Figure 69, cursor 781 is used to select selection "Paste" from pull down menu 783. The result, shown in Figure 70, is an icon for "New Shape", pointed to by cursor 781. In Figure 71, "New Shape (object 750) is shown to now be a child of NewWave Office (object 100), through a link 780.

In Figure 72, OMF 100 is shown to contain seven system files: system file 601, system file 602, system file 603, system file 604, system file 605, system file 606 and system file 607. OMF interface 599 serves as interface of OMF to other programs running on computer 18. System files 601-607 serve as a data base that provides various information. They provide information about object properties such as what class each object is what is the name of each object. System files 601-607 provide information about classes of objects such as what application is associated with each class of objects, what icon represents objects of a particular class and lists of what messages (such as those shown in Figure 2) can be processed by objects of a particular class. System files 601-607 also contain information about links between parent and child objects including a list of parents and reference names of each link from a parent for each object; a list of children and reference names of each link to a child for each object; and additional information to manage data exchange across data links. Additionally, system files 601-607 contain general information such as what files are installed in the operating system for each class that is installed, and what objects have requested automatic restart when the OMF 100 is restarted.

In the preferred embodiment of the present invention system file 601 is referred to as HPOMF.CAT, system file 602 is referred to as HPOMF.CLS, system file 603 is referred to as HPOMF.XRF, system file 604 is referred to as HPOMF.PRP, system file 605 is referred to as HPOMF.INS, system file 606 is referred to as HPOMF.SDF and system file 607 is referred to as HPOMFICO.NWE. A description of each system file is now given.

System file 601, HPOMF.CAT, is also referred to as SYSCAT. HPOMF.CAT is a catalog of all the existing objects in the system. In Figure 73, HPOMF.CAT is shown to be record oriented. HPOMF.CAT has a plurality of file records. In Figure 73, file record 0 through file record 8 are shown, although HPOMF.CAT may contain many more file records than are shown in Figure 73. File record 0 is a header which contains various signatures and is used to manage a list of free file records. A signature is some known value which if present indicates that the file is not corrupted. File record 1 through file record 8 and additional file records (not shown) either define an existing object, or are free. In the preferred embodiment HPOMF.CAT can grow dynamically, as more file records are needed, but cannot shrink.

File record 1 defines a special object called the global parent. The global parent has a form different than every other object, and may be regarded as a "pseudo" object. Figure 74 shows the global parent to be the parent of global object 250 through link 260, global object 251 through link 261, global object 252 through link 262, global object 253 through link 263, global object 254 through link 264 and global object 255 through link 265, as shown. Global objects 250-255 are also within HPOMF.CAT. Each global object 250-255 may be a parent of one or more objects in HPOMF.CAT. Each object in HPOMF.CAT which is not a global object, is a descendent of global object. Although Figure 74 shows only six global objects, the number of global objects operating on a system is a matter of system configuration. Any object in the system can refer to a global object by ~~by~~ using the reference name of the link to that global object from the global parent.

As may be seen from Figure 73, file records in HPOMF.CAT are numbered consecutively. These numbers serve as tags, which identify each object.

In the preferred embodiment of the present invention, each record is 128 bytes in length. The fields for file record 0 are listed in Table 1 below:

Table 1

lFirstFreeEntry	Contains the record number of the first free record in HPOMF.CAT, or "0" if there are no free records.
FileId	Contains the null terminated string "HPOMF.CAT". This serves as a signature.
Version	Contains the file format version number, which also serves as a signature.
lMaxRecordNumber	Contains the number of the highest record ever allocated from within HPOMF.CAT (this highest record may or may not be free).

Table 2, below, contains the fields for file records in HPOMF.CAT for file records other than file record 0:

Table 2

lFirstFreeEntry	Is "-1" if this record defines an object, otherwise this record is free and this field is the record number of the next free record, or "0" if there are no more free records. If the record is free, none of the other fields in the record is meaningful.
TypeInClass	Specifies the class of this object. This is the number of the record in HPOMF.CLS that indicates to which class the object belongs (see discussion of class above).
SysCatFlags	Specifies if the object is global if the bit masked by the number 20 (hexadecimal) is set in this byte. In the preferred embodiment all other bit positions must contain "0" and are not used.
properties	Specifies the number of properties, the length of the property names and the location in HPOMF.PRP of the object's properties. See the description of HPOMF.PRP below for further definition of the structure of this field.
fastprops	Certain object properties, such as name, are so heavily accessed that they are stored directly in this field, rather than indirectly in the properties file. Properties stored in this field are called "fast properties."

System file 602, HPOMF.CLS is also referred to as SYSCLASS. This system file is a list of all installed classes in the system. It is record oriented. The first record, numbered 0, is a header which contains various signatures (see above) and is used to manage a list of free records. All other records either define an installed class or are free. In the preferred embodiment HPOMF.CLS can grow dynamically, but cannot shrink.

Each file record in HPOMF.CLS is thirty-two bytes in length. HPOMF.CLS file record 0 (the header) contains the following fields listed in Table 3:

Table 3

1FirstFreeEntry	Contains the record number of the first free record in HPOMF.CLS, or "0" if there are no free records.
FileId	Contains the null terminated string "HPOMF.CLS"
Version	Contains the file format version number.
1MaxRecordNumber	Contains the number of the highest record ever allocated from within HPOMF.CLS (this highest record may or may not be free).

Table 4, below, contains the fields for file records in HPOMF.CLS for file records other than file record 0:

Table 4

lFirstFreeEntry	Is "-1" if this record defines an installed class, otherwise this record is free and this field is the record number of the next free record, or "0" if there are no more free records. If the record is free, none of the other fields in the record is meaningful.
ModuleFileName	Specifies the name of the application associated with objects of this class as a null-terminated string.
properties	Specifies the number of properties, the length of the property names and the location in HPOMF.PRP of the object's properties. See the description of HPOMF.PRP below for further definition of the structure of this field.

In Figure 75, the relationship of HPOMF.CAT and HPOMF.CLS is shown. Within each object entry within HPOMF.CAT, the record number, which is an object's tag, serves as an identifier 650 of data files in a mass storage memory 170 associated with the object. The field "TypeInClass" serves as an identifier 651 of the class entry in HPOMF.CLS, which identifies the class of each object. Within each class entry in HPOMF.CLS, the field "ModuleFileName" serves as an identifier 652 of the application file in mass storage memory 170 which is associated with the class.

In Figure 76, the organization of a portion of mass storage memory 170 is shown. A root directory 660 contains pointers to an HPNWDATA directory 661 and HPNWPROG directory

668. HPNWPROG directory 668 is the location of storage for applications files, represented by arrows 669. HPNWDATA contains a plurality of HPOMFddd directories, represented by directories 662, 663, 664, 665 and 666. In the HPOMFddd directories are stored data files associated with objects. The "ddd" in HPOMFddd stands for a three digit, leading zeros, hexadecimal number. Each HPOMFddd directory has a different "ddd" hexadecimal number. The "ddd" number indicates which HPOMFddd directory stores data files for a particular object. Data files for a particular object are stored in the HPOMFddd directory which has a "ddd" number equal to the tag for the object divided by an integer number, e.g., fifty four. Within each HPOMFddd directory, files are stored by tag numbers, e.g. data file names have the format xxxxxxxx.111, where "xxxxxxxx" is an eight digit leading zeros hexadecimal tag, and "111" are a reference chosen by the application.

System file 603, HPOMF.XRF is also referred to as SYSXREF. This file is a list of all the links existing in the system. It is record oriented, but does not have a header record. Each record file is either free, or defines an existing link, or is used as an overflow record from the previous record to specify additional view specification information. Records that contain view specifications are called view specification file records. View specification file records can be identified only by a previous record which defines an existing data link; view specification file records cannot be identified by the content within a view specification file record. HPOMF.XRF is increased in size 16K bytes at a time. A newly allocated portion of HPOMF.XRF is filled with zeros. File records within HPOMF.XRF which are free or which define a link have the following fields listed in Table 5:

Table 5

ParentTag	Contains the tag (HPOMF.CAT record number) of the parent object of this link. If this field is 0, then this record does not define a link and is free.
ChildTag	Contains the tag of the child object of this link. If ParentTag in this record is 0, and this field is also 0, then no record beyond this record in HPOMF.XRF defines a link.
RefName	Contains the reference name that the parent has assigned to the link. This field is meaningless if ParentTag or ChildTag is zero. Otherwise, if the top three bits of this value are 110, the next record in the file is a view specification.

File records within HPOMF.XRF which are view specification file records have the folloying fields listed in Table 5A:

Table 5A

DataId	Contains the value that the child has assigned to identify the part of itself that is being viewed through the link.
Snapshot	Contains the tag (HPOMF.CAT record number) of the object which is the view's snapshot, or if zero, the view has no snapshot. For further discussion of snapshots, see below.
Misc	Composed of several bit fields described below:
VS_NEWDATASET	Set if child has told OMF that new data is available, but has not been announced to the parent. The hexadecimal number 8000 0000 is a mask which indicates which bits are used for this bit field.
VS_NEWDATAANNOUNCED	Set if child has told OMF to announce new data to parent, but parent was inactive and was not notified. The hexadecimal number 4000 0000 is a mask which indicates which bits are used for this bit field.
VS_SNAPSHOTOLD	Set if child has told OMF that the view's snapshot is out-of-date. The hexadecimal number 2000 0000 is a mask which indicates which bits are used for this bit field.
VS_WANTMESSAGES	Set if child has told OMF that it wants to process view messages when snapshot is out-of-date. The hexadecimal number 1000 0000 is a mask which indicates which bits are used for this bit field.

VS_TEXTDISKLOC

File position in HPOMF.PRP where a view's 32 character textual data ID is located. This contains zero if no textual data ID has been defined by the child. The low order five bits of the file position are always zero and are thus not stored in the Misc field. The hexadecimal number OFFF FFEO is a mask which indicates which bits are used for this bit field.

VS_INITIALIZED

Set if the view specification has been initialized. If clear, all information in the view specification is zero. The hexadecimal number 0000 0010 is a mask which indicates which bits are used for this bit field.

VS_RESERVED

Reserved for future expansion. The hexadecimal number 0000 0008 is a mask which indicates which bits are used for this bit field.

VS_VIEWCLASS

Specifies the view class the child assigned to the view. The view class defines what view methods are available to the parent. The hexadecimal number 0000 0007 is a mask which indicates which bits are used for this bit field.

For example, in Figure 77, Object 671 is a folder and has a tag of "6". Object 671 is a parent of an object 672 through a link 674 and a parent of an object 673 through a link 675. Object 672 has a tag of "12". Link 674 as a reference name "1". Object 673 has a tag of "19". Link

675 has a reference name "7". Reference names are picked by the parent object and need to be unique for the particular parent object; however, other parents may have a link with the same reference name provided each reference name is unique for each parent.

Figure 79 shows a block diagram of HPOMF.XRF 603. HPOMF.XRF contains an entry for each link between parents and children. In HPOMF.XRF 603 column 731 contains the tag of the parent for each link. Column 732 contains the tag of the child for each link. Column 733 contains the reference name for each link. The first three bit positions of column 733, shown in Figure 79 as sub-column 734, indicate whether a view specification file record is present ("110") whether no view specification file record follows ("000") or whether the link is between is a link from the global parent to a global object ("100").

As may be seen, entry 735 is an entry which describes link 674 shown in Figure 77. That is, in column 731 of entry 735 there is the parent tag "6". In column 732 there is the child tag "12" and in column 733 there is the reference name "1". Since object 671 is a folder, there is no view, therefore the three bits within subcolumn 734 would be "000".

Similarly, entry 736 is an entry which describes link 675 shown in Figure 77. That is, in column 731 of entry 736 there is the parent tag "6". In column 732 there is the child tag "19" and in column 733 there is the reference name

"7". Since object 671 is a folder, there is no view, therefore the three bits within subcolumn 734 would be "000".

In Figure 78, Object 676 is a document and has a tag of "17". Object 676 is a parent of an object 677 through a link 679 and a parent of an object 678 through a link 680. Object 677 has a tag of "8". Link 679 as a reference name "1". Object 678 has a tag of "21". Link 680 has a reference name "3".

In Figure 79, an entry 737 describes link 679 shown in Figure 78. That is, in column 731 of entry 737 there is the parent tag "17". In column 732 there is the child tag "8" and in column 733 there is the reference name "1". Object 676 is a document, and assuming there is a view associated with link 679, the three bits within subcolumn 734 contain the three bits "110" and entry 738 is a view specification record.

Similarly, an entry 739 describes link 680 shown in Figure 78. That is, in column 731 of entry 739 there is the parent tag "17". In column 732 there is the child tag "21" and in column 733 there is the reference name "3". Assuming there is a view associated with link 680,, the three bits within subcolumn 734 contain the three bits "110" and entry 740 is a view specification record.

In Figure 80, view specification record 740 is shown to include a field 741 which contains a data identification for the view, a field 742 which indicates whether there is a snapshot used in the view, and a field 743 which contains miscellaneous information about the view. The data identification number is used by the child object of the link, to determine what data is sent through the link.

Figures 37 - 43 show the establishment of a link with a view. As has been discussed before, in Figure 37 window 791 for "Paste Up" (object 707) has been opened by double clicking on the icon for "Paste Up". In Figure 38, using Cursor 781, controlled by mouse 20, portion 790 of the text of "Sample Text" has been selected. The portion in inverse video stating "New Wave Office environment" is portion 790.

In Figure 39, cursor 781 is used to select the selection "Share" in a pull down menu 792. Once "Share" is selected, child object 709 ("Sample Text") creates a data identification number which identifies portion 790 of the text to child object 709. Child object 709 also causes OMF 100 to put a link to child object 709 on clipboard 720-- Child object 709 communicates to OMF 100 through command set forth in Appendix B, attached hereto--. Child object 709 also informs OMF 100 what data identification number is associated with the new link between the child 709 and clipboard 720. If there is a snapshot associated with the link, child 709 will also inform OMF 100 if there is a snapshot associated with the link. Snapshots are discussed more fully below. As a result OMF 100 will make an entry in HPOMF.XRF 603 for a link between clipboard 720 and child object 709. The view specification record for the link will include the data identification number given to OMF 100 by child 709.

In Figure 40, area 793 in window 791 is selected using cursor 781. In Figure 41, a selection "Paste" is selected from a pull down menu 794 using cursor 781. At this point parent object 707 ("Paste Up") requests OMF 100 for a link making him the parent of what is on clipboard 720. The view specification record for the link between clipboard 720 and child 709 is copied for link 729 between parent 707 and child 709. In Figure 43 "Sample Text" (object 709) is shown to be a child of "Paste Up" (object 707) through link 729.

In Figure 42, "displayed text 790 is displayed in "Paste Up" window 791. In accomplishing this, parent object 707 makes a call to OMF 100 asking that a message be sent to the object identified by the reference name for link 729. This message requests the child object 709 to display data from this link into a location specified by parent object 707. OMF 100 takes the message from parent 707, adds the data identification number from the view specification record for link 729, and delivers the message to child 709. Child 709 displays the data in the specified location, in this case area 793. The name of the message sent from parent 707 to OMF 100 to child 709 is "DISPLAY_VIEW", further described in Appendix B, attached hereto.

Another message "PRINT_SLAVE", also described in Appendix B, may be used when it is desired to print data on a printer rather than display data on a terminal screen.

In addition, Parent 707 may send a "GET_SIZE" message to child object 709. In a "GET_SIZE" message, parent object 707 identifies a reference name for link 729 and indicates coordinates for a display. OMF 100 takes the GET_SIZE message from parent 707, adds the data identification number from the view specification record for link 729, and delivers the message to child 709. Child 709 returns to parent 707 the size of the portion of the specified area that child 709 would use to display the data. This allows parent 707 to modify the region reserved for displaying data from child 709 when child 709 is not able to scale the data to fit in the region specified by parent 707.

When a data from a child object is being displayed by a parent object, and the child object changes the displayed data, the child objects notifies OMF 100 that there has been a change in the data object. For example, as described above, in Figure 47, data from "Star" (object 706) now displayed in region 795 of window 791. And, as may be seen in Figure 48, "Star" (object 706) is a child of "Paste Up" (object 707) through a link 726. Since data is being passed from child object 706 to parent object 707, link 726 is a data link which includes a view specification.

In Figure 49, the method for changing data in child object 706 is shown. A user places cursor 781 over region 795 of window 791 and double clicks a button on mouse 20. The result is the opening and display of "Star" (object 706) in a window 796. Using cursor 781 to select selection

"Ellipse" in a menu window 797 results in the data within "Star" (object 706) being changed from a star to an ellipse. As shown in Figure 51, the result is a change both in data displayed in window 796 and data displayed in region 795 of window 791.

Child object 706 accomplishes this change by making a call to OMF 100 stating that data associated with the data identification number associated with link 726 is changed. OMF 100 looks up all of the links that use the data identification number. If the parent object of any of the links is not active, OMF 100 sets the bit VS_NEWDATAANNOUNCED for that link in HPOMF.XRF. When the parent object is activated, the parent object can then request the new data.

If the parent object is active, OMF 100 will send a message to the parent object saying that new data is available. OMF 100 will identify to the parent object the reference name of the link for which there is additional data. The parent object sends a message to the child object if it wants the new data displayed. In the present case parent object 707 is active , and has requested the new data to be displayed in region 795 of window 791. A further description of the View Specifications are found in Appendixes B, C and D.

The advantage of the present invention is that parent object 707 is able to communicate with child object 706 through OMF 100, without parent object 707 or child object

706 knowing the identity or any other details about each other. The parent object identifies the link using only the reference name of the link. The child object identifies the link using just the data identification number of the link. OMF 100 does all the translation and identification of which links and which objects are involved.

System file 604, HPOMF.PRP, is also referred to as SYSPROP. HPOMF.PRP contains all the object and class properties except for the fast object properties which are contained in HPOMF.CAT. Each record in system file 601 (HPOMF.CAT) and system file 602 (HPOMF.CLS) has a properties field, as described above. Each properties field contains the fields described in Table 6 below:

Table 6

DirDiskLoc	Contains the position (byte offset) within HPOMF.PRP of the property list directory.
nProps	Contains the number of properties in the property list. This is the number of entries in the directory entry array described below.
PoolSize	Contains the combined length of all the names of the properties in the property list, including a null-terminating byte for each name. This is the size of the directory name pool described below.

For each object and for each class, at the DirDiskLoc position in the HPOMF.PRP file is the property directory for that object or that class. The directory has two major portions: the entry array, followed by the name pool. The entry array has one entry for each property in the property list. Each entry has fields set out in Table 7 below:

Table 7

ValueLen	Specifies the length in bytes of the associated property. This can be zero.
ValueDiskLoc	Contains the position within HPOMF.PRP of the value of the associated property. If ValueLen is zero, this is also zero, and there is no value stored anywhere.
CacheOffset	This field is only used at run time and is not meaningful in the file.

Immediately following the entry array is the name pool. This portion of HPOMF.PRP contains the null-terminated names of properties in the property list, in the same order as the entry array. Properties may include such things as titles, user comments, date and time of creation, the user who created the object, etc. For more information on properties, see Appendix D.

HPOMF.PRP grows dynamically as need. At the beginning of HPOMF.PRP there is a 128 byte bitmap which controls the allocation of the first 1024 pages of HPOMF.PRP. Each page is 32 bytes in length. These pages immediately follow the bit map. The bitmap is an array of words with the most significant bit of each word used first. Thus, bits 15 through C of the first word of the bitmap control the allocation of pages 0 through 15 of the file, respectively.

When storage in the first 1024 pages is insufficient, a second bitmap is added to the file following page 1023. This bitmap controls the allocation of pages 1024 through 2047, which immediately follow the second bitmap. Additional bitmaps and pages are added in the same way, as needed.

Each directory and property value is stored as a single block in the file, i.e., as a contiguous run of pages that are all allocated in the same bitmap. This causes the restriction that no directory or value can exceed 32K bytes (1024 times 32) in length.

System file 605, HPOMF.INS, is also referred to as SYSINSTL. HPOMF.INS contains a list of the files that were copied to the system when each class was installed. This information is used so that these files can be deleted when the class is de-installed.

The very beginning of HPOMF.INS is a double word value which serves as a validity/version identifier. In the preferred embodiment the value of this double word must be 0101ABCD hex to be valid. In Table 8, this number is stored as shown because of the protocols for storage in the particular processor used by the preferred embodiment, i.e. an 80286 microprocessor made by Intel Corporation.

Following the double word comes a series of variable length records. There is one record for each installed class. The first word of each record is the length of the rest of the record, in bytes. This is followed by the null-terminated class name of the installed class. Then follows the file names of the files copied to the OMF directories, each terminated by a null byte, and preceded by a byte which gives the length of the file name, including the length byte and the null terminator. If the file name begins with the special character "*", the file is assumed to be located in the HPNWPROG directory. If the file name begins with the special character "+" the file is assumed to be located in the HPNWDATA directory.

For example, assume two classes are installed: class "AB" and class "CDE". Class "AB" caused two files to be installed: "Z" to HPNWPROG directory 668 and "YY" to the HPNWDATA directory. Class "CDE" caused 1 file to be installed: "XXX" to HPNWPROG directory 668. Given this case Table 8 below shows the contents of HPOMF.INS for this example:

Table 8

offset	content	comments
0	CD AB 01 01	File header/version check
4	0C 00	Length of AB record (12 decimal)
6	41 42 00	"AB" + Null
9	04	Length of length byte "*Z" + Null
A	2A 5A 00	"*Z" + Null .
D	05	Length of length byte + "+YY" + Null
E	2B 59 59 00	"+YY" + Null
12	0A 00	Length of CDE record (10 decimal)
14	43 44 45 00	"CDE" + Null
18	06	Length of length byte + "*XXX" + Null
19	2A 58 58 58 00	"*XXX" + Null

System File 606, HPOMF.SDF is also referred to as the "shutdown file". HPOMF.SDF exists only when the system has been cleanly shut down. It is deleted as the system starts, and created as it shuts down. On startup, if this file is missing, OMF assumes that the last session ended abnormally, and so it goes through its crash recovery procedures to validate and repair the system files as best it can. The system files can be in an invalid but predictable state on a crash. These errors are corrected without user intervention. Certain other kinds of file consistency errors are detected, but are not really possible from an "ordinary" system crash. These errors are in general not correctable and the OMF will not allow the system to come up in this case.

If HPOMF.SDF is present, it contains a list of objects. When the system is being shut down normally, each object which is active at the time can request that the OMF restart them when the system is restarted. The list of objects, then is the list of tags of objects which have requested that they be restarted when the system is restarted.

The first word in HPOMF.SDF is a flag word. If this word is non-zero, OMF will execute its crash recovery code even though HPOMF.SDF exists. Normal shutdown will set this flag when producing the file if some serious error occurred in the session being ended.

After the first word, the rest of the file is a sequence of three byte records. The first two bytes of each record contain the tag of the object to be restored. The least significant byte is first. The third byte is not used in the preferred embodiment, and is zero.

For example, if the system is shut down cleanly in the last session and two objects, having tags of 2 and 7, respectively, have requested restart, the contents of HPOMF.SDF will be as set out in Table 9 below.

Table 9

offset	content	comments
0	00 00	Indicates no crash recovery needed
2	02 00	Tag of first object to restart
4	00	Unused and reserved
5	07 00	Tag of second object to restart
7	00	Unused and reserved

System file 7, HPOMFICO.NWE, is a Microsoft Windows dynamic library executable file which contains a dummy entry point and no data. Microsoft Windows is a program sold by Microsoft Corporation, having a business address at 16011 NE 36th Way, Redmond, WA 98073-9717. HPOMFICO.NWE also contains as "resources" the icons of each installed class. OMF modifies HPOMFICO.NWE directly during run time, and loads and unloads it to get the icon resources from it. The format of HPOMFICO.NWE is defined in Microsoft Windows documentation distributed by Microsoft Corporation.

Normally working with a view (see discussion on views above) causes a child's application to be invoked. Where large applications are involved, this can cause a lot of unnecessary overhead. The use of snapshots allow this overhead to be eliminated.

A snapshot is an object that uses executable code from a separate library referred to as a dynamic access library (or DAL) rather than using the full application executable code. The only data file associated with a snapshot contains data which is to be sent from a child object to a parent object. The code which encapsulates the data file although referred to as a dynamic library, is still stored in directory HPOMFPROG (directory 668).

For example, Figure 81 shows a parent object 501 linked to a child object 502 through a link 504. Associated with link 504 is a snapshot 503. Once child object has designated snapshot 503 in a view specification record for link 504, snapshot 503 is able to provide data from child object 502 to parent 501 without the necessity of invoking an application associated with child object 502.

As shown in Figure 82, when there is no snapshot, child object 502 must be active in order to send view data 522 to parent object 501, in order for parent object 501 to display view data 522 in a window display 521. In Figure 83, however, snapshot 503 is shown to provide view data 522 to parent object 501 without the necessity of child 502 being active. Further implementation details of snapshots are given is Appendix B, Appendix C and Appendix D.

Appendix A is a list of major data structures within OMF 100.

Appendix B is a description of functions which OMF interface 599 recognizes in the preferred embodiment of the present invention.

Appendix C (HP NewWave Environment: Program Design Examples) Gives examples of how the preferred embodiment of the present invention may be implemented.including detail as to how OMF 100 allows data to be viewed between windows displayed on monitor 14.

Appendix D (Chapter 2 of Programmer's Guide) gives a further overview of the preferred embodiment of the present invention. further detail as to the operation of the preferred embodiment of the present invention.

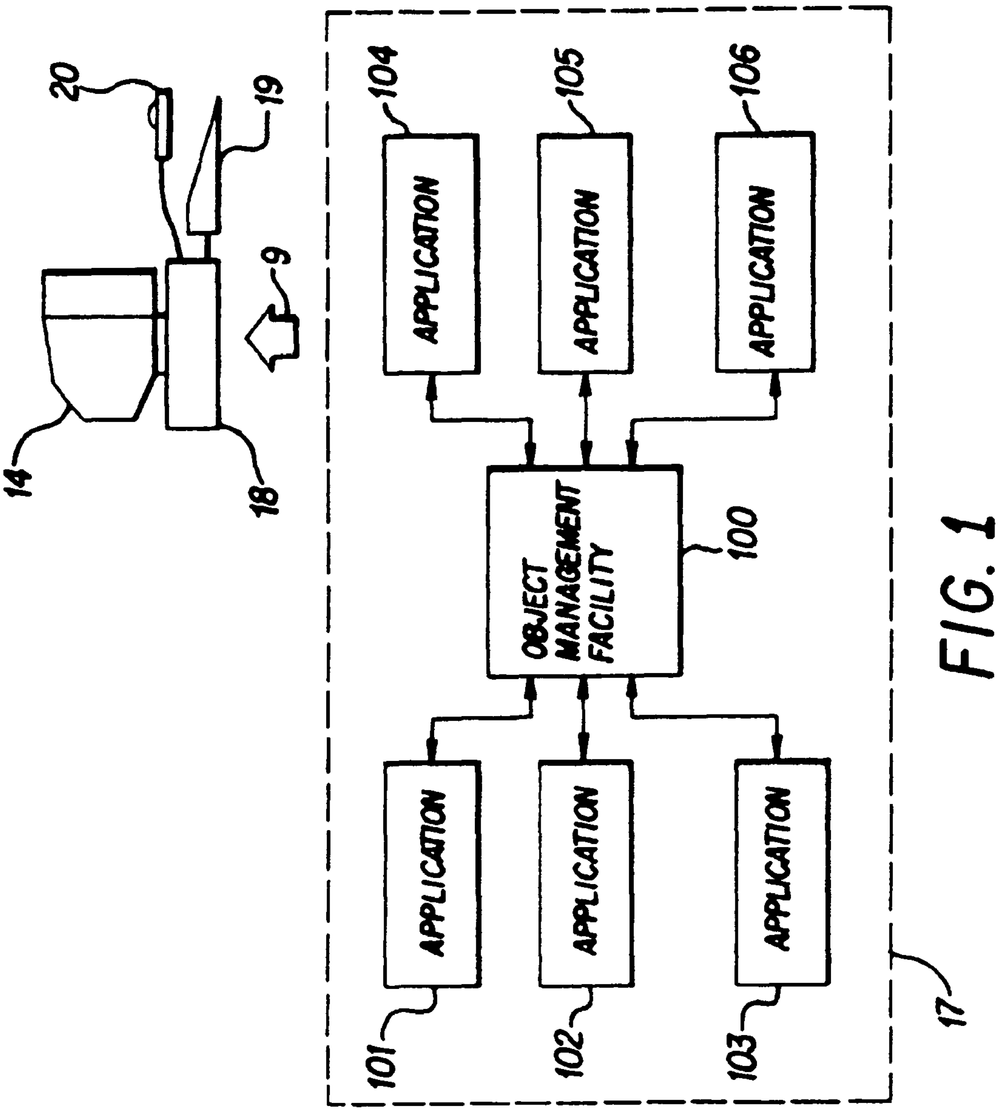

20
19
9
14
18
104
105
106
APPLICATION
APPLICATION
APPLICATION
OBJECT MANAGEMENT FACILITY
100
APPLICATION
APPLICATION
APPLICATION
101
102
103
17

FIG. 1

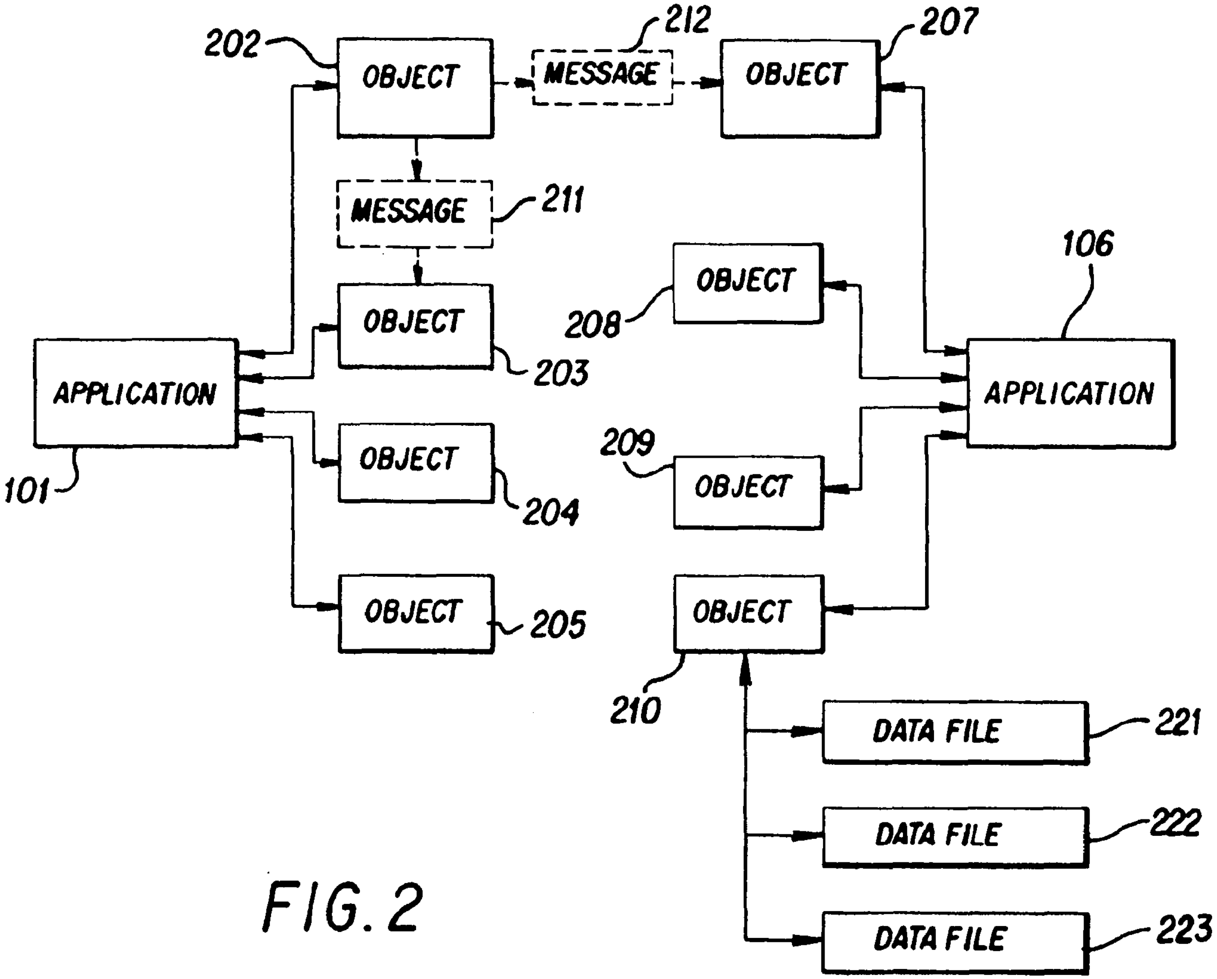

202
OBJECT
MESSAGE
212
OBJECT
207
MESSAGE
211
OBJECT
203
OBJECT
208
106
APPLICATION
APPLICATION
101
OBJECT
204
209
OBJECT
OBJECT
205
OBJECT
210
DATA FILE
221
DATA FILE
222
DATA FILE
223

FIG. 2

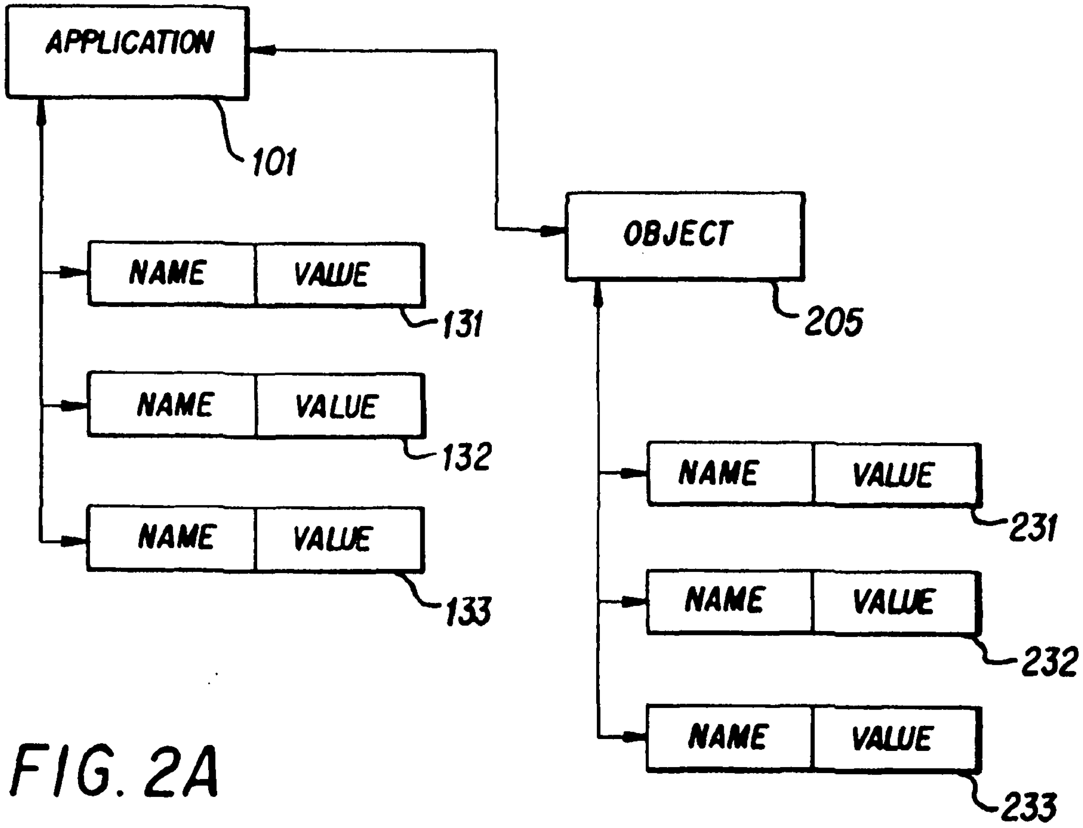

APPLICATION
101
NAME
VALUE
131
NAME
VALUE
132
NAME
VALUE
133
OBJECT
205
NAME
VALUE
231
NAME
VALUE
232
NAME
VALUE
233

FIG. 2A

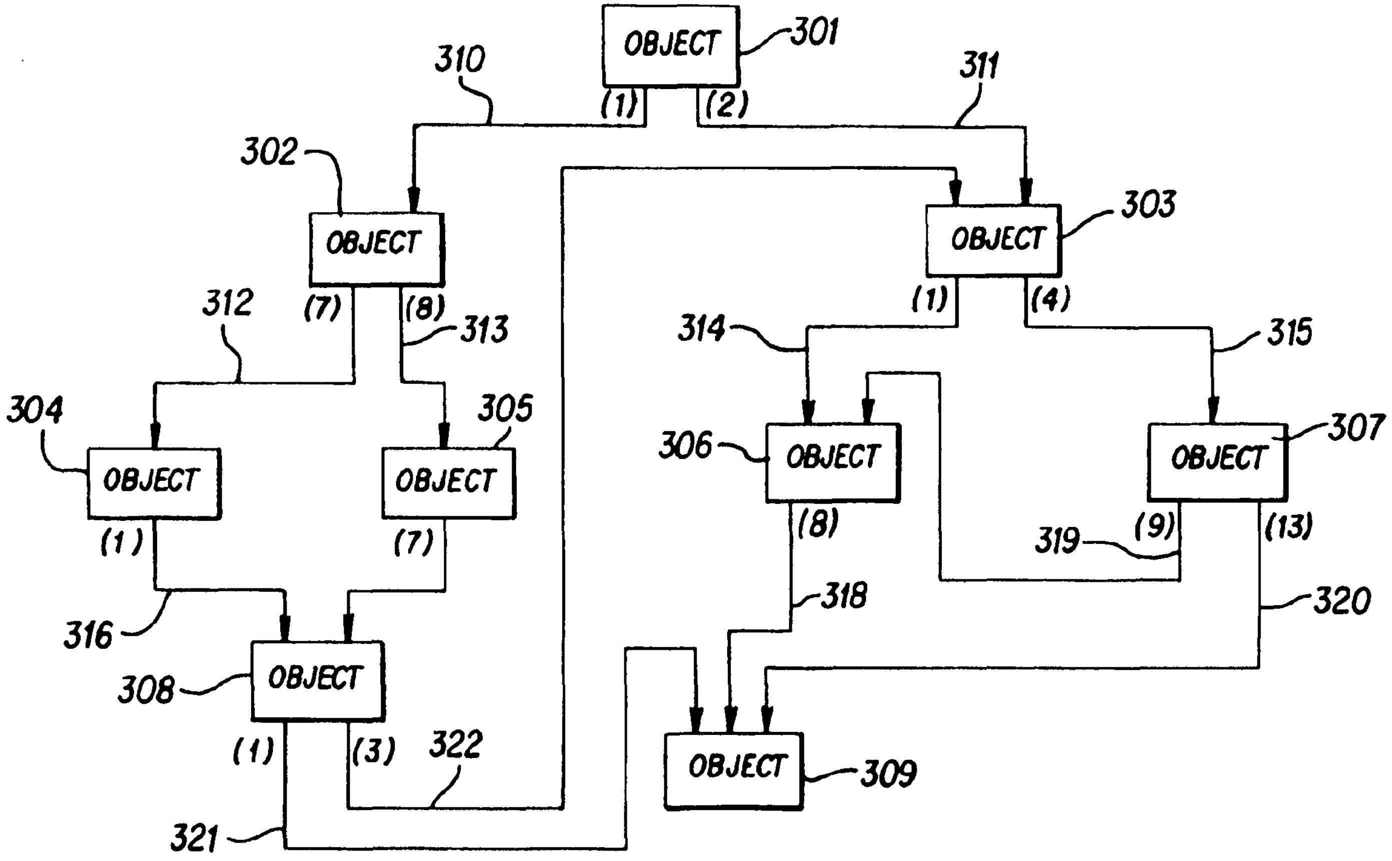

OBJECT
301
(1)
(2)
310
311
302
OBJECT
303
OBJECT
(7)
(8)
312
313
(1)
(4)
314
315
304
OBJECT
305
OBJECT
306
OBJECT
307
OBJECT
(1)
(7)
(8)
(9)
(13)
319
316
318
320
308
OBJECT
(1)
(3)
322
OBJECT
309
321

FIG. 3

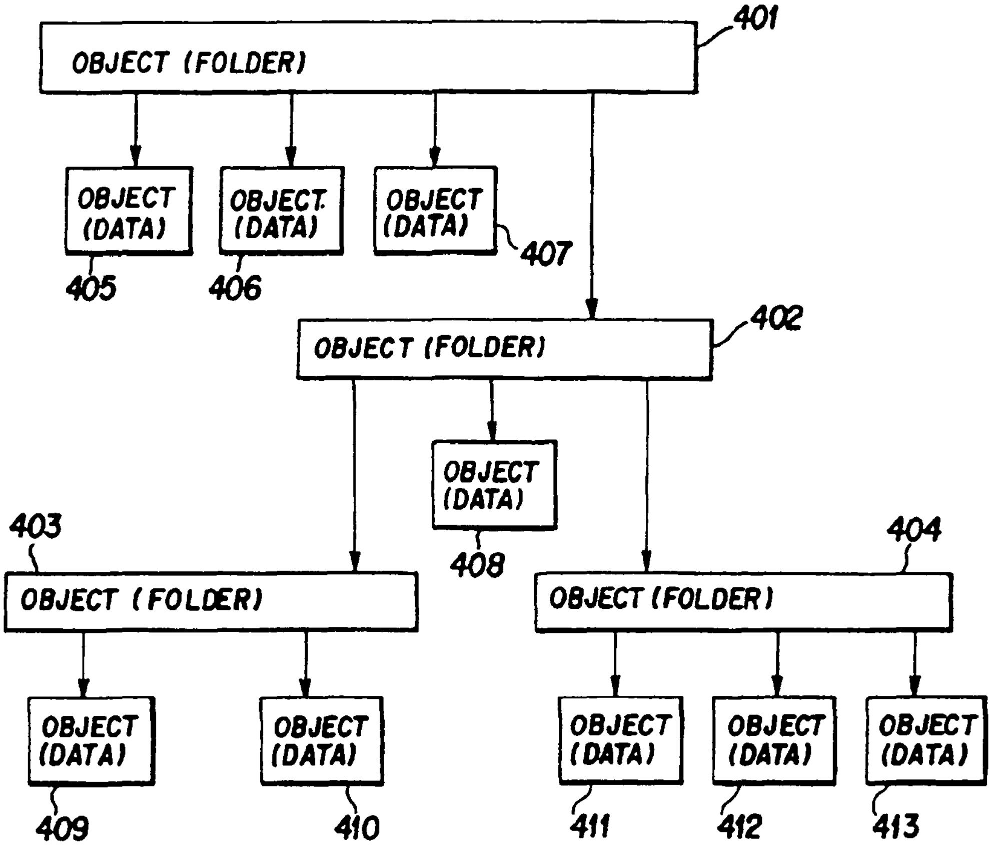
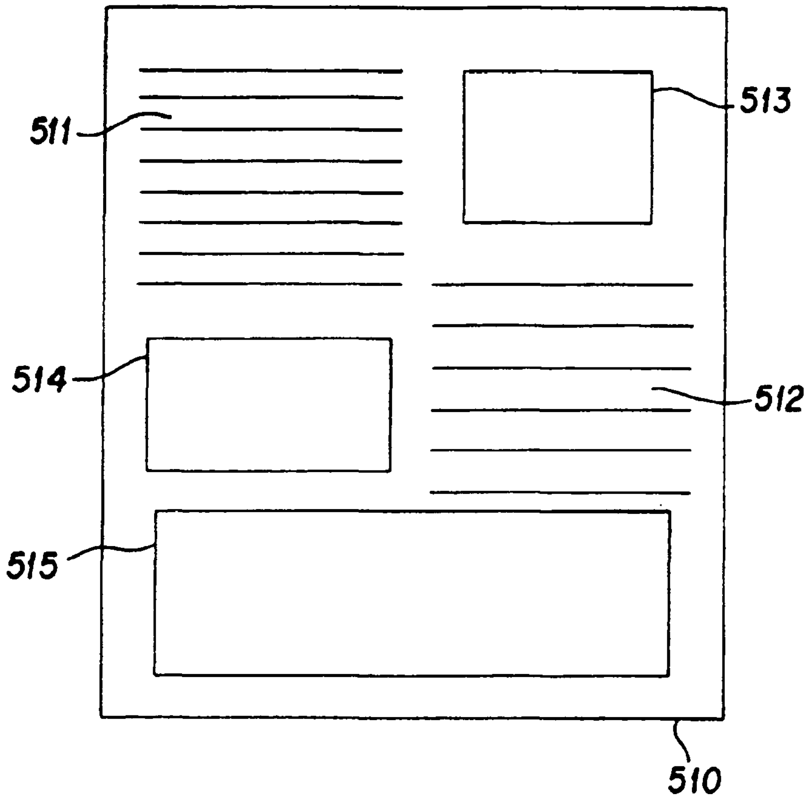

OBJECT (FOLDER)
401
OBJECT (DATA)
OBJECT (DATA)
OBJECT (DATA)
405
406
407
OBJECT (FOLDER)
402
OBJECT (DATA)
408
403
OBJECT (FOLDER)
404
OBJECT (FOLDER)
OBJECT (DATA)
OBJECT (DATA)
OBJECT (DATA)
OBJECT (DATA)
OBJECT (DATA)
409
410
411
412
413

FIG. 4

FIG. 5



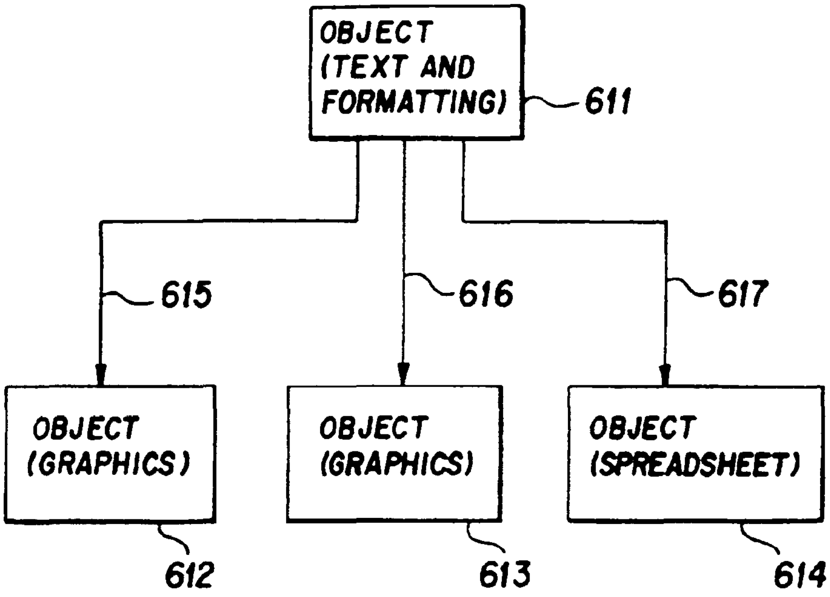

OBJECT (TEXT AND FORMATTING)
611
615
616
617
OBJECT (GRAPHICS)
OBJECT (GRAPHICS)
OBJECT (SPREADSHEET)
612
613
614

FIG.6

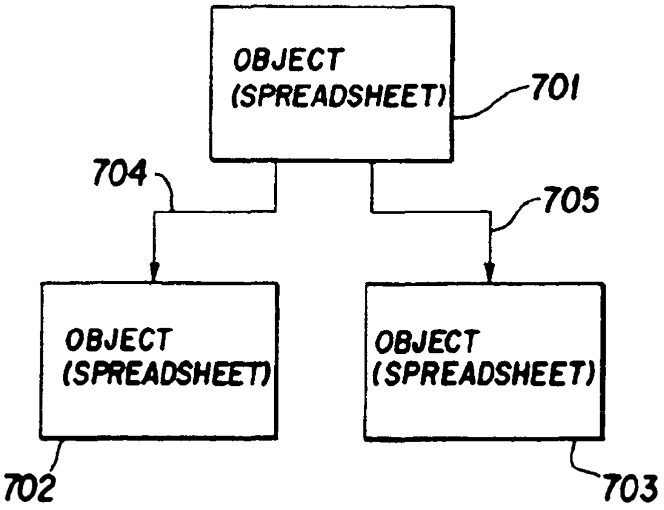

OBJECT (SPREADSHEET)
701
704
705
OBJECT (SPREADSHEET)
OBJECT (SPREADSHEET)
702
703

FIG.7

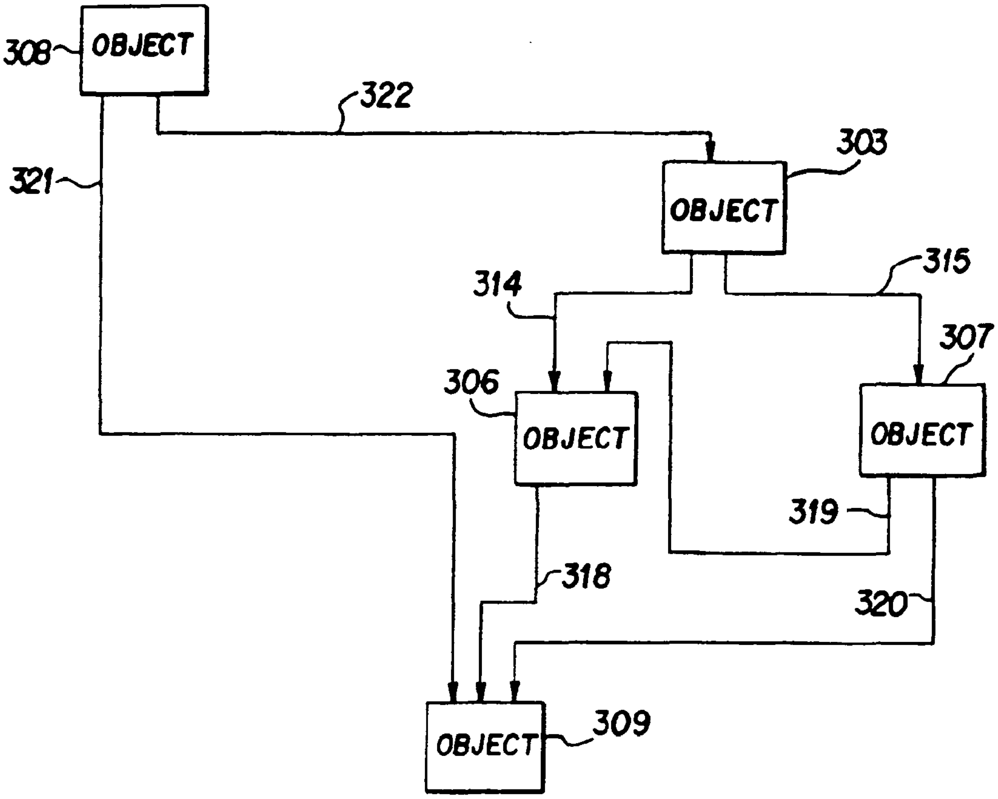

308
OBJECT
322
321
303
OBJECT
314
315
307
306
OBJECT
OBJECT
319
318
320
OBJECT
309

FIG.8

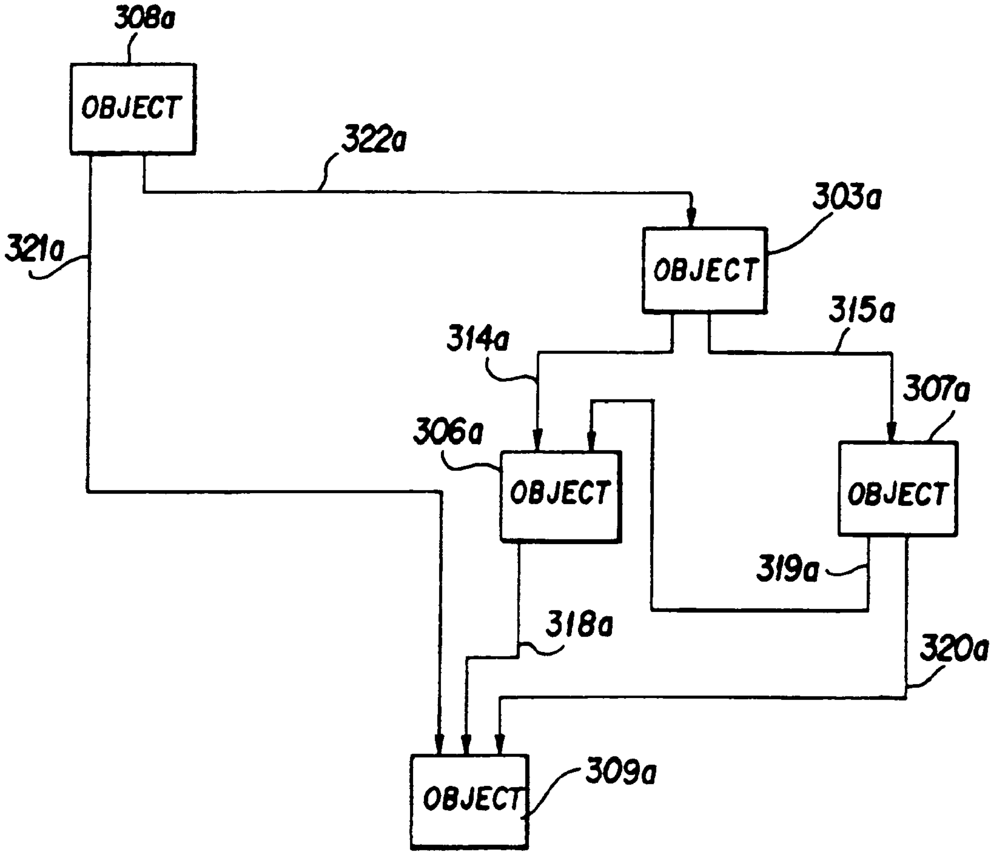

308a
OBJECT
322a
303a
321a
OBJECT
315a
314a
307a
306a
OBJECT
OBJECT
319a
318a
320a
OBJECT
309a

FIG.9

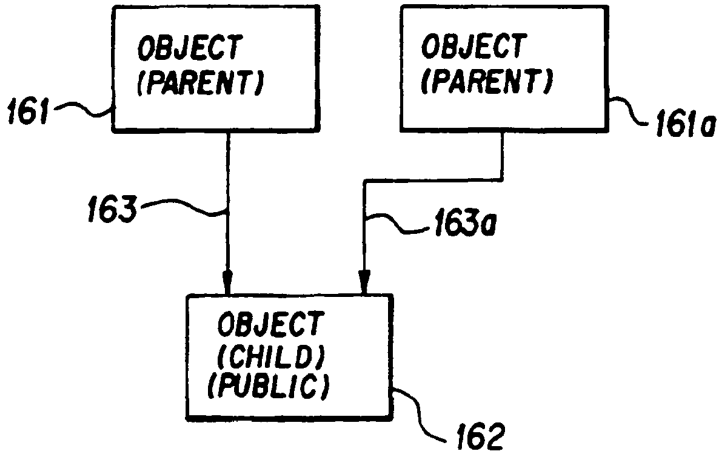

OBJECT
(PARENT)
161
OBJECT
(PARENT)
161a
163
163a
OBJECT
(CHILD)
(PUBLIC)
162

FIG. 11

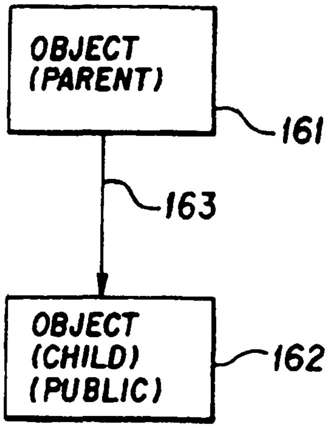

OBJECT
(PARENT)
161
163
OBJECT
(CHILD)
(PUBLIC)
162

FIG. 10

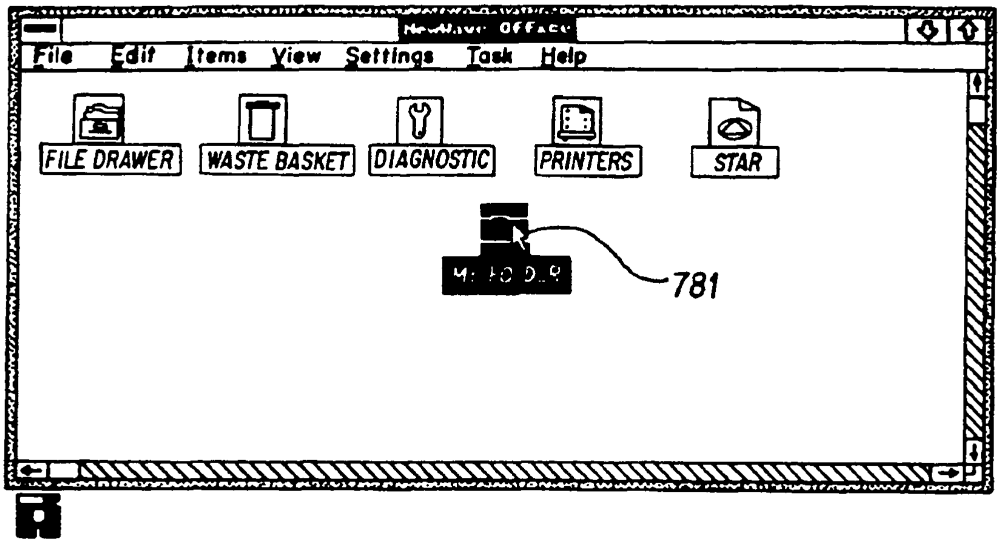

NewWave Office
File
Edit
Items
View
Settings
Task
Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
MY FOLDER
781

FIG. 12

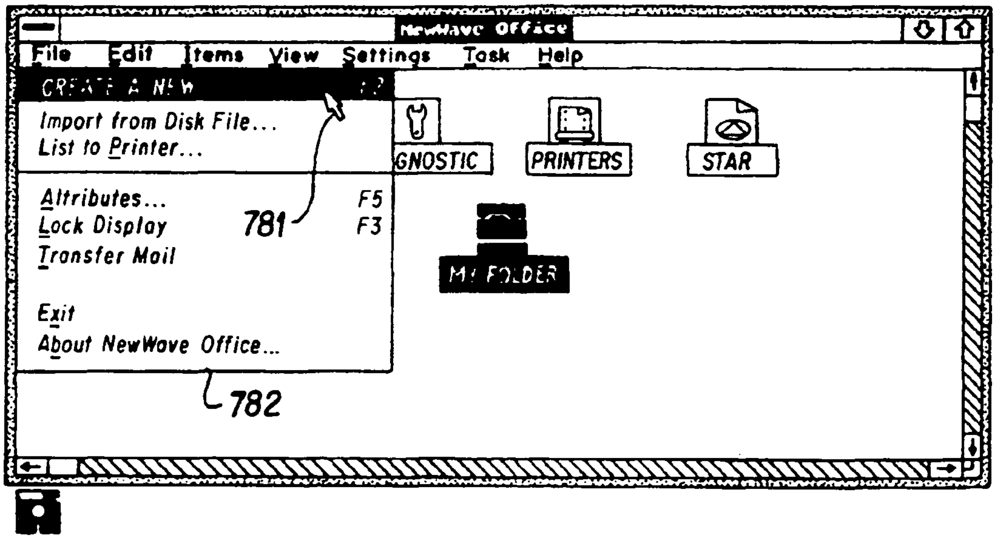

NewWave Office
File
Edit
Items
View
Settings
Task
Help
CREATE A NEW
Import from Disk File...
List to Printer...
Attributes... F5
Lock Display F3
Transfer Mail
Exit
About NewWave Office...
GNOSTIC
PRINTERS
STAR
MY FOLDER
781
782

FIG. 14

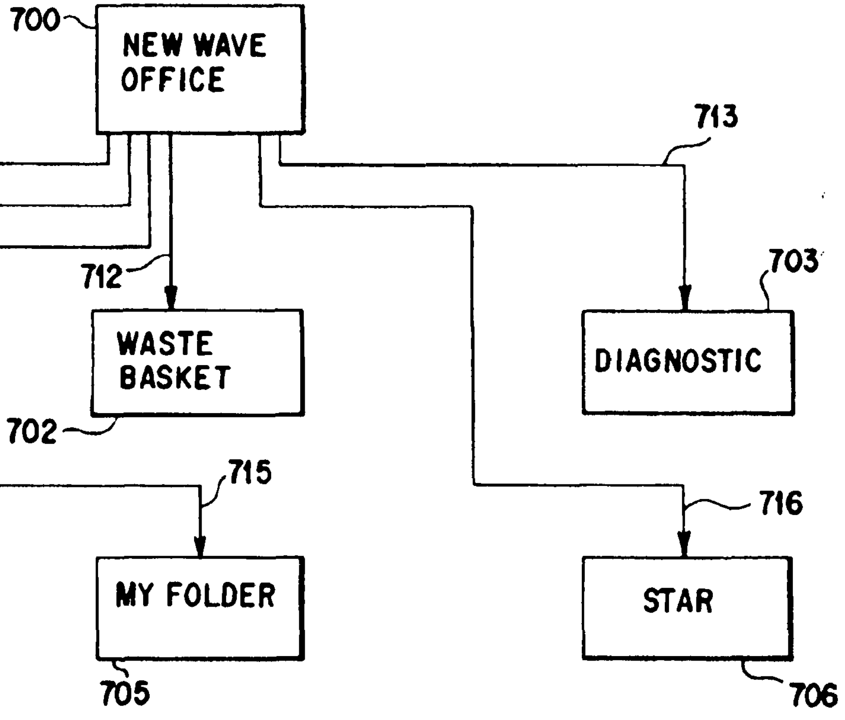
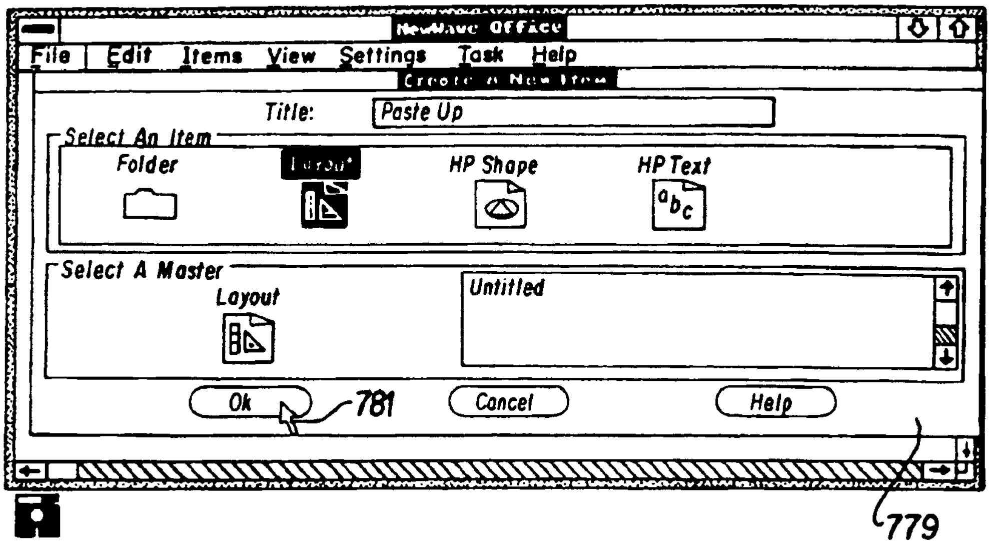

700
NEW WAVE OFFICE
711
713
701
712
703
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
702
715
714
716
PRINTERS
MY FOLDER
STAR
704
705
706

FIG. 13



Title: Paste Up

FIG. 15

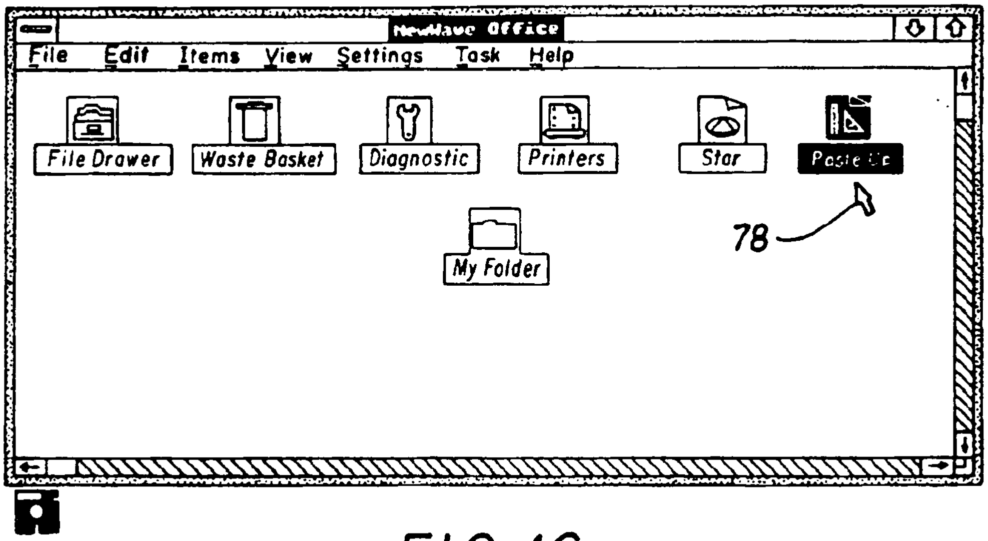


FIG. 16

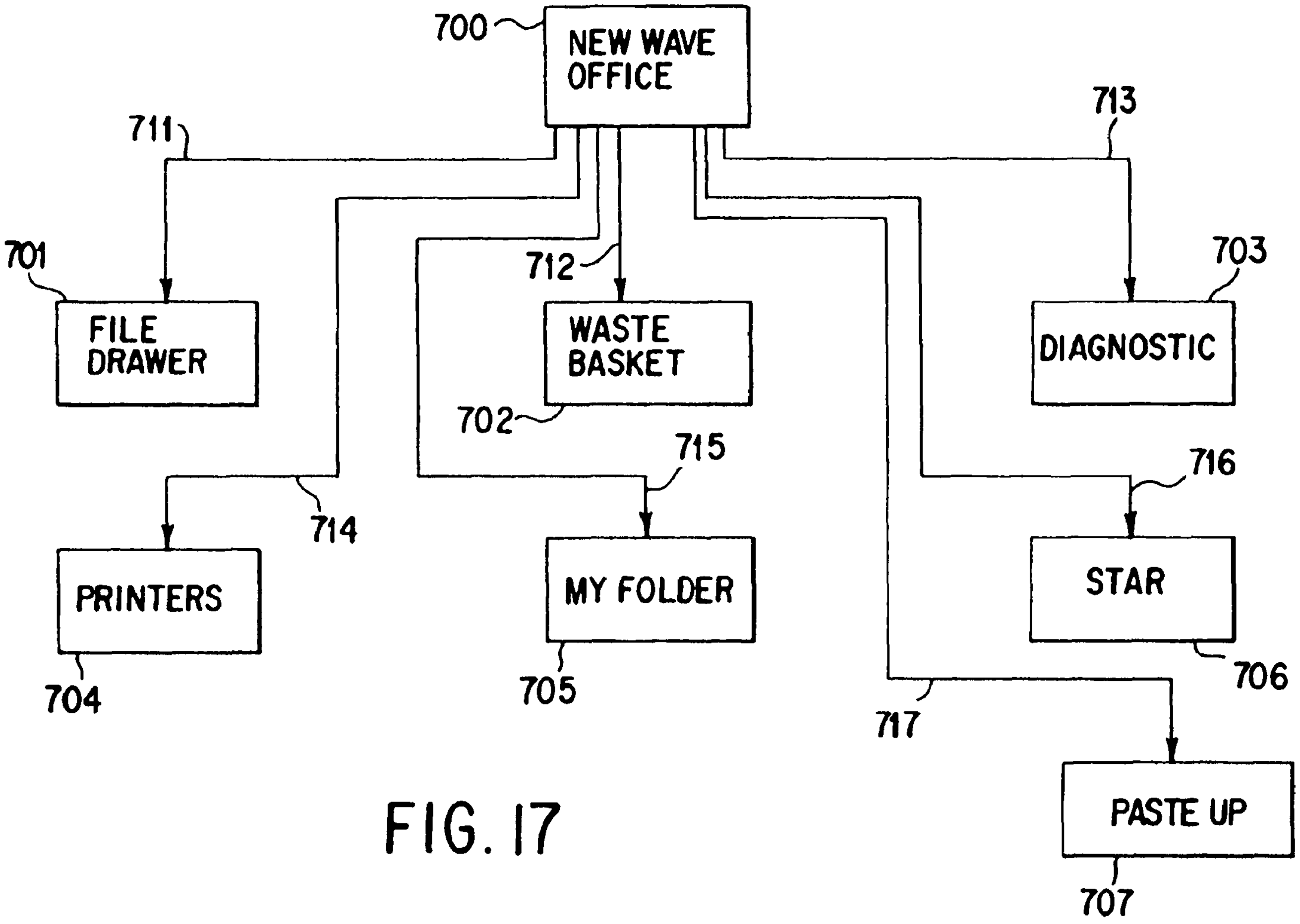

700
NEW WAVE OFFICE
711
713
701
712
703
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
702
715
716
714
PRINTERS
MY FOLDER
STAR
704
705
706
717
PASTE UP
707

FIG. 17

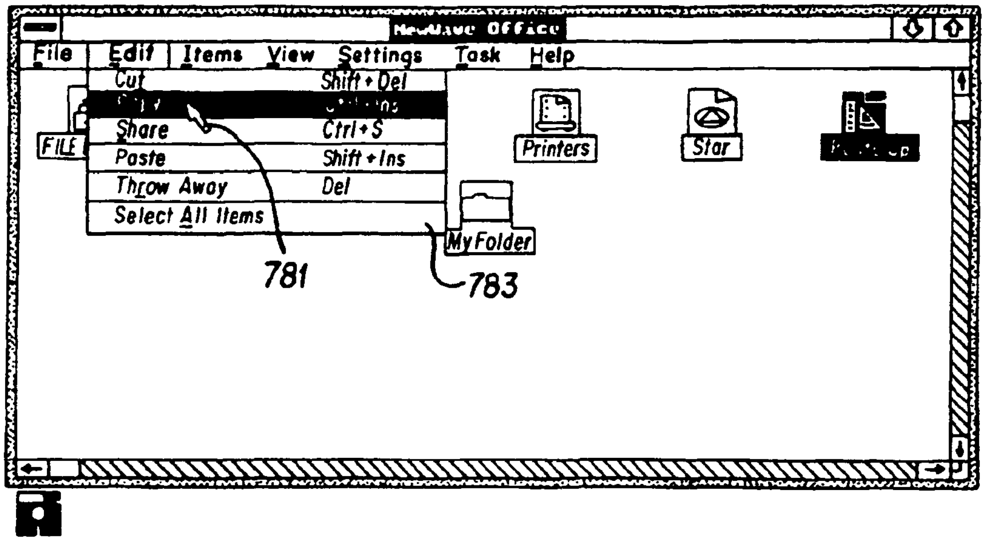

NewWave Office
File
Edit
Items
View
Settings
Task
Help
Cut Shift+Del
Share Ctrl+S
Paste Shift+Ins
Throw Away Del
Select All Items
FILE
Printers
Star
MyFolder
781
783

FIG. 18

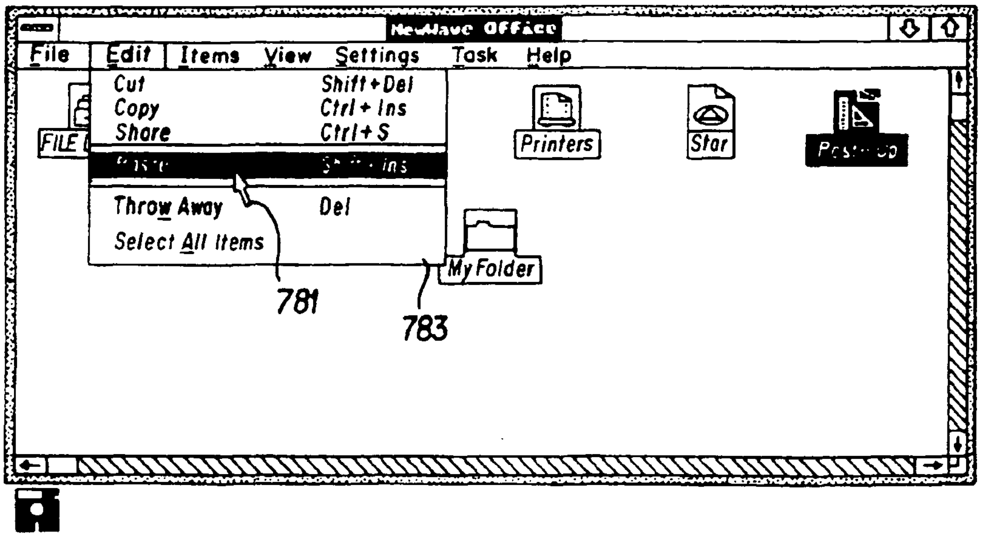

NewWave Office
File
Edit
Items
View
Settings
Task
Help
Cut Shift+Del
Copy Ctrl+Ins
Share Ctrl+S
Throw Away Del
Select All Items
FILE
Printers
Star
MyFolder
781
783

FIG. 19

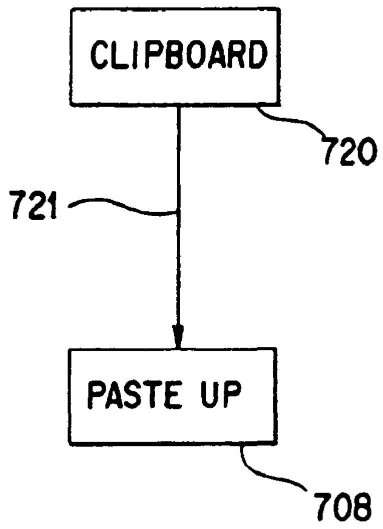


FIG. 18A

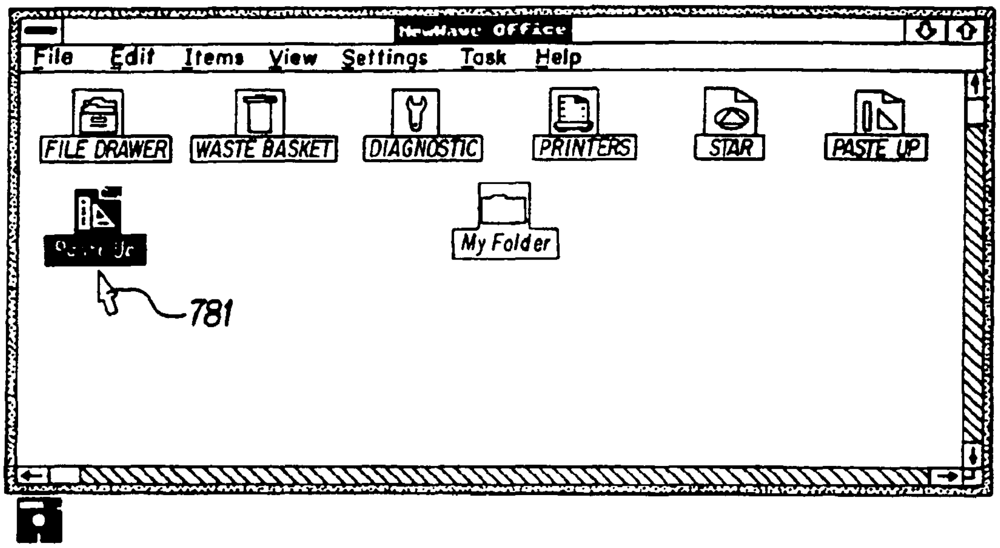

NewWave Office
File
Edit
Items
View
Settings
Task
Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
My Folder
781

FIG. 20

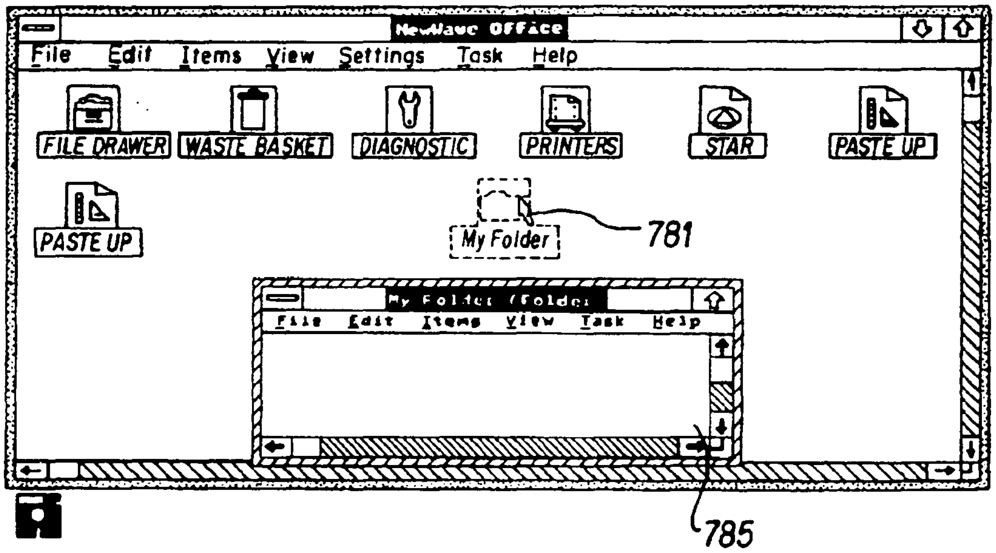

NewWave Office
File
Edit
Items
View
Settings
Task
Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
PASTE UP
My Folder
781
My Folder (Folder)
File
Edit
Items
View
Task
Help
785

FIG. 22

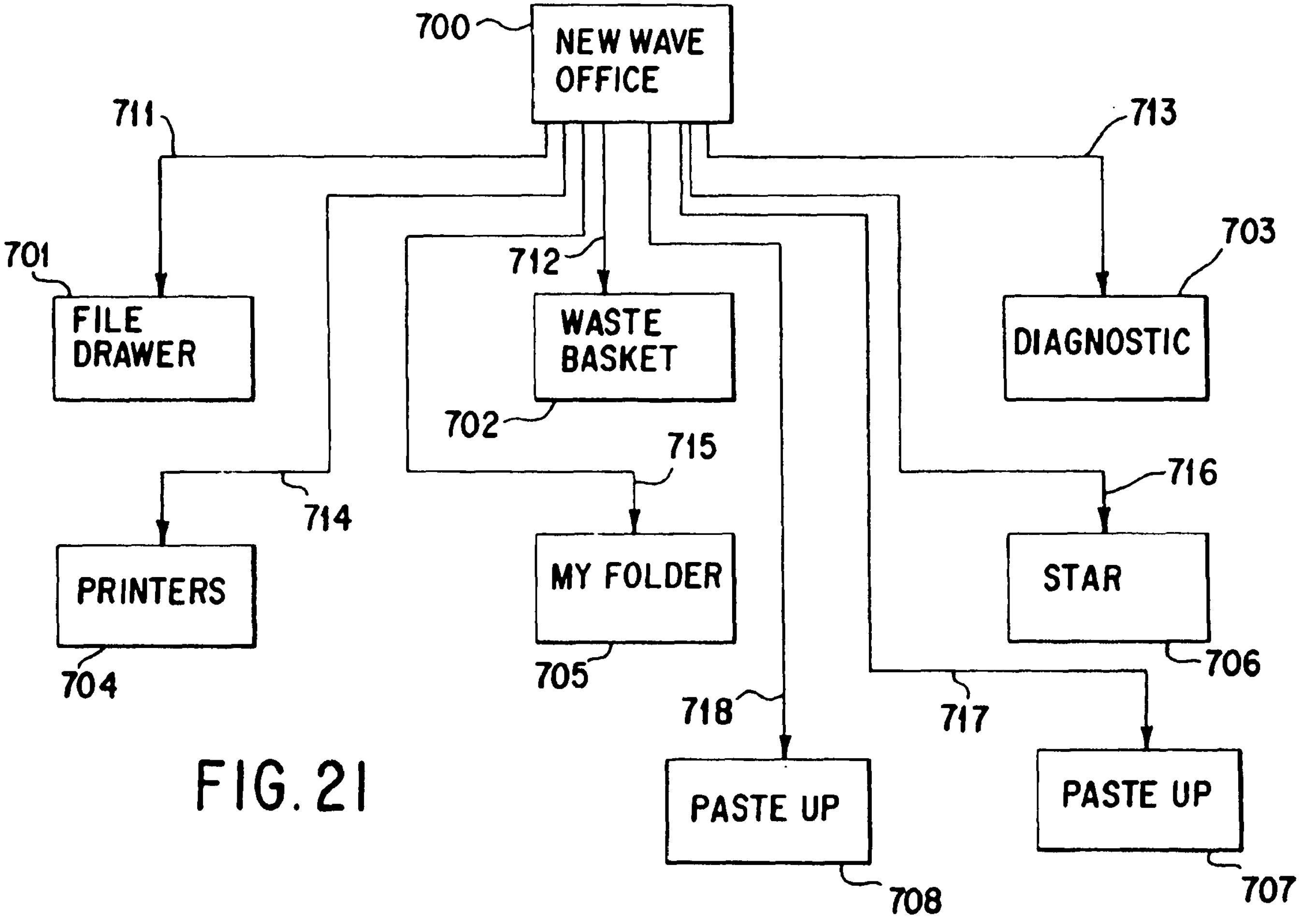

700
NEW WAVE OFFICE
711
713
701
712
703
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
702
715
716
714
PRINTERS
MY FOLDER
STAR
704
705
706
718
717
PASTE UP
PASTE UP
708
707

FIG. 21

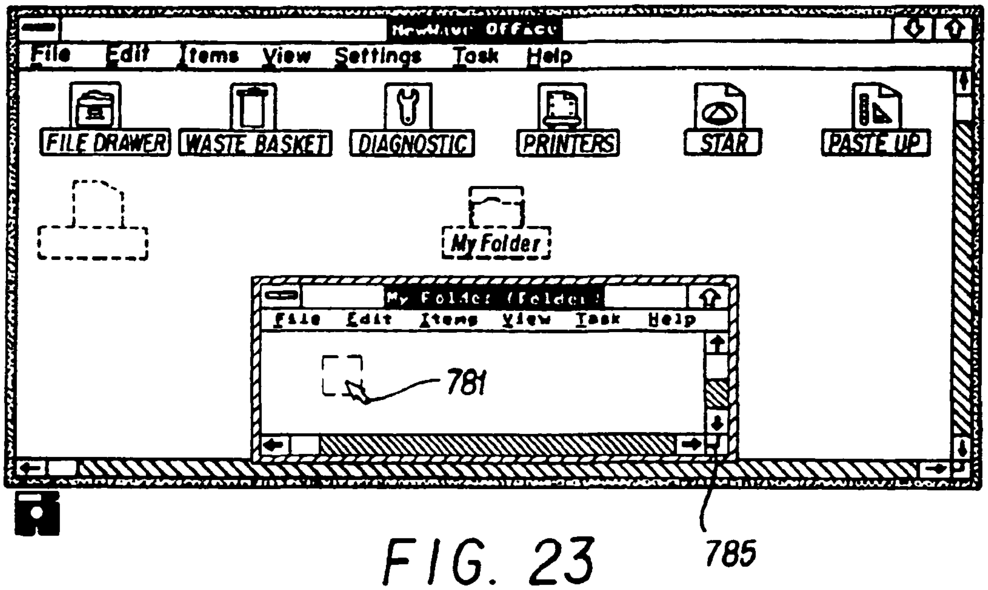

NewWave Office
File
Edit
Items
View
Settings
Task
Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
My Folder
My Folder (Folder)
File
Edit
Items
View
Task
Help
781
785

FIG. 23

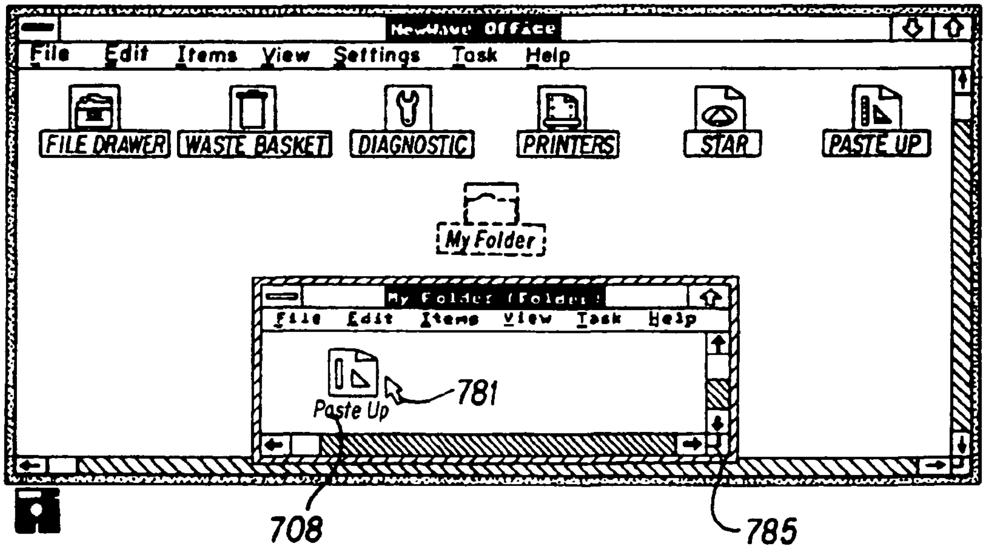

NewWave Office
File
Edit
Items
View
Settings
Task
Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
My Folder
My Folder (Folder)
File
Edit
Items
View
Task
Help
Paste Up
781
708
785

FIG. 24

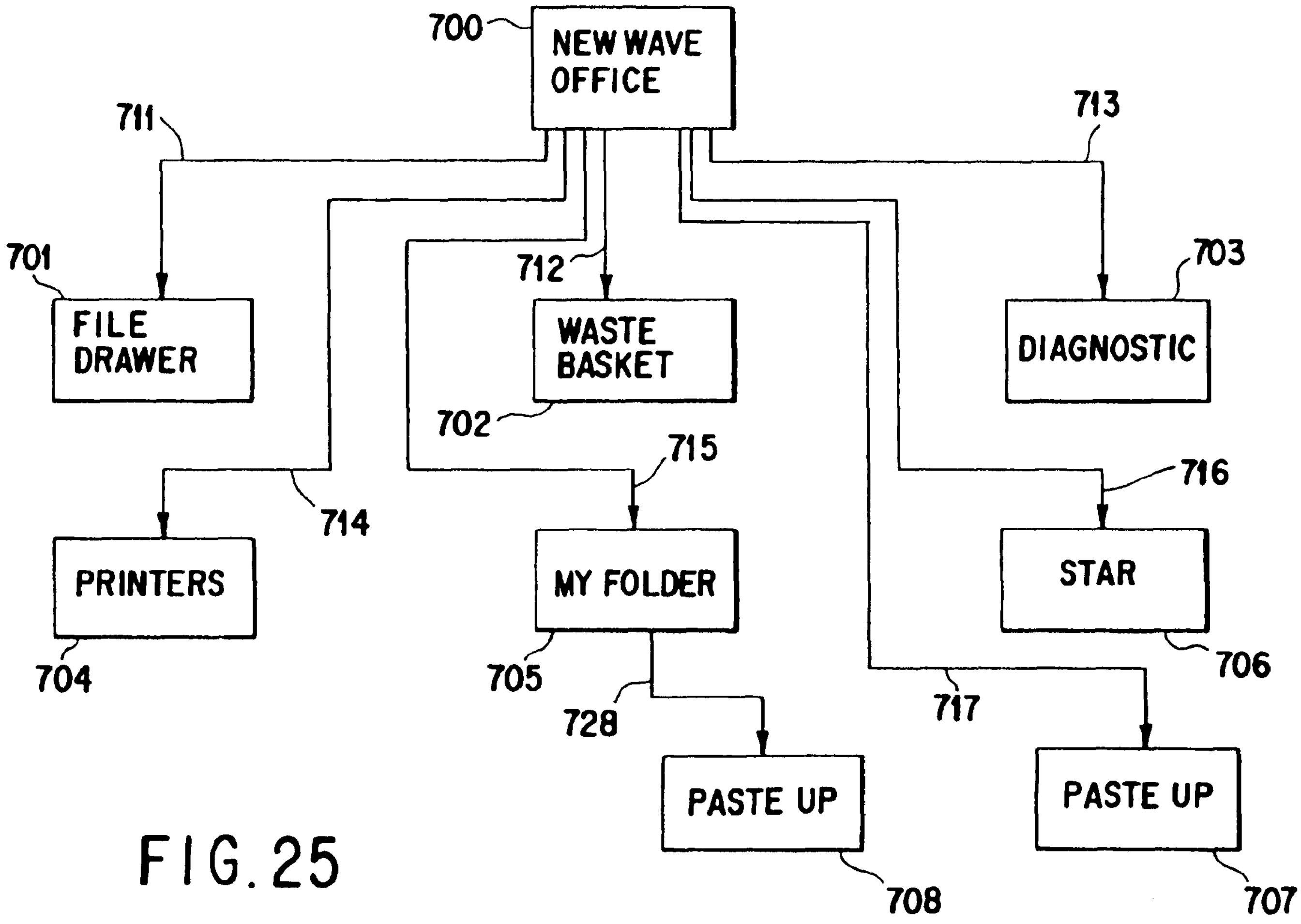

700
NEW WAVE OFFICE
711
713
701
FILE DRAWER
712
WASTE BASKET
703
DIAGNOSTIC
702
715
716
714
PRINTERS
MY FOLDER
STAR
704
705
706
728
717
PASTE UP
PASTE UP
708
707

FIG.25

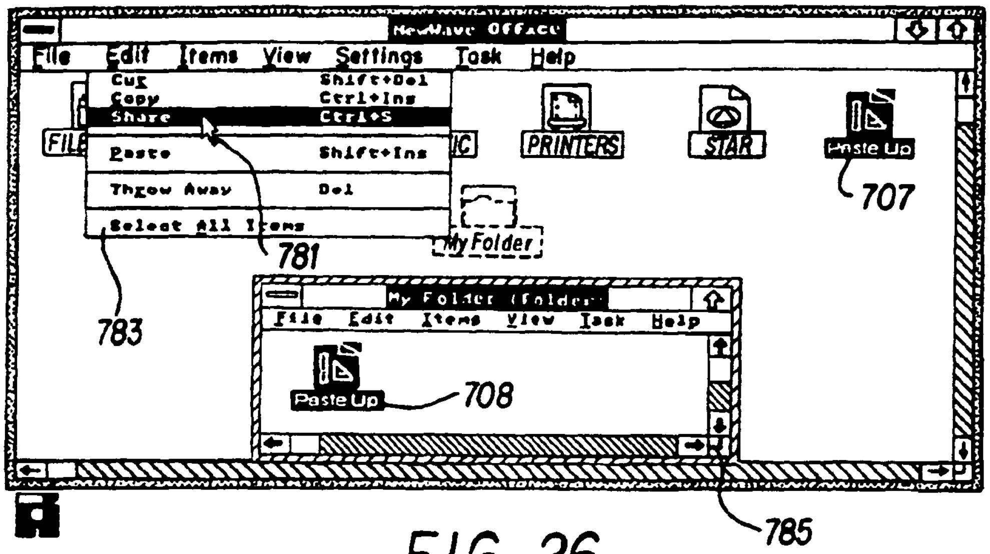

NewWave Office
File
Edit
Items
View
Settings
Task
Help
Cut Shift+Del
Copy Ctrl+Ins
Share Ctrl+S
Paste Shift+Ins
Throw Away Del
Select All Items
FILE
PRINTERS
STAR
Paste Up
707
781
My Folder
783
My Folder (Folder)
File
Edit
Items
View
Task
Help
Paste Up
708
785

FIG. 26

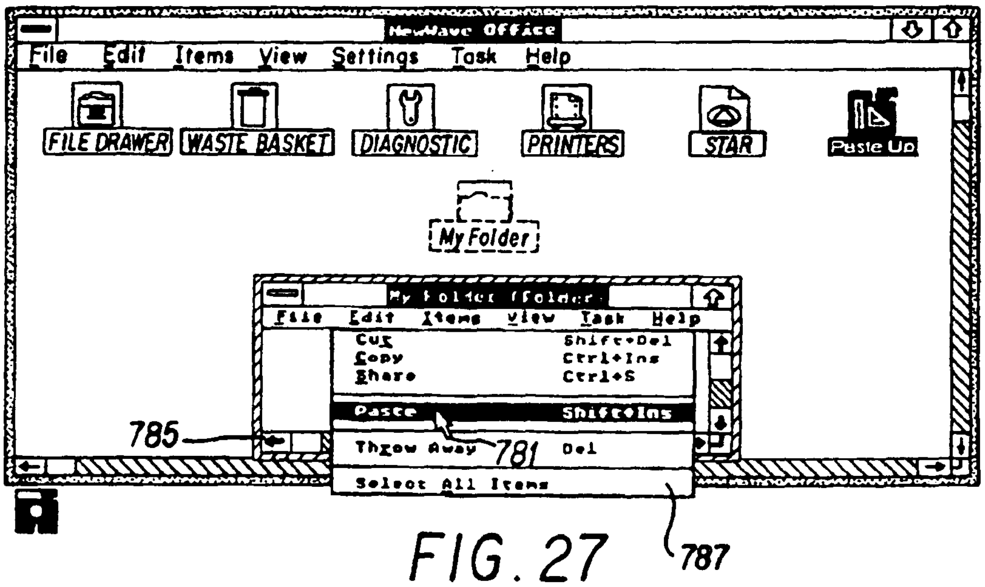

NewWave Office
File
Edit
Items
View
Settings
Task
Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
Paste Up
My Folder
My Folder (Folder)
File
Edit
Items
View
Task
Help
Cut Shift+Del
Copy Ctrl+Ins
Share Ctrl+S
Paste Shift+Ins
Throw Away Del
Select All Items
785
781
787

FIG. 27

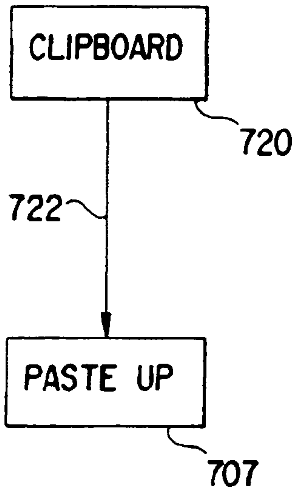


FIG.26A

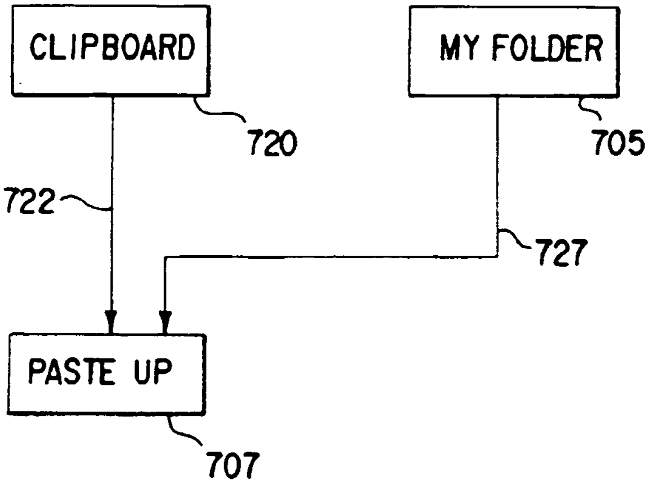


FIG.28A

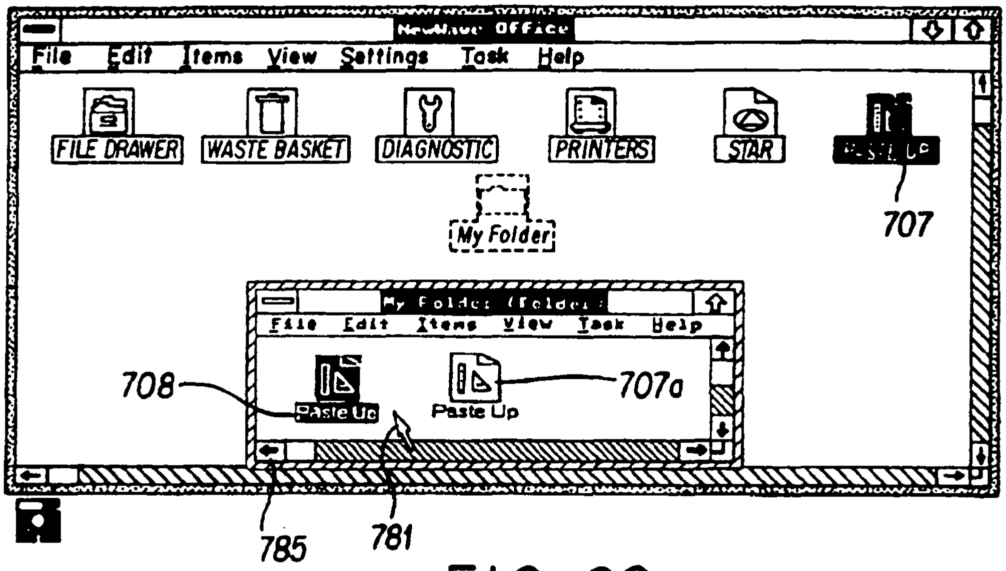

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
707
My Folder
My Folder (Folder)
File Edit Items View Task Help
708
Paste Up
Paste Up
707a
785
781

FIG. 28

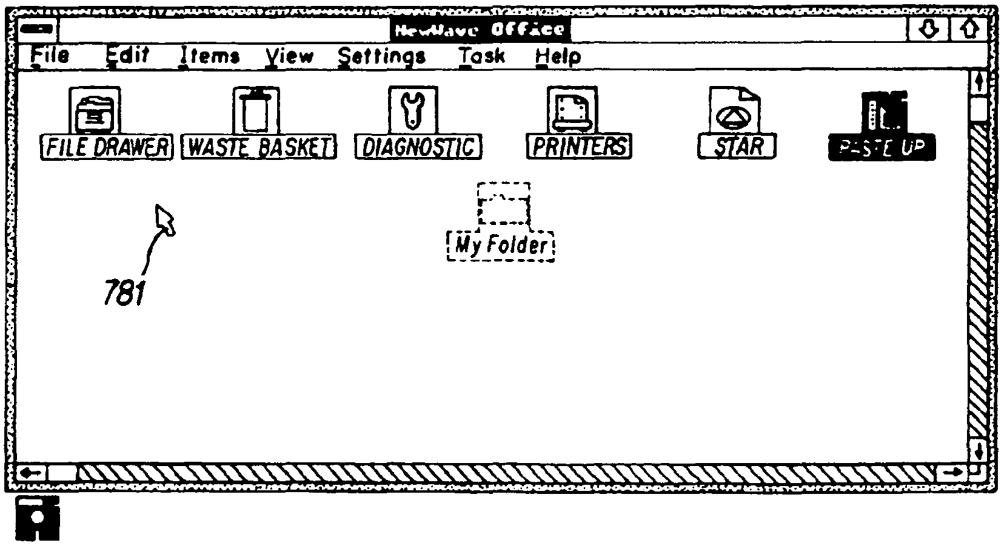

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
My Folder
781

FIG. 30

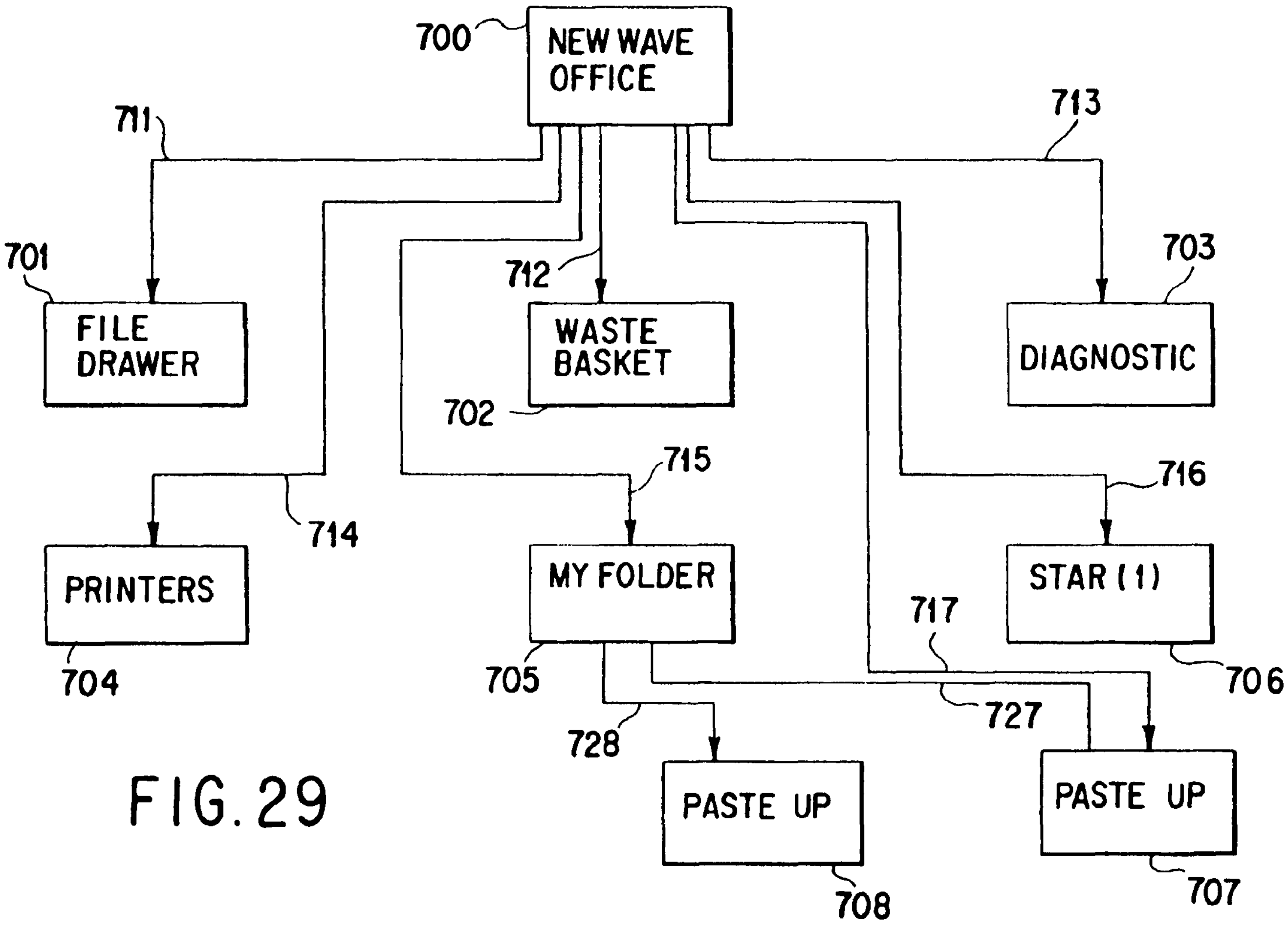

700
NEW WAVE OFFICE
711
713
701
FILE DRAWER
712
WASTE BASKET
703
DIAGNOSTIC
702
715
716
714
PRINTERS
MY FOLDER
STAR (1)
717
704
705
706
727
728
PASTE UP
PASTE UP
708
707

FIG.29

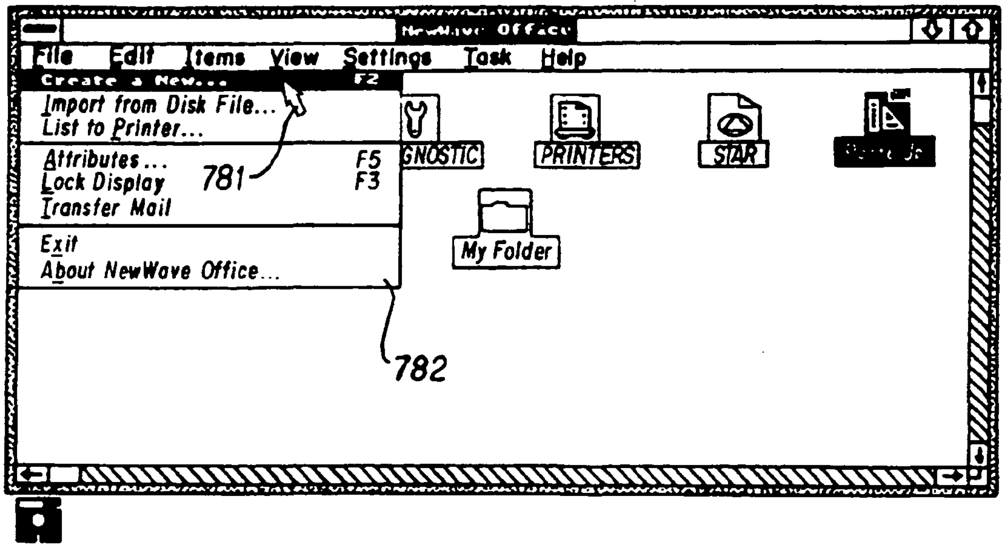


FIG. 31

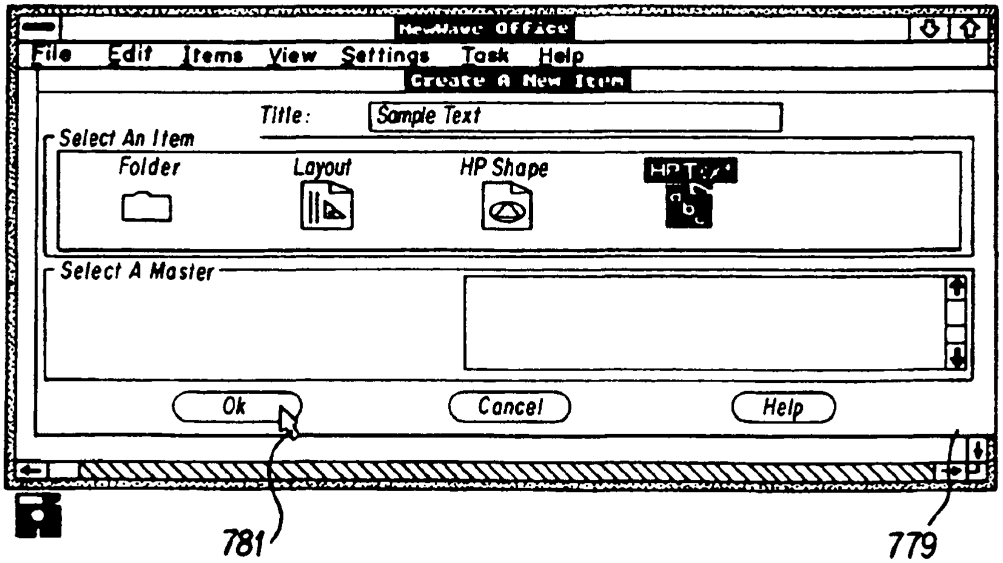

Title: Sample Text

FIG. 32

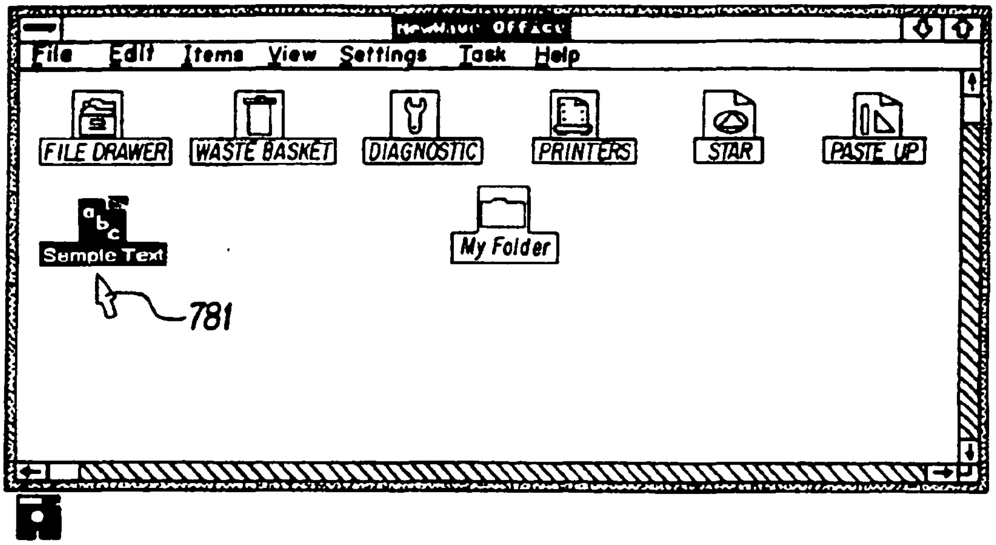

NewWave Office
File
Edit
Items
View
Settings
Task
Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
Sample Text
My Folder
781

FIG. 33

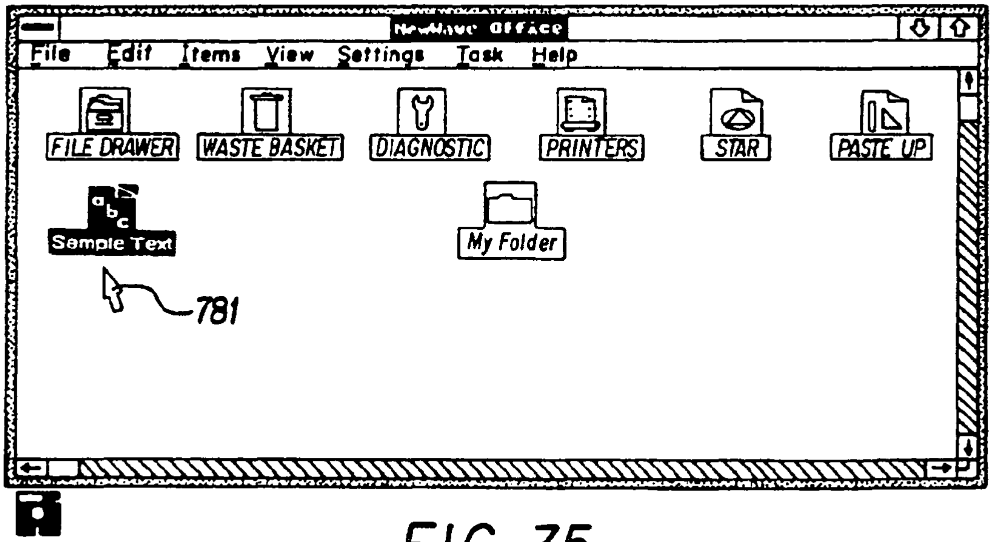

NewWave Office
File
Edit
Items
View
Settings
Task
Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
Sample Text
My Folder
781

FIG. 35

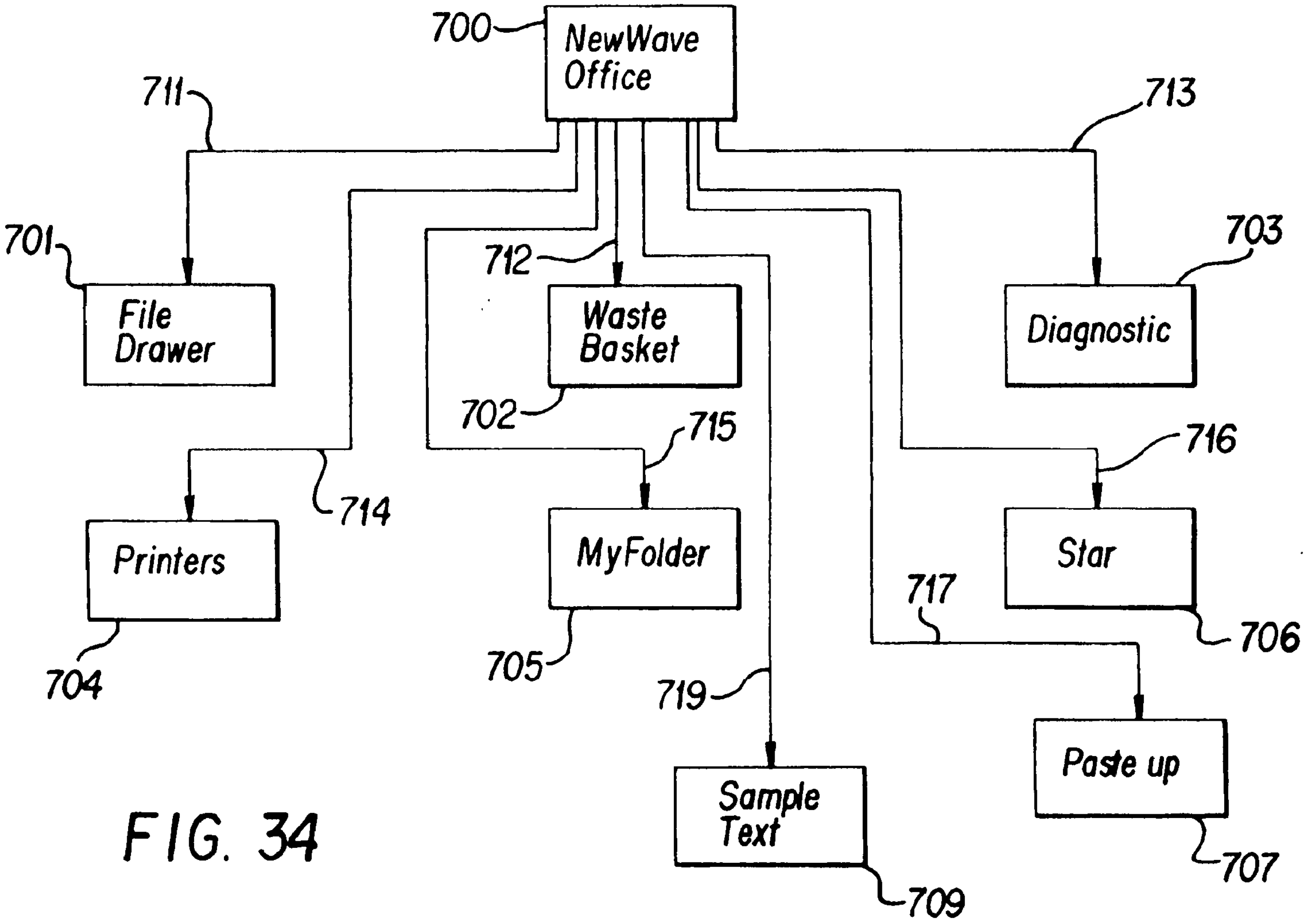
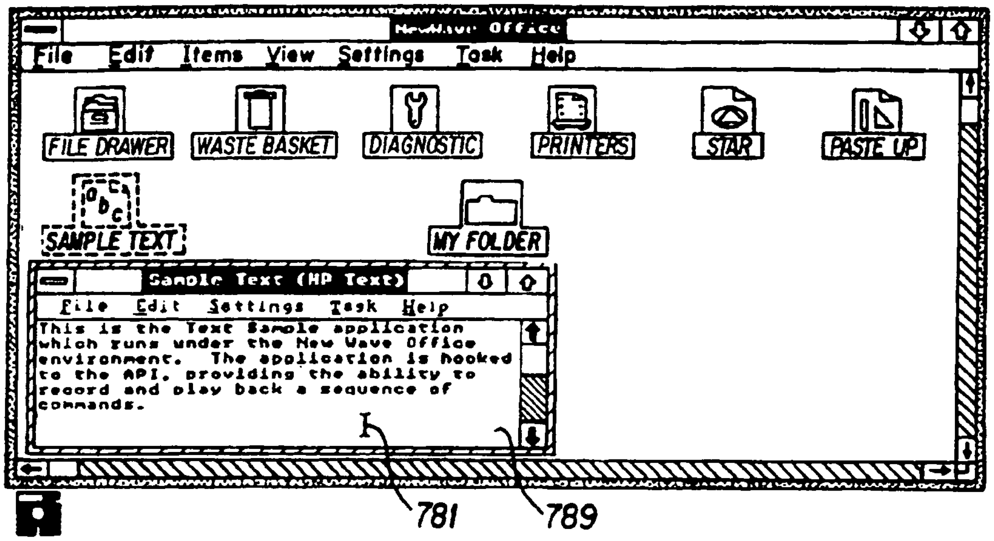
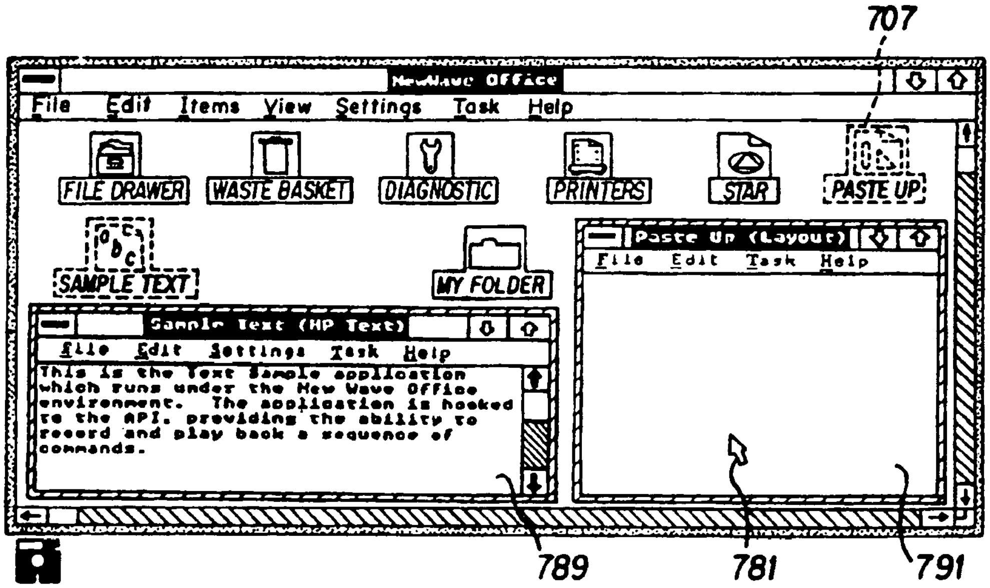

700
NewWave Office
711
713
701
File Drawer
712
Waste Basket
703
Diagnostic
702
715
716
714
Printers
MyFolder
Star
717
704
705
706
719
Paste up
Sample Text
707
709

FIG. 34



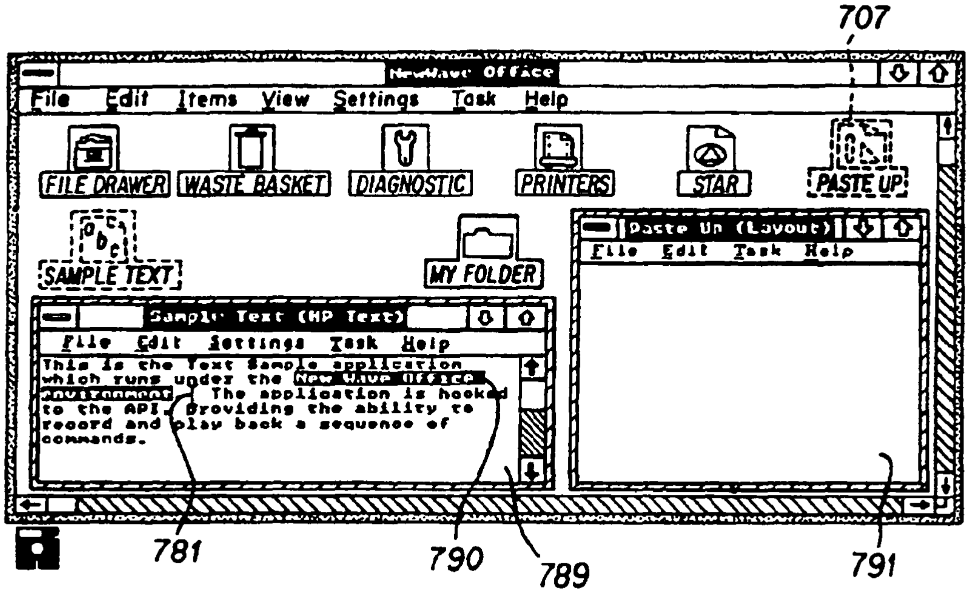
This is the Text Sample application which runs under the New Wave Office environment. The application is hooked to the API, providing the ability to record and play back a sequence of commands.

FIG. 36



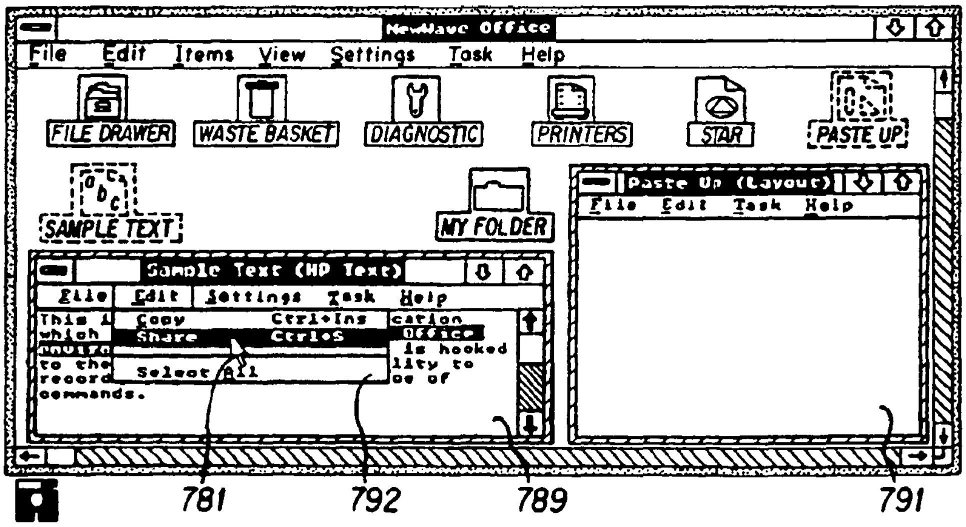
This is the Text Sample application which runs under the New Wave Office environment. The application is hooked to the API, providing the ability to record and play back a sequence of commands.

FIG. 37



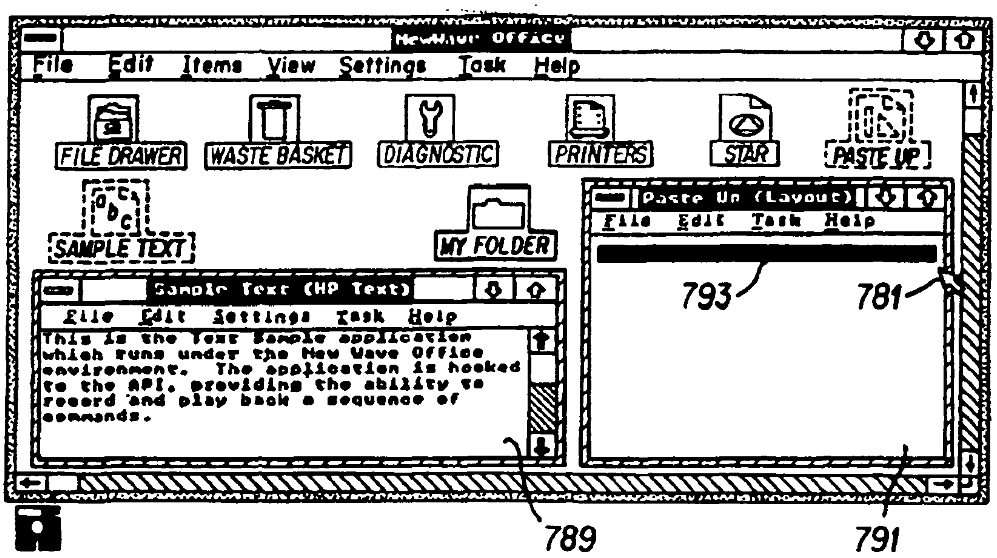
This is the Text Sample application which runs under the New Wave Office environment. The application is hooked to the API, providing the ability to record and play back a sequence of commands.

FIG. 38



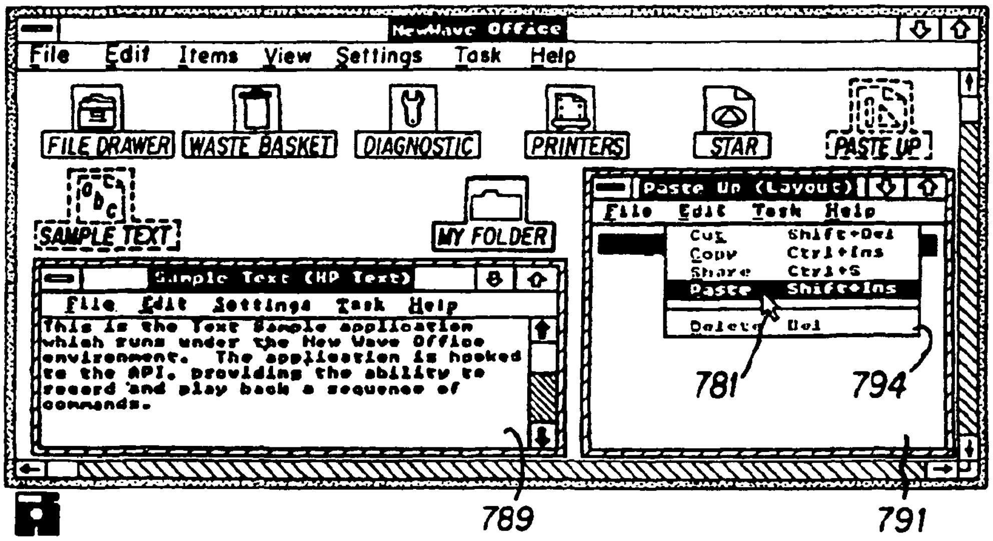
This i cation
which Office
enviro is hooked
to the lity to
record ce of
commands.

FIG. 39



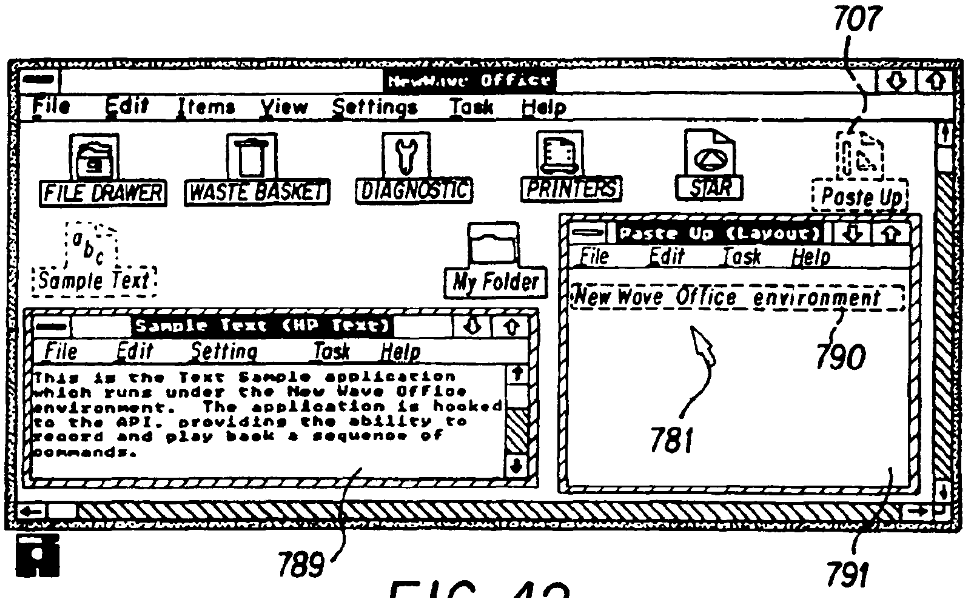
This is the Text Sample application which runs under the New Wave Office environment. The application is hooked to the API, providing the ability to record and play back a sequence of commands.

FIG.40



This is the Text Sample application which runs under the New Wave Office environment. The application is hooked to the API, providing the ability to record and play back a sequence of commands.

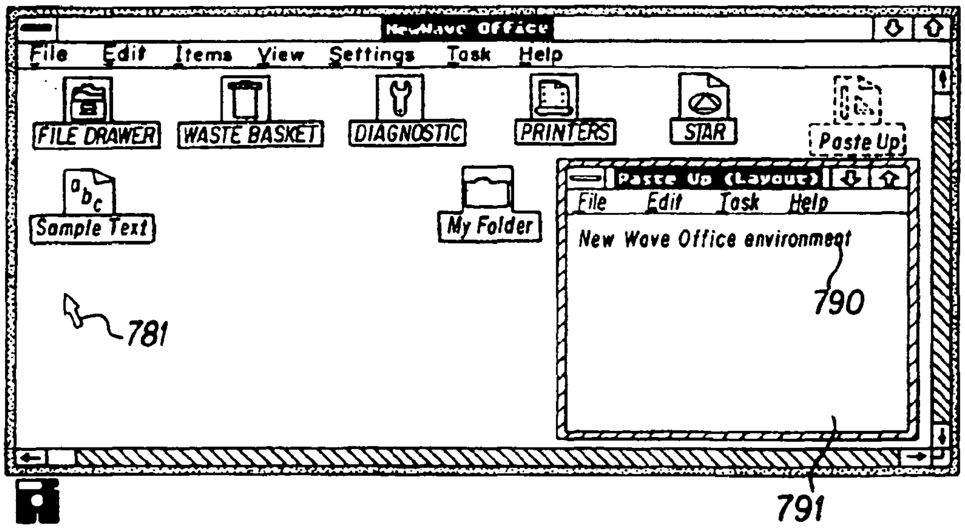
FIG. 41



This is the Text Sample application which runs under the New Wave Office environment. The application is hooked to the API, providing the ability to record and play back a sequence of commands.

New Wave Office environment

FIG. 42



New Wave Office environment

FIG. 44

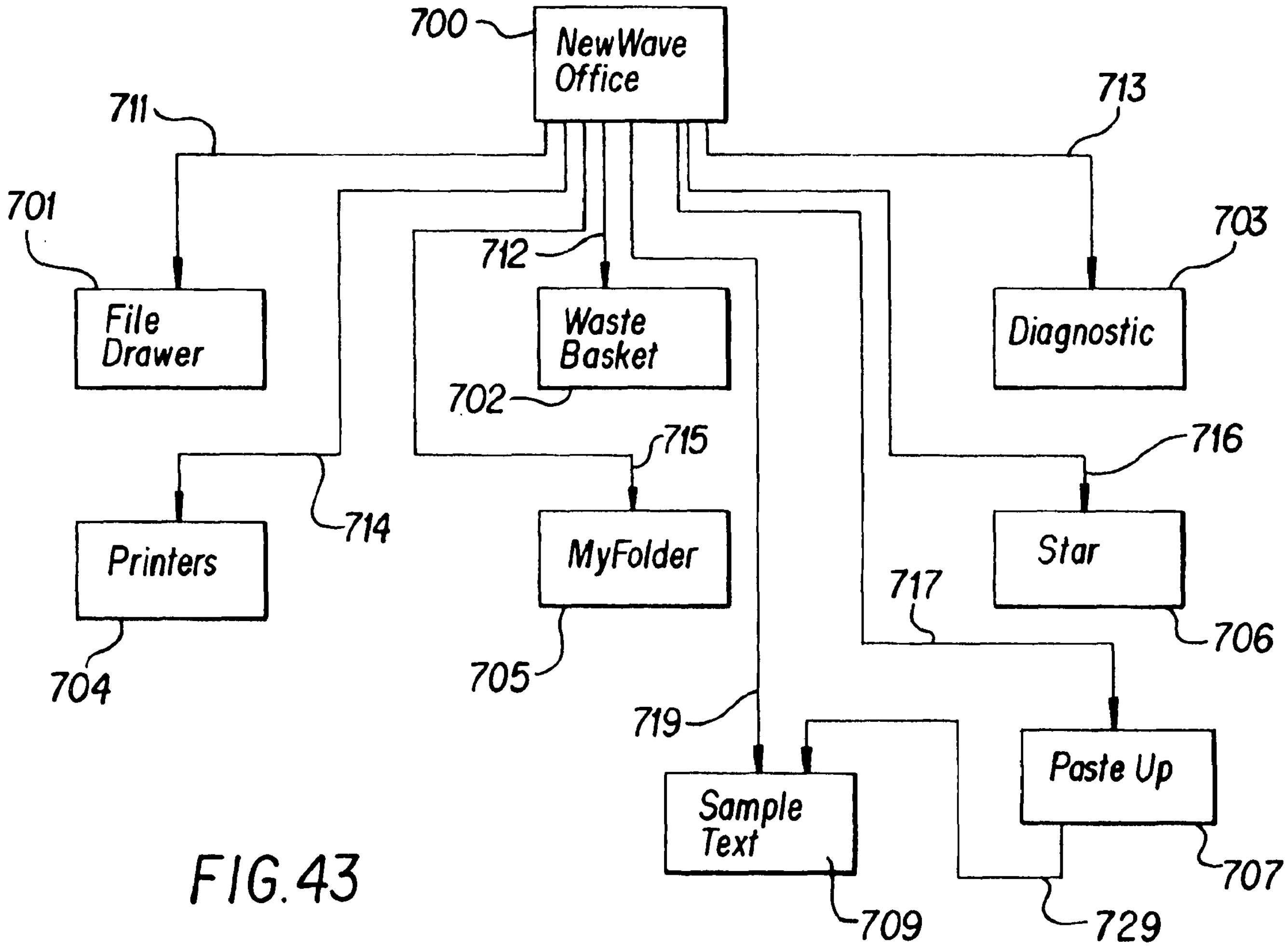

700
NewWave Office
711
701
File Drawer
712
Waste Basket
702
713
703
Diagnostic
715
714
Printers
704
MyFolder
705
716
Star
706
717
719
Sample Text
709
Paste Up
707
729

FIG.43

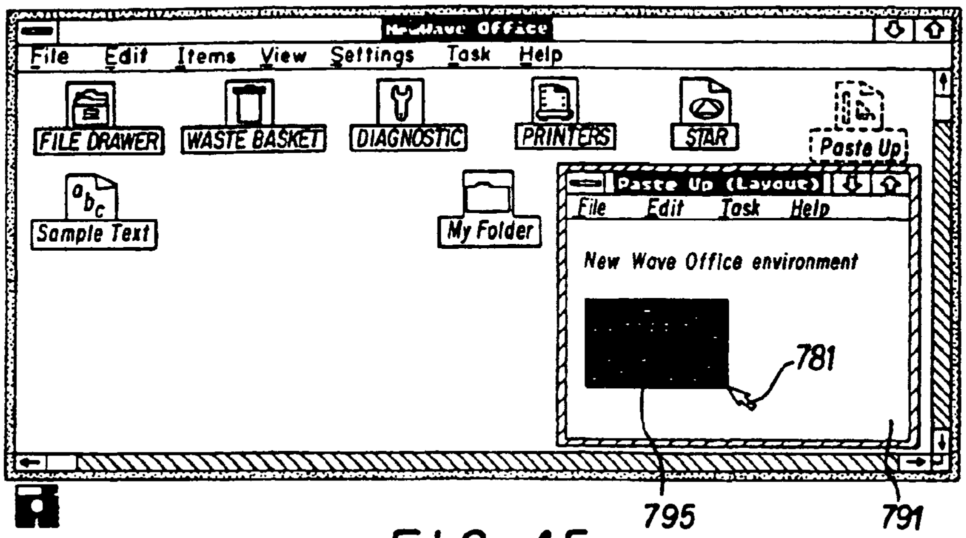

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
781
795
791

FIG. 45

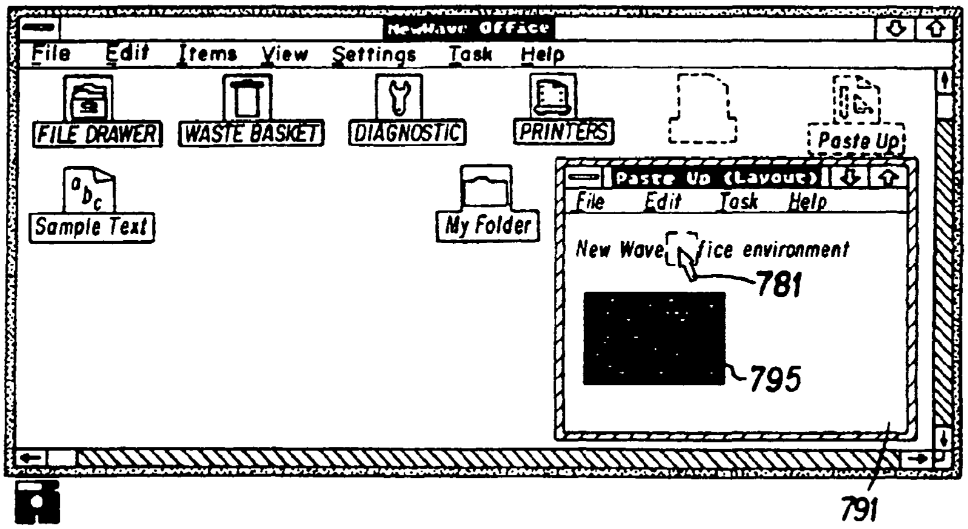

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave ffice environment
781
795
791

FIG. 46

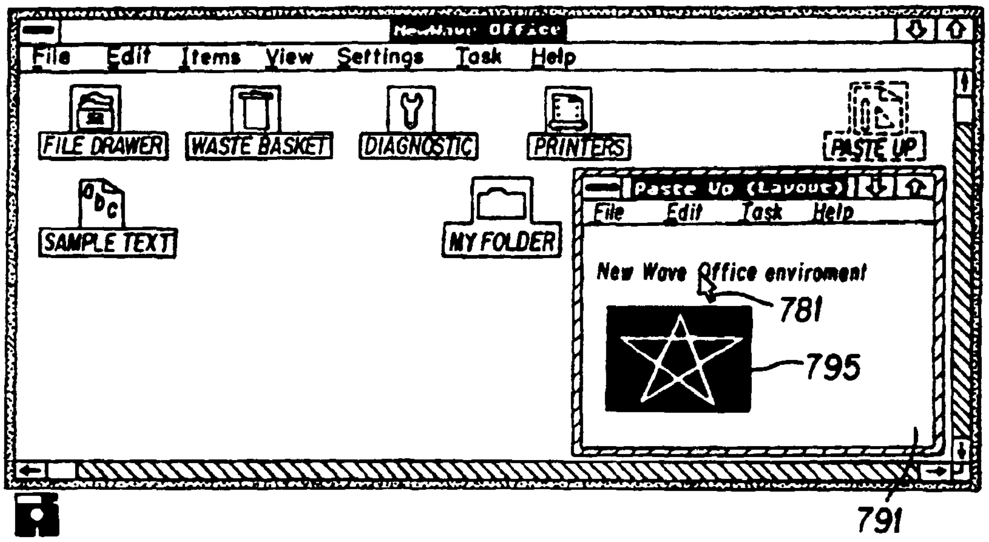

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
PASTE UP
SAMPLE TEXT
MY FOLDER
Paste Up (Layout)
File Edit Task Help
New Wave Office enviroment
781
795
791

FIG. 47

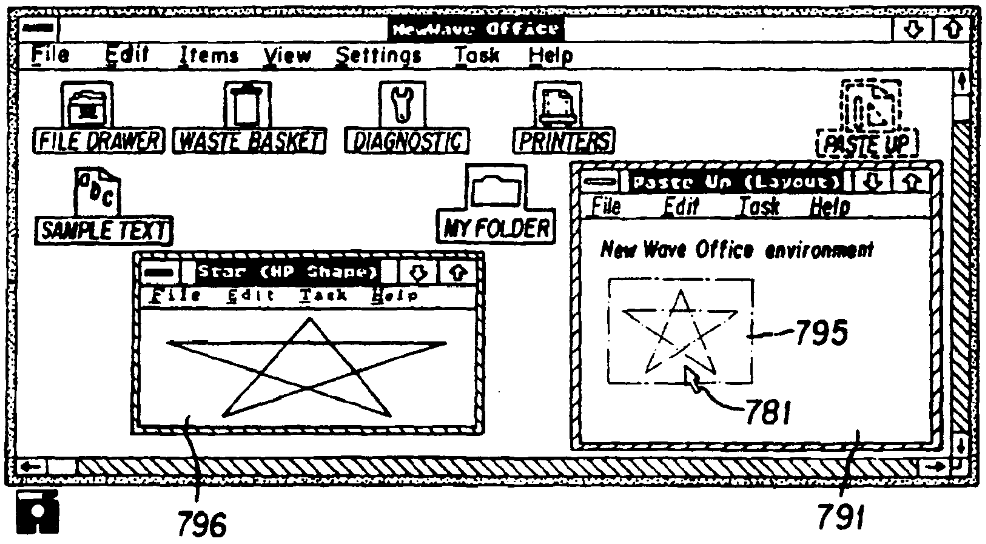

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
PASTE UP
SAMPLE TEXT
MY FOLDER
Star (HP Shape)
File Edit Task Help
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
795
781
796
791

FIG. 49

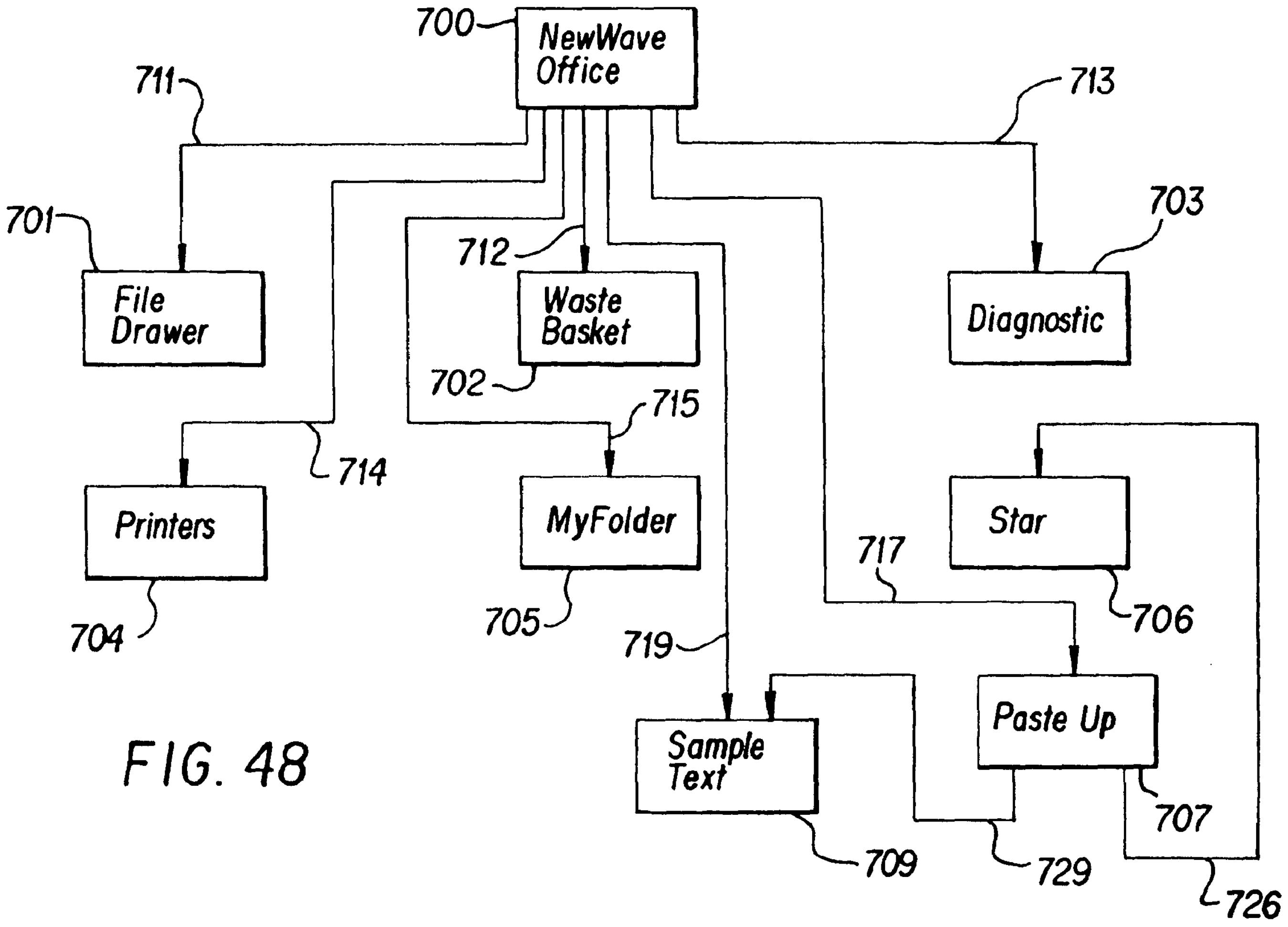

700
NewWave Office
711
713
701
File Drawer
712
Waste Basket
702
703
Diagnostic
715
714
Printers
MyFolder
Star
704
705
717
706
719
Paste Up
Sample Text
707
709
729
726

FIG. 48

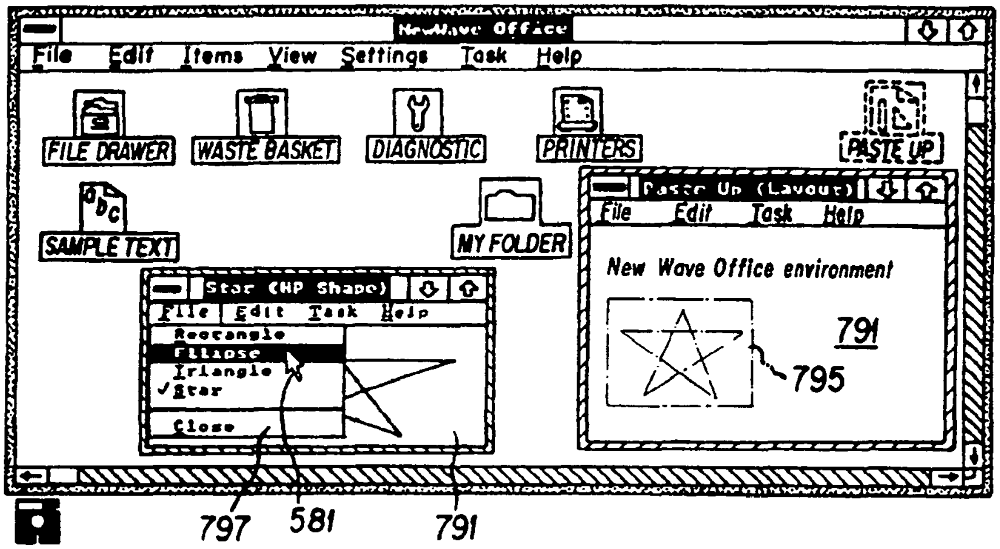

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
PASTE UP
SAMPLE TEXT
MY FOLDER
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
791
795
Star (HP Shape)
File Edit Task Help
Rectangle
Ellipse
Triangle
✓Star
Close
797
581
791

FIG. 50

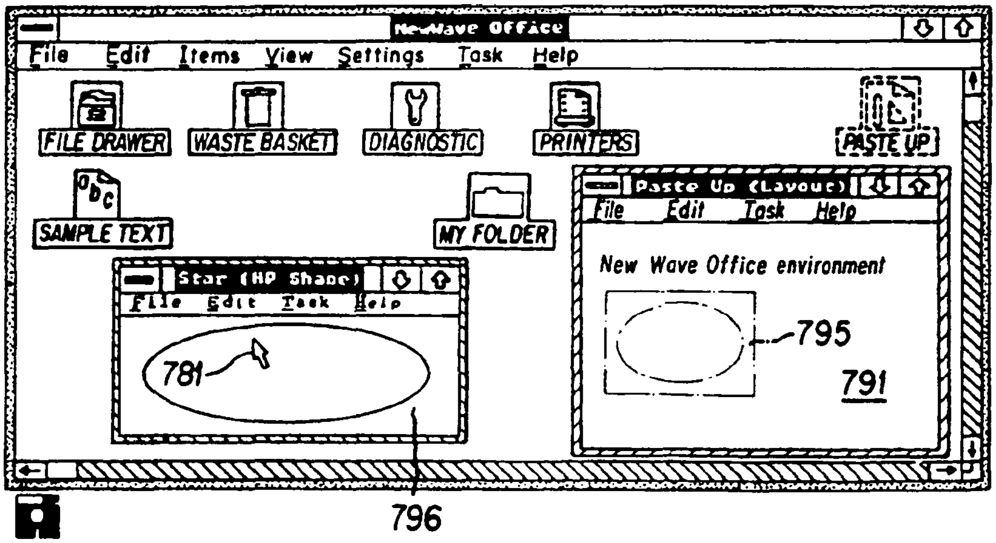
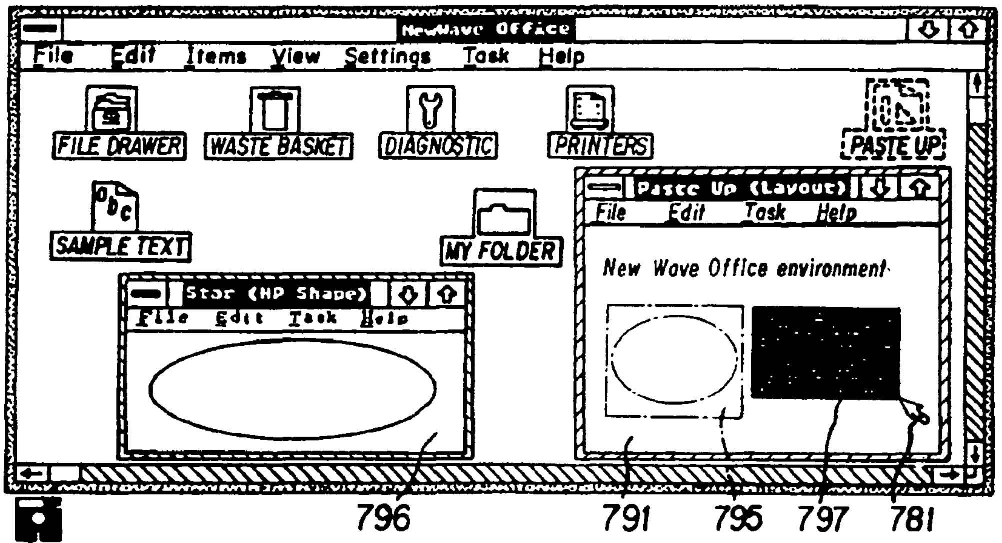

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
PASTE UP
SAMPLE TEXT
MY FOLDER
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
795
791
Star (HP Shape)
File Edit Task Help
781
796

FIG. 51



New Wave Office environment

FIG. 52

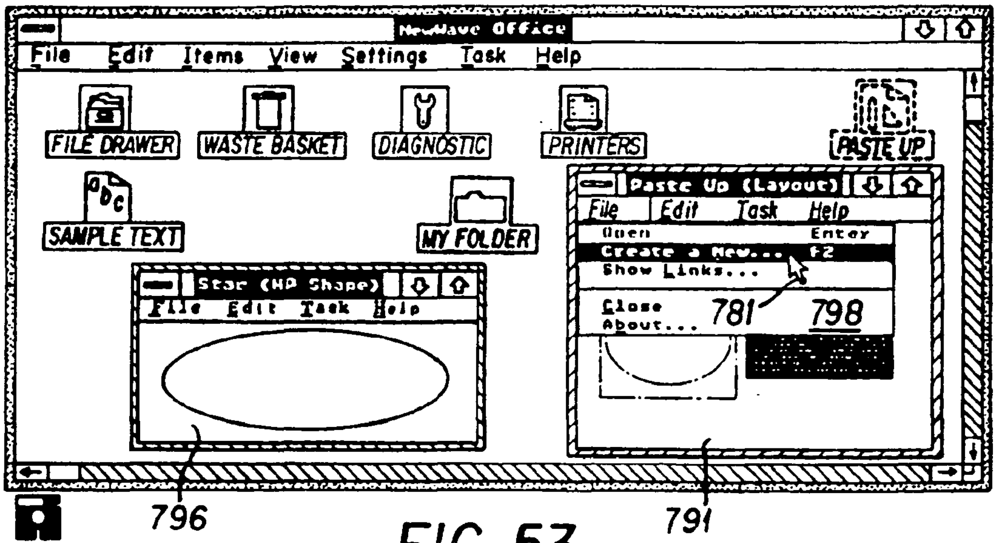


FIG. 53

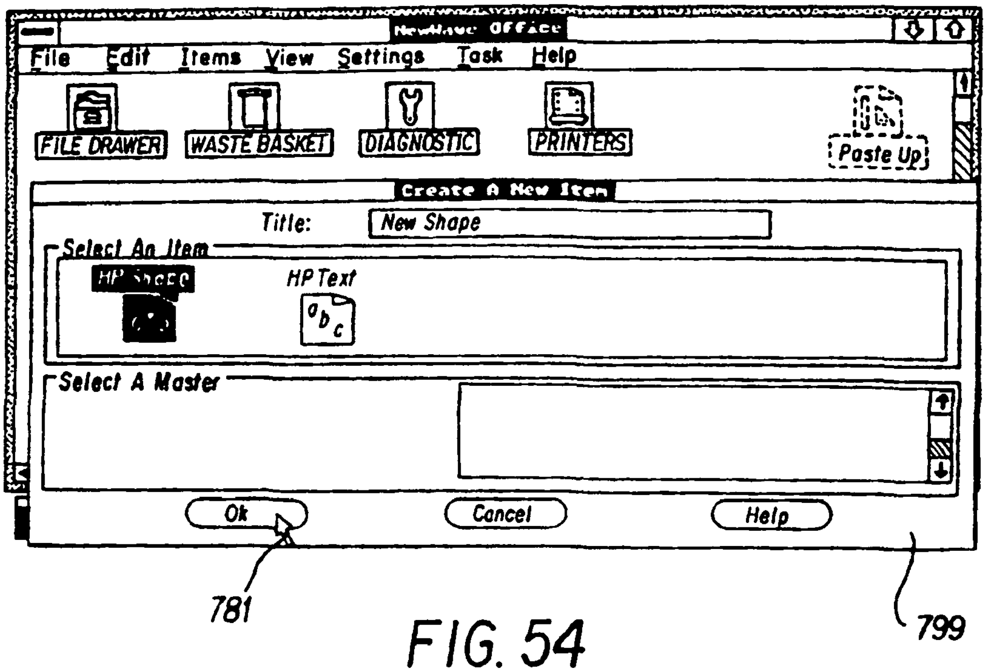

NewWave Office
File
Edit
Items
View
Settings
Task
Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Create A New Item
Title:
New Shape
Select An Item
HP Shape
HP Text
Select A Master
Ok
Cancel
Help
781
799

FIG. 54

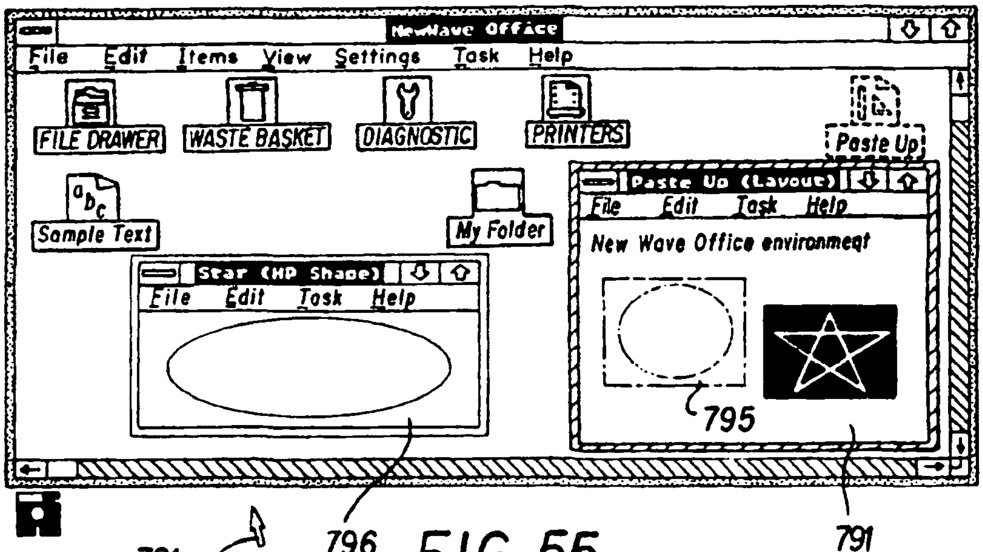

NewWave Office
File
Edit
Items
View
Settings
Task
Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Star (HP Shape)
File
Edit
Task
Help
Paste Up (Layout)
File
Edit
Task
Help
New Wave Office environment
795
781
796
791

FIG. 55

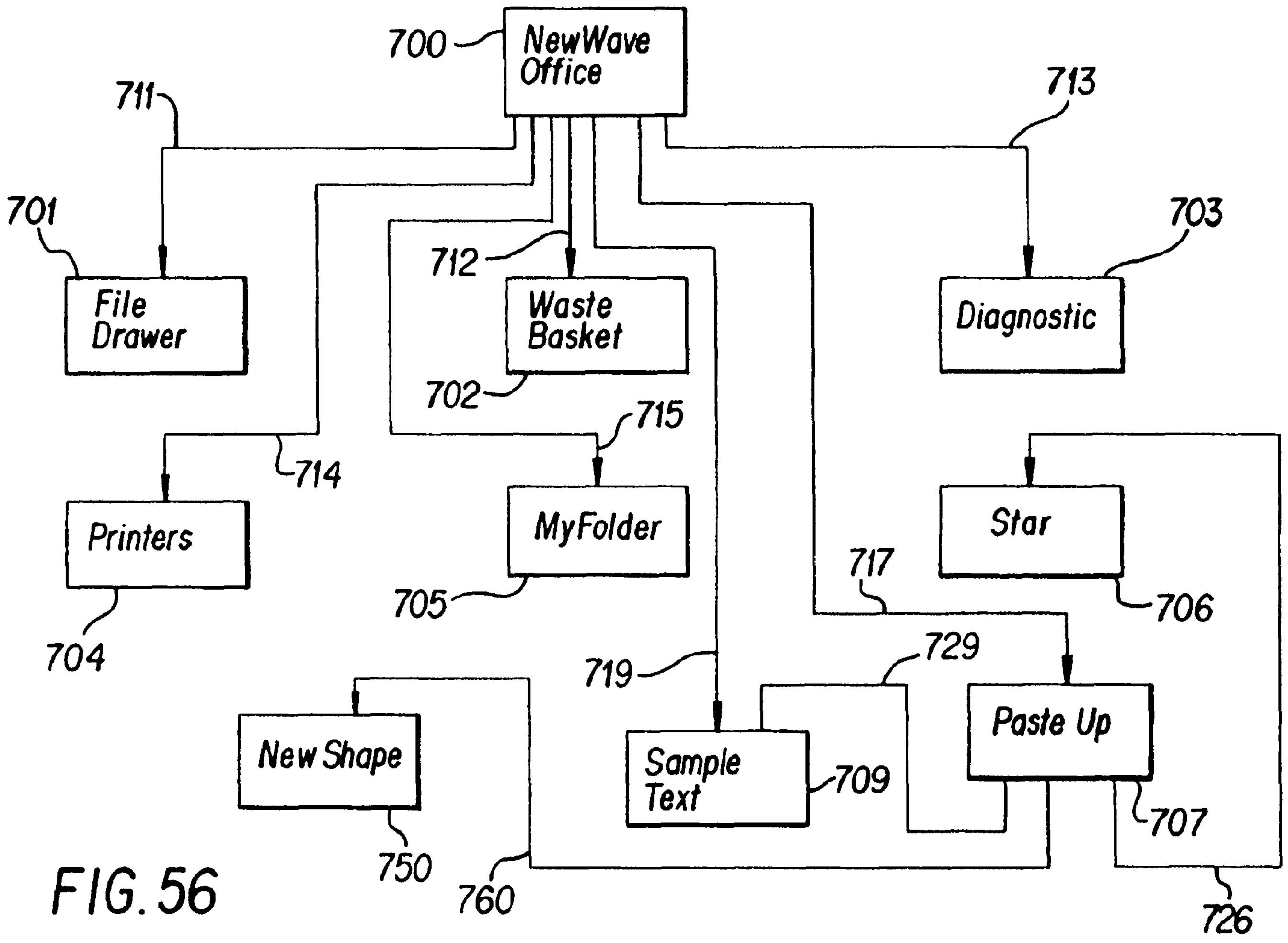

700
NewWave Office
711
713
701
File Drawer
712
Waste Basket
702
703
Diagnostic
715
714
Printers
704
MyFolder
705
Star
706
717
729
719
New Shape
750
760
Sample Text
709
Paste Up
707
726

FIG. 56

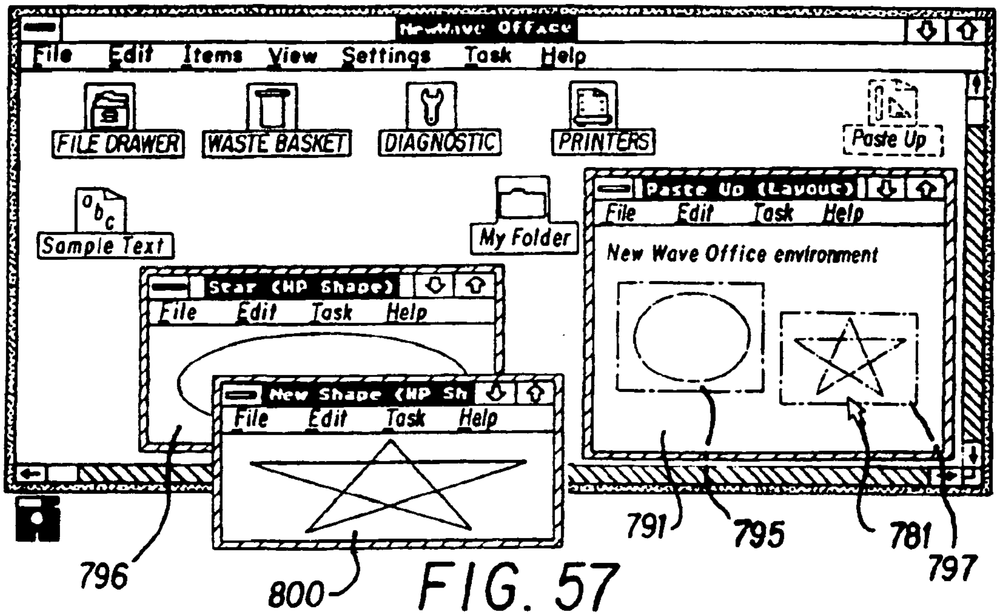

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
Star (HP Shape)
File Edit Task Help
New Shape (HP Sh
File Edit Task Help
791
795
781
797
796
800

FIG. 57

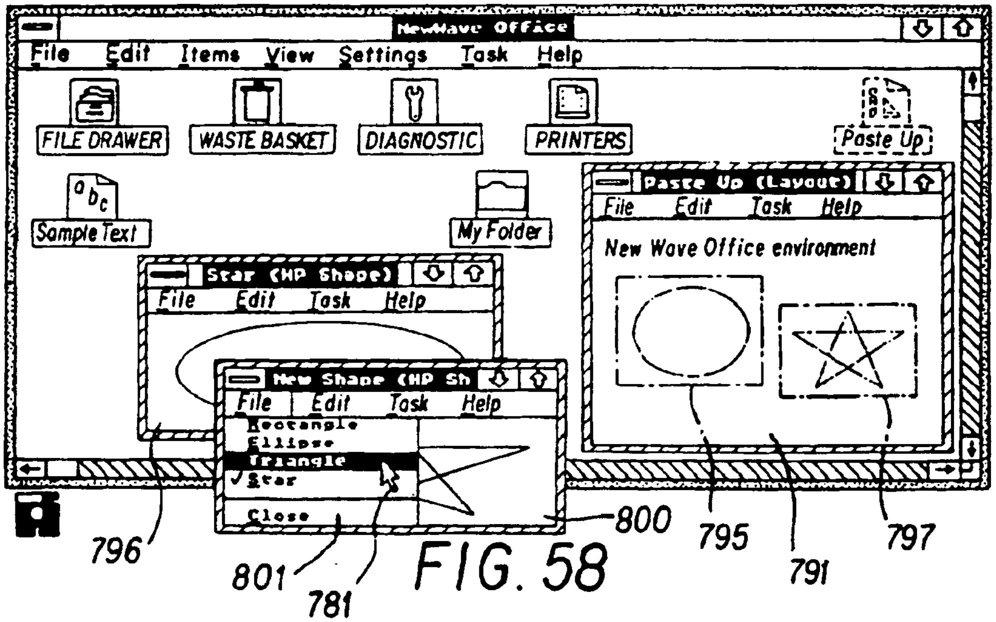

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
Star (HP Shape)
File Edit Task Help
New Shape (HP Sh
File Edit Task Help
Rectangle
Ellipse
Triangle
✓Star
Close
800
795
791
797
796
801
781

FIG. 58

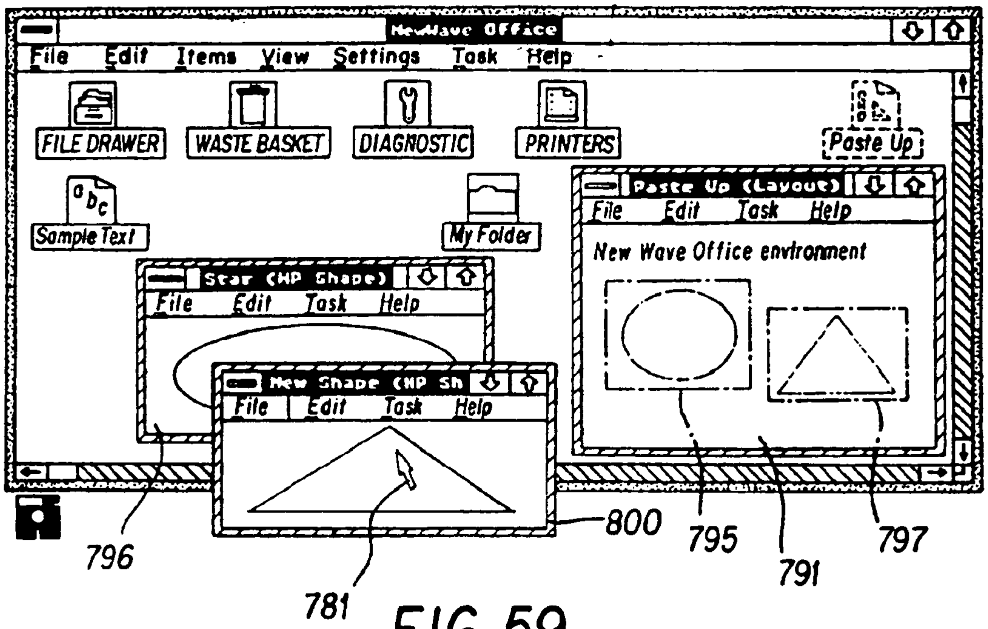

NewWave Office
File
Edit
Items
View
Settings
Task
Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File
Edit
Task
Help
New Wave Office environment
Star (HP Shape)
File
Edit
Task
Help
New Shape (HP Sh
File
Edit
Task
Help
800
796
795
791
797
781

FIG. 59

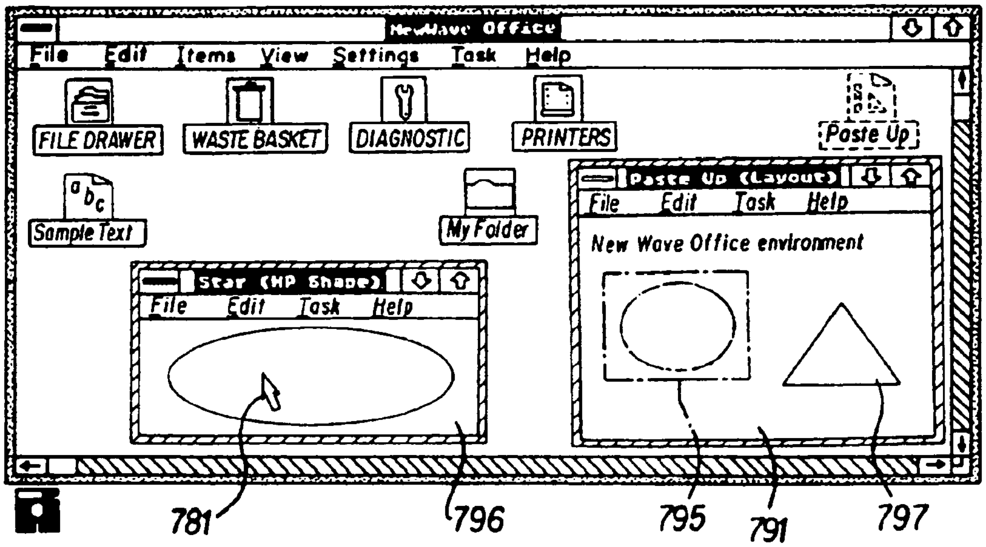

NewWave Office
File
Edit
Items
View
Settings
Task
Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File
Edit
Task
Help
New Wave Office environment
Star (HP Shape)
File
Edit
Task
Help
781
796
795
791
797

FIG. 60

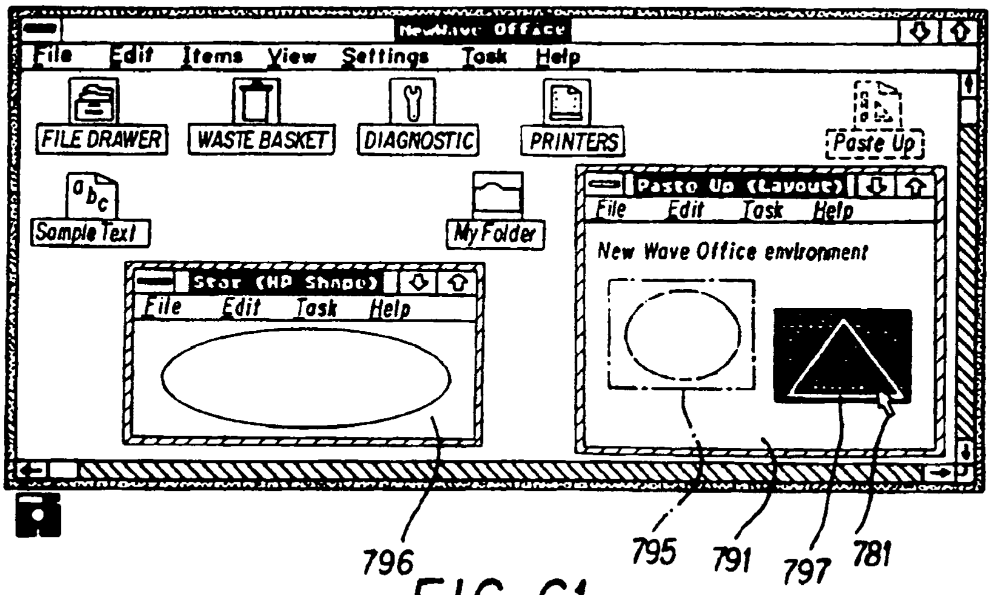

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
Star (HP Shape)
File Edit Task Help
796
795
791
797
781

FIG. 61

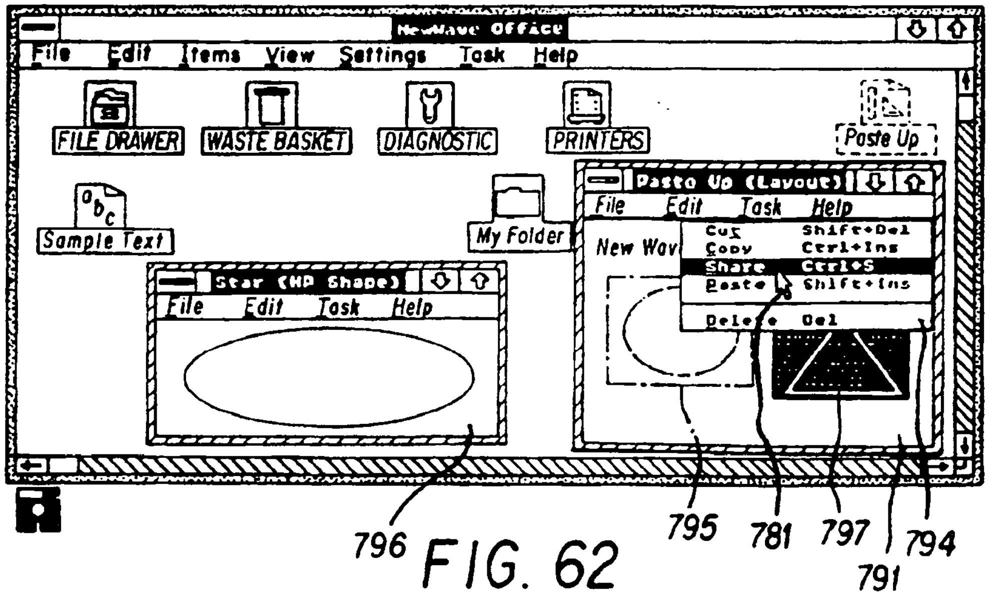

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wav
Cut Shift+Del
Copy Ctrl+Ins
Share Ctrl+S
Paste Shift+Ins
Delete Del
Star (HP Shape)
File Edit Task Help
796
795
781
797
791
794

FIG. 62

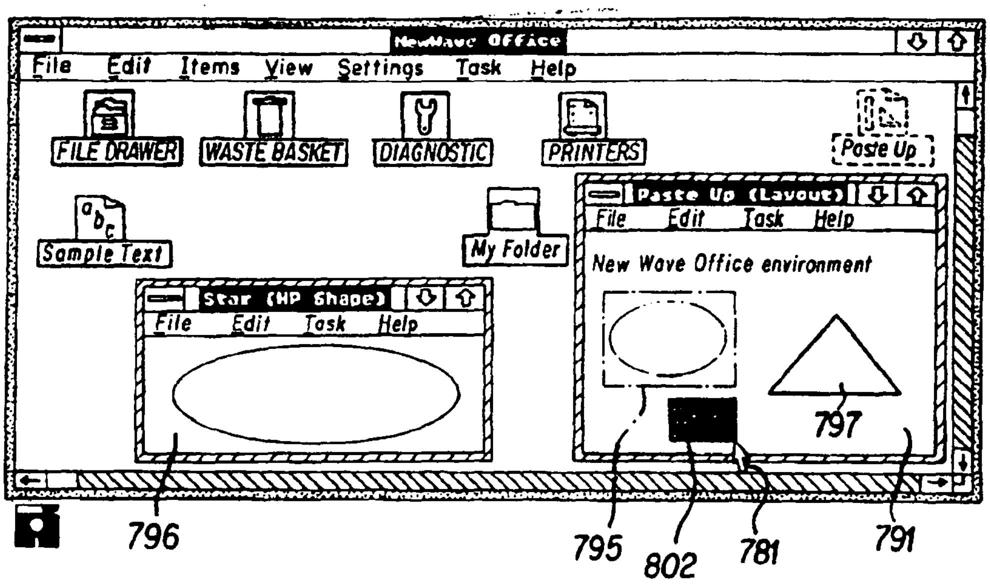

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
Star (HP Shape)
File Edit Task Help
797
796
795
802
781
791

FIG. 63

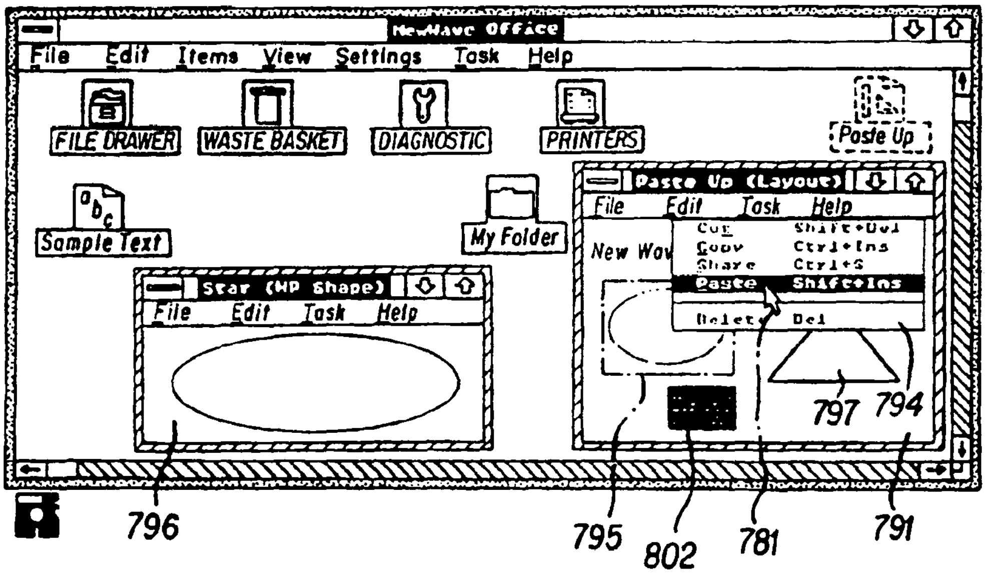

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wav
Cut Shift+Del
Copy Ctrl+Ins
Share Ctrl+S
Paste Shift+Ins
Delete Del
Star (HP Shape)
File Edit Task Help
797
794
796
795
802
781
791

FIG. 64

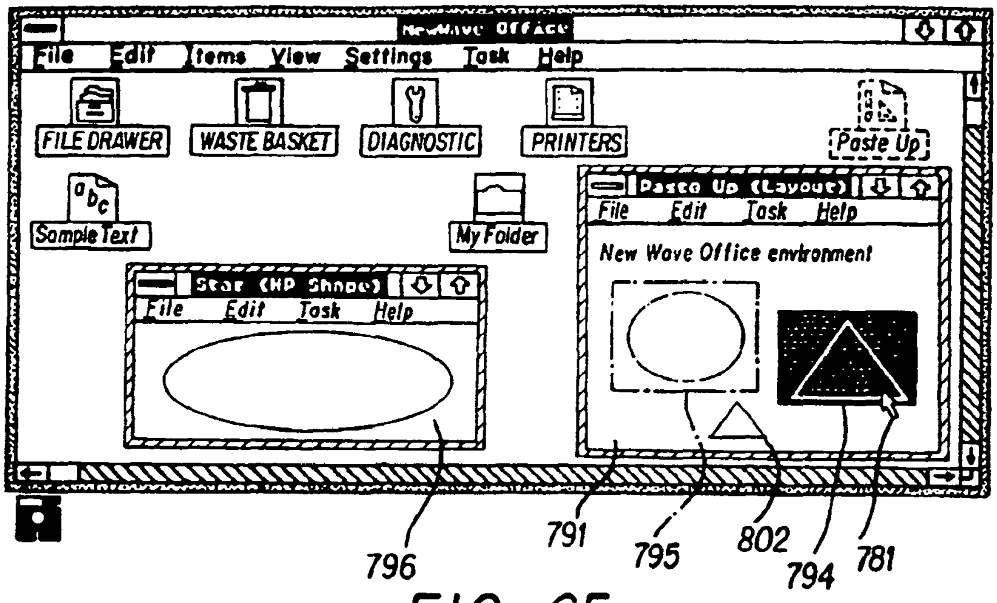

NewWave Office
File
Edit
Items
View
Settings
Task
Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File
Edit
Task
Help
New Wave Office environment
Star (HP Shape)
File
Edit
Task
Help
796
791
795
802
794
781

FIG. 65

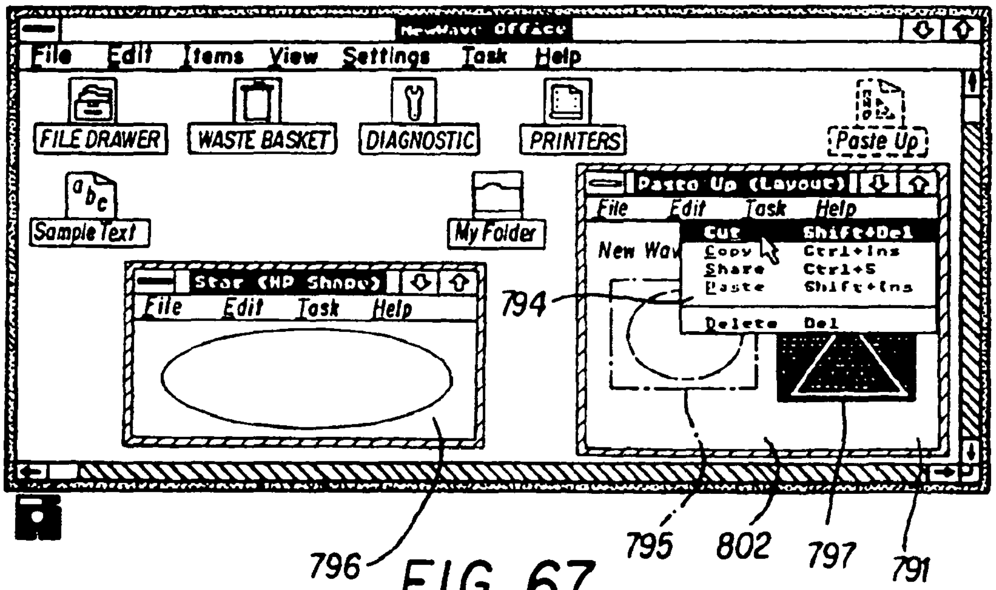

NewWave Office
File
Edit
Items
View
Settings
Task
Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File
Edit
Task
Help
New Wav
Cut Shift+Del
Copy Ctrl+Ins
Share Ctrl+S
Paste Shift+Ins
Delete Del
794
Star (HP Shape)
File
Edit
Task
Help
796
795
802
797
791

FIG. 67

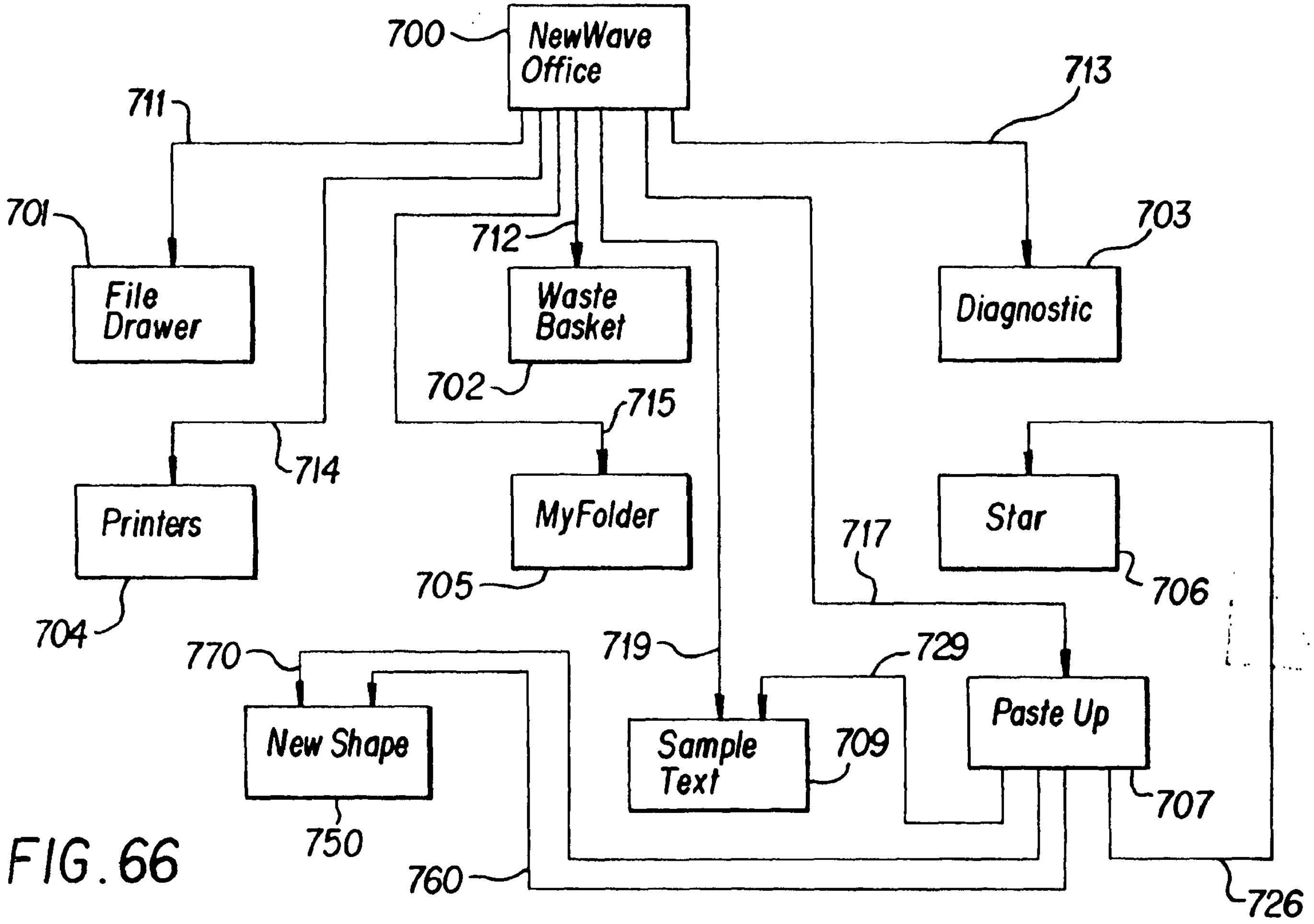

700
NewWave Office
711
713
701
712
703
File Drawer
Waste Basket
Diagnostic
702
715
714
Printers
MyFolder
Star
717
704
705
706
770
719
729
Paste Up
New Shape
Sample Text
709
707
750
760
726

FIG.66

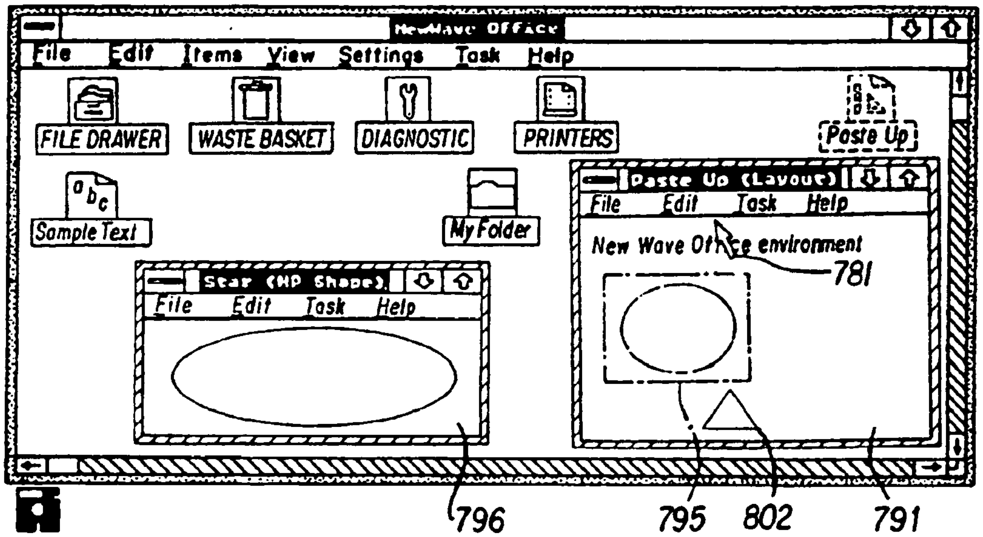

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
781
Star (HP Shape)
File Edit Task Help
796
795
802
791

FIG. 68

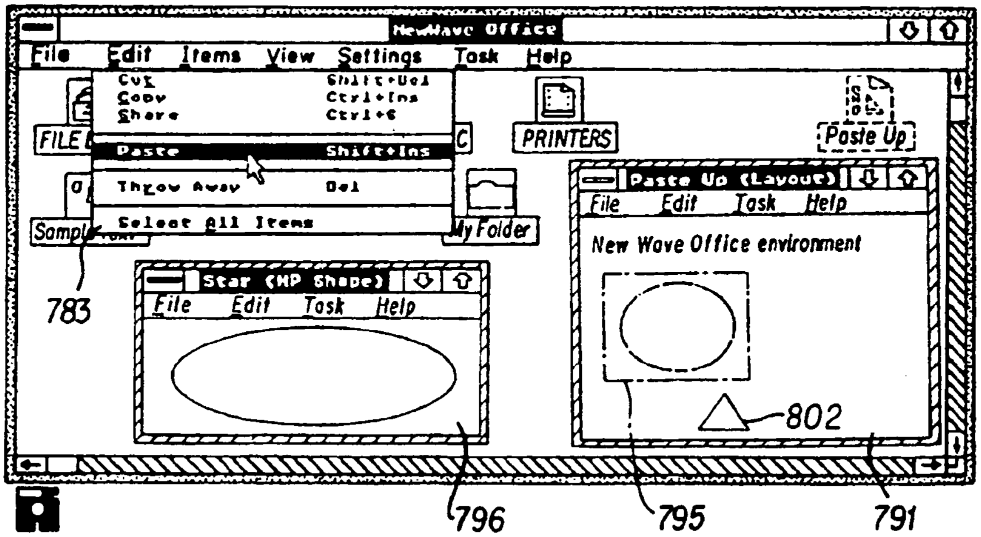

NewWave Office
File Edit Items View Settings Task Help
Cut Shift+Del
Copy Ctrl+Ins
Share Ctrl+C
Paste Shift+Ins
Throw Away Del
Select All Items
FILE
PRINTERS
Paste Up
Sample Text
My Folder
783
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
Star (HP Shape)
File Edit Task Help
802
796
795
791

FIG. 69

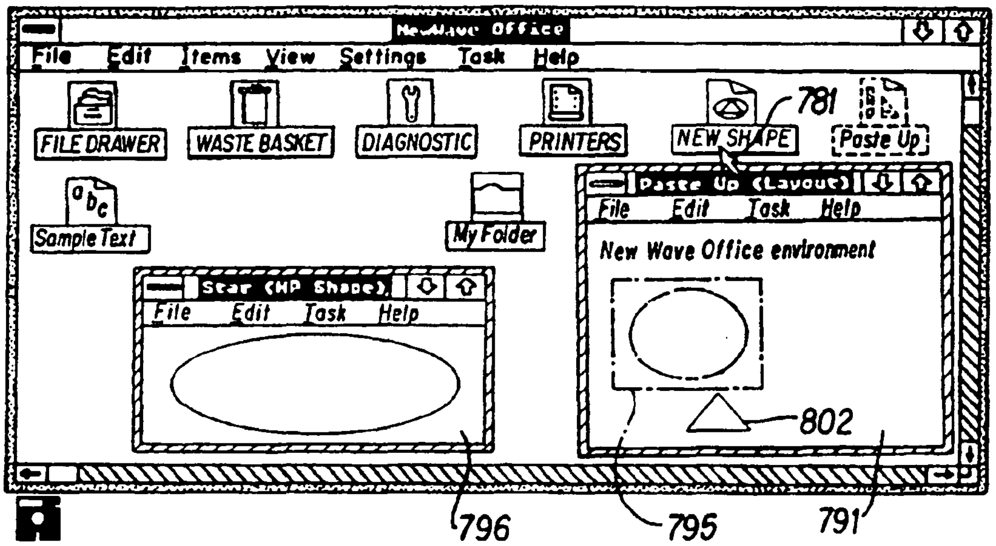

NewWave Office
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
NEW SHAPE
781
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
802
Star (HP Shape)
File Edit Task Help
796
795
791

FIG. 70

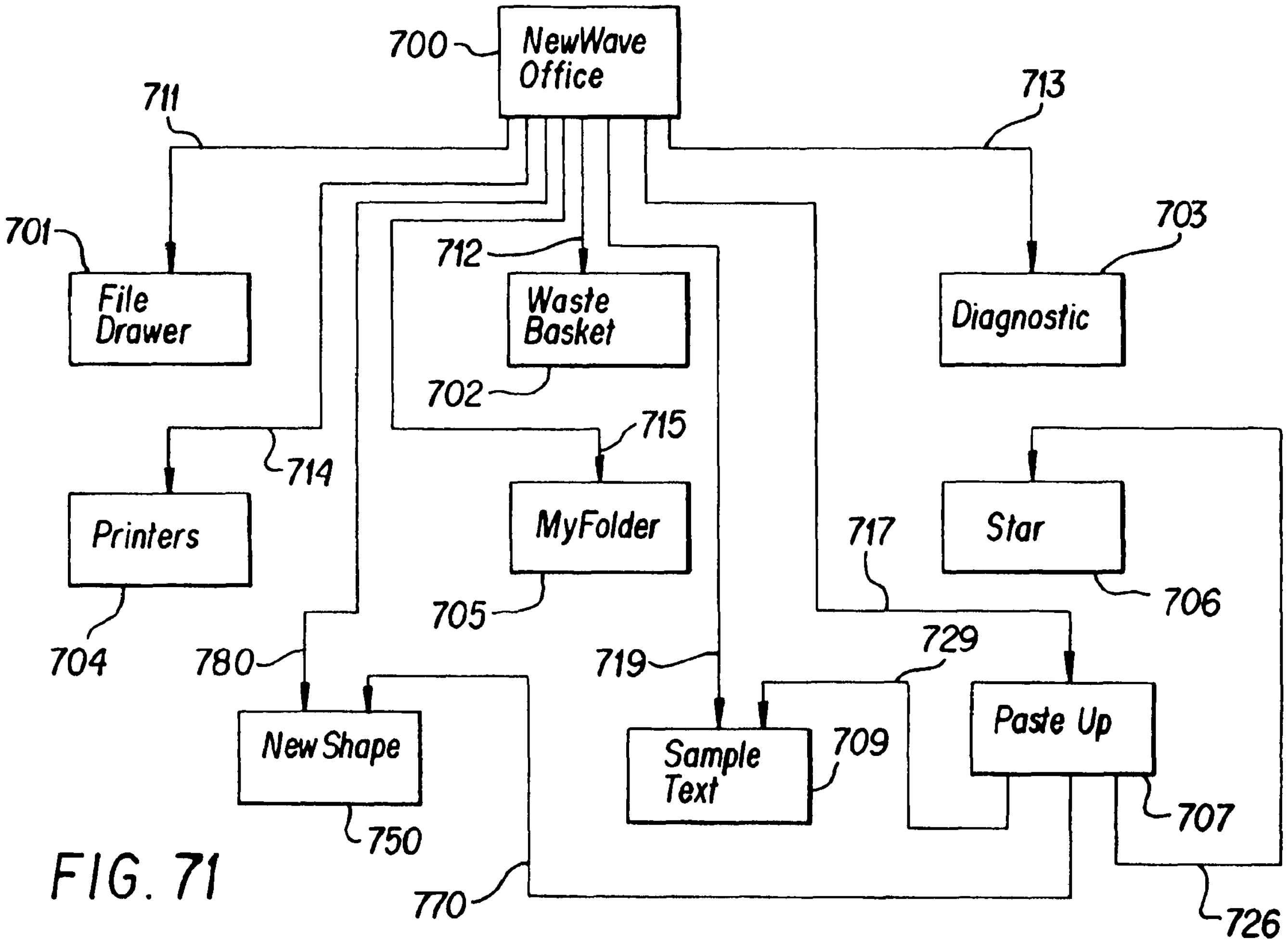

700
NewWave Office
711
713
701
File Drawer
712
Waste Basket
703
Diagnostic
702
715
714
Printers
MyFolder
717
Star
706
704
705
729
780
719
NewShape
Sample Text
709
Paste Up
707
750
770
726

FIG. 71

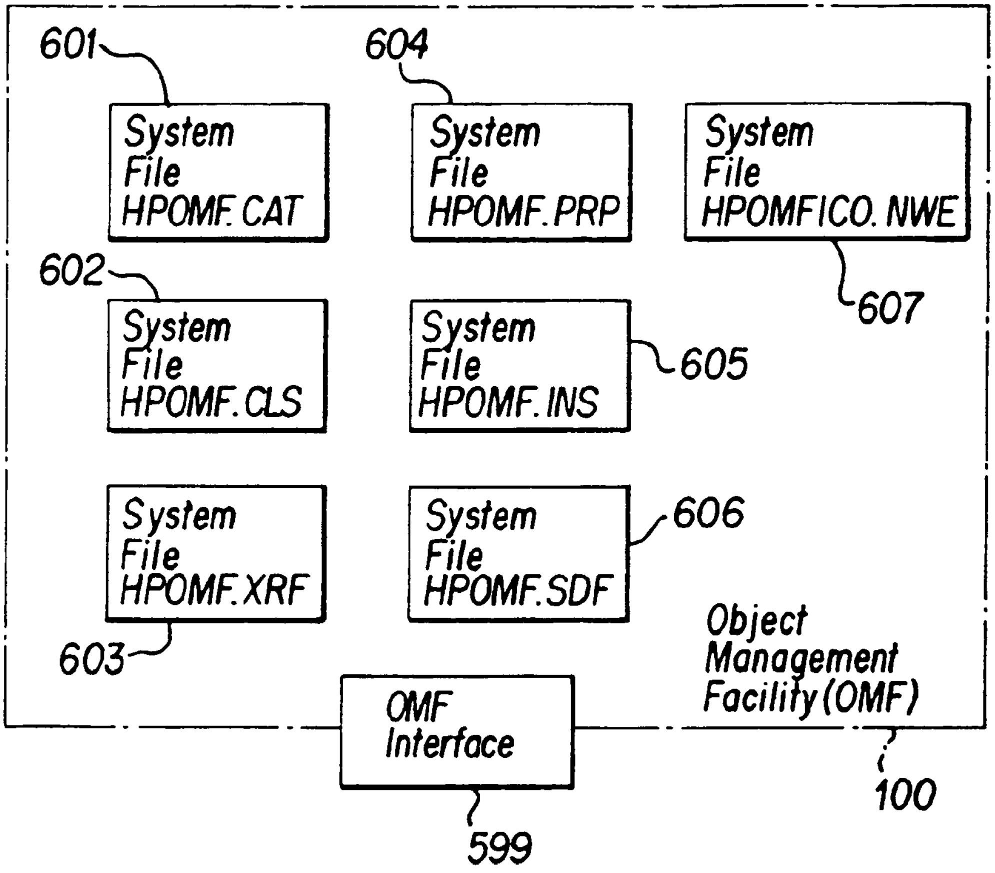

601
System File HPOMF.CAT
604
System File HPOMF.PRP
System File HPOMFICO.NWE
607
602
System File HPOMF.CLS
System File HPOMF.INS
605
System File HPOMF.XRF
System File HPOMF.SDF
606
603
Object Management Facility (OMF)
OMF Interface
599
100

FIG. 72

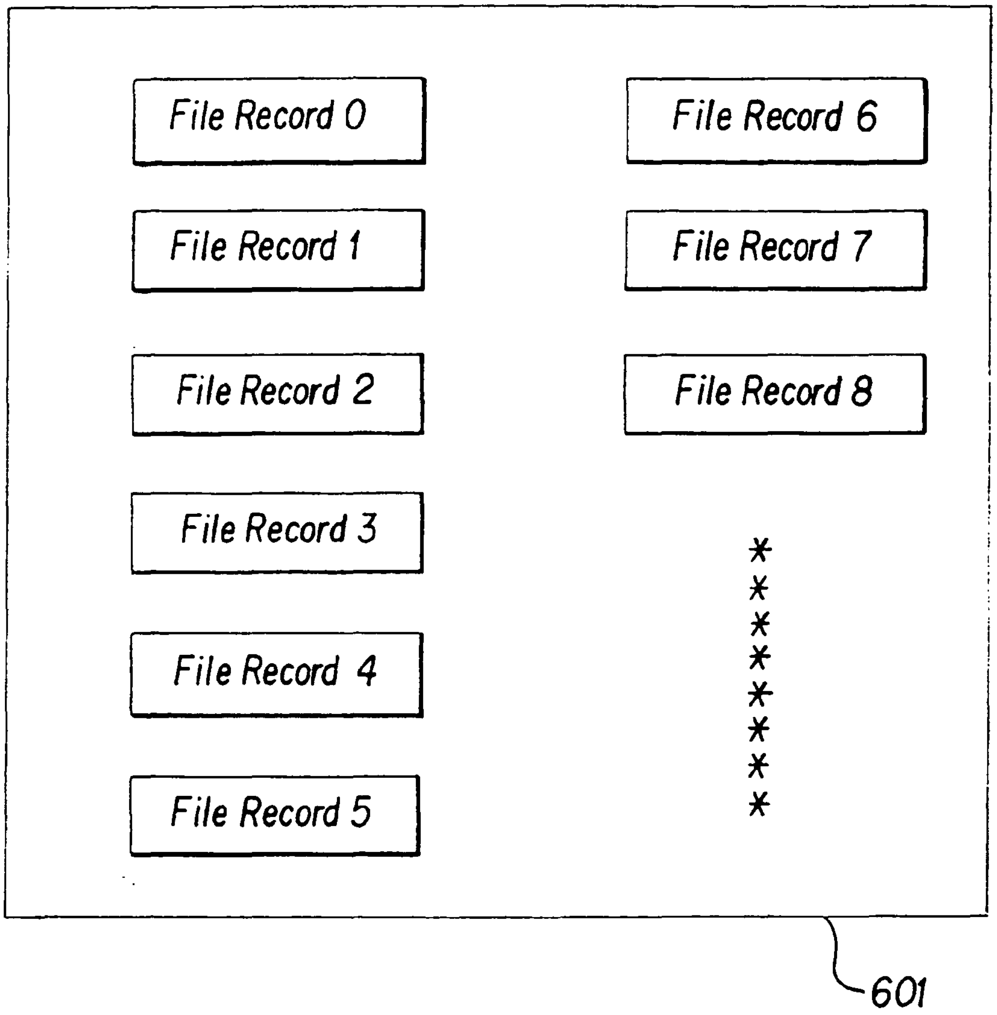

File Record 0
File Record 1
File Record 2
File Record 3
File Record 4
File Record 5
File Record 6
File Record 7
File Record 8
*
*
*
*
*
*
*
*
601

FIG. 73

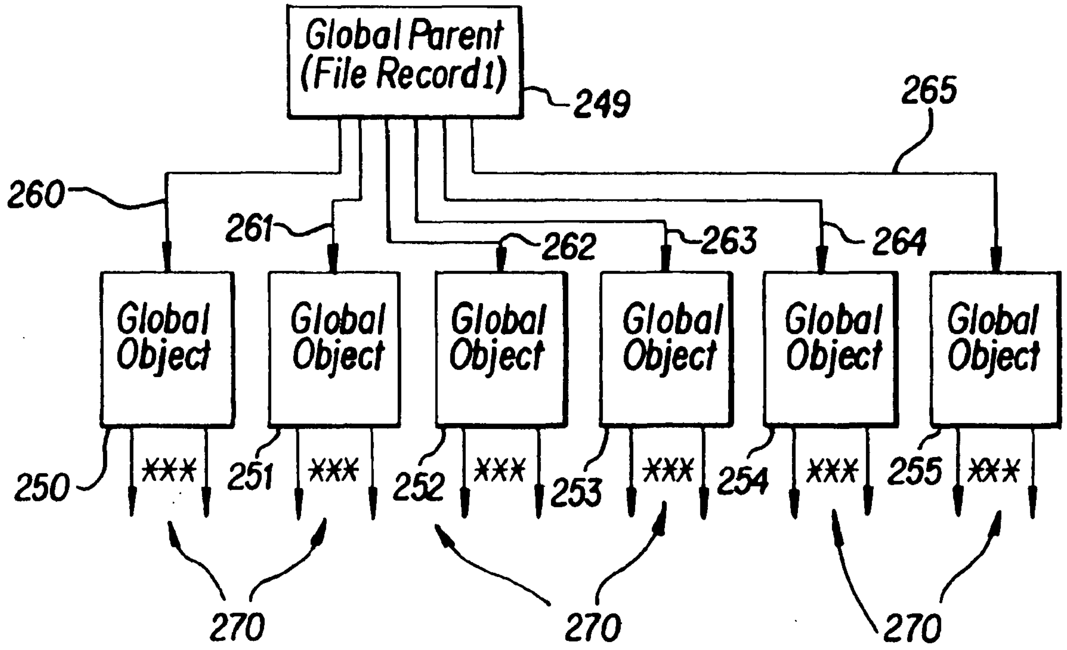

Global Parent (File Record1)
249
265
260
261
262
263
264
Global Object
Global Object
Global Object
Global Object
Global Object
Global Object
250
251
252
253
254
255
XXX
XXX
XXX
XXX
XXX
XXX
270
270
270

FIG. 74

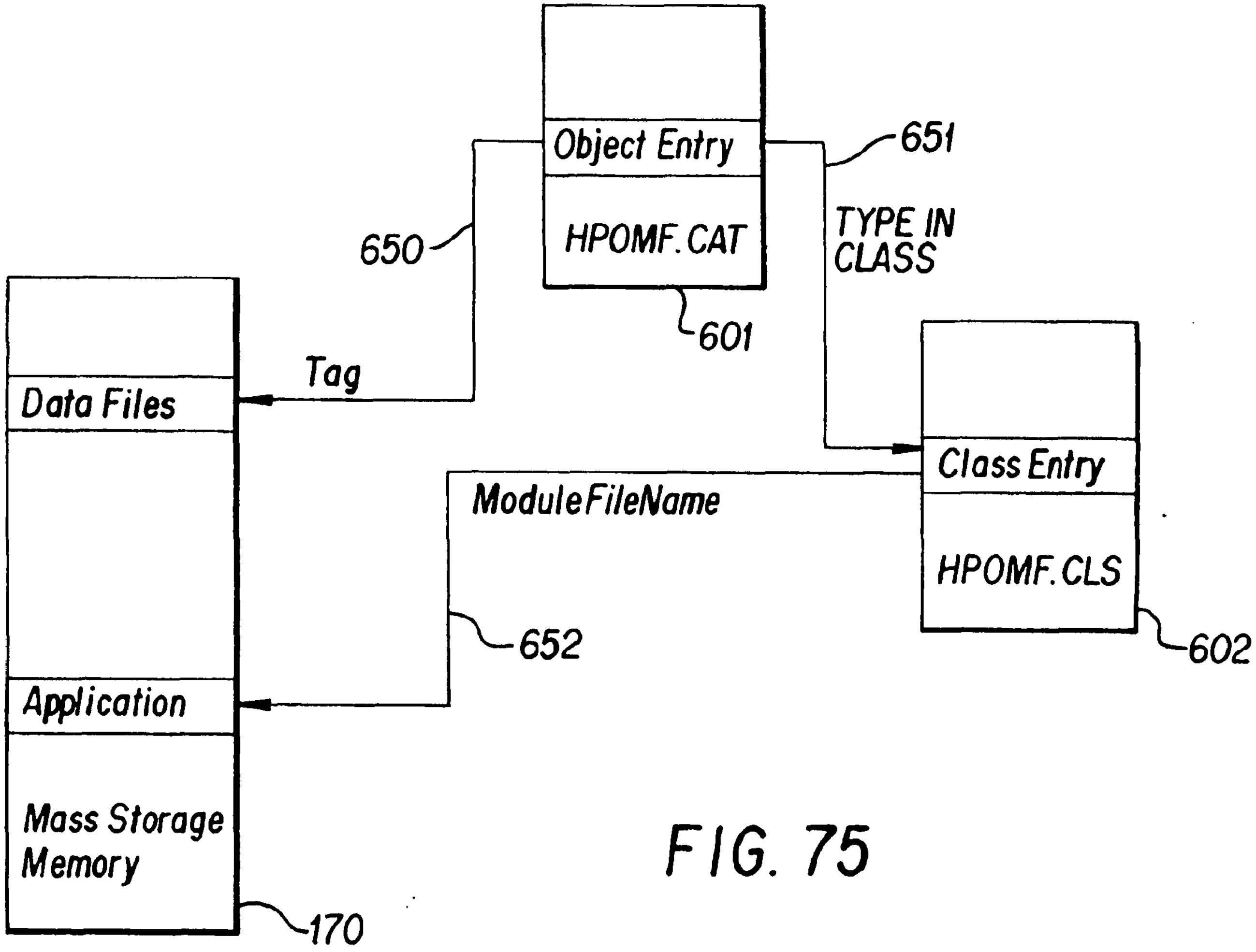

Object Entry
HPOMF.CAT
601
650
651
TYPE IN CLASS
Tag
Data Files
Class Entry
HPOMF.CLS
602
ModuleFileName
652
Application
Mass Storage Memory
170

FIG. 75

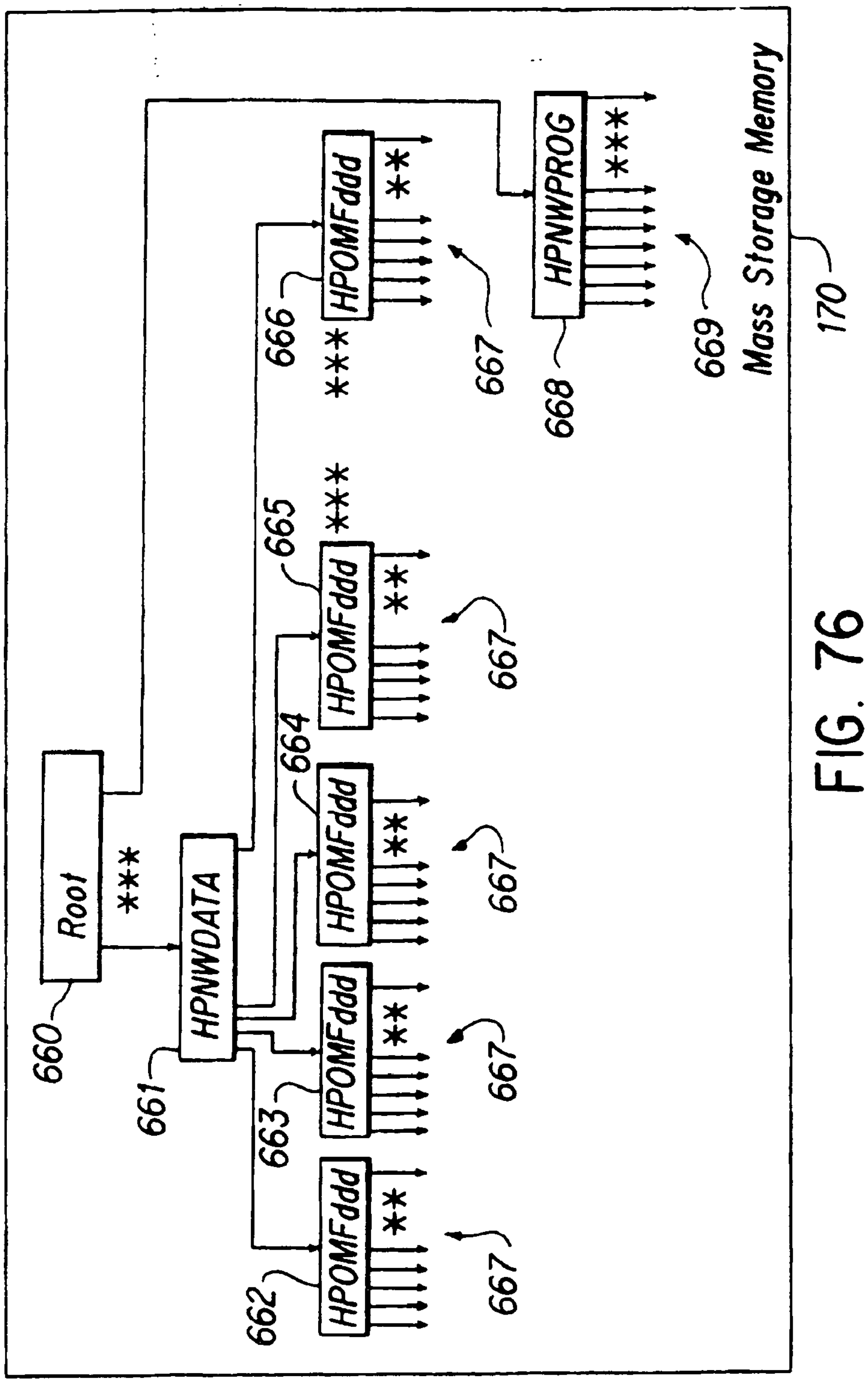

Root
660
***
HPNWDATA
661
HPOMFddd
662
**
667
HPOMFddd
663
**
667
HPOMFddd
664
**
667
HPOMFddd
665
**
***
667
HPOMFddd
666
***
**
667
HPNWPROG
668
***
669
Mass Storage Memory
170

FIG. 76

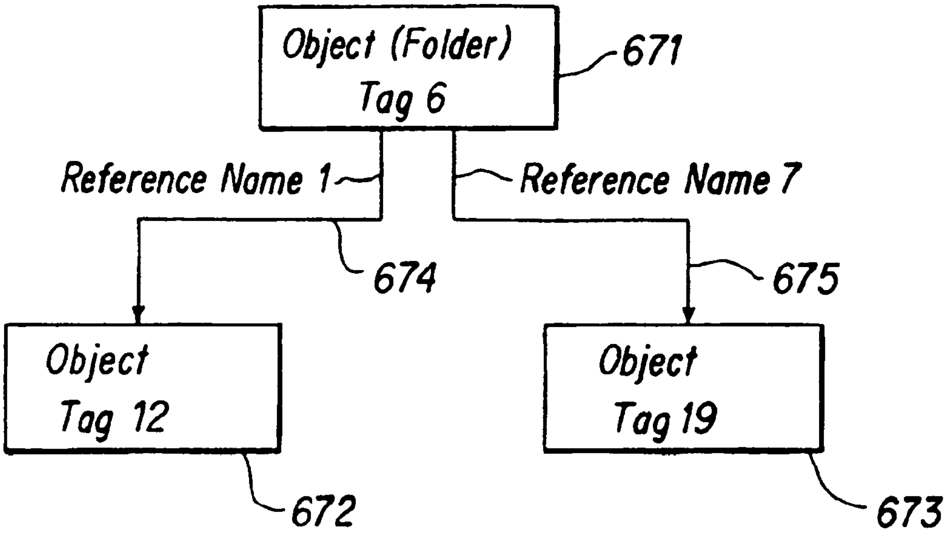

Object (Folder)
Tag 6
671
Reference Name 1
Reference Name 7
674
675
Object
Tag 12
Object
Tag 19
672
673

FIG. 77

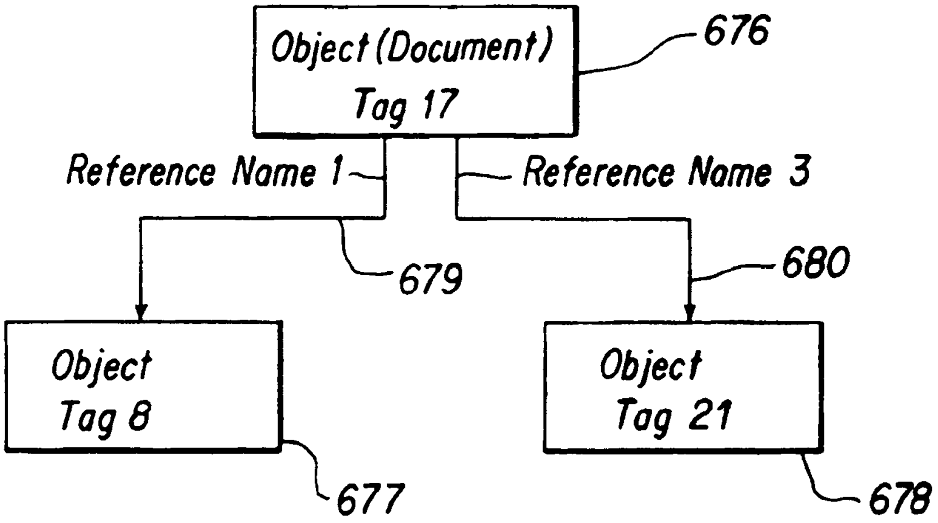

Object (Document)
Tag 17
676
Reference Name 1
Reference Name 3
679
680
Object
Tag 8
Object
Tag 21
677
678

FIG. 78

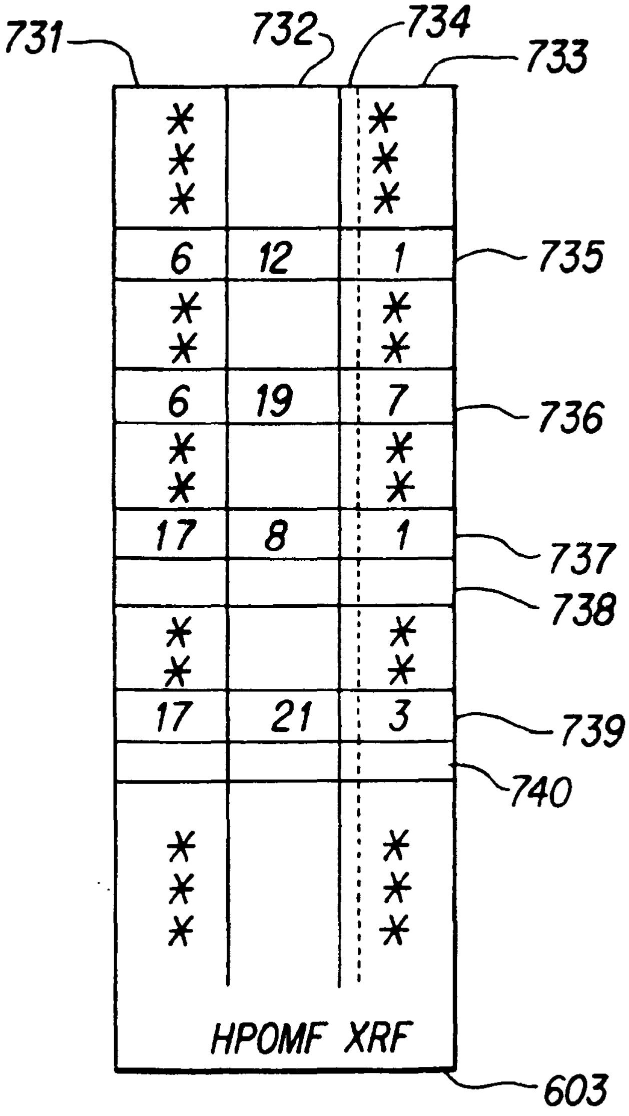
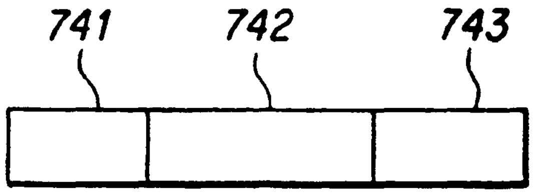

731
732
734
733
* * *
* * *
6
12
1
735
* *
* *
6
19
7
736
* *
* *
17
8
1
737
738
* *
* *
17
21
3
739
740
* * *
* * *
HPOMF XRF
603

FIG. 79



View Specification Record 740

FIG. 80

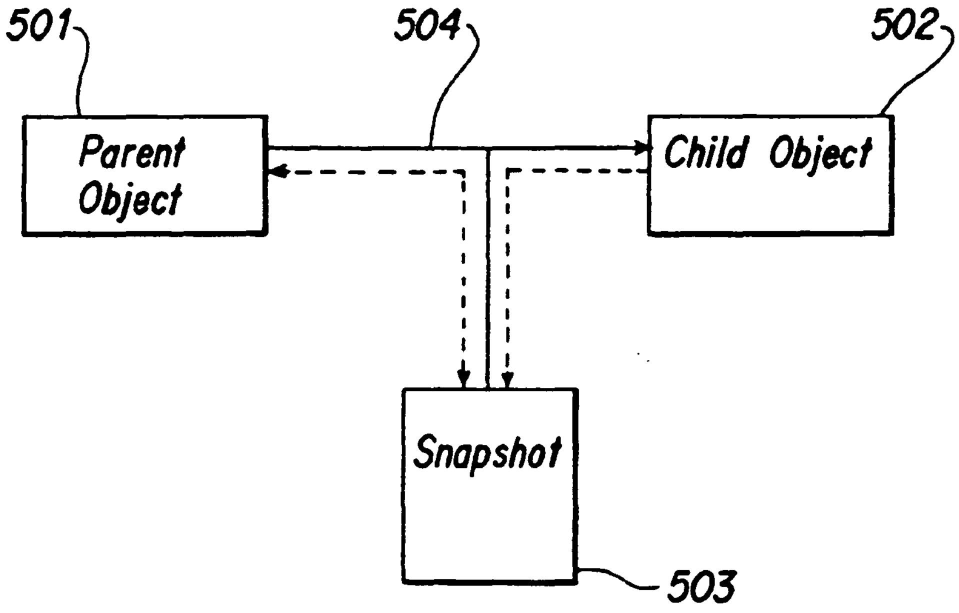


FIG. 81

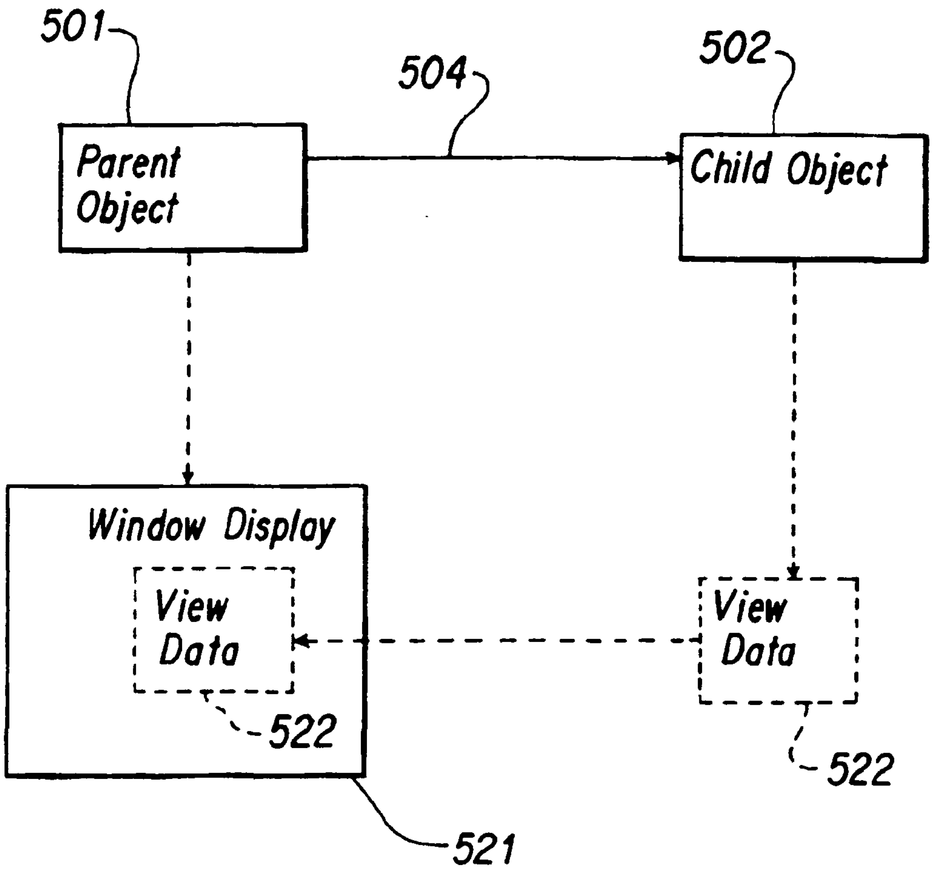

501
Parent Object
504
502
Child Object
Window Display
View Data
522
521
View Data
522

FIG. 82

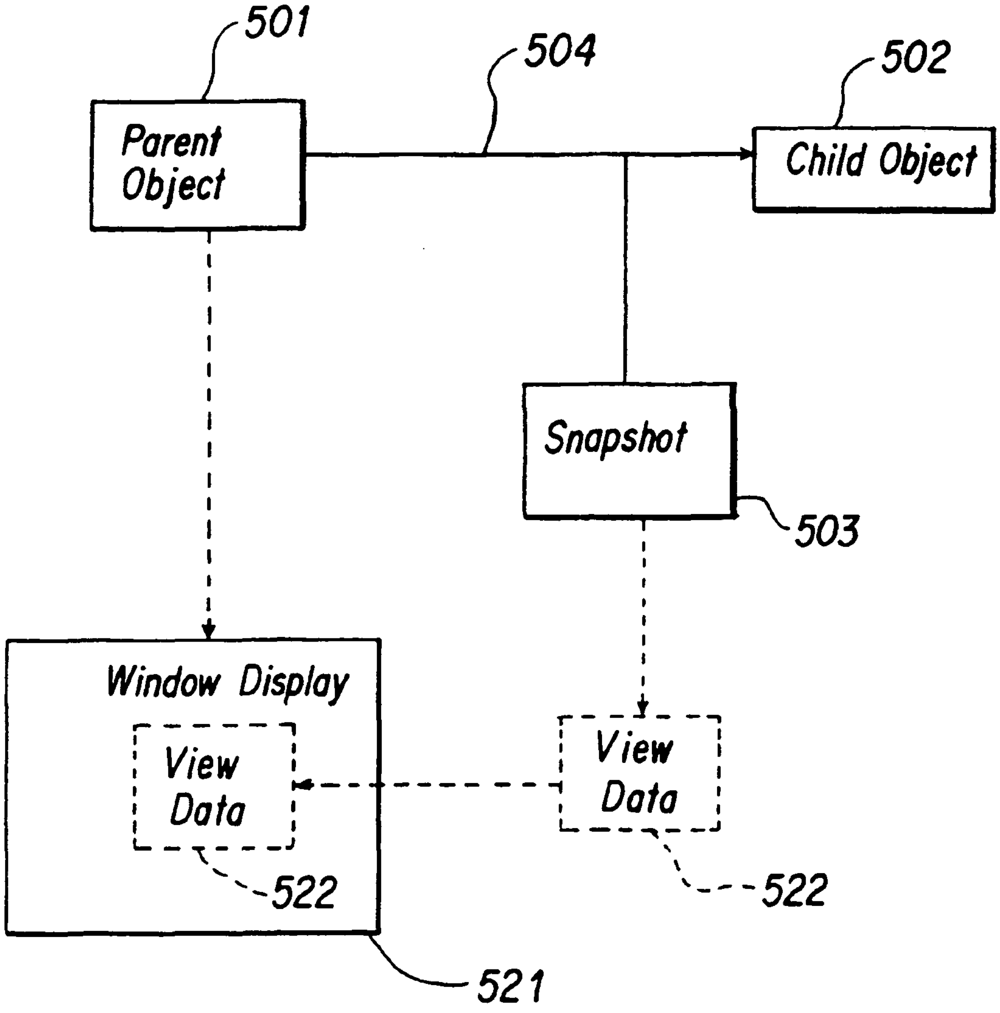


FIG. 83

Claims

1. An object based distributed computer system comprising a network of workstations and means for transmitting objects between workstations characterised by objects including a first object type for storing data and a second object type for presenting data to a user, wherein objects of the second type (V-c) reference an associated object of the first type (V-s) to enable a plurality of users of workstations to access data of the object of the first type, comprising means for transmitting an object of the second type (V-c) between workstations thereby to create a reference to the associated object of the first type (V-s) for each workstation receiving an object of the second type.

2. A system according to claim 1 comprising means for copying an object of the second type (V-c) between workstations.

3. A system according to claim 1 or claim 2 wherein transmitted objects of the second type (V-c)include an identifier (60) for the associated object of the first type (V-s).

4. A system according to any preceding claim in the form of a conferencing system comprising means enabling users of the workstations to participate in a meeting over the network wherein objects of the first type (V-s) store meeting data and objects of the second type (V-c) are for presenting meeting data.

5. A method of convening a meeting using a system as claimed in claim 4 comprising transmitting an object of the second type (V-c) between workstations thereby to create a reference to the associated object of the first type (V-s) for each workstation receiving an object of the second type.

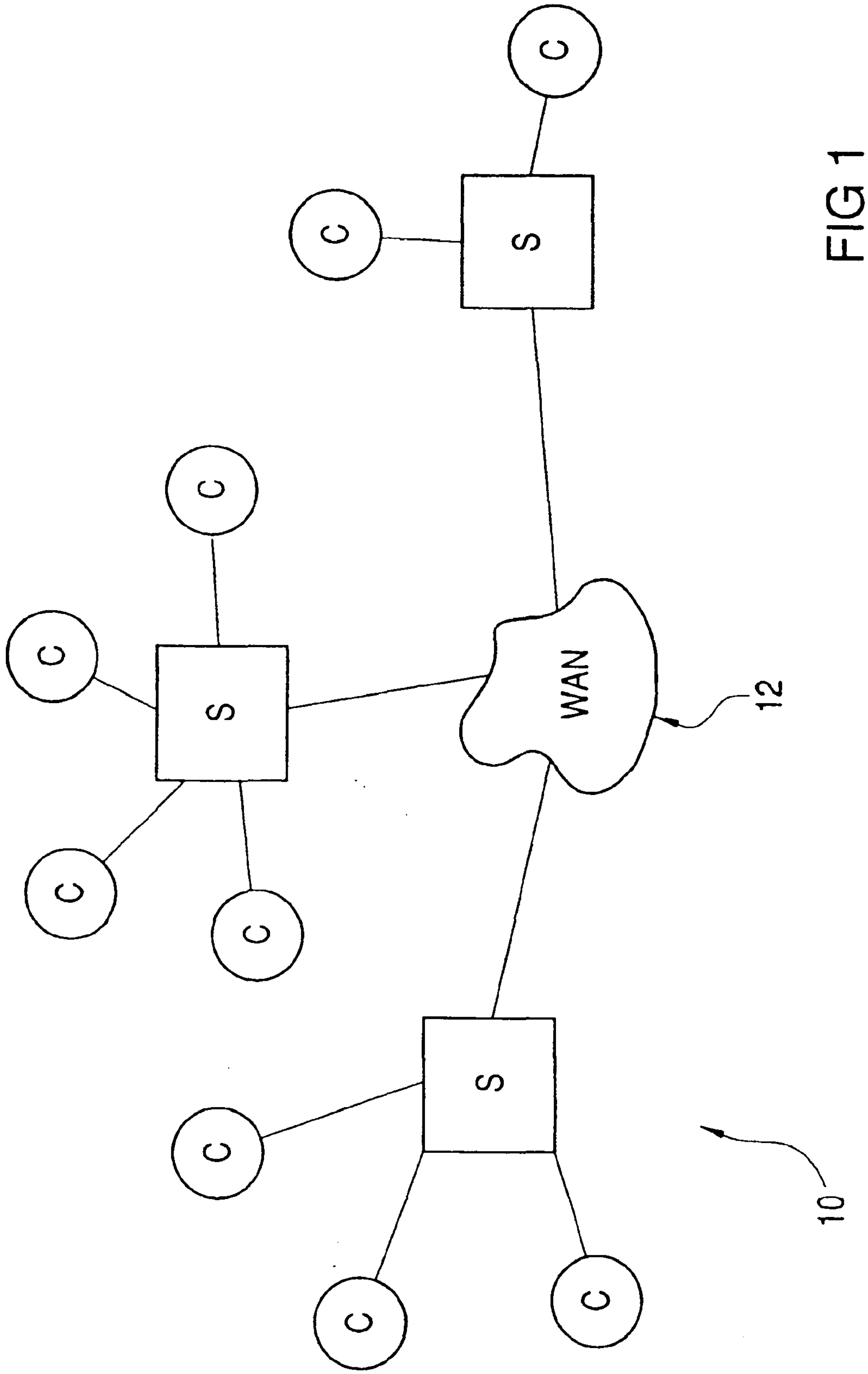


FIG 1

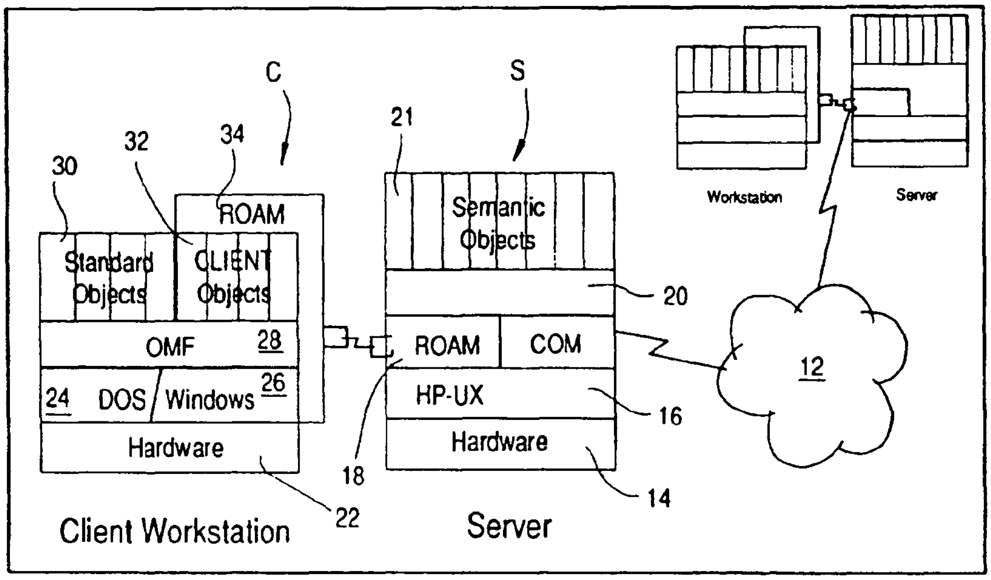

C
S
21
32
34
30
ROAM
Semantic Objects
Standard Objects
CLIENT Objects
20
OMF
28
ROAM
COM
12
24
DOS
Windows
26
HP-UX
16
Hardware
Hardware
18
14
22
Client Workstation
Server
Workstation
Server

FIG 2

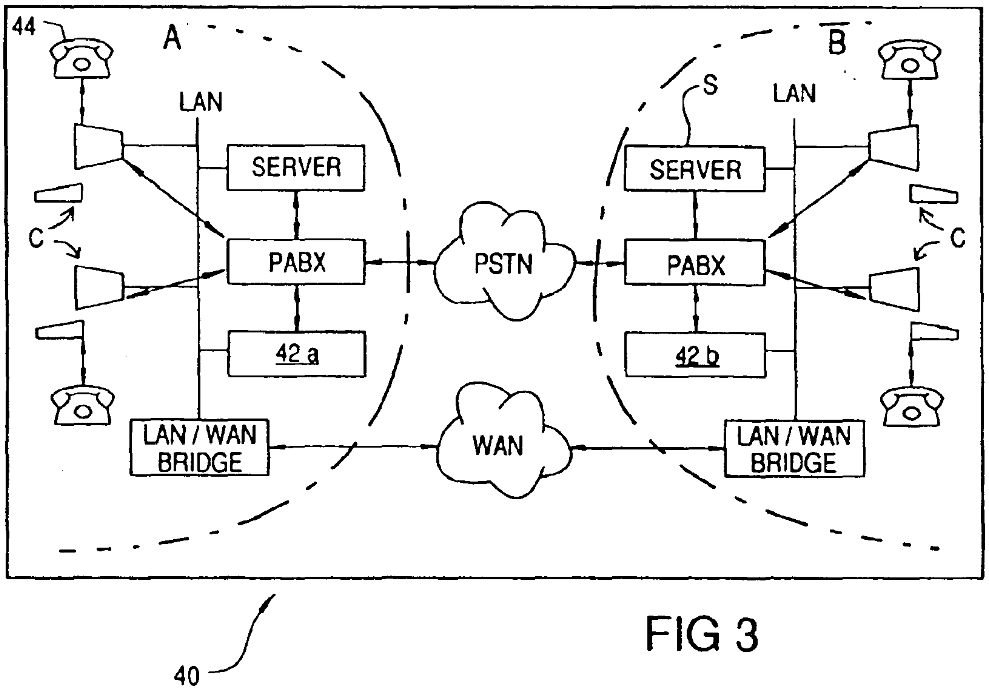

44
A
B
S
LAN
LAN
SERVER
SERVER
C
C
PABX
PSTN
PABX
42 a
42 b
LAN / WAN BRIDGE
WAN
LAN / WAN BRIDGE
40

FIG 3

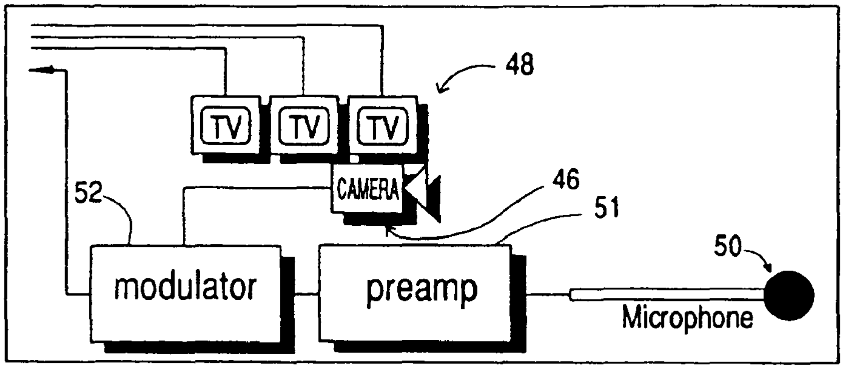

48
TV
TV
TV
CAMERA
52
46
51
50
modulator
preamp
Microphone

FIG 4

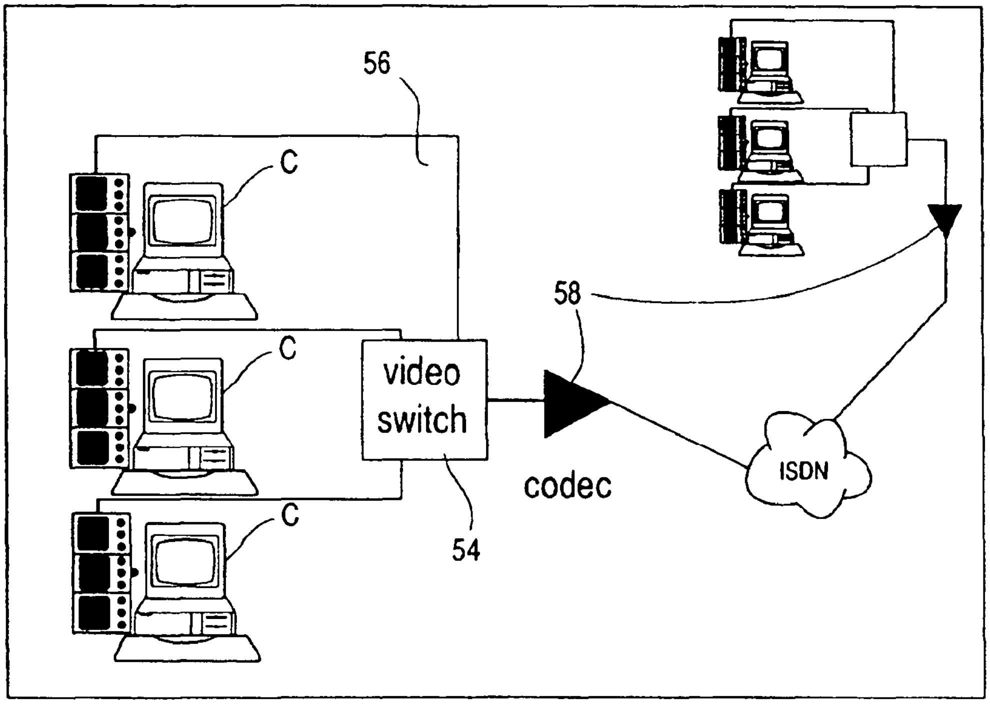

56
C
C
C
video switch
codec
54
58
ISDN

FIG 5

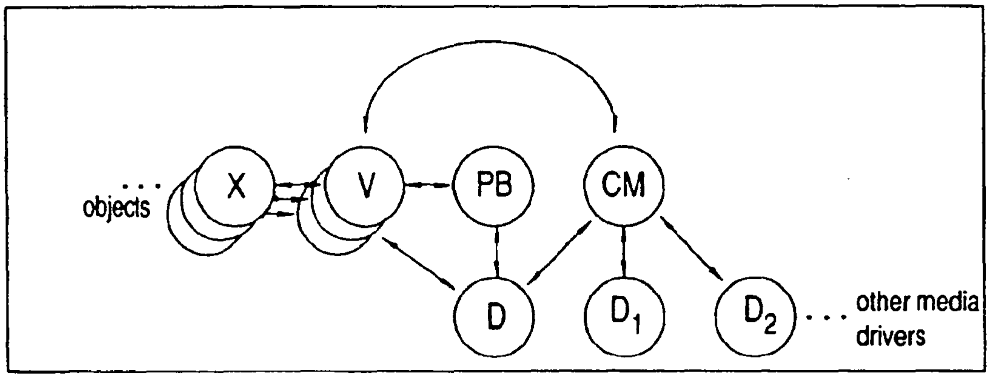

objects
X
V
PB
CM
D
D_1
D_2
other media drivers

FIG 6

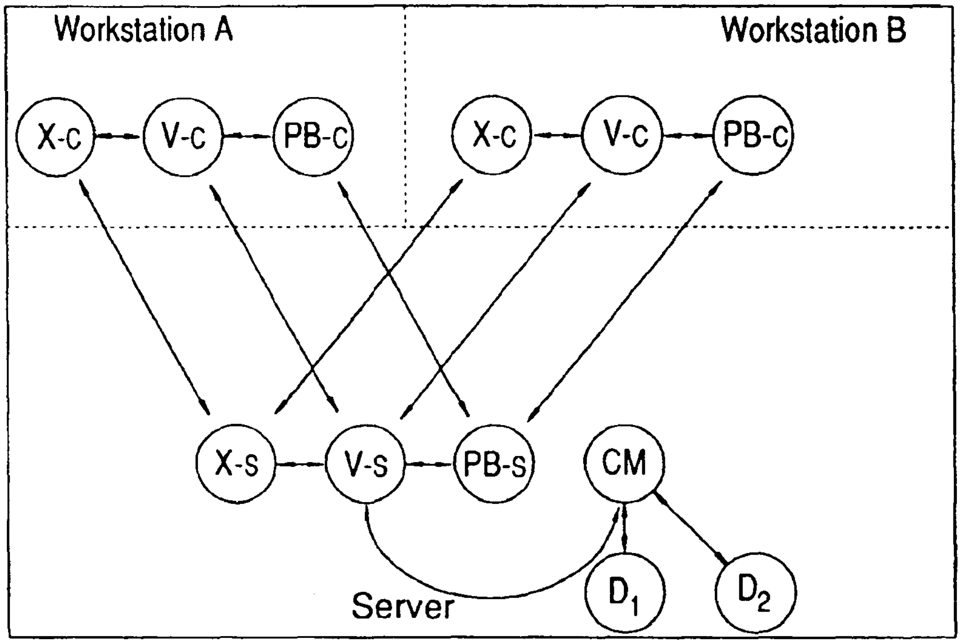

Workstation A
Workstation B
X-c
V-c
PB-c
X-c
V-c
PB-c
X-s
V-s
PB-s
CM
D_1
D_2
Server

FIG 7

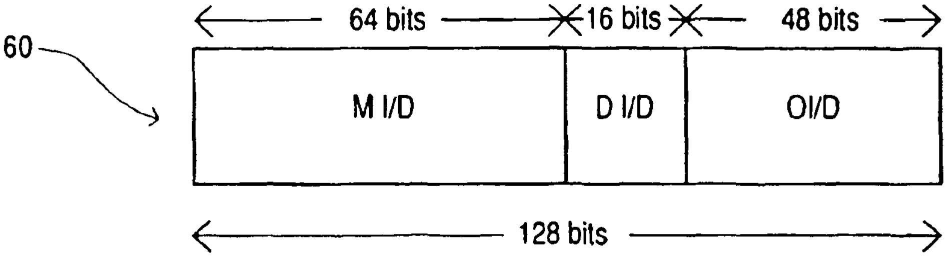


FIG 8

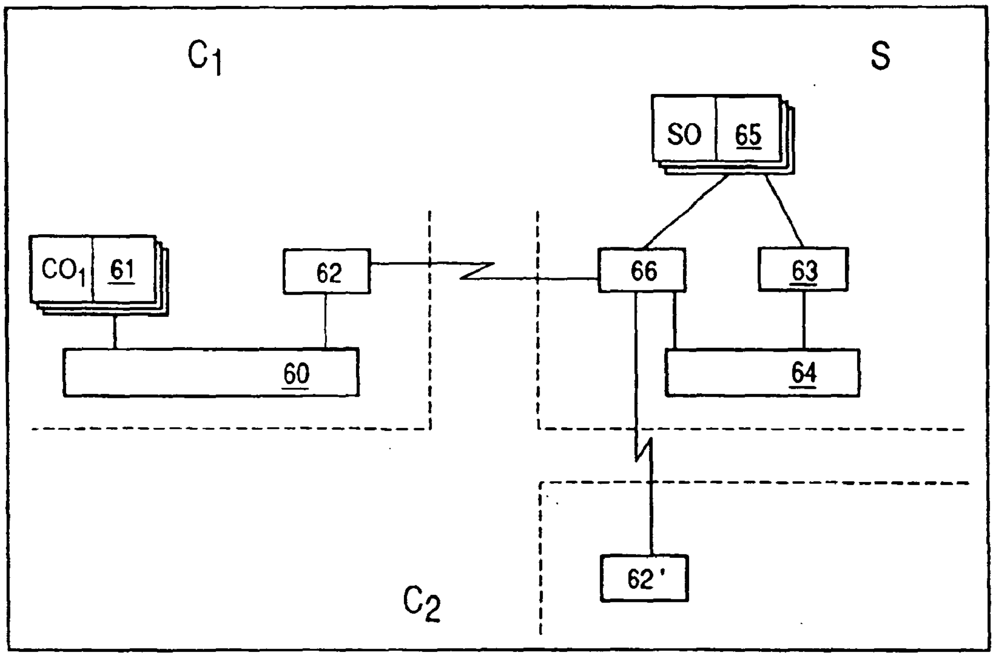


FIG 9

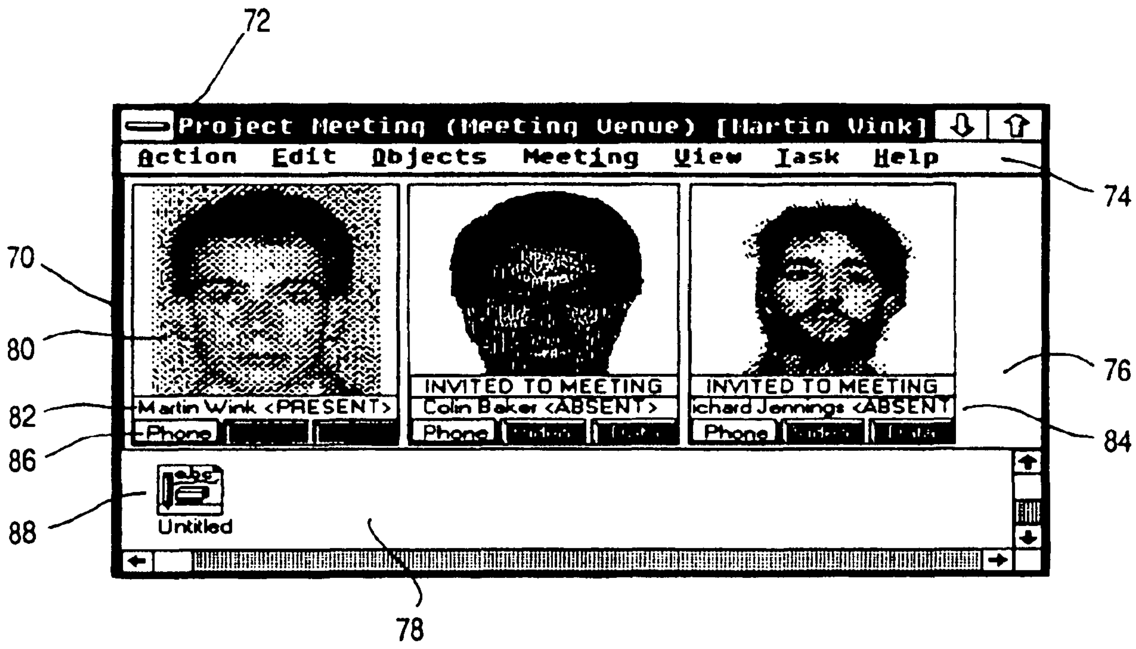
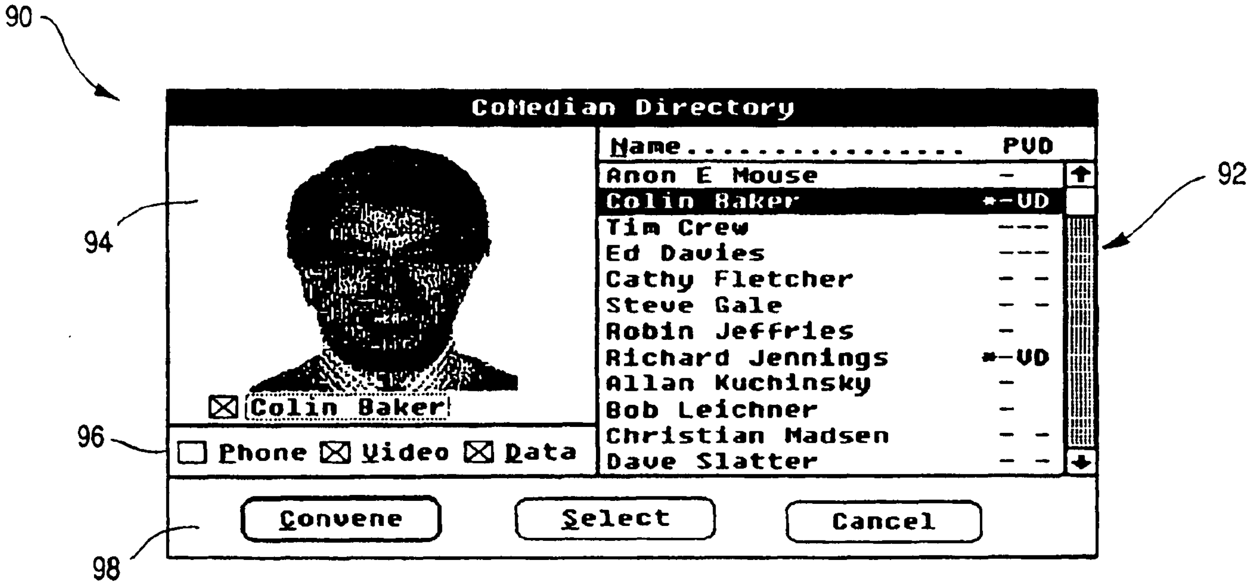

72
Project Meeting (Meeting Venue) [Martin Wink]
Action Edit Objects Meeting View Task Help
74
70
80
76
INVITED TO MEETING
INVITED TO MEETING
82
Martin Wink <PRESENT>
Colin Baker <ABSENT>
ichard Jennings <ABSENT
84
86
Phone
Phone
Phone
88
Untitled
78

FIG 10



CoMedian Directory

☒ Colin Baker

□ Phone ☒ Video ☒ Data

Name................	PVD
Anon E Mouse	-
Colin Baker	*-VD
Tim Crew	---
Ed Davies	---
Cathy Fletcher	- -
Steve Gale	- -
Robin Jeffries	-
Richard Jennings	*-VD
Allan Kuchinsky	-
Bob Leichner	-
Christian Madsen	- -
Dave Slatter	- -

FIG 11

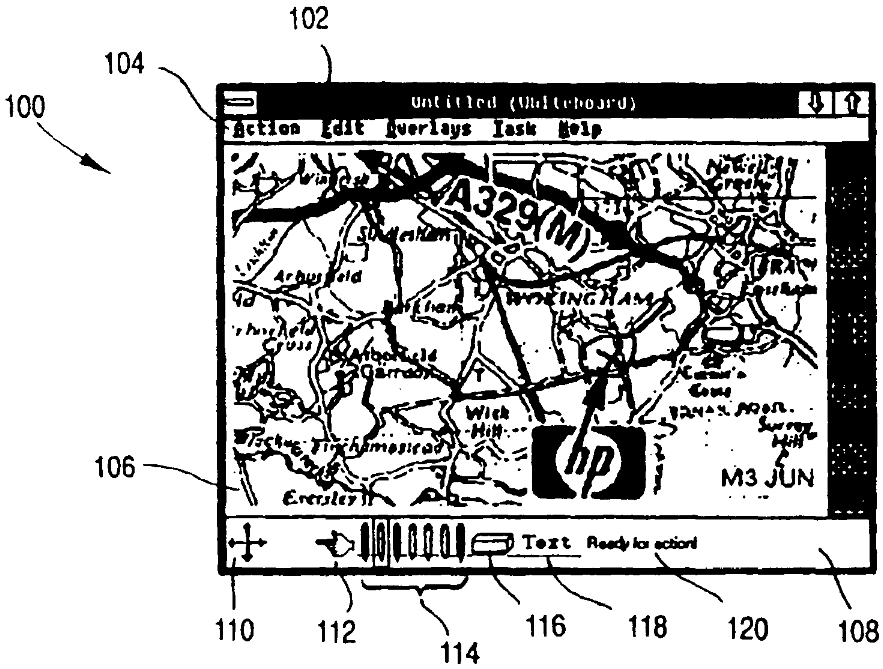

102
104
100
Untitled (Whiteboard)
Action Edit Overlays Task Help
A329(M)
WOKINGHAM
Wick Hill
Eversley
hp
M3 JUN
106
Text
Ready for action!
110
112
114
116
118
120
108

FIG 12

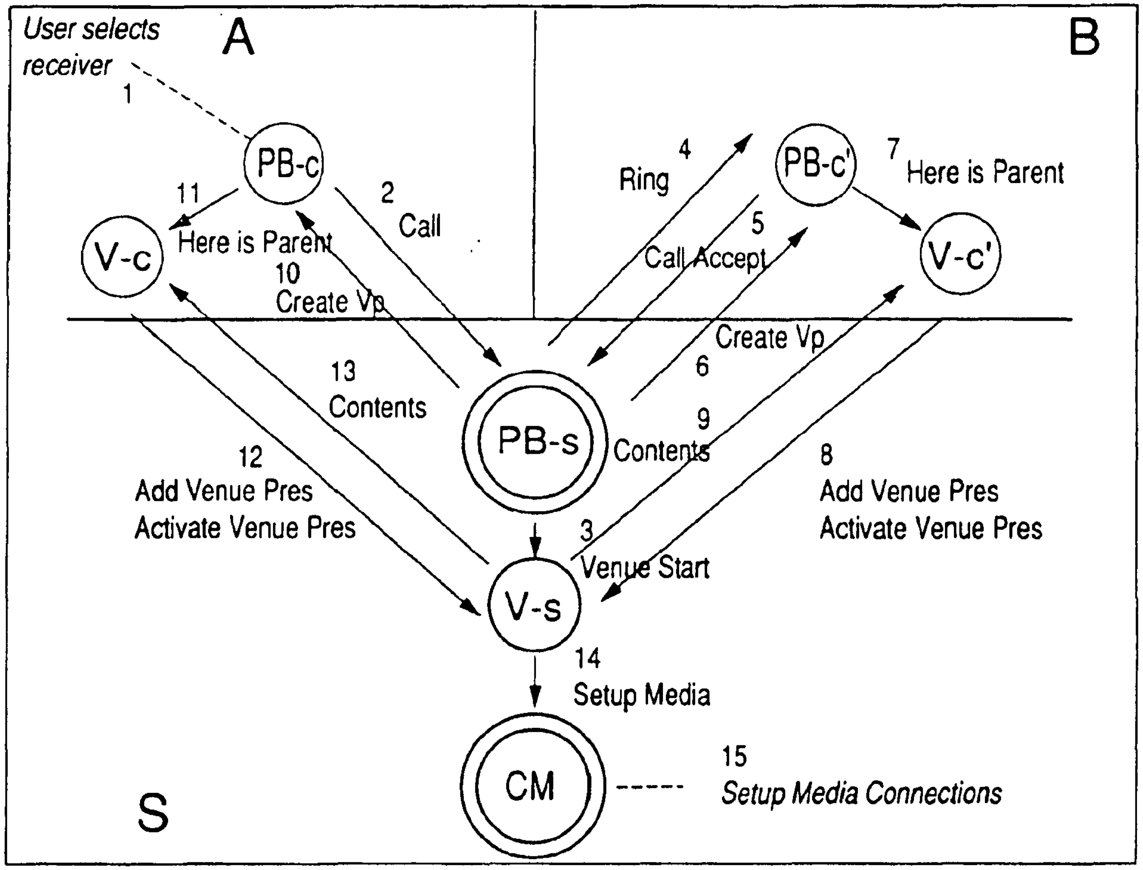

User selects
receiver
A
B
1
PB-c
2
Call
11
V-c
Here is Parent
10
Create Vp
4
Ring
PB-c'
7
Here is Parent
5
Call Accept
V-c'
Create Vp
6
13
Contents
PB-s
9
Contents
12
Add Venue Pres
Activate Venue Pres
8
Add Venue Pres
Activate Venue Pres
3
Venue Start
V-s
14
Setup Media
CM
15
Setup Media Connections
S

FIG 13

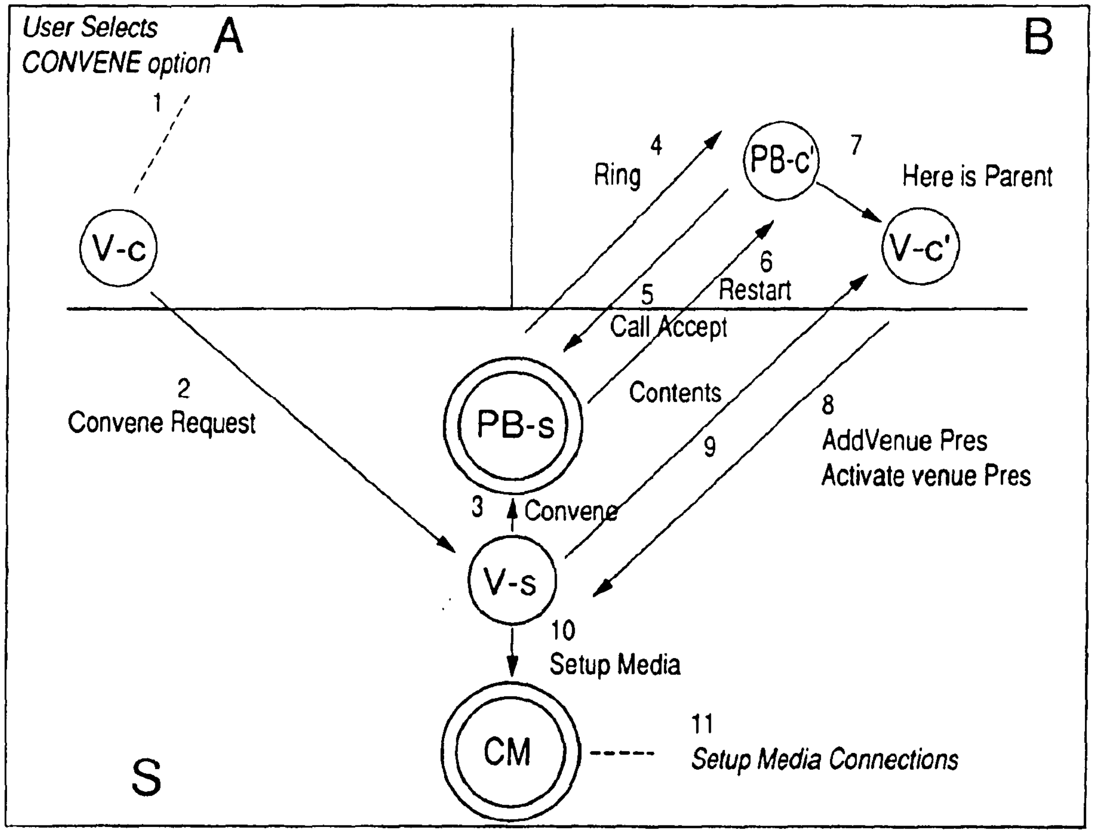

User Selects
CONVENE option
A
B
1
V-c
4
Ring
PB-c'
7
Here is Parent
V-c'
6
Restart
5
Call Accept
2
Convene Request
PB-s
Contents
8
AddVenue Pres
Activate venue Pres
9
3
Convene
V-s
10
Setup Media
CM
11
Setup Media Connections
S

FIG 14

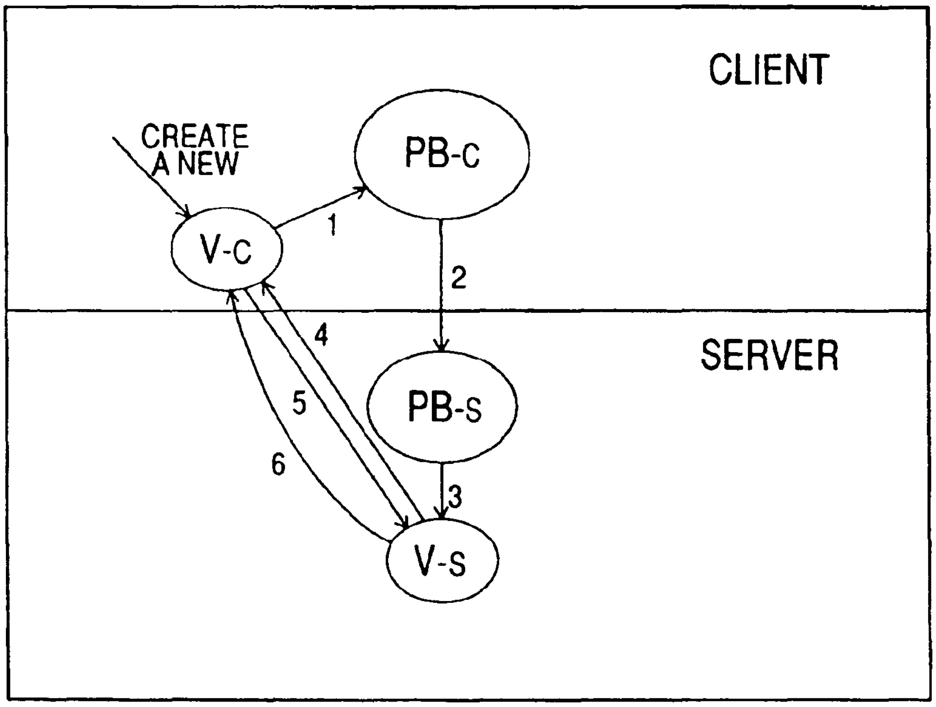


FIG 15

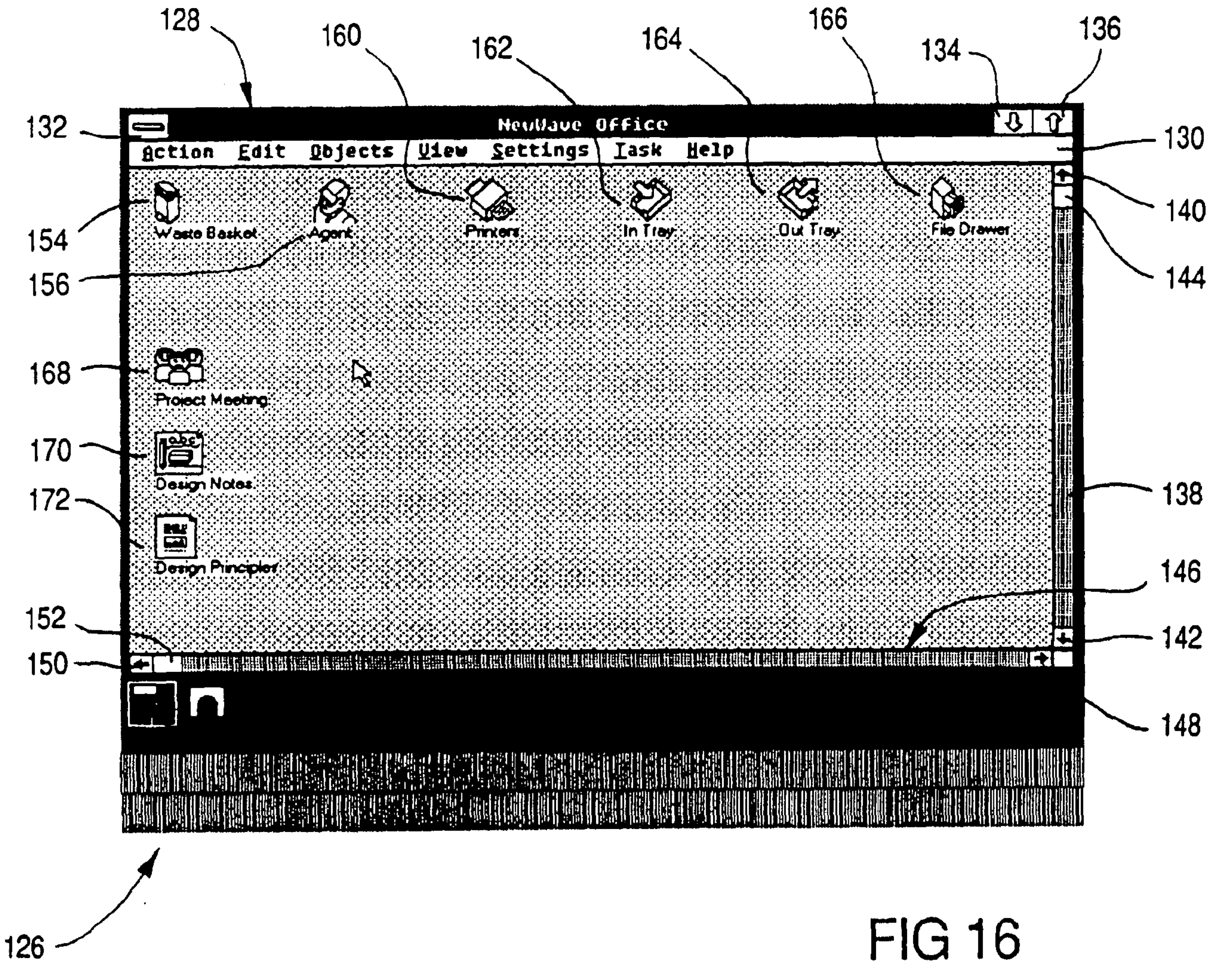

128
160
162
164
166
134
136
132
130
NewWave Office
Action
Edit
Objects
View
Settings
Task
Help
154
Waste Basket
Agent
Printers
In Tray
Out Tray
File Drawer
140
144
156
168
Project Meeting
170
Design Notes
172
Design Principles
138
146
152
142
150
148
126

FIG 16

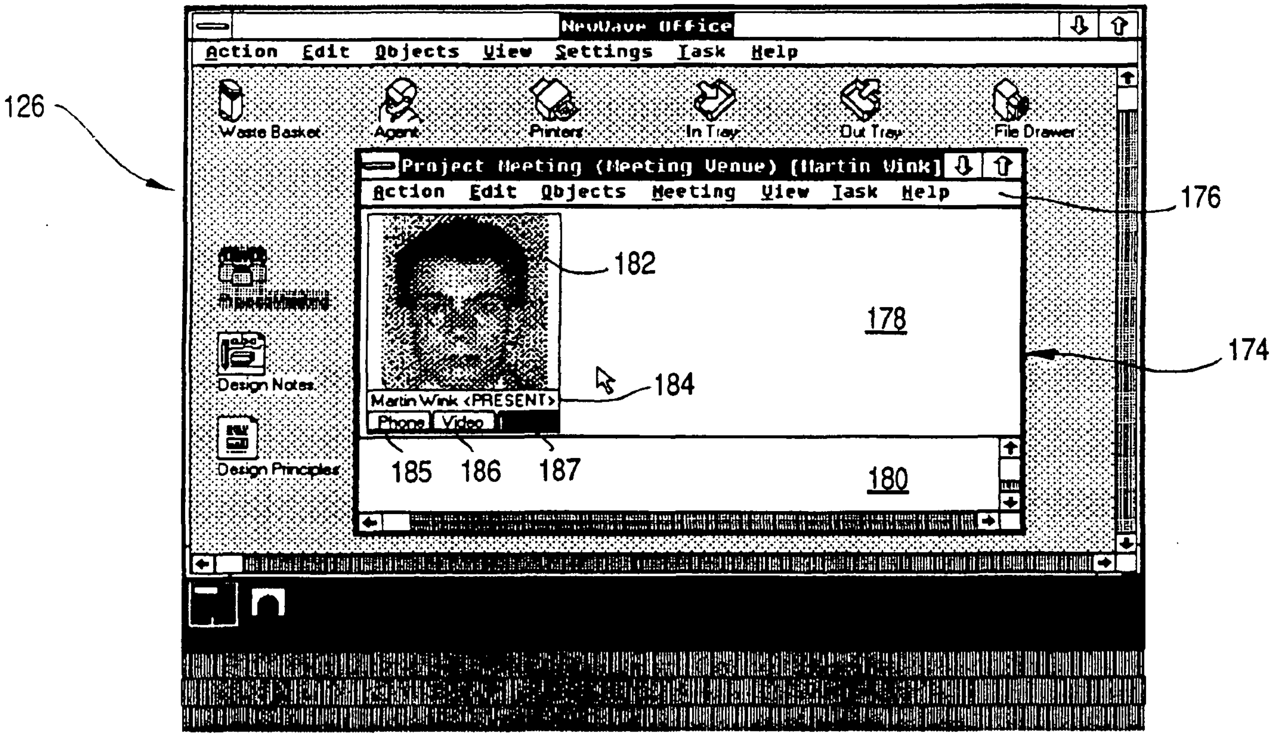
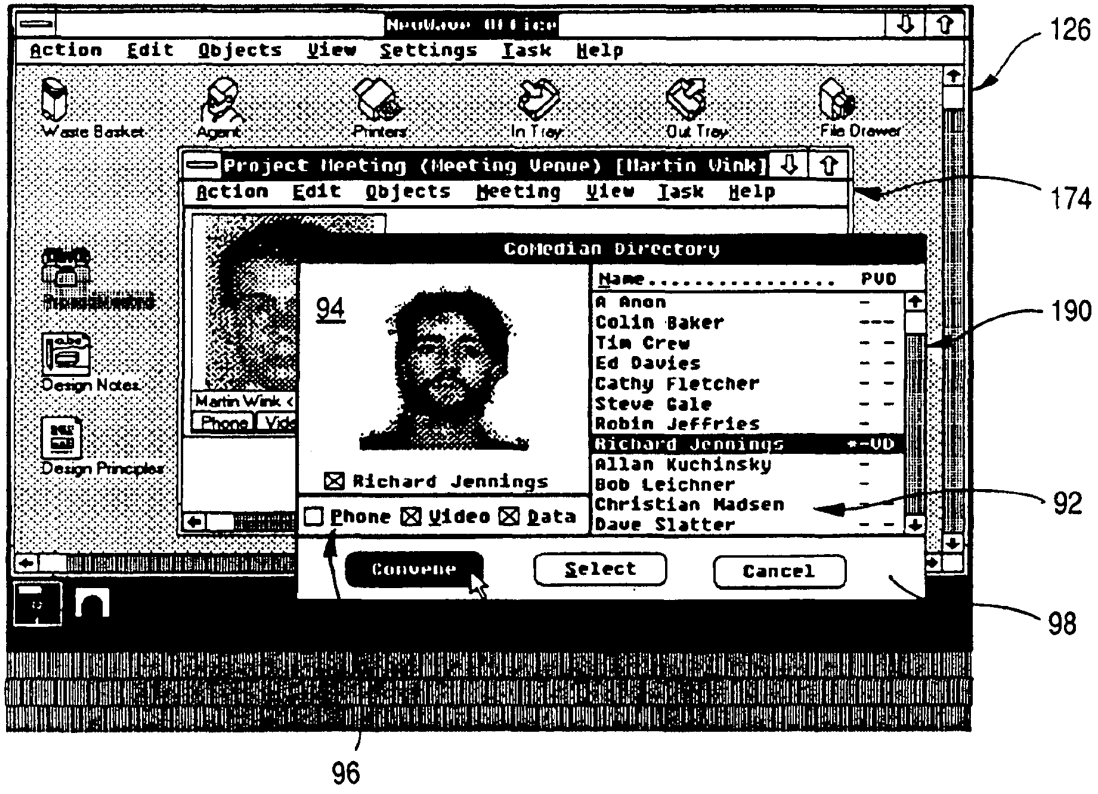

NewWave Office
Action Edit Objects View Settings Task Help
Waste Basket
Agent
Printers
In Tray
Out Tray
File Drawer
Project Meeting
Design Notes
Design Principles
Project Meeting (Meeting Venue) [Martin Wink]
Action Edit Objects Meeting View Task Help
Martin Wink <PRESENT>
Phone
Video
126
176
182
178
174
184
185
186
187
180

FIG 17



Name................	PVD
A Anon	-
Colin Baker	---
Tim Crew	- -
Ed Davies	- -
Cathy Fletcher	- -
Steve Gale	- -
Robin Jeffries	-
Richard Jennings	*-VD
Allan Kuchinsky	-
Bob Leichner	-
Christian Madsen	- -
Dave Slatter	- -

FIG 18

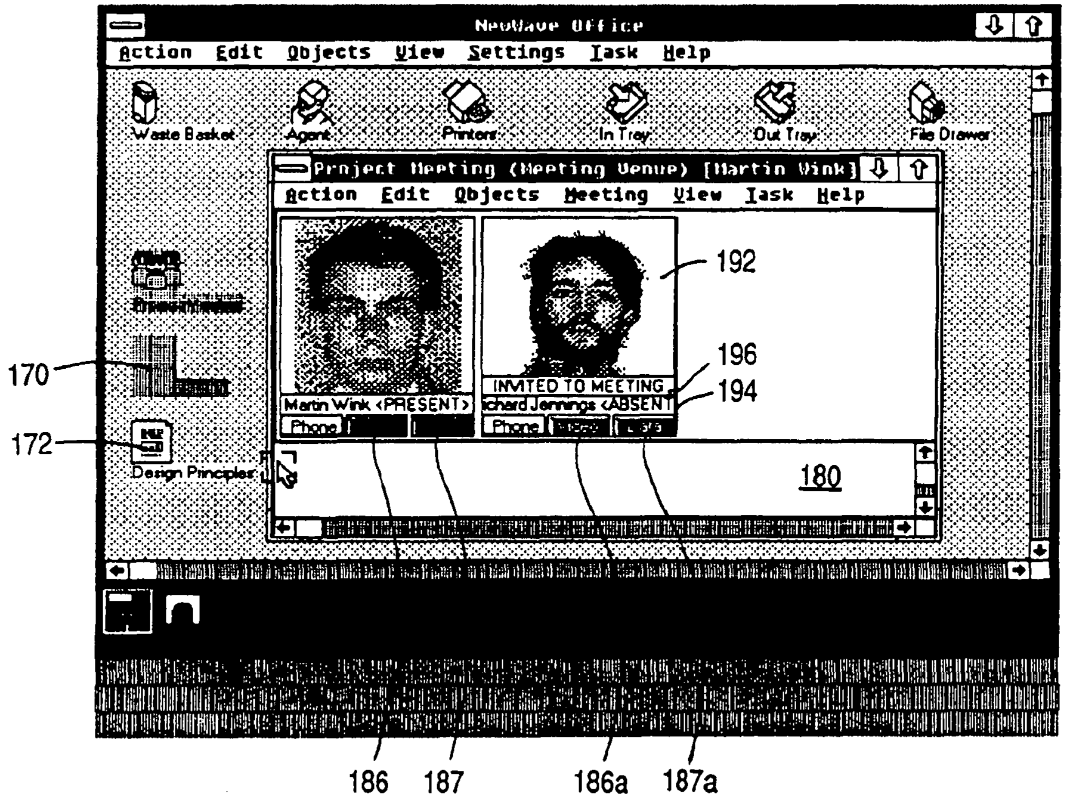

NewWave Office
Action Edit Objects View Settings Task Help
Waste Basket
Agent
Printers
In Tray
Out Tray
File Drawer
Project Meeting (Meeting Venue) [Martin Wink]
Action Edit Objects Meeting View Task Help
Martin Wink <PRESENT>
Phone
INVITED TO MEETING
ichard Jennings <ABSENT
Phone
Design Principles
170
172
192
196
194
180
186
187
186a
187a

FIG19

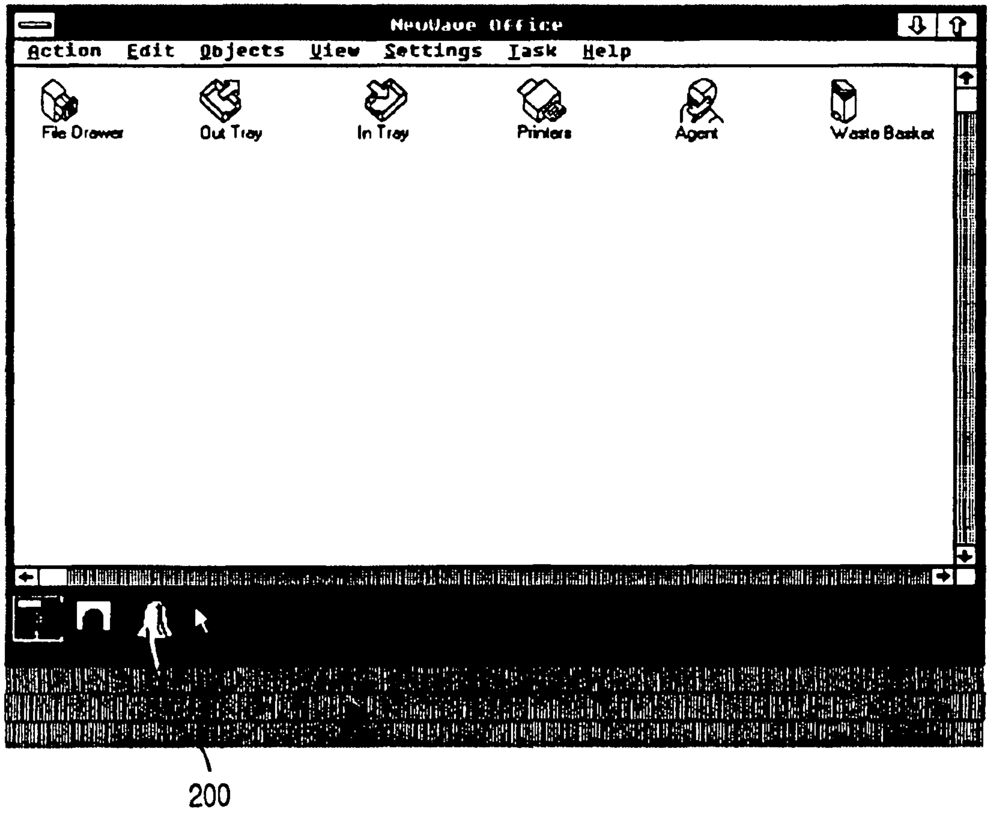
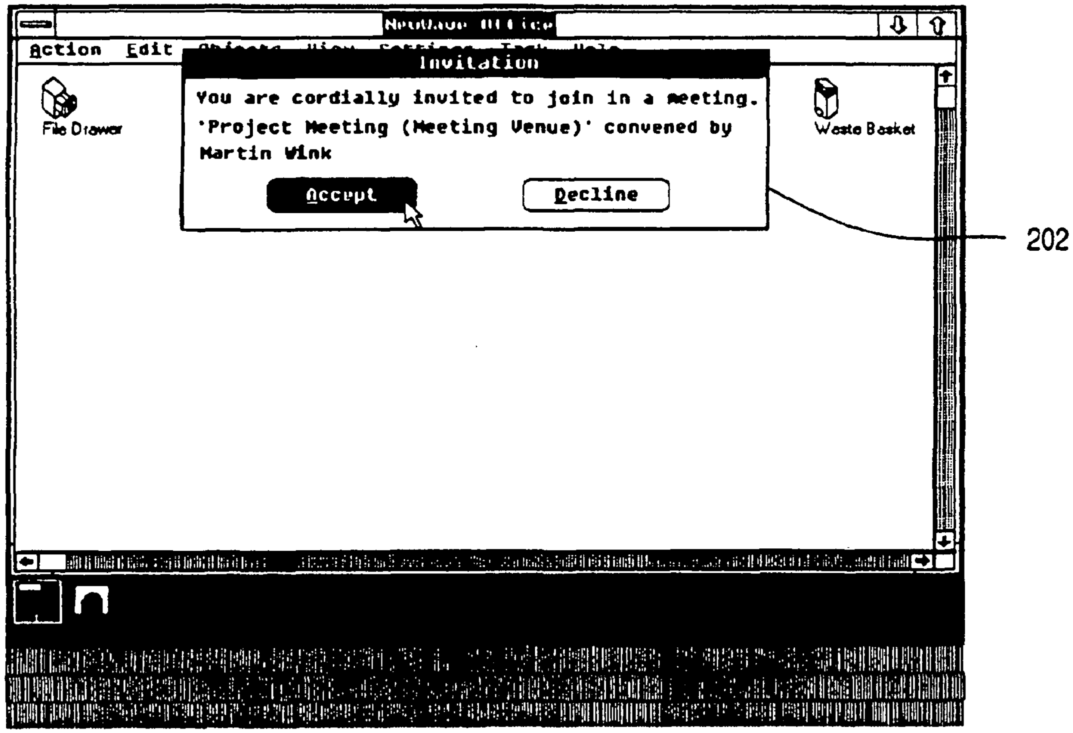

NewWave Office
Action Edit Objects View Settings Task Help
File Drawer
Out Tray
In Tray
Printers
Agent
Waste Basket
200

FIG 20



You are cordially invited to join in a meeting. 'Project Meeting (Meeting Venue)' convened by Martin Wink

FIG 21

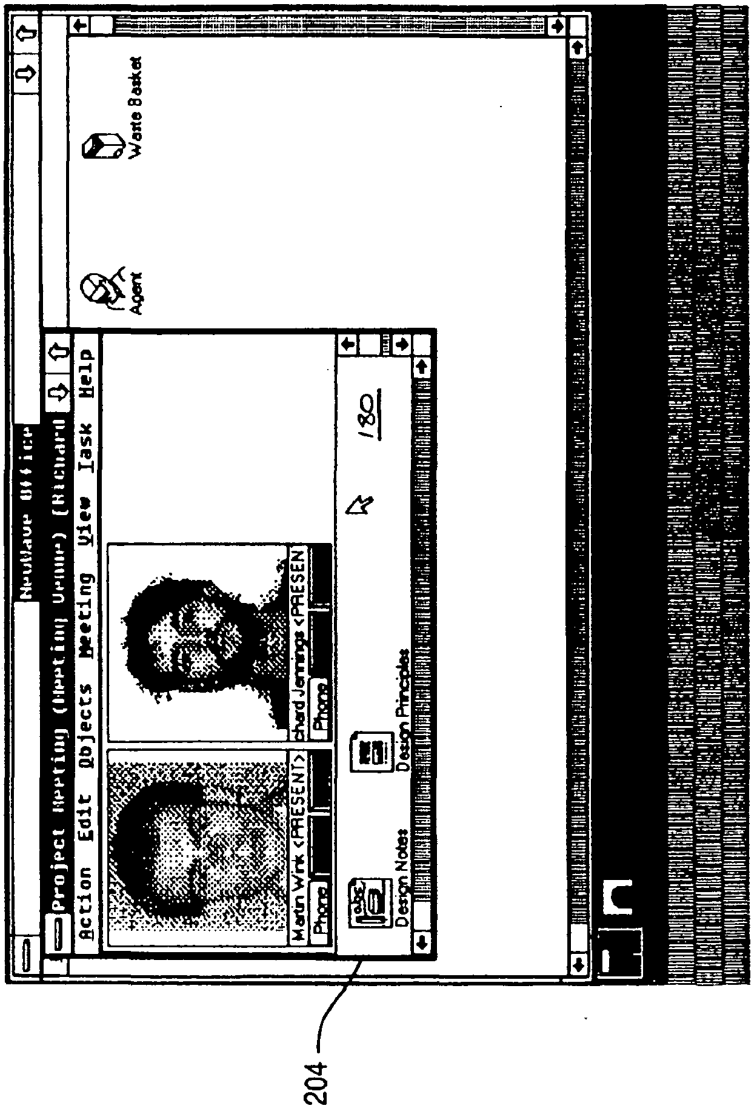

NeoWave Office
Project Meeting (Meeting Venue) (Richard
Action Edit Objects Meeting View Task Help
Martin Wink <PRESENT>
Phone
chard Jennings <PRESEN
Phone
Design Notes
Design Principles
180
Agent
Waste Basket
204

FIG 22

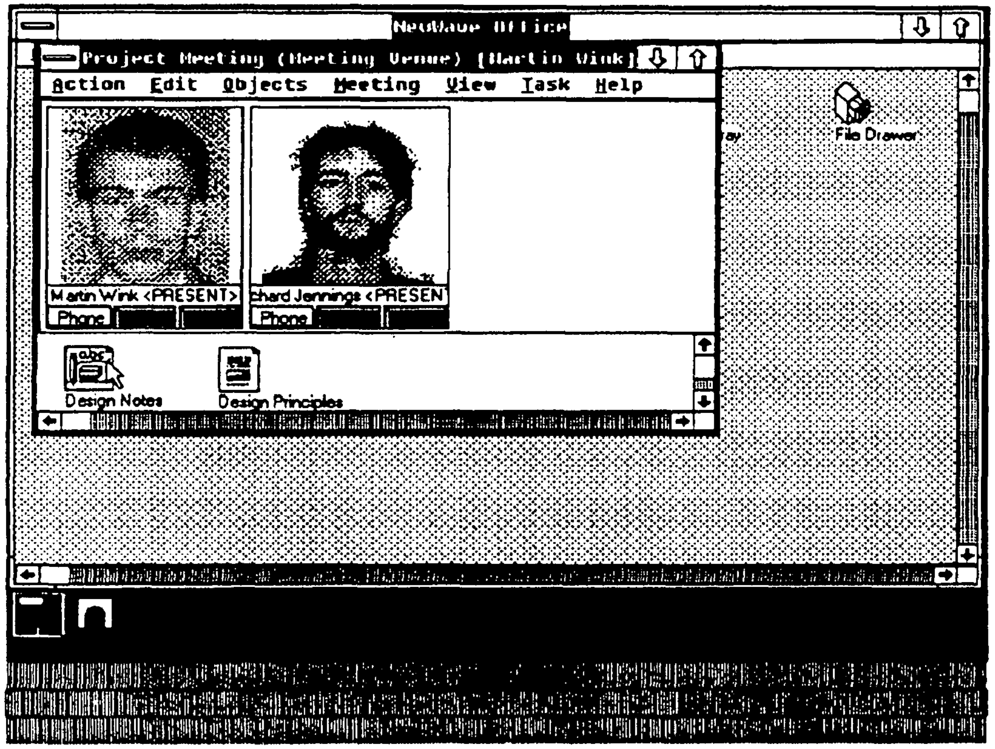
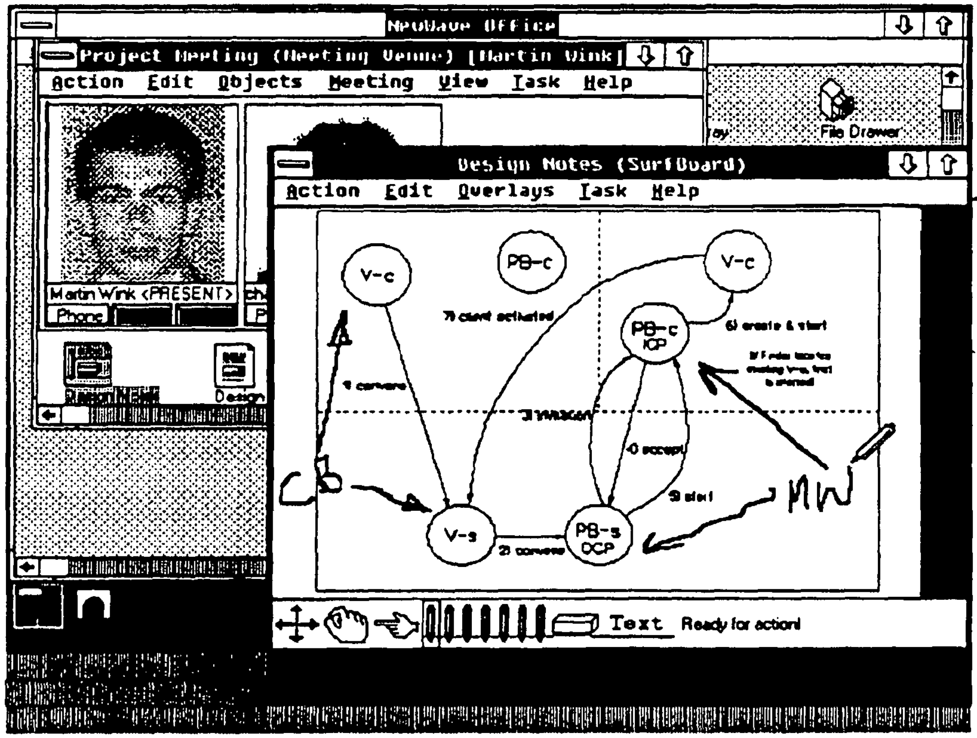

NewWave Office
Project Meeting (Meeting Venue) [Martin Wink]
Action Edit Objects Meeting View Task Help
Martin Wink <PRESENT>
Phone
chard Jennings <PRESEN
Phone
Design Notes
Design Principles
ay
File Drawer

FIG 23


NewWave Office
Project Meeting (Meeting Venue) [Martin Wink]
Action Edit Objects Meeting View Task Help
Martin Wink <PRESENT>
Phone
Design
File Drawer
Design Notes (SurfBoard)
Action Edit Overlays Task Help
V-c
PB-c
V-c
PB-c
ICP
7) client activated
6) create & start
1) convene
3) invocation
4) accept
5) start
V-s
PB-s
DCP
2) convene
CB
MW
Text
Ready for action!

206

FIG 24

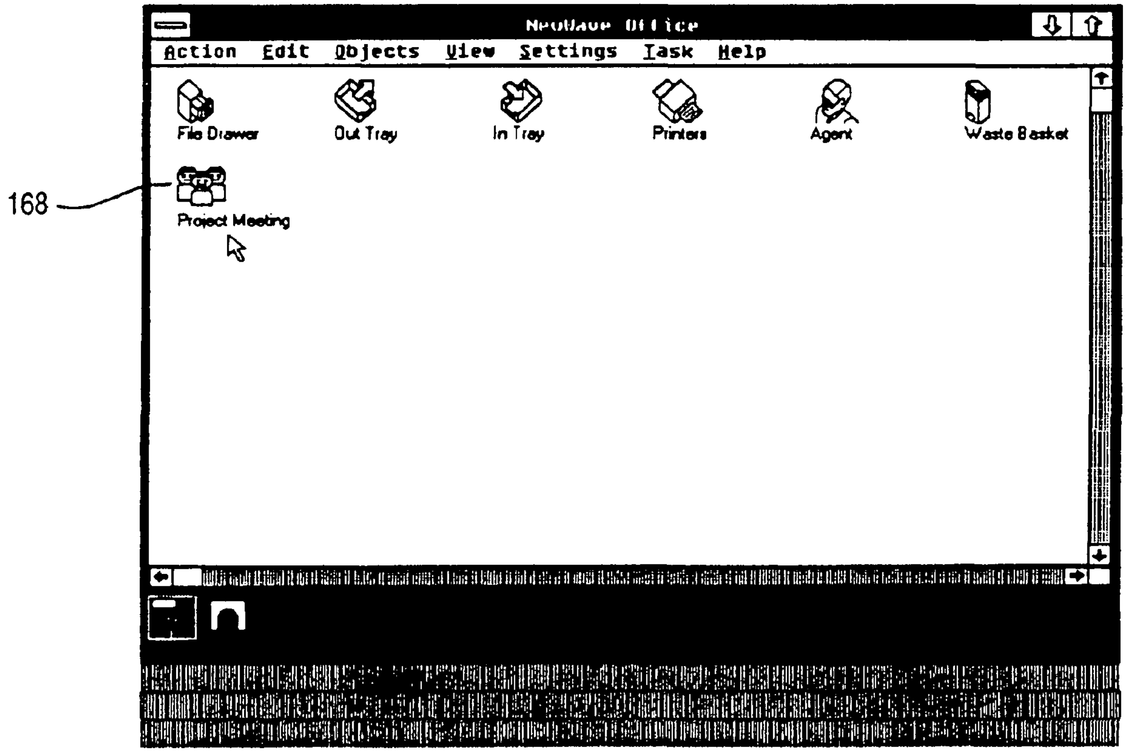

NewWave Office
Action Edit Objects View Settings Task Help
File Drawer
Out Tray
In Tray
Printers
Agent
Waste Basket
Project Meeting
168

FIG 25

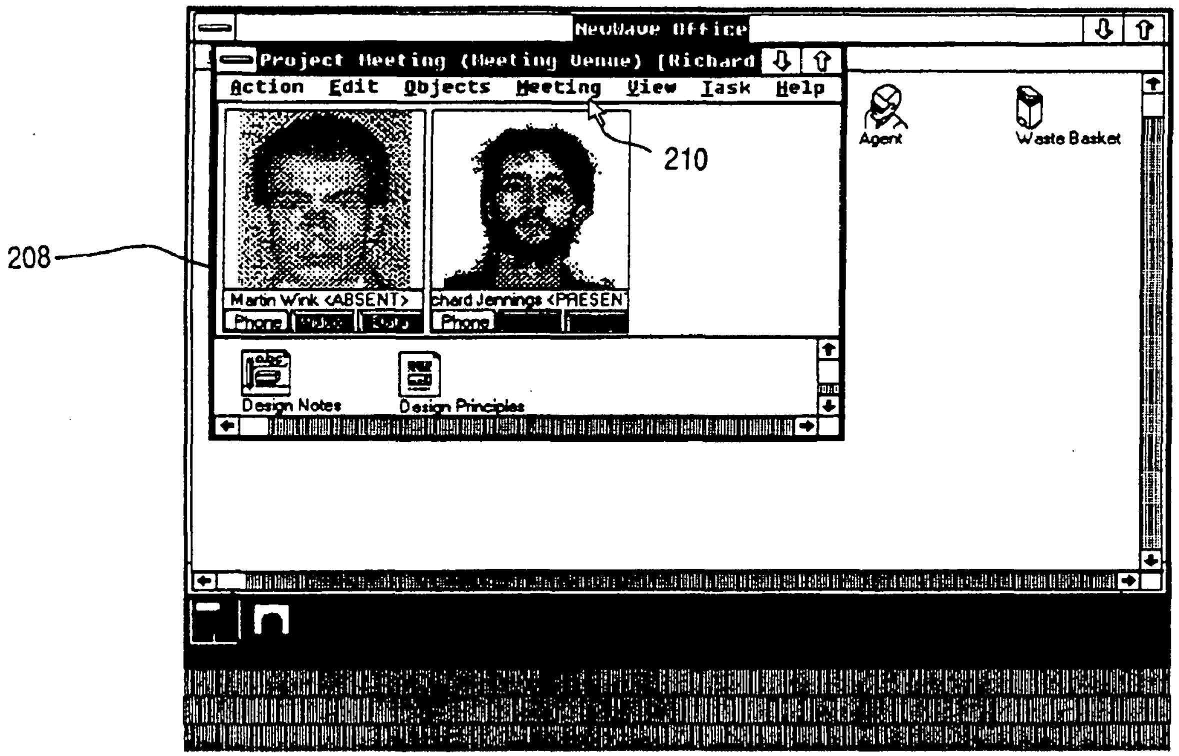
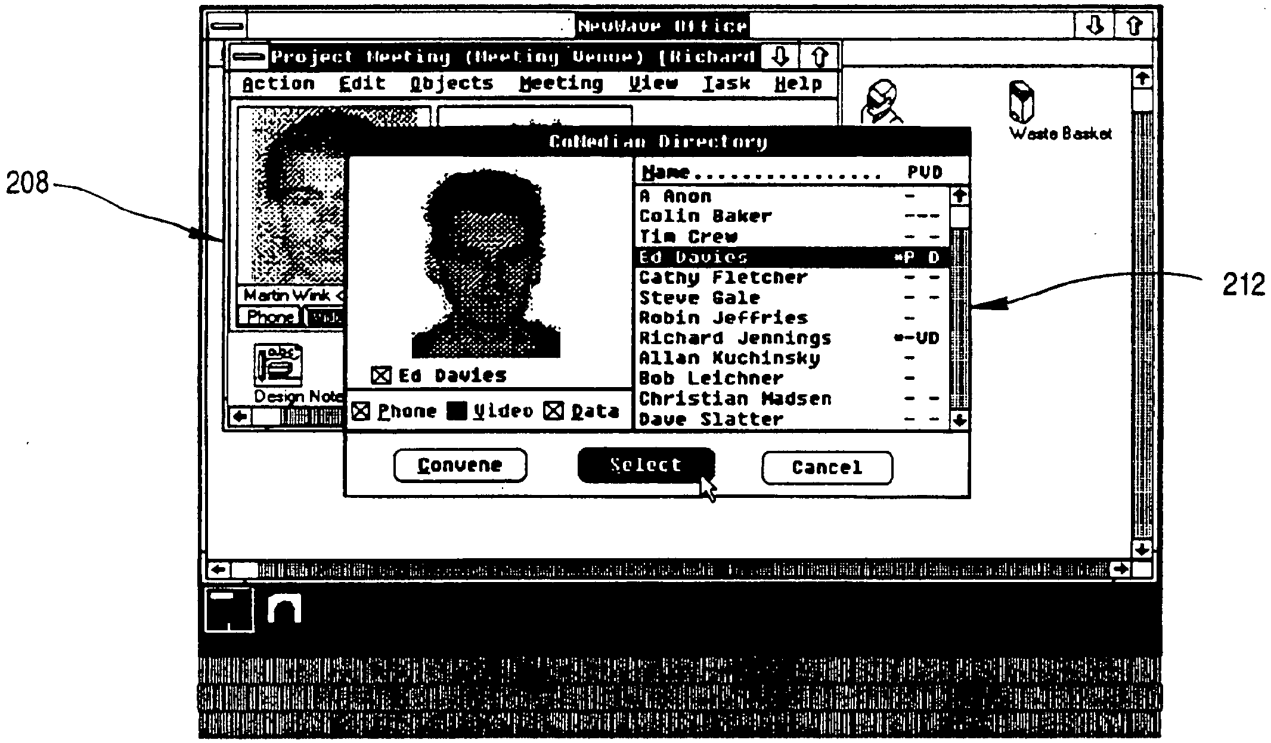

NewWave Office
Project Meeting (Meeting Venue) [Richard
Action
Edit
Objects
Meeting
View
Task
Help
Martin Wink <ABSENT>
Phone
chard Jennings <PRESEN
Phone
Design Notes
Design Principles
Agent
Waste Basket
208
210

FIG 26



☒ Ed Davies

☒ Phone ■ Video ☒ Data

Name.................	PVD
A Anon	-
Colin Baker	---
Tim Crew	- -
Ed Davies	*P D
Cathy Fletcher	- -
Steve Gale	- -
Robin Jeffries	-
Richard Jennings	*-VD
Allan Kuchinsky	-
Bob Leichner	-
Christian Madsen	- -
Dave Slatter	- -

FIG 27

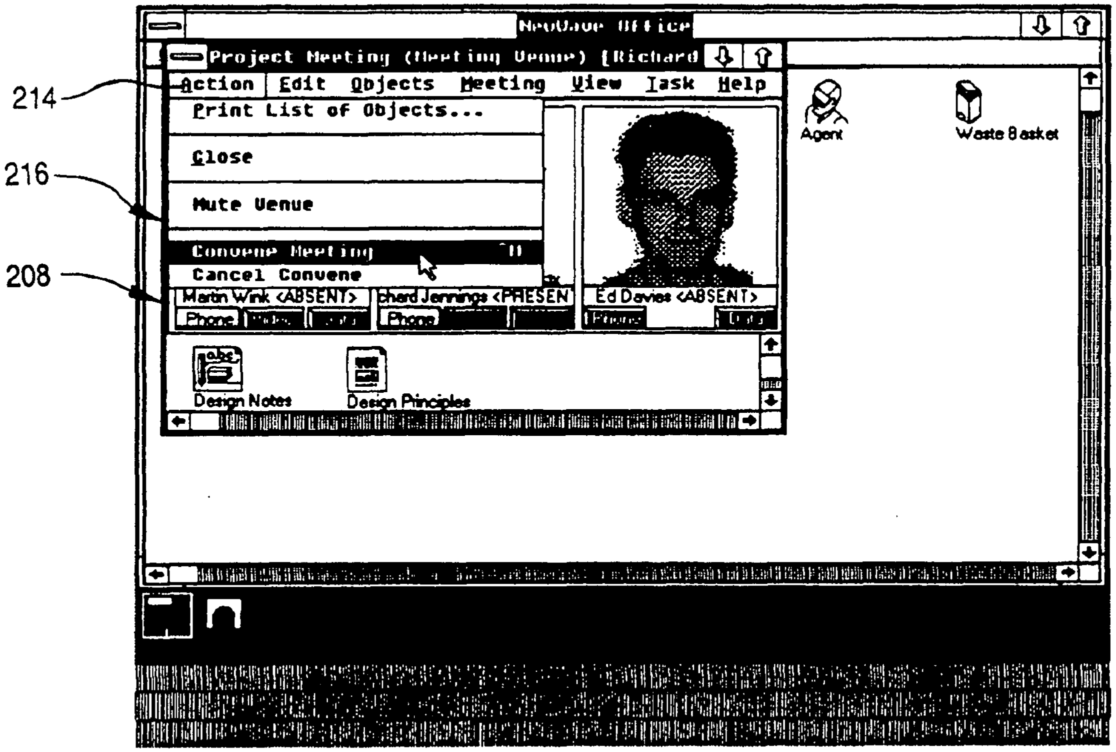

NewWave Office
Project Meeting (Meeting Venue) [Richard
Action
Edit
Objects
Meeting
View
Task
Help
Print List of Objects...
Close
Mute Venue
Convene Meeting
Cancel Convene
Martin Wink <ABSENT>
chard Jennings <PRESEN
Ed Davies <ABSENT>
Phone
Design Notes
Design Principles
Agent
Waste Basket
214
216
208

FIG 28

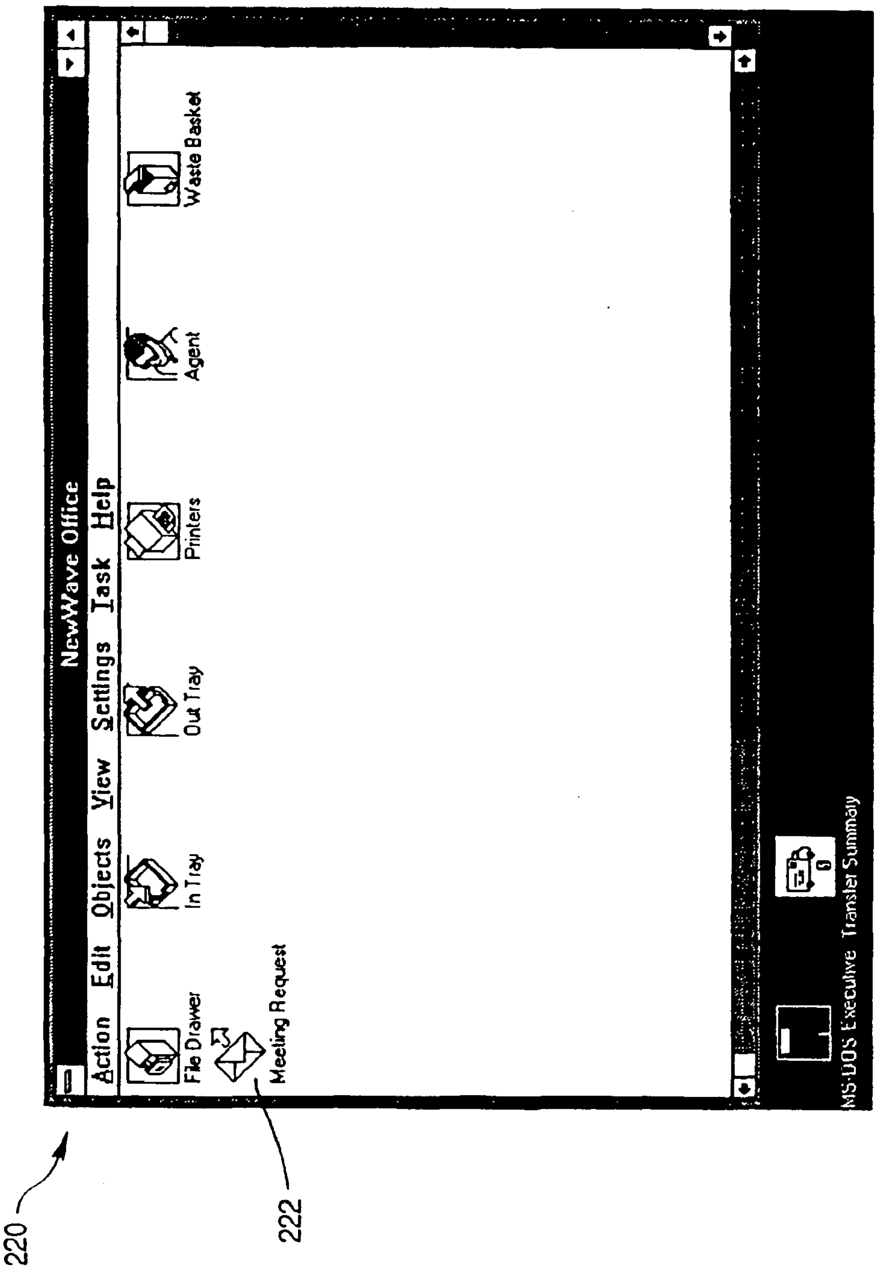
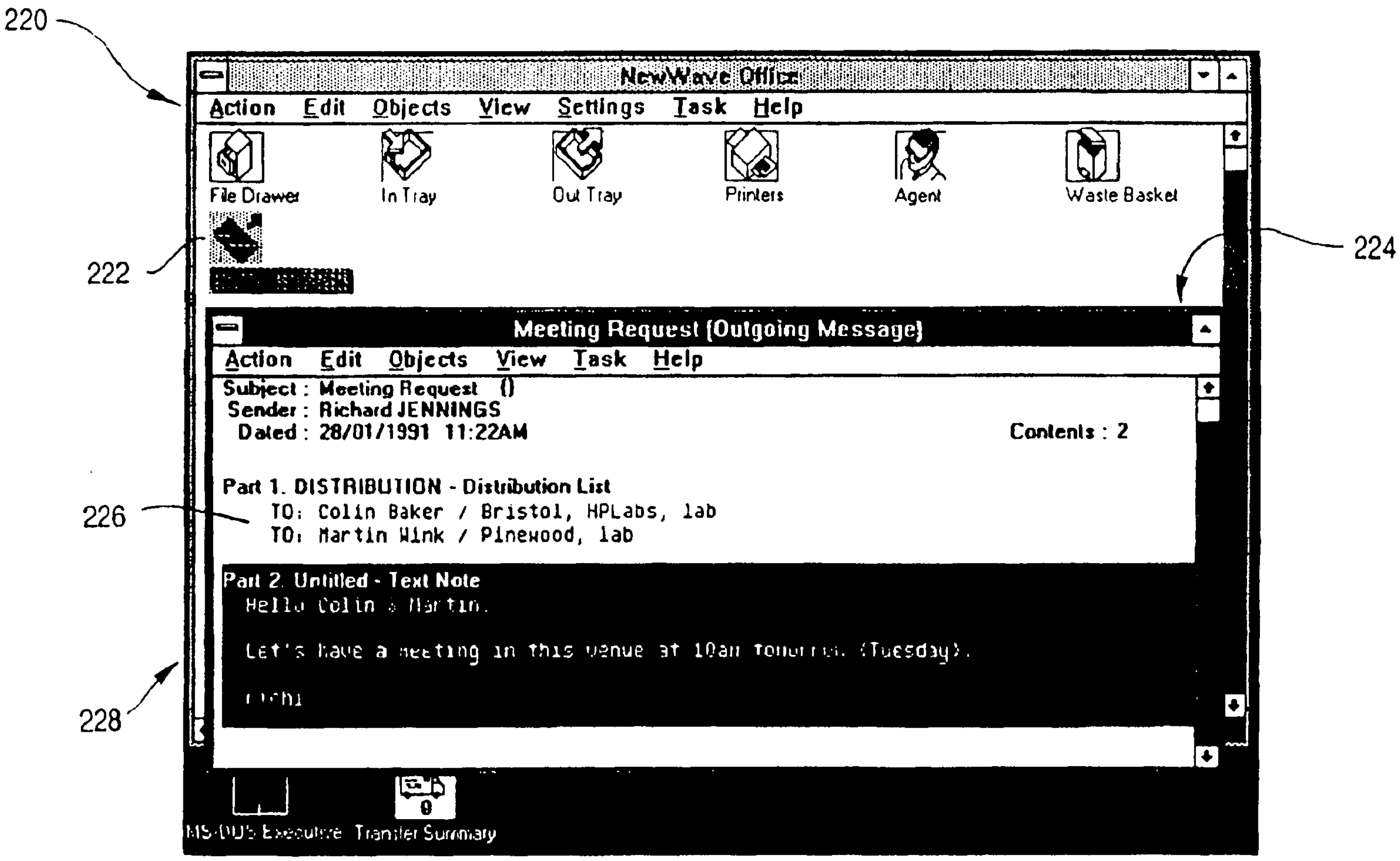

NewWave Office
Action
Edit
Objects
View
Settings
Task
Help
File Drawer
In Tray
Out Tray
Printers
Agent
Waste Basket
Meeting Request
MS-DOS Executive
Transfer Summary
220
222

FIG 29



Subject : Meeting Request ()
Sender : Richard JENNINGS
Dated : 28/01/1991 11:22AM
Contents : 2

Part 1. DISTRIBUTION - Distribution List

```
TO: Colin Baker / Bristol, HPLabs, lab
TO: Martin Wink / Pinewood, lab
```

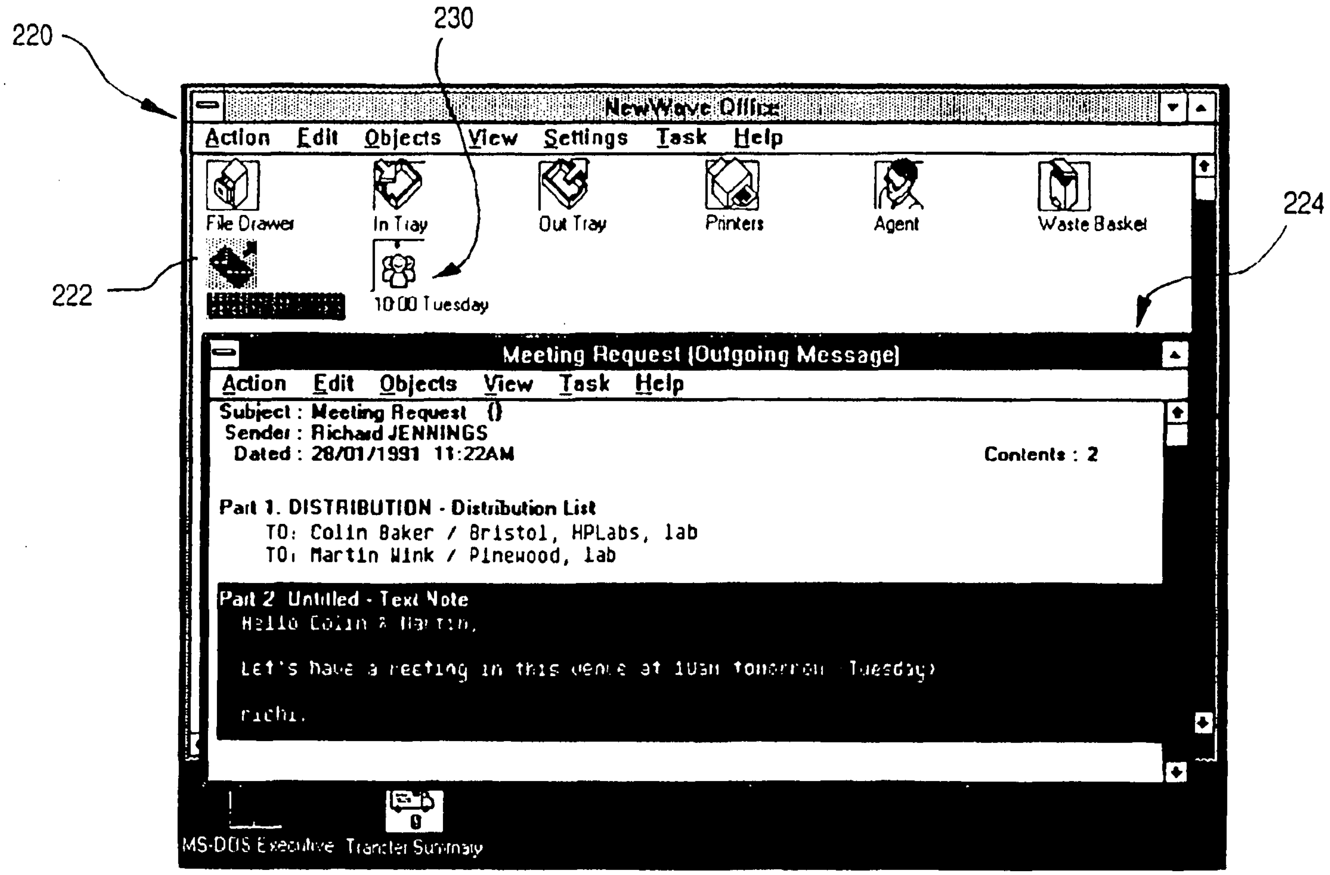
Part 2. Untitled - Text Note

```
Hello Colin & Martin.

Let's have a meeting in this venue at 10am tomorrow (Tuesday).

Richi
```

FIG 30



Subject : Meeting Request ()
Sender : Richard JENNINGS
Dated : 28/01/1991 11:22AM
Contents : 2

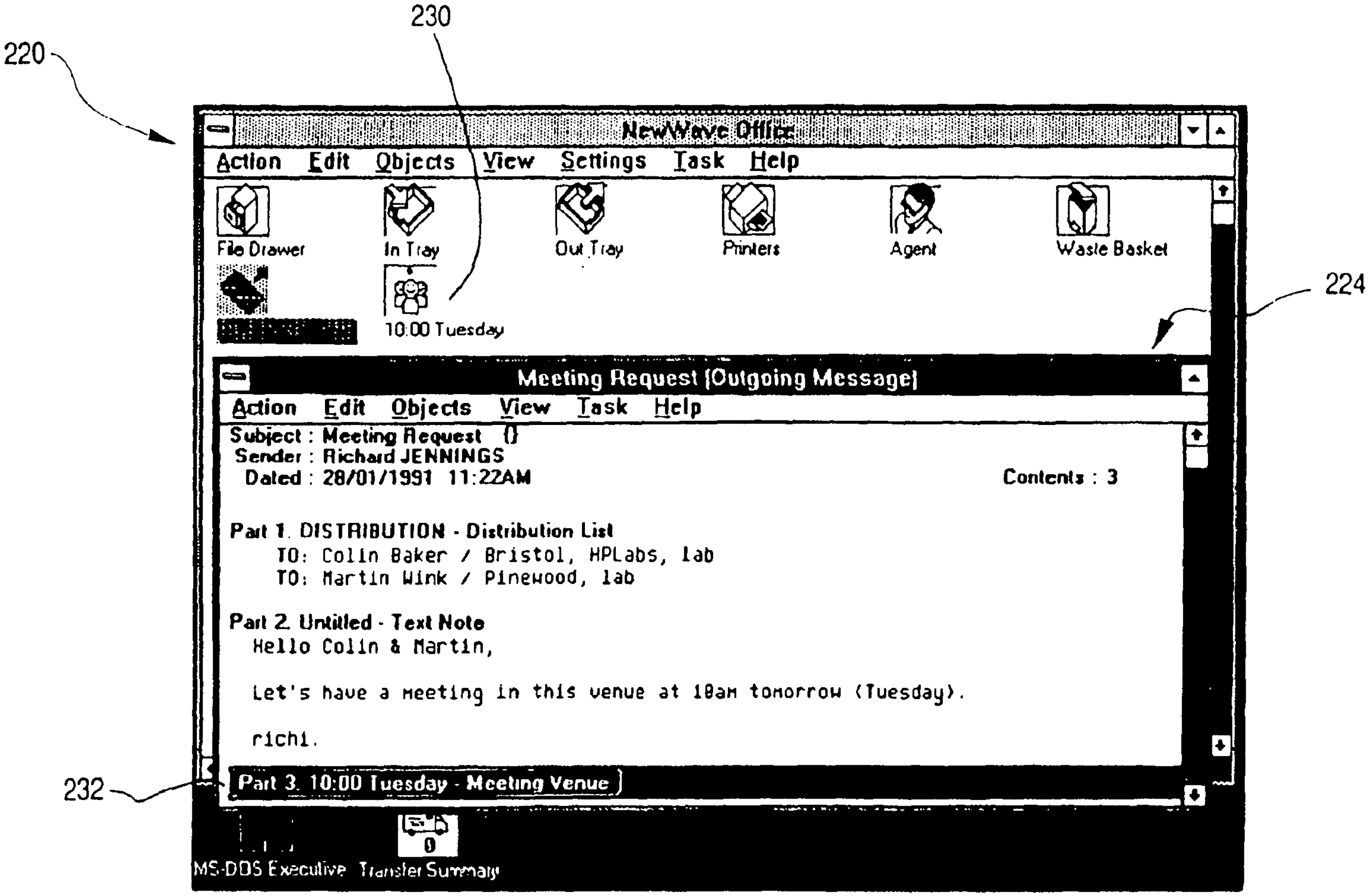
Part 1. DISTRIBUTION - Distribution List
TO: Colin Baker / Bristol, HPLabs, lab
TO: Martin Wink / Pinewood, lab

Part 2 Untitled - Text Note
Hello Colin & Martin,

Let's have a meeting in this venue at 10am tomorrow (Tuesday)

richi.

FIG 31



Subject : Meeting Request
Sender : Richard JENNINGS
Dated : 28/01/1991 11:22AM
Contents : 3

Part 1. DISTRIBUTION - Distribution List
TO: Colin Baker / Bristol, HPLabs, lab
TO: Martin Wink / Pinewood, lab

Part 2. Untitled - Text Note
Hello Colin & Martin,

Let's have a meeting in this venue at 10am tomorrow (Tuesday).

richi.

Part 3. 10:00 Tuesday - Meeting Venue

FIG 32

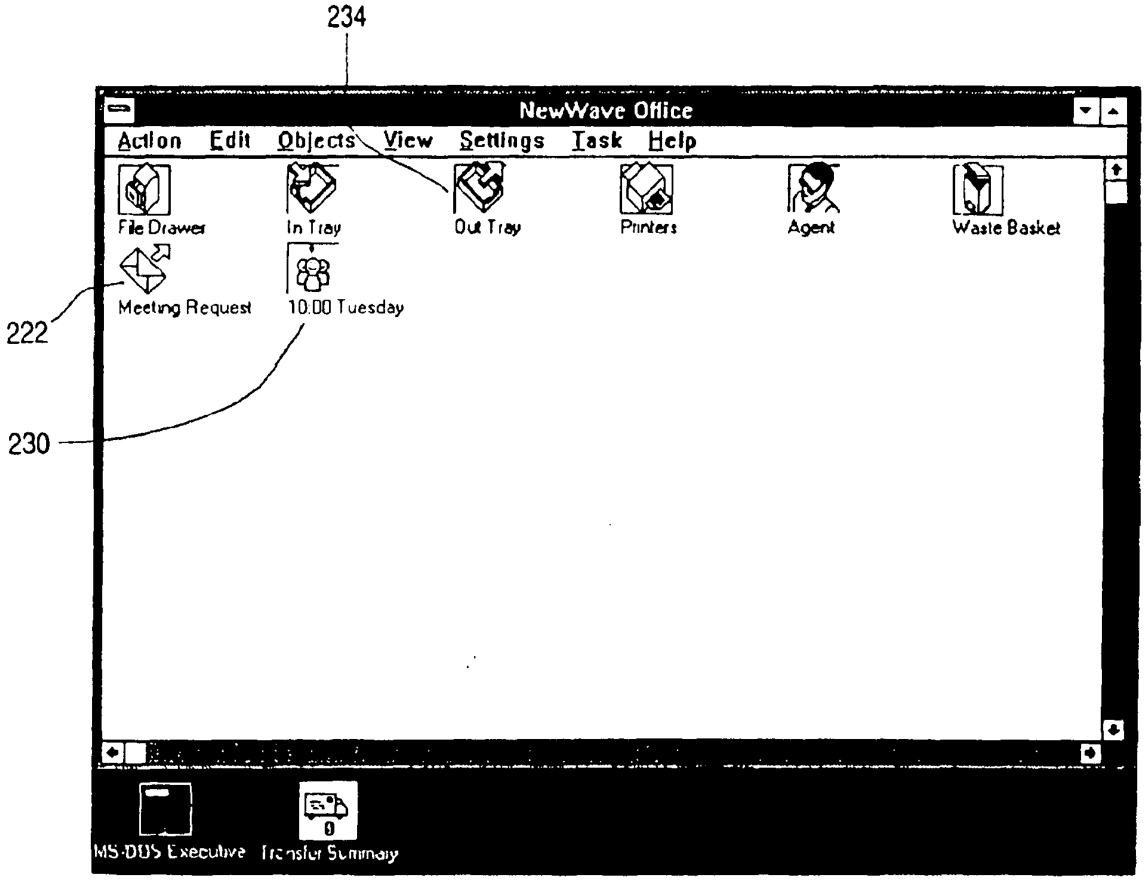

234
222
230
NewWave Office
Action Edit Objects View Settings Task Help
File Drawer
In Tray
Out Tray
Printers
Agent
Waste Basket
Meeting Request
10:00 Tuesday
MS-DOS Executive
Transfer Summary

FIG 33

European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 91 30 0772

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
X	EP-A-408 812 (HEWLETT PACKARD) * abstract * * page 3, line 29 - line 51 *	1-3	G06F15/40 G06F9/44 H04M3/56
Y	---	4,5	H04N7/14
Y	COMPUTER. vol. 18, no. 10, October 1985, SILVER SPRING, MARYLAND, US pages 33 - 45; SUNIL SARIN ET AL.: 'Computer-Based Real-Time Conferencing Systems ' * page 33, column 1, line 1 - page 38, column 1, line 44 * * page 39, column 3, line 31 - page 42, column 1, line 25 * * page 43, column 1, line 33 - page 44, column 1, line 25 * -----	4,5	

TECHNICAL FIELDS SEARCHED (Int. Cl.5)
G06F H04N H04M

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	18 SEPTEMBER 1991	KINGMA Y.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 581 722 A1**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : 93630052.4

(22) Date of filing : 19.07.93

(51) Int. Cl.^{5} : **H04L 12/58**

(30) Priority : 30.07.92 US 922314

(43) Date of publication of application :
02.02.94 Bulletin 94/05

(84) Designated Contracting States :
CH DE ES FR GB IT LI NL SE

(71) Applicant : YEDA RESEARCH AND DEVELOPMENT COMPANY, LTD.
The Weizmann Institute of Science
76 100 Rehovot (IL)

(72) Inventor : Shapiro, Ehud
Meonot Wolfson, Weizmann Institute of Science
Rehovot 76100 (IL)

(74) Representative : Waxweiler, Jean et al
OFFICE DENNEMEYER & ASSOCIATES Sàrl,
P.O. Box 1502
L-1015 Luxembourg (LU)

(54) A method for establishing an interactive communication between users at different workstations in a network.

(57) A method for establishing an interactive communication between at least first and second workstations in a computer network system having a communication protocol for despatching messages between different workstations and being further adapted to exchange batch messages by means of an electronic mail program stored in each of the workstations. The batch messages are categorized such that a batch message of a predetermined category informs a receiving workstation that a sending workstation wishes to establish an interactive communication between a specified first logical port in the sending workstation and a specified second logical port in the receiving station. A batch message of the predetermined category having therein a reference to the first logical port is sent from the first workstation to the second workstation so as to be received thereby and is stored in a storage device containing a list of batch messages. Upon noting the presence in the storage device of a batch message of the predetermined category, the communication protocol is utilized to send an initiation signal from the second logical port in the second workstation to the first logical port in the first workstation. Upon receipt of the initiation signal, an interactive two-way communication is established between the first logical port of the first workstation and the second logical port of the second workstation.

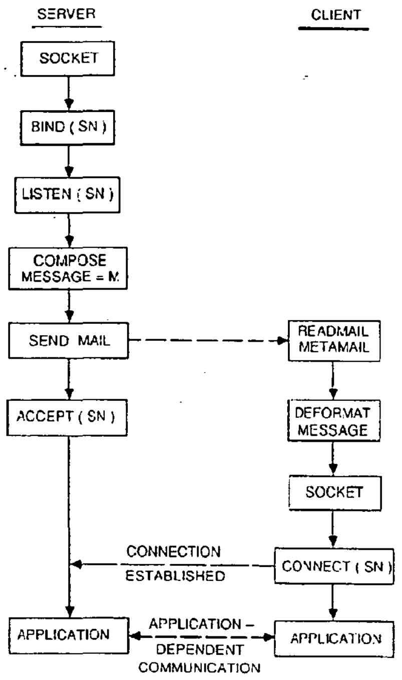


Fig.5

Jouve, 18, rue Saint-Denis, 75001 PARIS

FIELD OF THE INVENTION

The invention relates to the use of "electronic mail" for affording message exchange between computer users. In particular, the invention permits communication to be established between users at different terminals connected to a mini-computer or a main frame operating under a multi-tasking operating system, or between users at workstations or network stations (X-terminals) connected by a communication network. It should be noted that throughout the specification and claims, the term "workstation" is used generally to denote any device for communicating interactively with a computer and including a display monitor and an input device such as a keyboard, mouse and so on.

BACKGROUND OF THE INVENTION

In recent years, there has been a constant growth of computer network installations. This, coupled with the spread of minicomputers and main frames connected to a plurality of terminals, have encouraged the use of network orientated applications. A typical such application is "electronic mail", which enables at least two users to exchange messages.

Several standards, such as "X.400" for OSI or "Multipurpose Internet Mail Extension" (MIME) for "Internet" have been defined to permit message exchange of multiple data types. The messages may consist of different authorized types, such as computer-executable files, computer program-processable files (such as spreadsheets), audio and video sequences, or a combination thereof, as in the case of multi-media.

The electronic mail methods used nowadays are of off-line or batched character, whereby a message, e.g. a text file, is sent from User A to User B and stored in User B's data storage device, usually referred to as a "mailbox". User B, at his earliest convenience which may, of course, be some considerable time later, accesses his mailbox to read any pending messages, and so finds User A's text file. He may then respond by sending an acknowledgement message to User A. This simplified protocol demonstrates the off-line character of the connection.

It is possible to re-configure the computer prompt appearing on the screen of User B's computer at the moment the message arrives, whereby User B is provided with immediate feedback that a message is waiting for him. This, in turn, permits User B to generate an immediate response to User A. Nevertheless, the batched character of the communication is retained, since User B's response resides in User A's mailbox, and in order to retrieve it, User A must invoke a series of instructions, including entering the mailbox and selecting therefrom the message whose contents are to be displayed. If he wishes to answer it immediately, he must prepare a message of a type supported by the electronic mail program, and send the message through the network to User B.

It would clearly be preferable to invoke an interactive session in which a message sent by User A is immediately displayed on User B's screen for direct response by User B. Consider, for example, a firm in which each employee uses a PC all of which are interconnected by a standard network. Suppose an employee (User A) sends a draft of an important letter to his boss (User B). The boss, after reviewing the received draft, wishes to establish an interactive communication with User A, and optionally to involve in the discussion the department manager (User C). Obviously, a copy of User A's draft is sent to User C, so as to permit a discussion to be conducted between the three participants.

Alternatively, User A may be replaced by a "groupware application" running on a server. Such a groupware application is shared by several users as in the case of a document edited simultaneously by two or more authors. Conventional groupware applications permit each of the authors to effect simultaneous editing of the document whilst being logged into different workstations. However, suppose that during the course of editing, it is required to involve an additional author who is a specialist in a certain topic discussed in the groupware document. In such case, it is required to establish an interactive session between the groupware application (running on one computer) and the specialist user who is generally logged into a different workstation.

It is clear that the above requirements cannot be realized in currently available electronic mailing systems which, as explained above, are not interactive.

In contrast to batch-type electronic mailing systems, it is also known to provide an interactive on-line communication between users across a computer network. Thus, for computers operating under the UNIX operating system, there is provided a facility "TALK" whereby such interactive communication may be achieved between several users. However, "TALK" and other similar interactive communication methods are intrusive since only the user who establishes the communication has control as to when the communication is to be established, whilst all of the remaining users are likely to be disturbed during the performance of other tasks. Thus, typically, if User A invokes the "TALK" facility in order to initiate an interactive communication with User B, a message flashes on the screen of User B's workstation in order to inform him that User A wishes to establish a communication. If User B is otherwise preoccupied and ignores the message, it will reappear at regular intervals until finally User B also invokes the "TALK" facility in order to communicate with User A. Until such communication is established, the constant reappearance of warning-type messages on the screen of User B's workstation is inevitably intrusive and can be most irritating to User B.

Furthermore, if User B repeatedly ignores the messages which appear on his screen advising him of User A's desire to establish an interactive dialogue and User A responds in kind, by exiting from the "TALK" facility, no trace of the previous attempts to establish such dialogue is left in User A's workstation. Thus, if and when User B is eventually free to respond to the message originally dispatched by User A, there exists no trace on User A's workstation of his original attempts to establish such communication and it is thus now User B, and no longer User A, who must initiate the communication.

Additionally, facilities such as "TALK" are intended only for invoking interactive communications between users working at respective workstations and have no provision for establishing such communication between a user and an application or between two applications such as, for example, a groupware document being edited simultaneously by different users at respective workstations.

BRIEF SUMMARY OF THE INVENTION

It is an object of the invention to provide a method for establishing an interactive dialogue between two or more workstations in such a manner as to preserve the non-intrusive character of batch-type electronic mail systems, whilst nevertheless permitting an interactive multi-way communication to be conducted between the participants.

According to the invention there is provided for use in a computer network system comprising at least first and second workstations adapted to send and receive messages by utilizing a suitable communication protocol and further adapted to exchange batch messages by means of an electronic mail program stored in each of said at least first and second workstations;

a method for establishing an interactive communication between said at least first and second workstations, said method characterized by the steps of:

(i) categorizing said batch messages such that a batch message of a predetermined category informs a receiving workstation that a sending workstation wishes to establish an interactive communication between a specified first logical port in the sending workstation and a specified second logical port in the receiving workstation;

(ii) sending a batch message of the predetermined category having therein a reference to said first logical port from the first workstation to the second workstation so as to be received thereby and stored in storage means containing a list of batch messages;

(iii) monitoring at the second workstation all batch messages in said storage means at specified periods of time;

(iv) noting the presence in said storage means of a batch message of said predetermined category;

(v) utilizing the communication protocol to send an initiation signal from the second logical port in the second workstation to the first logical port in the first workstation; and

(vi) responsive to receipt of the initiation signal, establishing an interactive two-way communication between the first logical port of the first workstation and the second logical port of the second workstation.

In accordance with such a method, the message may be sent directly from an application running in the first workstation for subsequent storage in the mailbox in the second workstation. Upon scanning the mailbox in the second workstation, the user finds a message of the predetermined category, indicating that another user or application on the network wishes to establish a two-way interactive communication with him.

In normal batch-orientated electronic mail systems, each message in a user's mailbox has a corresponding title, by means of which the awaiting message can be identified. It is preferable to embed within the title of the awaiting message some indication that the message is adapted for establishing a two-way interactive communication with a sending workstation. Alternatively, this fact may not be apparent from the title of the message itself, in which case the receiving workstation will not afford the awaiting message any special priority, although the very act of reading the message will invoke the required interactive communication.

According to a preferred embodiment, the method is used in order to establish an interactive communication between a groupware application running on the first workstation and at least one user working at a second workstation who accesses the groupware application via a suitable interface window. In such a system, the interface window at the second workstation is associated with the second logical port thereof so that when the desired two-way interactive communication is established between the first and second logical ports of the first and second workstations, respectively, the groupware application running on the first workstation will interact directly with the user via the interface window of the second workstation.

BRIEF DESCRIPTION OF THE DRAWINGS

For a clearer understanding of the invention and to see how the same may be carried out in practice, some preferred embodiments will now be described, by way of non-limiting example only, with reference to the accompanying drawings, in which:

Figs. 1a and 1b are block diagrams showing functionally a computer network system and a detail of a workstation thereof for implementing an electronic mail method according to the invention;

Fig. 2 is a simplified flow diagram showing the principal operating steps associated with a sending workstation in the system shown in Fig. 1;

Fig. 3 is a simplified flow diagram showing the principal operating steps associated with a receiving workstation in the system shown in Fig. 1;

Fig. 4 is a flow diagram showing in somewhat greater detail the operating steps shown in Fig. 3; and

Fig. 5 is a composite flow diagram showing the principal operating steps of the system depicted in Fig. 1 operating under UNIX and utilizing the Internet communication protocol.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Figs. 1a and 1b show a computer network system 10 comprising a server 11 coupled, via a communication network depicted generally as 12 to a plurality of client computers 13 to 17, respectively. Each of the computers 13 to 17 constitutes a workstation associated with which is a storage device 19 and a display device 20. Typically, each of the computers 13 to 17 is adapted to run several tasks simultaneously, data associated with each task being displayed on the display device 20 in a corresponding window 21 thereof.

In accordance with a preferred embodiment, the communication network 12 utilizes the Internet standard as a communication protocol and further utilizes the so-called "Multipurpose Internet Mail Extension" (MIME) for the Internet standard. MIME provides means for exchanging messages between users in an Internet-orientated communication network. The messages may be one of several different categories such as, for example, computer executable files, computer program processable files (such as spreadsheets), audio and video sequences or a combination thereof. The MIME standard permits the definition of a application's specific category. This feature is exploited by the invention for defining a unique category to represent an electronic mail message which is configured to establish a two-way interactive communication between the server 11 and one or more of the client computers 13 to 17.

Typical mail exchange under the MIME standard between computers inter-connected in a network as shown in Fig. 1a is implemented as follows. The server 11 as well as each of the client computers 13 to 17 has access to a stored mail reader program. Upon receipt of a mail message sent from a user of one of the client computers, the received message is stored in a so-called "mailbox" in the storage device 19 associated with the receiving computer. The user of the receiving computer reads the stored message on the display device 20, and by doing so he activates a mail reader program which, in turn, calls a metamail program which assumes that the messages stored in the mailbox are in a format which conforms to the MIME standard.

Once the user has selected the desired mail message to be read, the metamail program accesses the desired message and retrieves therefrom the message category. It will be recalled that the message category typically defines the type of data associated with the transmitted message.

A list of authorized categories is stored in a database file designated as "mailcap" which instructs the metamail program what action to perform with respect to each of the authorized categories. Thus, if the message is a text file category, then a suitable entry in the mailcap file may cause a standard text editor to be invoked on the user's screen. If, on the other hand, the authorized category denotes that the message is an audio file, then a suitable entry within the mailcap file will specify an audio program which can play back the audio message.

In accordance with the invention, a new authorized category is stored in the mailcap denoting that a batched message associated therewith serves the purpose of establishing an interactive communication between a specified first logical port in the sending workstation and a window 21 in the receiving workstation bound to a specified second logical port. In the description which follows the new, authorized category will be denoted by the term "Active Mail". Associated with the Active Mail category is a procedure which performs a series of steps for establishing an interactive communication between the first and second logical ports in the sending and receiving workstations, respectively.

Referring now to Figs. 2 and 3 of the drawings, there are shown simplified flow diagrams relating to the principal operating steps associated with the establishment of an interactive communication by means of electronic mail in accordance with the modified MIME standard. Throughout the following explanation, it will be assumed that a groupware application running on the server 11 wishes to establish an interactive communication with the client computer 13. In a manner that will be described in detail below, the network address of the logical port in the server 11 is encoded and embedded in a suitably constructed mail message which further includes the message category.

Thus in accordance with the terminology introduced above, the message is categorized as "Active Mail" and specifies the name of the sending socket. The thus encoded message is then sent to the client computer 13 where it is stored in the mailbox thereof. Thereafter, the user working at the client computer 13 scans his mailbox and reads the awaiting message using the electronic mail program in the normal way. The awaiting message may, or may not, be flagged as being of type Active Mail in the title by means of which it is identified in the mailbox, thereby prompting the user to read the message contents with some urgency.

In any event, on reading the awaiting message, the message category is decoded and the mailcap is accessed in order to determine the operating instruction which must now be invoked responsive to a message category of the decoded type.

Referring to Fig. 4, this step will now be elaborated on assuming that the relevant instructions stored in the mailcap is a program called "am connect".

Initially, the "am connect" program decodes the network address of the logical port in the server 11 to which a connection is to be established and which is embedded in the received message. Thereafter, a suitable logical port is defined in the receiving workstation for communicating with the decoded network address of the logical port in the server 11. This is followed by a series of operating system primitives which open connection between the two network addresses of the respective logical ports.

Upon completion of the open communication, there exists a bi-directional communication channel between the respective network addresses of logical port in the server 11 and that in the client workstation 13. The "am connect" program may now open a window on the screen of the client workstation 13 for displaying and mediating an interactive communication between the user of client workstation 13 and the groupware application running on the server 11. Obviously, the application which mediates between the user of client workstation 13 and the groupware application running on the server 11 may independently conduct the interactive communication once the communication channel has been established. In such case, the "am connect" program may terminate upon establishment of a successful connection.

Referring now to Fig. 5, the situation described above generally with respect to Figs. 2 to 4 of the drawings will be described with particular reference to the UNIX operating system and with regard to a communication protocol and an electronic mail system which conform with the "Internet" and "MIME" standard, respectively. Suitable program modules written in the computer programming language "C" for carrying out the routines shown functionally in Fig. 5 are included in an Appendix hereto.

Before describing how an interactive communication is established using a suitably modified electronic mail program operating under UNIX, a further consideration should be understood. In UNIX, the logical ports are implemented using the so-called "socket" mechanism. The socket mechanism enables a workstation having one global Internet address to support a plurality of tasks. In order to distinguish between the various tasks running on the same workstation, a unique global identifier is associated with each of the running tasks so as to render the task identifiable by other tasks running on the network. Thus, a message which is sent to a specific task in a workstation has to encode both the physical destination address of the receiving workstation, as well as the logical port ("socket") which identifies the application running thereon.

In other words, since the UNIX operating system is a multi-tasking environment, it is not enough to define only the destination address of the receiving computer: a logical port associated with a specific task or application must also be specified. The combination of the physical address and the logical port is referred to as a "network address of the logical port". In the terminology of UNIX and Internet, a network address of the logical port is referred to as a "socket name".

The following description assumes that communication between two workstations employs the Internet "stream" communication protocol. However, it will be apparent that other protocols may equally well be employed such as, for example, the Internet "datagram" protocol and so on.

It should further be understood that, in accordance with the UNIX operating system, once a connection is established between respective sockets in different workstations, there exists a logical bidirectional connection between the sockets, whereupon the application which interacts through the respective socket may refer thereto as if it were a standard output or input stream. Thus, if a connection is established between a first socket in a first workstation and a second socket in a second workstation, the application associated with the first socket may interact with the application associated with the second socket simply by invoking "WRITE" or "READ" instructions. The underlying communication layers in the operating system structure will take care to ensure that the message is properly routed to the required destination.

It should also be added, for the sake of completeness, that the socket mechanism is well known to those versed in the UNIX operating system and is therefore not discussed in greater detail.

In the description which follows, whenever a service supported by the UNIX operating system is invoked, the service name will be symbolically indicated, omitting, for the sake of simplicity, reference to any parameters transferred to the service or received therefrom. The precise syntax for calling the UNIX services is familiar to those skilled in the art and likewise, since the services themselves are not a specific feature of the present invention, a detailed description thereof is unnecessary.

At the outset, the groupware application running on the server 11 invokes the "socket" primitive supplied by the operating system in order to obtain a socket number which will be associated therewith. This having been achieved, it is required to bind the socket number to a socket name (SN) and this is achieved by the "bind" service which is fed with the socket number obtained as a parameter in the previous stage. The "bind" service performs a series of steps, some of which are responsible for the binding of the logical socket to the global Internet address. To this end, the standing groupware application has a defined logical port or socket bound to which is a global Internet address through which communication with another workstation in the network may be established.

Thereafter, the "listen" service is called, this being responsible for controlling the number of simultaneous communication acknowledgements that the server 11 can handle. Upon completion of the initialization phase, the groupware application running on the server 11 prepares an Active Mail message which conforms to the MIME standard. The message is categorized as "Active Mail" and has embedded therein the encoded socket name obtained by the previous step.

The message is now sent across the network to its destination, i.e. client workstation 13. The server 11 now performs the "accept" service which, when invoked, permits the server 11 to receive communication requests addressed to a specific socket name so as to establish communication with a calling workstation.

Meanwhile, at the client workstation 13, the user activates the Read Mail program for accessing his mailbox. Upon selecting the message sent from the server 11, the "Metamail" service is activated which assumes that the message conforms to the MIME standard and retrieves therefrom the message category, i.e. "Active Mail".

The thus decoded message category is cross-referenced in the mailcap file in order to determine which service is to be invoked responsive to a message of category Active Mail.

In this case, it is assumed that the mailcap file includes an entry which specifies that the Active Mail category corresponds to the service "am connect". As a result, the "am connect" program is invoked which performs several steps.

First, the encoded socket number embedded in the message is decoded. Thereafter, the "socket" service is invoked in order to obtain a socket number in client workstation 13 which will be connected to the pre-defined socket of the server 11. The "connect" service is now called whereby a bi-directional communication channel between the two respective sockets is established.

From the perspective of the "connect" service, the originating computer is the client workstation 13 and the destination computer is the server 11. Since the server 11 is in the "Accept" status awaiting a communication request for coupling a remote workstation to the same socket of the server 11 as is now requested by the "connect" service, the required bi-directional connection is now established. At this stage, a window is opened on the user's screen of the client workstation 13, whereby the application in the client workstation 13 mediates between the underlying socket connection and the thus-defined window. This gives rise to application-dependent communication between the groupware application running on the server 11 and the mediating application in the client workstation 13, whereby the user of the client workstation 13 may interactively communicate with the groupware application running on the server 11 through the corresponding sockets in the server 11 and the client workstation 13.

In the case, as described above, where a groupware application initiates the communication so as to permit multiple, simultaneous processing thereof by a plurality of independent users, there is an implicit assumption that the server, on which the groupware application is loaded, is logged on or active when the user at the second workstation reads the appropriate mailbox message. Such an assumption is likely to be valid, particularly in cases in which the groupware application initiates the communication from a server.

It should further be noted that, whilst in the preferred embodiment, only two workstations are interconnected for two-directional interactive communication, in general a sending workstation can be connected to any number of receiving workstations in an analogous manner to that described. Thus, for example, the first workstation may be associated through a third logical port, different to the first logical port, to a fourth logical port associated with a third workstation. In similar manner, each of the second and third workstations may likewise be linked to yet further workstations.

Thus, the invention as described, permits not only simultaneous, real-time editing, for example, of a groupware document but allows another user not presently involved to be invited to participate. The invitation to participate, being effected through electronic mail, is non-intrusive, although the very act of reading the dispatched message causes the desired two-way interactive communication to be established.

In the preferred embodiment described above, only a single groupware application is connected to a single window in a receiving workstation. However, it will be readily appreciated that any number of applications can be connected to corresponding windows via appropriate logical ports in either a single workstation or, indeed, in a plurality of workstations.

It will further be noted that the invention produces a bi-directional communication which at its most general is between a server application and a client application, as shown in Fig. 5. In such case, if the client application operates within a window on the client's workstation, then the client application must perform additional steps in order to route the communication to the appropriate window.

However, if the window system of the receiver of an Active Mail message run a network-based window system, such as X under UNIX, then a simpler variant of the above protocol is available. In accordance with such a protocol, upon establishing the two-way interactive communication, the receiver notifies the sender of the global Internet address of its workstation (or X terminal) and executes a command which allows the server to interact directly with the receiver's window system. In such a case, the interactive communication is not between an application running on the server and a process on the client's workstation which then talks to the window, but rather a direct connection between the application running on the server and the window on the receiver's screen.

Furthermore, whilst in the preferred embodiment the two-way interactive communication or dialogue is effected through the computer network, this is not a requirement of the invention. Thus, consider a receiving user whose workstation is connected to the receiving user's telephone line either directly or via a PBX. The act of reading his mailbox and finding a message of the Active Mail category, may, for example, automatically dial the sender and permit the receiving user to establish a dialogue with the sender via the telephone. This approach can, likewise, be extended to any number of participants using shared telephone or conferencing techniques.

Yet a further use of the invention is to effect an interactive communication between two applications running on respective workstations, whilst obviating the need for human interference. Thus, for example, consider a program which prompts a user to enter information and then continues operation along different branches, in accordance with the data entered by the operator.

Instead of a human operator providing the desired information, it is clearly possible to incorporate the responses in a data file for remote reading by the application. In such case, the invention may be employed to initiate an interactive communication between the workstation on which the application is loaded to the remote workstation on which the data file is loaded. The data file itself is, of course, incorporated within an application which, upon sensing the presence of an Active Mail-type message in its mailbox, automatically reads the message so as to establish the required interactive communication with the sending workstation.

The present invention also permits a logical port to be forwarded from a linked user to a non-linked user, so as to connect the non-linked user to the application. Thus, suppose that User B receives from User A a message of the Active Mail category having embedded therein the logical port associated with User A's workstation. He may perform the steps according to the invention in order to establish an interactive communication with User A. Additionally, or alternatively, providing that the "listen" service will manage a sufficient number of connection requests, he may forward the Active Mail message to a third user, User C who will then see the message in his mailbox as if it were directed from User A, there being no reference at all to the fact that this message was, in fact, dispatched by User B.

User C can then establish an interactive communication with User A in the normal manner. This facility is rendered possible because the logical port associated with the sending workstation is embedded in the message dispatched thereby: the embedding being effected when the message is configured and not when it is dispatched.

Such an approach might be used, for example, when User A dispatches a message to User B who reads the Active Mail-type message in order to establish the required interactive communication, and then decides that it would be beneficial to involve a third participant, User C. In such case, he need only forward the Active Mail message to User C, there being no requirement for User B to enter the electronic mail program, construct a suitable Active Mail-type message and dispatch it to User C.

In the detailed description of a preferred embodiment, no mention has been made of the type of data associated with the message other than that the message itself must, of course, be of the Active Mail category. However, in addition to the Active Mail message which simply establishes the required interactive communication, there may be attached thereto any other message of a supported category such as, for example, text, audio, graphics and so on. Such an approach obviates the need for two separate messages to be sent: one for establishing the required interactive communication and the other for dispatching electronic mail in the normal manner.

It will be understood that such an approach is possible only under electronic mail standards which support multiple message categories and attachments. Whilst this true of the MIME standard for "Internet" it is not, of course, universally true. Nevertheless, with slight modification, the invention is applicable even to electronic mail standards which are less versatile than MIME.

Thus, suppose that the electronic mail standard supports only text messages which may include typed attachments. Then the mail reading program can be upgraded to handle Active Mail attachments by calling AM Connect to process them. To invoke Active Mail, a suitably coded text message is written and despatched by electronic mail from the sending to the receiving workstations. Such a message might include an attachment of type "Active Mail" and the sending socket number (SN) might be included in the attachment. On reading such a message, the mail reading program reads the attachment and knows to connect the receiving workstation to socket SN of the sending workstation.

Suppose, however, that only text is supported by the electronic mail standard. In this case, to invoke Active Mail, a suitably coded text message is written and despatched by electronic mail from the sending to the receiving workstation. Such a message might include a banner reading: "Active Mail" and the sending socket name (SN) might be included as part of the message. The mail reading program can then be modified so that on reading such a message, the mail reading program sees the banner "Active Mail" and, upon decoding the socket name (SN) from the remainder of the message, knows to connect the receiving workstation to socket SN of the sending workstation.

Although the invention described with particular reference to Fig. 5 of the drawings relates to a network operating under UNIX and employing the Internet communications protocol, the more general description relating to Figs. 2 to 4 is applicable both to the Internet protocol and to other network protocols. Thus, it is contemplated that the arrangement described with reference to Figs. 2 to 4 may be easily adapted to any system having mail capabilities and a Point-to-Point communication, such as NOVELL and Net-Bios based networks, including LAN-Manager.

No mention has been made so far with regard to disconnecting a client from the server after having established an interactive communication in accordance with the invention. It should be noted that standard means may be employed by the client and/or the server in order to effect such disconnection.

Finally, whilst the invention has been described with reference to establishing an interactive communication between two or more workstations in the same network, it will be clear that it is equally feasible for the workstations to be in different interconnected networks. All that is required is for a unique network name to be reserved for the logical port in each workstation.

APPENDIX

Example of Server Code

```
/*
 *               Server Demo.
 * Send mail messages containing the local addr to several mail clients
 * and create a hand-shake.
 *
 */

#define     MarkError(Str,Code)  {printf("Error : %s\n",Str); exit(Code); }

#define     MAX_SIZE     1024
#define     QUEUE_LEN    5
#define     TMP_FILE_NAME "/tmp/am.tmp"
#define     SEND_MAIL_CMD "/usr/lib/sendmail"
#define     SUBJECT      "Subject: ActiveMail connection\n"
#define     CONTENT_TYPE  "Content-Type: ActiveMail\n\n"

#include      <errno.h>
#include      <stdio.h>
#include      <string.h>
#include      <fcntl.h>
#include      <signal.h>
#include      <sys/time.h>
#include      <sys/ioctl.h>
#include      <sys/types.h>
#include      <sys/stat.h>
#include      <sys/socket.h>
#include      <netinet/in.h>
#include      <netdb.h>
#include      <sys/param.h>

char   Buffer[MAX_SIZE];   /* General purpose buffer */


main()
{

  struct hostent          *hostP;
  struct sockaddr_in      ServerSa;
  char                    HostName[MAXHOSTNAMELEN];
  int                     ServerLen, ServerFd;

/*
 *  Get host name and host network parameters. Fill the socket name
 *  with the network addr.
 */

  if ( gethostname(HostName,MAXHOSTNAMELEN) )
    MarkError("Cannot get host name", 1);
  if ( (hostP = gethostbyname(HostName)) == NULL)
    MarkError("Cannot get host network params", 1);
  if (hostP->h_addrtype != AF_INET)
    MarkError("Invalid address type", 1);

/*
 * Set the socket name parameters, and let the UNIX choose a port for you.
 */

  bzero((char *)(&ServerSa), sizeof(ServerSa));
  ServerSa.sin_family = AF_INET;
  bcopy(hostP->h_addr, &(ServerSa.sin_addr.s_addr), hostP->h_length);
  ServerSa.sin_port = 0;

  if ((ServerFd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    MarkError("Cannot open socket", 1);
  if ( bind(ServerFd, &ServerSa, sizeof ServerSa) < 0)
    MarkError("Cannot bind socket", 1);

/*
 * Check that we got a valid port.
 */

  ServerLen = sizeof(ServerSa);
  if ( getsockname(ServerFd, (struct sockaddr *) &ServerSa, &ServerLen) < 0)
    MarkError("Cannot get socket name", 1);
  if ( ntohs(ServerSa.sin_port) == 0)
    MarkError("Uncorrect port number", 1)

/*
 * Listen on the socket and call the requests handling routine.
 */

  if ( listen(ServerFd, QUEUE_LEN) < -1 )
    MarkError("Cannot listen", 1);

  HandleRequests(ServerFd, &ServerSa);

} /* main */


/*****************************************************************************
 *
 * For input Socket and its name SocketName, send a formatted form of the
 * name to mail clients, and wait for connection requests from the client
 * on Socket. Upon such a request establish a hand-shake.
 *
 *****************************************************************************/

HandleRequests(Socket, SocketName)
int   Socket;
struct sockaddr_i *SocketName;
{

  struct sockaddr_in     ClientSa;
 char          MailMsg[30];
 char          *p;
 char          SendMailCmd[MAX_SIZE];
 int           ClientLen;
 int           ClientFd, TmpFd;
 int           NumOfClients=0;
 short         StrLen, BytesRead;

/*
 * Format the Socket-Name, and write it to a temporary file.
 * The file will be sent by mail to the client.
 */

 SocketNameToStr(SocketName, MailMsg);

 if ( (TmpFd = open( TMP_FILE_NAME, O_WRONLY | O_CREAT)) < 0)
   MarkError("Cannot open tmp file",TmpFd);
 if ( write(TmpFd,SUBJECT,strlen(SUBJECT)) != strlen(SUBJECT) )
      MarkError("Cannot write to tmp file", 1);
 if ( write(TmpFd,CONTENT_TYPE,strlen(CONTENT_TYPE)) != strlen(CONTENT_TYPE) )
      MarkError("Cannot write to tmp file", 1);
 if ( write( TmpFd, MailMsg, strlen(MailMsg)) != strlen(MailMsg) )
   MarkError("Cannot write to tmp file", 1);
 close(TmpFd);
 chmod( TMP_FILE_NAME, S_IRUSR | S_IWUSR | S_IROTH);


/*
 * Loop :
 * Get mail target, send it the local socket name and wait for a
 * connection request.
 * Once a connection is established, create a hand-shake.
 */

 while(1) {

   printf("\nEnter mail address [name@addr] > ");
   scanf("%s", Buffer);
   printf("Sending mail to %s ...\n", Buffer);
   sprintf( SendMailCmd, "%s %s < %s", SEND_MAIL_CMD, Buffer, TMP_FILE_NAME);
   system(SendMailCmd);

   ClientLen = sizeof(ClientSa);
   if ( (ClientFd = accept(Socket, &ClientSa, &ClientLen)) < 0 )
     MarkError("Cannot accept", 1);

   /*
    * Write a message to the client.
    */

   sprintf(Buffer, "You are client number %d", NumOfClients++);
   StrLen = strlen(Buffer);
   write(ClientFd, (char *)(&StrLen), sizeof(StrLen));
   write(ClientFd, Buffer, strlen(Buffer));
```

```
    /*
     * Read the acknowledgment from the client.
     */

    BytesRead = recv(ClientFd, (char *)(&StrLen), sizeof(StrLen), 0);
    if ( BytesRead != sizeof(StrLen) )
      MarkError("Cannot read message size", 1);

    BytesRead = recv(ClientFd, Buffer, StrLen, 0);
    if ( BytesRead != StrLen )
      MarkError("Cannot read message size", 1);
    Buffer[BytesRead] = '\0';
    printf("Recieved message : %s\n", Buffer);

  } /* while */

} /* HandleRequests */


/***************************************************************************
 *
 * Format SocketName to a string containing the family, port and address.
 * Output the formatted form in Str.
 *
 ***************************************************************************/

SocketNameToStr(SocketName, Str)
struct sockaddr_in *SocketName;
char   *Str;
{

  sprintf( Str, "%4hx %4hx %8lx\n", SocketName->sin_family,
                     SocketName->sin_port,
                     SocketName->sin_addr.s_addr );
} /* SocketNameToStr */
```

Example of Client Code

```
/*
 *                  Client Demo.
 * On input fn in the command line, connect to a server whose address is
 * formatted in the file. Establish a hand-shake and terminate.
 *
 */

#define     MarkError(Str,Code)  {printf("Error : %s\n",Str); exit(Code); }

#define     MAX_SIZE    1024
#define     HDR_LEN     2

#include     <errno.h>
#include     <stdio.h>
#include     <string.h>
#include     <fcntl.h>
#include     <signal.h>
#include     <sys/time.h>
#include     <sys/ioctl.h>
#include     <sys/types.h>
#include     <sys/socket.h>
#include     <netinet/in.h>
#include     <netdb.h>
#include     <sys/param.h>

char   Buffer[MAX_SIZE];   /* General purpose buffer */

main(argc,argv)
int   argc;
char  *argv[];
{

  struct sockaddr_in     ServerSa;
  char                   *p;
  int                    ClientFd;
  short                  BytesRead, StrLen;
  FILE                   *Fin;

  if (argc != 2) {
    printf("Usage : client FileName\n");
    exit(0);
  }

 /*
  *  Open the file and read the formatted internet addr from it.
  */

  if ( (Fin = fopen( argv[1], "r")) == NULL )
    MarkError("Cannot open file", 1);

  fscanf(Fin, "%4hx %4hx %8lx", &(ServerSa.sin_family),
                           &(ServerSa.sin_port),
                           &(ServerSa.sin_addr.s_addr) );
  fclose(Fin);
/*
 * Connect with the server.
 */

  if ((ClientFd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    MarkError("Cannot open socket", 1);
  if ( connect(ClientFd, &ServerSa, sizeof(ServerSa)) < 0)
    MarkError("Cannot connect to server", 1);

/*
 * Read a message from the server.
 */

  BytesRead = recv(ClientFd, (char *)(&StrLen), sizeof(StrLen), 0);
  if ( BytesRead != sizeof(StrLen) )
    MarkError("Cannot read message size", 1);

  BytesRead = recv(ClientFd, Buffer, StrLen, 0);
  if ( BytesRead != StrLen )
    MarkError("Cannot read message size", 1);
  Buffer[BytesRead] = '\0';
  printf("Received message : %s\n", Buffer);

/*
 * Write an acknowledgment to the server.
 */

  strcpy(Buffer, "Got Your message");
  StrLen = strlen(Buffer);
  write(ClientFd, (char *)(&StrLen), sizeof(StrLen));
  write(ClientFd, Buffer, StrLen);

  close(ClientFd);

} /* main */
```

Claims

1. For use in a computer network system (10) comprising at least first and second workstations (11, 13, 14, 15, 16, 17) adapted to send and receive messages by utilizing a suitable communication protocol and further adapted to exchange batch messages by means of an electronic mail program stored in each of said at least first and second workstations;

   a method for establishing an interactive communication between said at least first and second workstations, said method characterized by the steps of:

   (i) categorizing said batch messages such that a batch message of a predetermined category informs a receiving workstation that a sending workstation wishes to establish an interactive communication between a specified first logical port in the sending workstation and a specified second logical port in the receiving workstation;

   (ii) sending a batch message of the predetermined category having therein a reference to said first logical port from the first workstation to the second workstation so as to be received thereby and stored in a storage means (19) containing a list of batch messages;

   (iii) monitoring at the second workstation all batch messages in said storage means (19) at specified periods of time;

   (iv) noting the presence in said storage means (19) of a batch message of said predetermined category;

   (v) utilizing the communication protocol to send an initiation signal from the second logical port in the second workstation to the first logical port in the first workstation; and

(vi) responsive to receipt of the initiation signal, establishing an interactive two-way communication between the first logical port of the first workstation and the second logical port of the second workstation.

2. The method according to Claim 1, wherein the first logical port of the first workstation is associated with a groupware application.

3. The method according to Claim 1, wherein the first logical port of the first workstation and the second logical port of the second workstation are each associated with respective applications which interactively communicate with each other.

4. The method according to Claim 1, wherein the computer network system comprises at least two interconnected networks and said first and second workstations are located in different ones of said interconnected networks.

5. The method according to Claim 1, wherein said batch message includes an attachment having therein data of a category supported by the electronic mail program, whereby upon reading the message, said data is automatically output to the second workstation.

6. The method according to Claim 1, wherein the first logical port in the first workstation serves only to establish a communication whereupon the communication is subsequently routed via a third logical port to the first workstation, thereby permitting multiple connections to be established to the first workstation via said first logical port.

7. The method according to Claim 1, wherein the first workstation establishes said interactive two-way communication with said second workstation and with at least one third workstation whereby all three workstations communicate simultaneously.

8. The method according to Claim 7, wherein the second workstation and said at least one third workstation are each coupled to different logical ports in the first workstation.

9. The method according to Claim 1, wherein the computer network system operates under UNIX.

10. The method according to Claim 1, wherein the computer network system operates under NOVELL.

11. The method according to Claim 1, wherein the computer network system operates under LAN Manager.

12. The method according to Claim 1, wherein:
multiple message categories are supported by the electronic mail program, and
a unique message category is defined indicating that the sending workstation wishes to establish said communication with the receiving workstation.

13. The method according to Claim 1, wherein:
multiple message categories are not supported by the electronic mail program,
the electronic mail program supports attachments, and
said reference to the first logical port is included within an attachment which serves as an argument to the electronic mail program on reading the batch message at the receiving workstation.

14. The method according to Claim 1, wherein:
multiple message categories are not supported by the electronic mail program,
the electronic mail program does not support attachments, a banner is included within the batch message to inform the electronic mail program that the sending workstation wishes to establish said communication with the receiving workstation, and
the first logical port of the sending workstation is encoded within the batch message and serves as an argument to the electronic mail program on reading the batch message at the receiving workstation.

15. The method according to Claim 1, wherein:
the second workstation runs a network based window system associated with a global network address of the second workstation and having means for determining whether a process running on a remote workstation is authorized to open a window (21) on the second workstation and communicate with said window (21), and

prior to step (v) there is included the further step of:

(ivb) authorizing the first logical port of the first workstation to open a window (21) on the second workstation;

whereby:

in step (v) the communication protocol is utilized in order to supply the global network address of the second workstation to the first logical port in the first workstation, and

upon performing step (vi), a signal is sent from the first workstation to the second workstation in order to open a window (21) on the second workstation with which the first workstation-may interact with a user on the second workstation.

16. The method according to Claim 15, wherein:

the network based window system is X said means for determining whether a process running on a remote workstation is authorized to open a window (21) on the second workstation and communicate with said window (21) comprising a list of addresses each in respect of a remote workstation which is authorized to open said window (21) and communicate therewith, and

in step (ivb) the first workstation is added to the list of addresses in the second workstation.

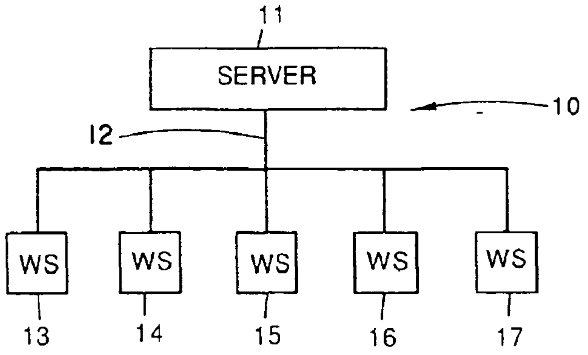

11
SERVER
10
12
WS
WS
WS
WS
WS
13
14
15
16
17

Fig.1a

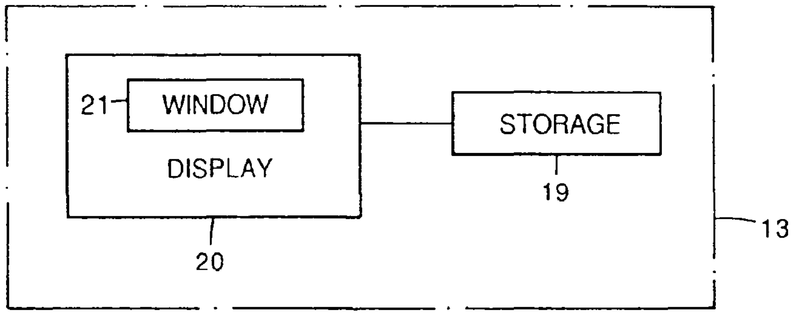

21
WINDOW
DISPLAY
STORAGE
19
13
20

Fig.1b

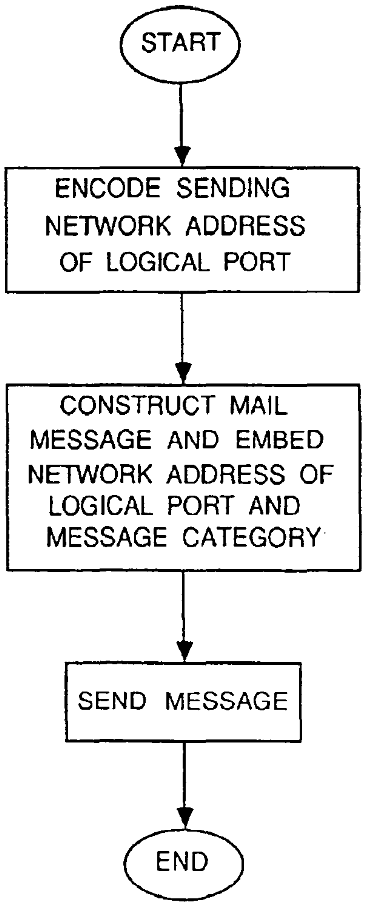

START
ENCODE SENDING NETWORK ADDRESS OF LOGICAL PORT
CONSTRUCT MAIL MESSAGE AND EMBED NETWORK ADDRESS OF LOGICAL PORT AND MESSAGE CATEGORY
SEND MESSAGE
END

Fig.2

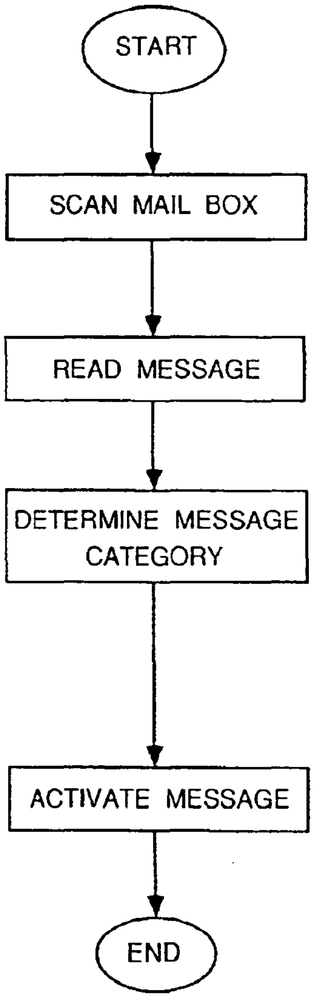

START
SCAN MAIL BOX
READ MESSAGE
DETERMINE MESSAGE CATEGORY
ACTIVATE MESSAGE
END

Fig.3

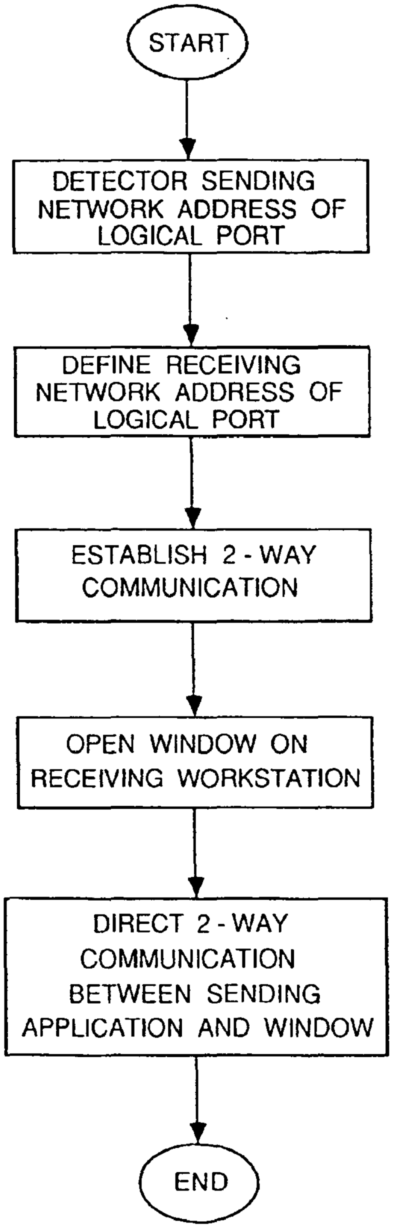

START
DETECTOR SENDING NETWORK ADDRESS OF LOGICAL PORT
DEFINE RECEIVING NETWORK ADDRESS OF LOGICAL PORT
ESTABLISH 2 - WAY COMMUNICATION
OPEN WINDOW ON RECEIVING WORKSTATION
DIRECT 2 - WAY COMMUNICATION BETWEEN SENDING APPLICATION AND WINDOW
END

Fig.4

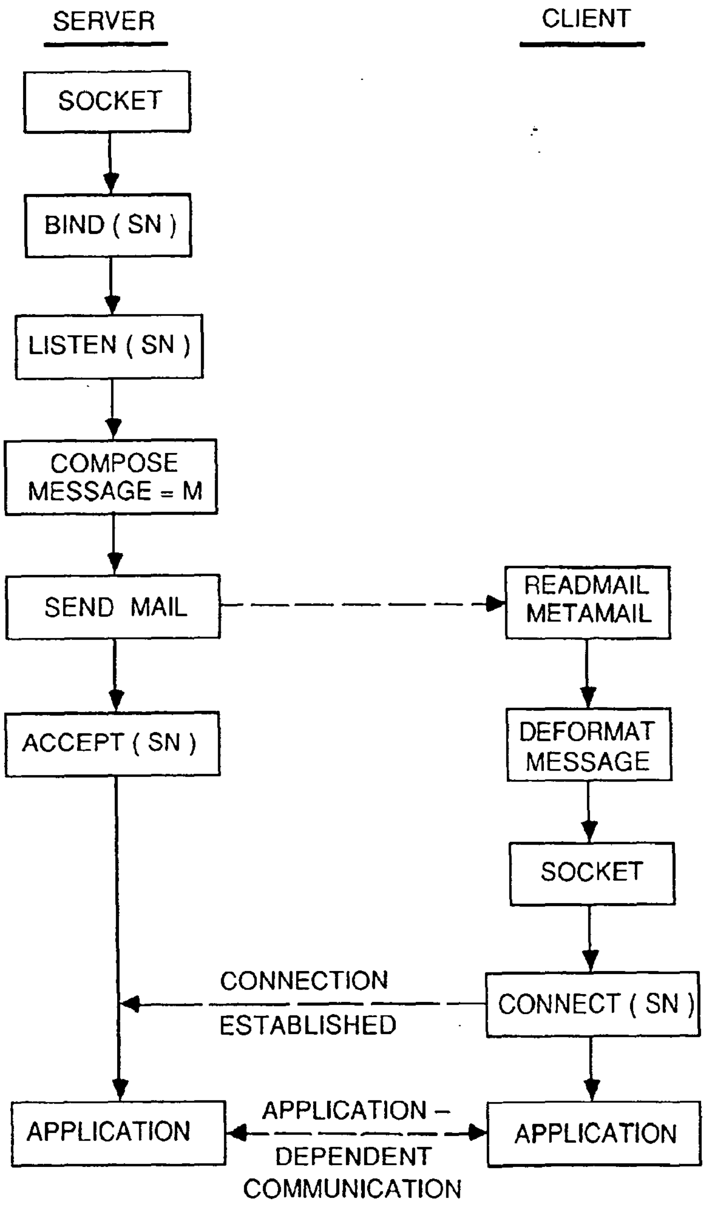

SERVER
CLIENT
SOCKET
BIND (SN)
LISTEN (SN)
COMPOSE MESSAGE = M
SEND MAIL
READMAIL METAMAIL
ACCEPT (SN)
DEFORMAT MESSAGE
SOCKET
CONNECTION ESTABLISHED
CONNECT (SN)
APPLICATION
APPLICATION – DEPENDENT COMMUNICATION
APPLICATION

Fig.5

European Patent
Office

EUROPEAN SEARCH REPORT

Application Number
EP 93 63 0052

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
A	WO-A-9 003 074 (CAPRICOM, S.A.) * page 2, line 14 - page 3, line 25 * * page 5, line 20 - page 7, line 3 * ---	1-16	H04L12/58
A	US-A-5 040 141 (K.YAZIMA ET AL) * column 3, line 12 - line 51 * ---	1-16	
A	US-A-5 127 003 (W.J.DOLL ET AL) * column 7, line 64 - column 8, line 49 * * column 10, line 5 - line 21 * -----	1-16	
			TECHNICAL FIELDS SEARCHED (Int. Cl.5)
			G06F H04L

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	21 OCTOBER 1993	CANOSA ARESTE C.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

Re-Exam

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re PATENT APPLICATION OF:
Net2Phone, Inc.
Control No.: 90/010,416
Issue Date: August 22, 2000
Title: **POINT-TO-POINT INTERNET PROTOCOL**

Attorney Docket: 2655-0188
Group Art Unit: 3992
Examiner: KOSOWSKI, Alexander J.
Date: June 11, 2009
Confirmation No.: 1061

INFORMATION DISCLOSURE STATEMENT

Hon. Commissioner of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Pursuant to 37 C.F.R. § 1.56, the attention of the Patent and Trademark Office is hereby directed to the reference(s) listed on the attached PTO-1449. One copy of each non-U.S. Patent reference is attached. It is respectfully requested that the information be expressly considered during the prosecution of this application, and that the reference(s) be made of record therein and appear among the "References Cited" on any patent to issue therefrom.

The submission of any document herewith, which is not a statutory bar, is not intended that any such document constitutes prior art against any of the claims of the present application or is considered to be material to patentability as defined in 37 C.F.R. § 1.56(b). Applicants do not waive any rights to take any action which would be appropriate to antedate or otherwise remove as a competent reference against the claims of the present application.

☒ This Information Disclosure Statement is being filed within three (3) months of the U.S. filing date OR before the mailing date of a first Office Action on the merits. No certification or fee is required.

□ This Information Disclosure Statement is being filed more than three (3) months after the U.S. filing date AND after the mailing date of the first Office Action on the merits, but before the mailing date of a Final Rejection or Notice of Allowance.

□ I hereby certify that each item of information contained in this Information Disclosure Statement was cited in a communication from a foreign patent office in a counterpart foreign application not more than three (3) months prior to the filing of this Information Disclosure Statement. 37 C.F.R. § 1.97(e)(1).

□ I hereby certify that no item of information in this Information Disclosure Statement was cited in a communication from a foreign patent office in a counterpart foreign application or, to my knowledge after making reasonable inquiry, was known to any individual designated in 37 C.F.R. § 1.56(c) more than three (3) months prior to the filing of this Information Disclosure Statement. 37 C.F.R. § 1.97(e)(2).

□ Attached is our check no. _____ in the amount required under 37 C.F.R. § 1.17(p). Please credit or debit Deposit Account No. 501860 as needed to ensure consideration of the disclosed information. A duplicate copy of this paper is attached.

□ This Information Disclosure Statement is being filed more than three (3) months after the U.S. filing date and after the mailing date of a Final Rejection or Notice of Allowance, but before payment of the Issue Fee. Applicant(s) hereby requests that the Information Disclosure Statement be considered. Attached is our check in the amount required under 37 C.F.R. § 1.17(p). Please credit or debit

Deposit Account No. 501860 as needed to ensure consideration of the disclosed information. A duplicate copy of this paper is attached.

□ I hereby certify that each item of information contained in this Information Disclosure Statement was cited in a communication from a foreign patent office in a counterpart foreign application not more than three (3) months prior to the filing of this Information Disclosure Statement. 37 C.F.R. § 1.97(e)(1).

□ I hereby certify that no item of information in this Information Disclosure Statement was cited in a communication from a foreign patent office in a counterpart foreign application and, to my knowledge after making reasonable inquiry, was known to any individual designated in 37 C.F.R. § 1.56(c) more than three (3) months prior to the filing of this Information Disclosure Statement. 37 C.F.R. § 1.97(e)(2).

□ Relevance of the non-English language reference(s) is/are discussed in the present specification.

□ The reference(s) was/were cited in a counterpart foreign application. An English language version of the foreign search report is attached for the Examiner's information.

□ A concise explanation of the relevance of the non-English language reference(s) appear(s) in the Appendix hereto.

□ The Examiner's attention is directed to co-pending U.S. Patent Application No. _____, filed _____, which is directed to related technical subject matter. The identification of this U.S. Patent Application is not to be construed as a waiver of secrecy as to that application now or upon issuance of the present application as a patent. The Examiner is respectfully requested to consider the cited application and the art cited therein during examination.

□ Copies of the references were cited by or submitted to the Office in parent Application No. _____, filed _____, which is relied upon for an earlier filing date under 35 U.S.C. 120. Thus, Form PTO 1449 is attached without copies of these references. 37 C.F.R. § 1.98(d).

CHARGE STATEMENT: Deposit Account No. 501860, order no. **2655-0188**.

The Commissioner is hereby authorized to charge any fee specifically authorized hereafter, or any missing or insufficient fee(s) filed, or asserted to be filed, or which should have been filed herewith or concerning any paper filed hereafter, and which may be required under Rules 16-18 (missing or insufficiencies only) now or hereafter relative to this application and the resulting Official Document under Rule 20, or credit any overpayment, to our Accounting/Order Nos. shown above, for which purpose a duplicate copy of this sheet is attached

This CHARGE STATEMENT does not authorize charge of the issue fee until/unless an issue fee transmittal sheet is filed.

CUSTOMER NUMBER
42624

Davidson Berquist Jackson & Gowdey LLP
4300 Wilson Blvd., 7th Floor,
Arlington Virginia 22203
Main: (703) 894-6400 • FAX: (703) 894-6430

Respectfully submitted,

By:
Michael R. Casey, Ph.D.
Registration No.: 40,294

CERTIFICATE OF SERVICE

The undersigned hereby certifies that, on June 11, 2009, the Information Disclosure Statements (with references in electronic format, as agreed by requestor) filed in Reexam Control Numbers:

1) 90/010,422;

2) 90/010,424;

3) 90/010,421;

4) 90/010,416; and

5) 90/010,423

were served by FedEx (tracking no. 796686808721), on Requestor:

Blakely, Sokoloff, Taylor & Zafman LLP
1279 Oakmead Parkway
Sunnyvale, CA 94085-4040

Michael R Casey

Michael R. Casey, Ph.D.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 1 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

U.S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document No.	Publication/ Issue Date	Name of Patentee or Applicant of Cited Document
	1-1	US-4313035	1982/01/26	Jordan et al.
	1-2	US-4423414	1983/12/27	Bryant et al.
	1-3	US-4491693	1985/01/01	Sano et al.
	1-4	US-4602132	1986/07/22	Nagatomi et al.
	1-5	US-4653090	1987/03/24	Hayden, C.
	1-6	US-4658093	1987/14/04	Hellman
	1-7	US-4706274	1987/11/10	Baker et al.
	1-8	US-4754479	1988/06/28	Bicknell et al.
	1-9	US-4755985	1988/07/05	Jayapalan et al.
	1-10	US-4756020	1988/07/05	Fodale
	1-11	US-4759056	1988/07/19	Akiyama
	1-12	US-4800488	1989/24/01	Agrawal et al.
	1-13	US-4823374	1989/04/18	Verlohr
	1-14	US-4827411	1989/05/02	Arrowood
	1-15	US-4899333	1990/06/02	Roediger
	1-16	US-4899373	1990/02/06	Lee et al.
	1-17	US-4914571	1990/04/03	Baratz et al.
	1-18	US-4928306	1990/05/22	Biswas et al.
	1-19	US-4953159	1990/08/28	Hayden, C.
	1-20	US-4962449	1990/10	Schlesinger
	1-21	US-5109403	1992/04/28	Sutphin
	1-22	US-5113499	1992/05	Ankney et al.
	1-23	US-5127001	1992/30/06	Steagall, et al.
	1-24	US-5134648	1992/07/28	Hochfield et al.

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 2 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

U.S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document No.	Publication/ Issue Date	Name of Patentee or Applicant of Cited Document
	2-1	US-5136716	1992/08/04	Harvey et al.
	2-2	US-5153908	1992/10/06	Kakizawa et al.
	2-3	US-5159592	1992/10	Perkins
	2-4	US-5164988	1992/11/17	Matyas et al.
	2-5	US-5185860	1993/02/09	Wu
	2-6	US-5195086	1993/03/16	Baumgartner et al.
	~~2-7~~	US-5301324	1994/04	Dewey et al.
	2-8	US-5315705	1994/24/05	Iwami Naoko et al
	2-9	US-5319705	1994/07/06	Halter et al.
	2-10	US-5325524	1994/06/28	Black et al.
	2-11	US-5329619	1994/07/12	Page et al.
	2-12	US-5388213	1995/02/07	Oppenheimer et al.
	2-13	US-5402477	1995/03/28	McMahan et al.
	2-14	US-5402528	1995/03/28	Christopher et al.
	2-15	US-5408526	1995/04/18	McFarland et al.
	2-16	US-5408619	1995/04/18	Oran
	2-17	US-5425028	1995/13/06	Britton et al.
	2-18	US-5434913	1995/07	Tung et al.
	2-19	US-5440632	1995/08/08	Bacon et al.
	2-20	US-5452289	1995/09/19	Sharma et al.
	2-21	US-5461668	1995/10/24	Zdenek et al.
	2-22	US-5469500	1995/21/11	Satter et al.
	2-23	US-5475819	1995/12	Miller et al.
	2-24	US-5481720	1996/02/01	Loucks et al.

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 3 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

U.S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document No.	Publication/ Issue Date	Name of Patentee or Applicant of Cited Document
	3-1	US-5499295	1996/12/03	Cooper
	3-2	US-5502727	1996/03/26	Catanzaro et al.
	3-3	US-5515508	1996/05/07	Pettus et al.
	3-4	US-5533110	1996/02/07	Pinard et al.
	3-5	US-5555290	1996/09/10	McLeod et al.
	3-6	US-5608786	1997/04/03	Gordon
	3-7	US-5615257	1997/03/25	Pezzullo et al.
	3-8	US-5621789	1997/04/15	McCalmont et al.
	3-9	US-5627978	1997/05/06	Altom et al.
	3-10	US-5649194	1997/07	Miller et al.
	3-11	US-5671412	1997/09/23	Christiano
	3-12	US-5684951	1997/04/11	Goldman et al.
	3-13	US-5689641	1997/11/18	Ludwig et al.
	3-14	US-5692180	1997/11	Lee
	3-15	US-5724648	1998/03/03	Shaughnessy et al.
	3-16	US-5734828	1998/31/03	Pendse et al.
	3-17	US-5774656	1998/06/30	Hattori et al.
	3-18	US-5790803	1998/04/08	Kinoshita et al.
	3-19	US-5815665	1998/29/09	Teper et al.
	3-20	US-5819084	1998/10/08	Shapiro, E.
	3-21	US-5825865	1998/20/10	Oberlander et al.
	3-22	US-5844978	1998/12/01	Reuss et al.
	3-23	US-5883956	1999/03/16	Le et al.
	3-24	US-5953350	1999/09	Higgins

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 4 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

U.S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document No.	Publication/ Issue Date	Name of Patentee or Applicant of Cited Document
	4-1	US-5956485	1999/09/21	Perlman
	4-2	US-6009469	1999/12	Mattaway et al.
	4-3	US-6031836	2000/02	Haserodt
	4-4	US-6047054	2000/04	Bayless et al.
	4-5	US-6067350	2000/05/23	Gordon, A.
	4-6	US-6108704	2000/08	Hutton et al.
	4-7	US-6131121	2000/10	Mattaway et al.
	4-8	US-6360266	2002/03/19	Pettus
	4-9	US-6513066	2003/01	Hutton et al.
	4-10	US-6701365	2004/02/03	Glenn W. Hutton
	4-11			
	4-12			
	4-13			
	4-14			
	4-15			
	4-16			
	4-17			
	4-18			
	4-19			
	4-20			
	4-21			
	4-22			
	4-23			
	4-24			

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT FORM PTO-1449 (modified)	Reexam number	90/010,416
	First Named Inventor	Hutton
	Patent Under Re-Exam	6108704
	Issue Date	2000/08/22
	Group Art Unit	3992
	Examiner Name	KOSOWSKI, ALEXANDER J
	Attorney Docket No.	2655-0188
Sheet 5 of 67	Confirmation No.	1061

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document No.	Publication Date	Name of Patentee or Applicant of Cited Document	Notes
	5-1	EP-0455402-A2	11-06-1991	Hewlett-Packard Company	
	5-2	EP-0556012-A2	08-18-1993	Wada et al.	
	5-3	EP-0581722	02/02/1994	Yeda R&D Co., Ltd.	
	5-4	EPO 0497022A1	19920508	Jennings et al.	
	5-5	WO-9219054	10-29-1992	Ferdinand et al.	
	5-6				
	5-7				
	5-8				
	5-9				
	5-10				
	5-11				
	5-12				
	5-13				
	5-14				
	5-15				
	5-16				
	5-17				
	5-18				
	5-19				
	5-20				
	5-21				
	5-22				
	5-23				
	5-24				

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 6 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	6-1	"A Low Cost Solution for: Using your WAN as a Voice Communication Tool" VocalTec White Paper (dated 06/03/94)	
	6-2	"CyberPhone Annoucement" Internet Posting in Newsgroups comp.speech, June 8, 1995.	
	6-3	"CyberPhone!" Internet Posting in Newsgroups comp.speech, April 14, 1995.	
	6-4	"Electric Magic Company Provides Internet Alternative to Long-Distance Calls", Electric Magic Company Press Release (March 13, 1995)	
	6-5	"Electric Magic Company Releases NetPhone 1.2 and Netpub Server", Electric Magic Company Press Release (June 1995)	
	6-6	"Frequently-Asked Questions about Tribal Voices PowWow" Version 0.34, March 4, 1996.	
	6-7	"Frequently-Asked Questions about Tribal Voices PowWow" Version 0.43, May 1, 1996.	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 7 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	7-1	"Frequently-Asked Questions about Tribal Voices PowWow" Version 0.45, May 31, 1996.	
	7-2	"Frequently-Asked Questions about Tribal Voices PowWow" Version 0.47, June 12, 1996.	
	7-3	"Frequently-Asked Questions about Tribal Voices PowWow" Version 0.48, June 25, 1996.	
	7-4	"Frequently-Asked Questions about Tribal Voices PowWow" Version 0.59, October 30, 1996.	
	7-5	"NetPhone Gets Internet Users Talking at Local Rates" MacUser UK, March 3, 1995, pg. 27.	
	7-6	"NetPhone Gives Your Mac Voice Over the Internet" Inside the Internet Rocket Science for the Rest of Us. Vol. 2 Num. 3, June 1995.	
	7-7	"NetPhone" MacWorld, July 1995.	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 8 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	8-1	"NetPhone" West Coast Online, Ver. 3.02 (#26), April 1995.	
	8-2	"PowWow 1.3b Now Available!" Google Newsgroup comp.os.ms-windows.misc Discussion Posting (dated April 22, 1995)	
	8-3	1996-1997 Buyer's Guide, CTI for Management	
	8-4	Abbe Cohen, Inessential Zephyr (Aug. 23, 1993).	
	8-5	Adam Gaffin, VocalTec Ware Lets Users Make Voice Calls over 'Net, NETWORK WORLD (Feb. 13, 1995).	
	8-6	Alexander Schill, ed., DCE—The OSF Distributed Computer Environment: Client/Server Model and Beyond, Lecture Notes in Computer Science 731, Karlsruhe University (1993).	
	8-7	Analysis of DCE Security Draft (Sept. 18, 1996).	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

FORM PTO-1449 (modified)

Sheet 9 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	9-1	Andrew D. Birrell, et al., Grapevine: An Exercise in Distributed Computing, Communications of the ACM (April 1982).	
	9-2	Andrew D. Birrell, et al., Grapevine: An Exercise in Distributed Computing, COMMUNICATIONS OF THE ACM, vol. 25, No. 4, Apr. 1982.	
	9-3	Andrew D. Birrell, et al., Implementing Remote Procedure Calls, ACM Transactions on Computer Systems (Feb. 1984).	
	9-4	Andrew S. Tanenbaum, COMPUTER NETWORKS, 2d ed. (Prentice-Hall, 1988).	
	9-5	Andy Patrizio, Telecom, Digital Limits Begin to Blur with 'Phone Calls' Across Internet, PC WEEK, vol. 12, no. 6 (Feb. 13, 1995).	
	9-6	Antonio Ruiz, Voice and Telephony Applications for the Office Workstation, IEEE 1st International Conference on Computer Workstations, San Jose, California (Nov. 11-14, 1985).	
	9-7	AVC-650: Technical Issues Concerning Real-Time Protocol in H.32Z Systems in ATM and Other Packet-Switched Computer Networks, July 9, 1994.	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 10 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	10-1	AVC-655: Communication Procedure for H.222.1, July 1, 1994.	
	10-2	AVC-666: H.32X Communication Modes, Terminal Types and Interworking Scenarios, July 1994.	
	10-3	AVC-683: Update Draft H.32Z Following Grimstad Meeting, Nov. 1994.	
	10-4	AVC-696: An Example of Call Setup Procedure in a H.32Z Terminal, Nov. 1994.	
	10-5	AVC-702: Terminal to Terminal Signaling in H.32X, Oct. 24, 1994.	
	10-6	AVC-707R: Report of the Seventeenth Experts Group Meeting in Singapore (1-11 July 1994) – Part 1 and Part II, Nov. 11, 1994.	
	10-7	AVC-716: Draft Recommendation H.32X, Jan. 1995.	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT FORM PTO-1449 (modified) Sheet 11 of 67	Reexam number	90/010,416
	First Named Inventor	Hutton
	Patent Under Re-Exam	6108704
	Issue Date	2000/08/22
	Group Art Unit	3992
	Examiner Name	KOSOWSKI, ALEXANDER J
	Attorney Docket No.	2655-0188
	Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	11-1	AVC-718: Draft H.32X, Jan. 1995.	
	11-2	AVC-743R: Report of the Eighteenth Experts Group Meeting in Kamifukuoka (24-27 January 1995), Jan. 27, 1995.	
	11-3	AVC-748: Update of Draft Recommendation H.322, May 1995.	
	11-4	AVC-750: Report of the Study Group 15 Meeting Held During 6-17 February 1995, Feb. 24, 1995.	
	11-5	AVC-752: Open Issues Towards the Stockholm Meeting, Mar. 17, 1995.	
	11-6	AVC-758: Draft Recommendation H.323 Visual Telephone Systems and Terminal Equipment for Local Area Networks Which Provide A Non-Guaranteed Quality of Service, Rev. May 12, 1995.	
	11-7	AVC-767: Logical Channel Set-up Procedure, Apr. 28, 1995.	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

FORM PTO-1449 (modified)

Sheet 12 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	12-1	AVC-799: Comments on Draft H.323 and H.22Z, May 11, 1995.	
	12-2	AVC-800R: Report of the Nineteenth Experts Group Meeting in Haninge (15-18 May 1995), May 18, 1995	
	12-3	AVC-813: Signaling Recommendation Within the Scope of H.323, Sept. 10, 1995.	
	12-4	AVC-819: LAN Addressing Plan in H.323, Sept. 10, 1995	
	12-5	AVC-830: Connection Management Procedures for H.323, Oct. 24-27, 1995.	
	12-6	AVC-842: Gateway, Gatekeeper and Terminal Procedures in H.323, Oct. 17, 1995.	
	12-7	Avnish Aggarwal, et al., RFC 1002: Protocol Standard for a NetBIOS Service on a TCP/UDP Tansport: Detailed Specifications (Mar. 1987).	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT FORM PTO-1449 (modified)		
	Reexam number	90/010,416
	First Named Inventor	Hutton
	Patent Under Re-Exam	6108704
	Issue Date	2000/08/22
	Group Art Unit	3992
	Examiner Name	KOSOWSKI, ALEXANDER J
	Attorney Docket No.	2655-0188
Sheet 13 of 67	Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	13-1	Barbara Darrow, Internet Phone Chat Software Prompts Spat; IRC Operators Rebuffed Use of Their Systems, COMPUTER RESELLER NEWS (Mar. 20, 1995).	
	13-2	Barry Michael Arons, The Audio-Graphical Interface to a Personal Integrated Telecommunications System, Masters Thesis, Massachusetts Institute of Technology (June 1984).	
	13-3	Barry Phillips, Casting the Net for New Media, OEM MAGAZINE, no. 320 (1995).	
	13-4	Belville, Sharon, "Zephyr on Athena", Athena Documentation, September 10, 1991, Version 3.	
	13-5	Ben Mesander, et al., The Client-To-Client Protocol (Aug. 12, 1994).	
	13-6	Bill Welsh, H.245 Implementors' Guide (undated but references April 1996)	
	13-7	Bob Blakley's Email to sig-dce-security, DCE Delegation Proposal Review, July 7, 1992.	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT FORM PTO-1449 (modified) Sheet 14 of 67	Reexam number	90/010,416
	First Named Inventor	Hutton
	Patent Under Re-Exam	6108704
	Issue Date	2000/08/22
	Group Art Unit	3992
	Examiner Name	KOSOWSKI, ALEXANDER J
	Attorney Docket No.	2655-0188
	Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	14-1	Brad Curtis Johnson, A Distributed Computing Environment Framework: An OSF Perspective (1991).	
	14-2	Brent Nordin, et al., Remote Operation Across a Network of Small Computers (Association of Computing Machinery, 1986).	
	14-3	Brian Fox, et al., GNU Finger program documentation, Free Software Foundation (1992).	
	14-4	Bruce Brown, BugNet Bug/Fix List, NEWSBYTES (Dec. 13, 1995).	
	14-5	Bruce Brown, BugNet Bug/Fix List, NEWSBYTES (Dec. 13, 1996).	
	14-6	Butler W. Lampson, et al., A Distributed Systems Architecture for the 1990's (Dec, 17, 1989),	
	14-7	Buy Memory Configured Expressly for Your Computer, SAN JOSE MERCURY NEWS (July 16, 1995).	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 15 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	15-1	C. Anthony DellaFera, et al., Section E.4.1: Zephyr Notification Service, ATHENA TECHNICAL PLAN (July 29, 1988).	
	15-2	C. Anthony DellaFera, et al., Section E.4.1: Zephyr Notification Service, Project Athena Technical Plan (June 5, 1989).	
	15-3	C. Anthony DellaFera, et al., The Athena Notification Service: Zephyr (1987).	
	15-4	C. Anthony DellaFera, et al., The Athena Notification Service: Zephyr (Dec. 31, 1987).	
	15-5	C. Anthony DellaFera, et al., The Zephyr Notification Service (undated).	
	15-6	C. Anthony DellaFera, et al., The Zephyr Notification Service, USENIX Winter Conference, Feb. 9-12, 1988	
	15-7	C. Anthony DellaFera, The Zephyr Notification Service, MIT Project Athena, Winter Usenix Conference (Feb. 12, 1988).	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT FORM PTO-1449 (modified)	Reexam number	90/010,416
	First Named Inventor	Hutton
	Patent Under Re-Exam	6108704
	Issue Date	2000/08/22
	Group Art Unit	3992
	Examiner Name	KOSOWSKI, ALEXANDER J
	Attorney Docket No.	2655-0188
Sheet 16 of 67	Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	16-1	C. Malamud, et al., RFC 1528: Principles of Operation for the TPC.INT Subdomain: Technical Procedures (Oct. 1993).	
	16-2	C. Malamud, et al., RFC 1530: Principles of Operation for the TPC.INT Subdomain: General Principles and Policy (Oct. 1993).	
	16-3	C. Sunshine, et al., IEN 135: Addressing Mobile Hosts in the ARPA Internet Environment (Oct. 1985).	
	16-4	C. Yang, RFC 1789: INETPhone: Telephone Services and Servers on Internet (Apr. 1995).	
	16-5	Calls Waiting on the Internet Although Telephone Software Makes 'Free' Long Distance Possible, it's a Long Way from Practical, KANSAS CITY STAR (July 14, 1996).	
	16-6	Carl Sunshine, IEN 178: Addressing Problems in Multi-Network Systems (Apr. 1981).	
	16-7	Charles E. Perkins, et al., A Mobile Networking System Based on Internet Protocol, IEEE PERSONAL COMMUNICATIONS (First Quarter 1994).	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 17 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	17-1	Charlie Kaufman's Email to dmackey re DCE 1.1 Delegation Proposal for Review, June 22, 1992.	
	17-2	Chii-Ren Tsai, et al., Distributed Audit with Secure Remote Procedure Calls (1991).	
	17-3	Christopher Schmandt, et al., An Audio and Telephone Server for Multi-Media Workstations, IEEE (1988).	
	17-4	Christopher Schmandt, et al., Phone Slave: A Graphical Telecommunications Interface, Society for Information Display, 1984 International Symposium Digest of Technical Papers (June 1984).	
	17-5	Chuck Kane, List of IRC servers as of Feb. 1, 1995, available at http://ftp.funet.fi/pub/unix/irc/does/servers.950201.	
	17-6	Clinton Wilder, Pulling in the Net – InfoSeek, VocalTec Offer Search and Voice Options to Internet Users Online, INFORMATIONWEEK, no. 516 (1995).	
	17-7	Common Desktop Environment 1.0—Advanced User's and System Administrator's Guide, Addison-Wesley Publishing Co. (1995).	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 18 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	18-1	Common Desktop Environment 1.0—User's Guide, Addison-Wesley Publishing Co. (1995).	
	18-2	Communications Connectivity Networking, MICROSOFT SYSTEMS JOURNAL, vol. 10, no. 1 (Jan. 1995).	
	18-3	Comp.Speech FAQ Archive; Comp.Speech FAQ Web Page, COMP.SPEECH NEWSGROUP. July 17, 1995)	
	18-4	Comp.Speech FAQ Weekly Reminder, COMP.SPEECH NEWSGROUP (June 21, 1995)	
	18-5	Contents, Preface, and Index to Open Software Foundation, X/Open Preliminary Specification—X/Open DCE: Authentication and Security Services (March 1996).	
	18-6	Conversation Excerpt from ftp://svr-ftp.eng.camac.uk/pub/pub/comp.speech/archive/subject5xxx.txt accessed on 11/28/2007	
	18-7	Craig Crossman, Free Calls on Internet are CB-Style No Longer, MIAMI HERALD (June 26, 1995).	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT FORM PTO-1449 (modified)	Reexam number	90/010,416
	First Named Inventor	Hutton
	Patent Under Re-Exam	6108704
	Issue Date	2000/08/22
	Group Art Unit	3992
	Examiner Name	KOSOWSKI, ALEXANDER J
	Attorney Docket No.	2655-0188
Sheet 19 of 67	Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	19-1	Craig Crossman, Make Long Distance Calls Via the Internet, RECORD (July 3, 1995).	
	19-2	D. O'Mahoney, 1st Generation Internet Phones (1998).	
	19-3	D. Reed, RFC 1324: A Discussion on Computer Network Conferencing (May 1992).	
	19-4	D. Zimmerman, RFC 1288: The Finger User Information Protocol (Dec. 1991).	
	19-5	Dale Skran, Draft ITU-T Recommendation H.225.0— Line Transmission of Non-Telephone Signals, Media Stream Packetization and Synchronization on Non-Guaranteed Quality of Service LANs (May 28, 1996).	
	19-6	Dale Skran, ed. ASN.1 for H.225.0 (June 18, 1996).	
	19-7	Dan Cohen, IEN 31: On Name, Addresses and Routings (II) (Apr. 28, 1978).	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 20 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	20-1	Dan Keating, Ring! It's Computer Calling Phone By Internet Has Gotten Better, MIAMI HERALD (May 22, 1996).	
	20-2	Daniel C. Swinehart, Telephone Management in the Etherphone System, IEEE (1987).	
	20-3	Daniel H. Craft, Resource Management in a Decentralized System, OPERATING SYSTEMS REVIEW, vol. 17, no. 5 (Association for Computing Machinery, Oct. 1983).	
	20-4	Danny Cohen, IEN 23: On Name, Addresses and Routings (Jan. 23, 1978).	
	20-5	Dave Lindbergh, H.323 Encryption, Document: CNC-96-22 (April 15, 1996).	
	20-6	David D. Clark, RFC 814: Name, Addresses, Ports, and Routes (July 1982).	
	20-7	David Gertler, Hardware and Software Tidbits from CEBIT, SEYBOLD REPORT ON DESKTOP PUBLISHING, vol. 9, no. 8 (Apr. 3, 1995).	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 21 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	21-1	David Hafke, New on the Net -- Talk It Up, WINDOWS MAGAZINE, no. 711 (1996).	
	21-2	David Harvey, All the News That's Fit to Speak, NETGUIDE, no. 301 (1996).	
	21-3	David R. Cheriton, et al., A Decentralized Naming Facility (Stanford University, Feb. 1, 1986).	
	21-4	David R. Cheriton, The V Distributed System, COMMUNICATIONS OF THE ACM, vol. 31, no. 3 (Apr. 1988).	
	21-5	David Rapp, I've Got to Get a Message to You, Instant Messaging Started as an MIT Computer-Science Department Project, TECHNOLOGY REVIEW (2002).	
	21-6	DCE 1.0 Security Technology, architectural overview documents, Walter Tuvell, Feb 1997	
	21-7	DCE 1.1 Security Technology, architectural overview documents May 1994	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 22 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	22-1	DCE RPC Internals and Data Structures (Aug. 1993).	
	22-2	Dean Adams, ed., Security Survival: An indispensable guide to securing your business, X/Open Co. (1996).	
	22-3	Decided H.225.0 (June 19, 1996).	
	22-4	Derek C. Oppen, et al., The Clearinghouse: A Decentralized Agent for Locating Named Objects in a Distributed Environment (Association for Computing Machinery, 1983).	
	22-5	Description of New Zephyr Protocol (undated).	
	22-6	Digiphone Specifications, from Q1.11 of Section 1 of the comp.speech FAQ Home Page (dated Jan. 6, 1997)	
	22-7	Douglas B. Terry, et al., The Berkeley Internet Name Domain Server, USENIX Association Software Tools Users Group, Summer Conference, Salt Lake City, Utah (June 12-15, 1984).	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 23 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	23-1	Douglas B. Terry, Structure freeName Management for Evolving Distributed Environments, IEEE 6th International Conference on Distributed Computing Systems, Cambridge, Massachusetts (May 19-23, 1986).	
	23-2	Douglas Brian Terry, Distributed Name Servers: Naming and Caching in Large Distributed Computing Environments, Ph.D. Thesis, University of California, Berkeley (Feb. 21, 1985).	
	23-3	Douglas E. Comer, INTERNETWORKING WITH TCP/IP: VOL. 1: PRINCIPLES, PROTOCOLS, AND ARCHITECTURE, 1st ed. (Prentice-Hall, 1988).	
	23-4	Douglas E. Comer, INTERNETWORKING WITH TCP/IP: VOL. 1: PRINCIPLES, PROTOCOLS, AND ARCHITECTURE, 3d ed. (Prentice-Hall, 1995).	
	23-5	Douglas E. Comer, INTERNETWORKING WITH TCP/IP: VOLUME 1: PRINCIPLES, PROTOCOLS, AND ARCHITECTURES, 2d ed. (Prentice-Hall, 1991).	
	23-6	Douglas W. Johnson, Internet-Connected Phone Calls Dial in to Lower Prices, COMPUTERWORLD (Feb. 19, 1996).	
	23-7	Draft ITU-T Recommendation G.723—Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 & 6.3 KBIT/S (Oct. 17, 1995).	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 24 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	24-1	Draft ITU-T Recommendation H.323 Line Transmission of Non-Telephone Signals: Visual Telephone Systems and Equipment for Local Area Networks Which Provide a Non-Guaranteed Quality of Service, (May 28, 1996).	
	24-2	Draft Recommendation H.323 Visual Telephone Systems and Terminal Equipment for Local Area Networks Which Provide A Non-Guaranteed Quality of Service, Sept. 8, 1995.	
	24-3	Draft Recommendation H.323—Visual Telephone Systems and Equipment for Local Area Networks Which Provide A Non-Guaranteed Quality of Service (May 28, 1996).	
	24-4	E.D. Sykas, et al., Overview of the CCITT X500 Recommendations Series (Butterworth-Heinemann, 1991).	
	24-5	Electric Magic Company Sales Invoices, February 23, 1995 thru December 3, 1995.	
	24-6	Electric Magic Company, Beta Test License Agreement (dated May 30, 1995)	
	24-7	Elizabeth Feinler, et al., RFC 810: DoD Internet Host Table Specification (Mar. 1, 1982).	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT FORM PTO-1449 (modified) Sheet 25 of 67		
	Reexam number	90/010,416
	First Named Inventor	Hutton
	Patent Under Re-Exam	6108704
	Issue Date	2000/08/22
	Group Art Unit	3992
	Examiner Name	KOSOWSKI, ALEXANDER J
	Attorney Docket No.	2655-0188
	Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	25-1	Ellen Massmer, PictureTel Brings Video to the Lan Network World (Sept. 4, 1995).	
	25-2	E-mail from Dale Skran to jtoga@ibeam.jf. intel.com, phone numbers for email list (Jan. 6, 1997).	
	25-3	E-mail from Dale Skran to jtoga@ibeam.jf.intel.com, mailing list to enter (Jan. 6, 1997).	
	25-4	E-mail from Ofer Shapiro to Bob Bell, et al., RE: Destination side gateway problem (July 29, 1996).	
	25-5	E-mail from Sakae Okubo to Experts of ITU-T SG16 Q.12/16, Q.13/16 and Q.14/16, Notice of the Q.12-14/16 Sunriver meeting (July 17, 1997).	
	25-6	Email from Sakae Okubo to yves.robin-champigneu10issy.cnet.fr, et al., Working tools of SG16 experts groups (May 8, 1997).	
	25-7	E-mail from Vineet Kumar to h323implementors@mailbag.jf.intel.com Receiver associating a logical channel with a RTP stream (Aug. 5, 1996).	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT FORM PTO-1449 (modified)	Reexam number	90/010,416
	First Named Inventor	Hutton
	Patent Under Re-Exam	6108704
	Issue Date	2000/08/22
	Group Art Unit	3992
	Examiner Name	KOSOWSKI, ALEXANDER J
	Attorney Docket No.	2655-0188
Sheet 26 of 67	Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	26-1	Erdos, Marlena and Pato, Joseph, "Extending the OSF DCE Authorization System to Support Practical Delegation," February 11, 1993	
	26-2	Eric C. Rosen, IEN 183: Logical Addressing (May 1981).	
	26-3	Eric C. Rosen, IEN 188: Issues in Internetting Part 3: Addressing (June 1981).	
	26-4	Etherphone: Collected Papers 1987-1988, Xerox PARC, CSL-89-2 (May 1989).	
	26-5	Eve M. Schooler, Case Study: Multimedia Conference Control in a Packet-Switched Teleconferencing System, JOURNAL OF INTERNETWORKING: RESEARCH AND EXPERIENCE, vol. 4, no. 2 (June 1993).	
	26-6	Eve M. Schooler, et al., An Architecture for Multimedia Connection Management, Proceedings IEEE 4th Comsoc International Workshop on Multimedia Communications, MM '92, Monterey, California (Apr. 1992).	
	26-7	Eve M. Schooler, The Connection Control Protocol: Architecture Overview (Jan. 28, 1992).	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
FORM PTO-1449 (modified)

Sheet 27 of 67

Reexam number	90/010,416
First Named Inventor	Hutton
Patent Under Re-Exam	6108704
Issue Date	2000/08/22
Group Art Unit	3992
Examiner Name	KOSOWSKI, ALEXANDER J
Attorney Docket No.	2655-0188
Confirmation No.	1061

NON-PATENT REFERENCES

Examiner Initials*	Cite No.	Non-patent Reference bibliographic information, where available	Notes
	27-1	Eve M. Schooler, The Connection Control Protocol: Specification, Version 1.1 (Jan. 29, 1992).	
	27-2	Eve M. Schooler, The Impact of Scaling on Multimedia Connection Architecture, MULTIMEDIA SYSTEMS, vol. 1 (Association for Computing Machinery, 1993).	
	27-3	Exportability of DCE Multi-Crypto Feature by Walter Tuvell, March 5, 1996.	
	27-4	F. Anklesaria, et al., RFC 1436: The Internet Gopher Protocol (A Distributed Document Search and Retrieval Protocol) (Mar. 1993).	
	27-5	FAQ: How Can I Use the Internet as a Telephone, Ver. 0.2 (Apr. 27, 1995)	
	27-6	FAQ: How Can I Use the Internet as a Telephone, Ver. 0.4 (Feb. 23, 1996)	
	27-7	Fax from Ryan Holmquist to Dale Skran (May 30, 1996).	

Examiner Signature		Date Considered	

*Examiner: Initial if reference was considered, whether or not citation is in conformance with MPEP 609. Draw a line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant. Notes: If identified, the following is provided: EA = English Abtract, T = Translation, PF = Patent Family.