United States Patent *5,758,072*
Filepp , et al. **May 26, 1998**

Interactive computer network and method of operation

Abstract

A distributed processing, interactive computer network and method of operation is described. The network is designed to provide very large numbers of simultaneoususers access to large numbers of applications which feature interactive text/graphic sessions. The network includes one or more host computers having application data stores; a plurality of concentrator computers, also including application data stores, the concentrator computers being connected in groups of one of more to each of the host computers; and a plurality of reception system computers connected in groups of one or more to each of the concentrator computers, the reception system computers being arranged so that respective users can request interactive applications at the reception system computers. In accordance with the design, the reception system computers also include application data stores. The method for operating the network includes steps for generating the interactive text/graphic sessions from objects that include data and/or program instructions. Additionally, the method features steps for distributing objects among the data stores of the network computers, and, thereafter, permitting the reception system computer at which an application is requested to selectively collect object required for the application from the network and the respective reception system so that the requested application may be presented at the reception system based on the objects collected. This operation decreases processing demand on the higher-level network elements, permitting them to function primarily as data supply and maintenance resources, thereby reducing network complexity, cost and response time.

Inventors: **Filepp; Robert** (Springfield, NJ); **Gordon; Michael L.** (Dobbs Ferry, NY); **Bidwell; Alexander W.** (New York, NY); **Young; Francis C.** (Pearl River, NY); **Wolf; Allan M.** (Ridgefield, CT); **Meo; Sam** (New York, NY); **Tiemann; Duane** (Ossining, NY); **Abrahams; Lawrence** (Hastings-on-Hudson, NY); **Silfen; Michael J.** (Croton-on-Hudson, NY); **Dalsass; Aldo R.** (Oakland, NJ); **Lee; Florence M.** (Stamford, CT); **Appleman; Kenneth H.** (White Plains, NY)

Assignee: **International Business Machines Corp.** (Armonk, NY)

Appl. No.: **740043**

Filed: **October 23, 1996**

Current U.S. Class:	**709/220**; 709/217
Intern'l Class:	G06F 013/38
Field of Search:	345/800 395/200.11,200.43,200.46,200.47,200.48,200.49,200.5,200.8

References Cited [Referenced By]

U.S. Patent Documents

4289930	Sep., 1981	Connolly et al.	
4575679	Mar., 1986	Simon et al.	
4691340	Sep., 1987	Maeda et al.	
4805119	Feb., 1989	Maeda et al.	
4805134	Feb., 1989	Calo et al.	
5239662	Aug., 1993	Danielson et al.	395/800.

Other References

Adams et al, "Object Management in a CASE Environment," IEEE ACM, May 89, pp. 154-163.
Manson et al, "Remote Control," BYTE Magazine, Jul. 89, pp. 235-240.
Mark J. Stejik et al, "Integrating Access Oriented Programming into a Multiparadigm Environment", IEEE Software, 1986 pp. 10-18.
B. Melamed et al, "Visual Simulation: The Performance Analysis Workstation", IEEE Computers, Aug. 1985 pp. 87-94.
Robert J.K. Jacob, "A State Transition Diagram Language for Visual Programming", IEEE Computer, Aug. 1985, pp. 51-59.
Mark Moriconi et al, "Visualizing Program Designs Through Pega Sys", IEEE Computer, Aug. 1985 pp. 72-85.
Jan Gecsei The Architecture Of Videotex Systems Mar. 25, 1983 Prentice-Hall, Inc.
RW Scheifler & J Getts The X Windows System ACM Trans. On Graphics vol. 5, No. 2 Apr. 1986.

Primary Examiner: Geckil; Mehmet B.
Attorney, Agent or Firm: Seifo; Paul C.

Parent Case Text

This is a division of application Ser. No. 158,026, filed Nov. 26, 1993, which issued as U.S. Pat. No. 5,594,910, application Ser. No. 158,026 being a division of application Ser. No. 388,156 filed Jul. 28, 1989, which issued Sep. 13, 1994, as U.S. Pat. No. 5,387,632, application Ser. No. 388,156 being a continuation in part of U.S. Pat. No. 5,347,632 application Ser. No. 328,790, filed Mar. 23, 1989, which itself was a continuation in part of application Ser. No. 219,931, filed Jul. 15, 1988.

Claims

What we claim is:

1. A method for operating a computer network having a multiplicity of reception systems at which respective users can request applications that include interactive services, the method comprising the steps of:

a. organizing the applications as objects, the objects collectively including data and executable program instructions for generating the applications;

b. distributing objects within the network in accordance with a predetermined plan; and

c. supplying objects to a respective reception system at which an application is requested to enable the respective reception system to selectively collect objects required for the application from the respective reception system and to the extent objects required for the application are unavailable at the respective reception system, to collect required objects not at the respective reception system from the network other than the respective reception system so that the requested application can be presented at the respective reception system based on the objects collected.

2. The method of claim 1 wherein collecting objects includes execution of application software operating at the respective reception system which is capable of interpreting the program instructions included in the objects.

3. The method of claim 2 wherein the organization of the applications into objects includes structuring the objects to include a first section having information for processing the object and a second section including one or more subunits of information including the presentation data and program instructions for executing the object.

4. The method of claim 3 wherein in collecting objects for presenting the requested applications, the respective reception system application software determines if the requested application can be constituted from objects stored at the respective reception system and to the extent it is determined that objects not stored at the respective reception system are required, requesting the required objects from the network.

5. The method of claim 4 wherein the organizing of the applications into objects includes providing the program instructions necessary to execute the requested application in a high-level language having verbs adapted to be interpreted by the application software maintained at the respective reception system.

6. The method of claim 5 further including supplying information to the reception system other than objects, and wherein such other information includes messages.

7. The method of claim 6 wherein the supplying of messages includes structuring the messages to have a first section including information for processing the message, and at least a second section including information representing the message.

8. The method of claim 4 wherein distributing the objects within the network includes storing objects depending upon the likelihood that its associated application will be requested so that objects for applications likely to be requested are likely to be placed at the respective reception system computer and objects for applications least likely to be requested are not likely to be placed at the respective reception system.

9. The method of claim 8 wherein distributing the objects within the network includes storing objects depending in part upon preferences of the respective reception system users.

10. The method of claim 9 wherein distributing objects within the network depends in part upon user preference determined by user previous application requests.

11. The method of claim 10 wherein organizing the application information into objects includes formulating the objects so that they can be used in one or more applications.

12. The method of claim 3 wherein formulating the objects includes formulating the object subunits as elements that can be used in one or more objects.

13. The method of claim 4 wherein the organizing of applications into objects includes collecting the objects in sets which include the information necessary to present a display to the user at a monitor provided at the respective reception system, and wherein one or more screens of display are used to make up an application.

14. The method of claim 4 wherein supplying objects to the respective reception system computer includes downloading objects from the network to the respective reception system when the user logs onto a host so as to maintain the store of objects at the respective reception system computer current.

15. The method of claim 1 wherein distributing objects in accordance with the predetermined plan includes providing the objects with a storage candidacy parameter that determines at least in part the eligibility of the object for retention at respective reception systems.

16. A computer network for providing a multiplicity of users access to a multiplicity of applications, the applications each including data, the apparatus comprising:

a. one or more host computers, each including a data store containing data used in generating applications, at least some of the data used in generating at least some applications being organized as objects;

b. a plurality of reception system computers at which respective users can request applications, each of the respective reception system computers in communication with a host computer, the reception system computers each including a data store containing data used in generating applications and means for processing objects so that applications may be presented at the respective reception systems;

c. data distribution means for distributing data in the network such that data required for an application requested at a respective reception system can be collected from the data store of the respective reception system and to the extent required data is unavailable at the respective reception system, such that required data not at the respective reception system can be collected from the network other than the respective reception system so that the requested application can be presented at the respective reception system based on the objects collected.

17. The computer network apparatus of claim 16 wherein the objects collectively including data and executable program instructions for generating the respective applications; and wherein the object processing means includes means for interpreting the objects so that the respective applications generated from objects can be presented at the respective reception system at which the applications generated from objects are requested.

18. The computer network of claim 17 wherein the object interpreting means includes elements for interpreting objects having a first section including information for processing the object and a second section including one or more subunits of information having presentation data and program instructions for executing the object.

19. The computer network of claim 18 wherein the program instructions for executing applications generated from objects are provided in a high-level language having verbs, and wherein the object interpreting means includes elements for interpreting verbs of the high-level language.

20. The computer network of claim 19 wherein the objects include information for identifying the type and purpose of the object, and wherein the first section of the object includes the object type and purpose information, and wherein the interpreting means includes elements for interpreting the object type and purpose.

Description

BACKGROUND OF THE INVENTION

1. Field of Use

This invention relates generally to a distributed processing, interactive computer network intended to provide very large numbers of simultaneous users; e.g. millions, with access to an interactive service having large numbers; e.g., thousands, of applications which include pre-created interactive text/graphic sessions; and more particularly, to a computer network in which the interactive text/graphic sessions are comprised of pre-created blocks of data and program instructions which may be distributed downwardly in the network for execution at software-enhanced user terminals that decrease processing demand on the higher-level network elements, thus permitting the higher-level elements to function primarily as data supply and maintenance resource and, thereby, reduce network complexity, cost and response time.

2. Prior Art

Interactive computer networks are not new. Traditionally they have included conventional hierarchical architectures wherein a central, host computer responds to the information requests of multiple users. An illustration would be a time-sharing network in which multiple users, each at a remote terminal, log onto a host that provides data and software resource for sequentially receiving user data processing requests, executing them and supplying responses back to the users.

While such networks have been successful in making the processing power of large computers available to many users, problems have existed with them. For example, in such networks, the host has been required to satisfy all the user data processing requests. As a result, processing bottle-necks arise at the host that cause network slowdowns and compel expansion in computing resources; i.e., bigger and more complex computer facilities, where response times are sought to be held low in the face of increasing user populations.

Host size and complexity, however, are liabilities for interactive networks recently introduced to offer large numbers of the public access to transactional services such as home shopping, banking, and investment maintenance, as well as informational services concerning entertainment, business and personal matters.

As can be appreciated, commercial interactive networks must provide interesting and desirable transactional and informational services at low cost and with minimal response times in order to be successful. As a result, unlike military and governmental networks where, because of the compulsory nature of the service performed, costs and content are of secondary concern, in commercial services, the network capital and maintenance expenses must be kept low in order to make the network affordable and, the content maintained interesting to attract both users who would subscribe to the network and merchandisers who would rely on the service as a channel of distribution for their good and services. Further, in addition, to maintaining capital and operating costs low, and quality of content high, it is also essential that network response time be kept to a minimum in order to not only capture and hold the user's attention, but also, quickly free the network to satisfy the requests of other users. Accordingly, and as will be appreciated, the ability of the network to satisfy large numbers of user requests with minimal resources is fundamental to the ultimate success of a commercial, interactive network.

While conventional, previously known time-sharing network designs have attempted to alleviate host complexity and response time problems by providing some processing at the user site; i.e., "smart terminals", still, the storage of the principal data and software resources needed for processing applications at the host continues to create a burden on network complexity and response time which renders the conventional approach unsuited for the large numbers of users contemplated for a commercially viable interactive, informational and transactional network.

SUMMARY OF INVENTION

Accordingly, it is an object of this invention to provide method and apparatus which permit a very large number of users to obtain access to a large number of applications which include interactive text/graphic sessions that have been created to enable the users to obtain informational and transactional services.

It is a further object of this invention to provide method and apparatus which permit the data and program instructions necessary to support applications sessions to be distributed over a computer network.

It is still a further object of this invention to provide method and apparatus that would permit a user to access informational and transactional services available over an electronic gateway.

It is yet a further object of this invention to provide method and apparatus which permit the data and program instructions necessary to support applications sessions to be updated while at the user cites.

It is another object of this invention to provide method and apparatus that would permit informational and transactional services to be provided to users based upon predetermined parameters such as user demographics and/or locale.

It is yet another object of the invention to provide method and apparatus capable of collecting data regarding usage of the network and applications and to condition distribution in the network of data for supporting applications on user reaction to the applications.

Briefly, to achieve the above and other objects and features, the invention includes method and apparatus for operating an interactive network that includes a multiplicity of computer-based user reception systems at which respective users can request applications that include informational and transactional services. In preferred form, the method aspect of the invention includes steps for organizing the applications into objects that collectively include data and executable program instructions for generating the applications, as well as steps for distributing selected objects within the network in accordance with a predetermined plan based on the likelihood a user will request a particular

application. Further, in preferred form, the method includes steps for supplying objects to a reception system requesting an application to enable the requesting reception system to selectively collect objects required for the application from the network and the requesting reception system so that the requested application may be presented based on the objects collected.

Further, in the apparatus aspect of the invention, the network in the preferred form includes one or more host computers, a plurality of concentrator computers connected in groups of one or more to each of the host computers, and a plurality of reception system computers connected in groups of one or more to each of the concentrator computers, the reception system computers being configured to permit respective users to enter requests for interactive applications. Additionally, the method aspect of operating the preferred form of the network apparatus includes steps for establishing data stores at the host computers, the concentrator computers and the reception system computers and, thereafter, distributing application data to data stores maintained, respectively, at the host computers, the concentrator computers and the reception system computers in accordance with a predetermined plan designed to reduce the time required to present a requested application

Still further, the method aspect of operating the preferred form of the network apparatus includes supplying application data to a reception system computer requesting an application so that the requesting reception system computer the data stores being substantially the same in form can assemble the data which makes up the requested application by selectively collecting data from its own data store and the data stores of the respective host computer and concentrator computer to which it is connected.

Further, in preferred form, the method aspect of the invention, features use, of specially structured messages that harmonize and facilitate communications between the different elements of the network and computing elements external to the network that may be called upon to supply information to support the applications.

Also in preferred form, the method aspect of the invention features specially prepared program instructions within the objects that permit the objects to be executed at the reception system in conjunction with the application software.

DESCRIPTION OF THE DRAWINGS

The above and further objects, features and advantages of the invention will become clear from the following more detailed description when read with reference to the accompanying drawings in which:

FIG. 1 is a block diagram of the interactive computer network in accordance with the invention;

FIG. 2 is a schematic diagram of the network illustrated in FIG. 1;

FIGS. 3a and 3b are plan views of a display screen presented to a user in accordance with the invention;

FIGS. 4a, 4b, 4c and 4d are schematic drawings that illustrate the structure of objects, and object segments utilized within the interactive network in accordance with the invention;

FIG. 5a is a schematic diagram that illustrates the configuration of the page template object in accordance with the invention;

FIG. 5b is a schematic diagram that illustrates page composition in accordance with the invention;

FIG. 6 is a schematic diagram that illustrates the protocol used by the reception system to support user

applications in accordance with the invention;

FIG. 7 is a schematic diagram that illustrates major layers of the reception system in accordance with the invention;

FIG. 8 is a block diagram that illustrates native code modules of the reception system in accordance with the invention;

FIG. 9 is a schematic diagram that illustrates an example of a partitioned application to be processed by the reception system in accordance with the invention;

FIG. 10 illustrates generation of a page with a page processing table in accordance with the invention; and

FIG. 11 is a flow diagram for an aspect of the navigation method in accordance with the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT GENERAL SYSTEM DESCRIPTION

With reference to FIGS. 1 and 2, the invention features a network 10 including a plurality of reception units within reception layer 401 for displaying information and providing transactions services. In this arrangement, many users each access network 10 with a conventional personal computer; e.g., one of the IBM or IBM-compatible type, which has been provided with application software in accordance with a preferred form of the invention to constitute a reception system (RS) 400.

As shown in FIG. 1, interactive network 10 uses a layered structure that includes an information layer 100, a switch/file server layer 200, and cache/concentrator layer 300 as well as reception layer 401. This structure maintains active application databases and delivers requested parts of the databases on demand to the plurality of RSs 400, shown in FIG. 2. As seen in FIG. 2, cache/concentrator layer 300 includes a plurality of cache/concentrator units 302, each or which serve a plurality of RS 400 units over lines 301. Additionally, switch/file server layer 200 is seen to include a server unit 205 connected to multiple cache/concentrator units 302 over lines 201. Still further, server unit 205 is seen to be connected to information layer 100 and its various elements, which act as means for producing, supplying and maintaining the network databases and other information necessary to support network 10. Continuing, switch/filer layer 200 is also seen to include gateway systems 210 connected to server 205. Gateways 210 couple layer 200 to other sources of information and data; e.g., other computer systems. As will be appreciated by those skilled in the art, layer 200, like layers 401 and 300 could also include multiple servers, gateways and information layers in the event even larger numbers of users were sought to be served.

Continuing with reference to FIG. 2, in preferred form, each RS 400 is seen to include a personal computer 405 having a CPU 410 including a microprocessor (as for example the type made by INTEL Corporation in its X 86 family of microprocessors), comparison RAM and ROM memory and other associated elements, monitor 412 with screen 414 and a keyboard 424 Further, personal computer 405 may also include one or two floppy disk drives 416 for receiving diskettes 426 containing application software in accordance with this invention for supporting the interactive sessions with network 10 and diskettes 428 containing operating systems software; e.g., MS-DOS, suitable for the personal computer 405 being used. Personal computer 405 may also include a hard-disk drive 420 for storing the application software and operating system software which may be transferred from diskettes 426 and 428 respectfully.

Once so configured, each RS 400 provides: a common interface to other elements of interactive

computer network 10; a common environment for application processing; and a common protocol for user application conversation which is independent of the personal computer brand used. RS 400 thus constitutes a universal terminal for which only one version of all applications on network 10 need be prepared, thereby rendering the applications interpretable by a variety of brands of personal computers of the IBM or IBM-compatible type.

RS 400 formulated in this fashion is capable of communication with the host system to receive information containing either of two types of data, namely objects and messages. Objects have a uniform, self-defining format known to RS 400, and include data types, such as interpretable programs and presentation data for display at monitor screen 414 of the user's personal computer. Applications presented at RS 400 are partitioned into objects which represent the minimal units available from the higher levels of interactive network 10 or RS 400. In this arrangement, each application partition typically represents one screen or a partial screen of information, including fields filled with data used in transactions with network 10. Each such screen, commonly called a page, is represented by its parts and is described in a page template object, discussed below.

Applications, having been partitioned into minimal units, are available from higher elements of network 10 or RS 400, and are retrieved on demand by RS 400 for interpretive execution. Thus, not all partitions of a partitioned application need be resident at RS 400 to process a selected partition, thereby raising the storage efficiency of the user's RS 400 and minimizing response time. Each application partition is an independent, self-contained unit and can operate correctly by itself. Each partition may refer to other partitions either statically or dynamically. Static references are built into the partitioned application, while dynamic references are created from the execution of program logic using a set of parameters, such as user demographics or locale. Partitions may be chosen as part of the RS processing in response to user created events, or by selecting a key word of the partitioned application (e.g., "JUMP" or "INDEX," discussed below), which provides random access to all services represented by partitioned applications having key words.

Objects provide a means of packaging and distributing partitioned applications. As noted, objects make up one or more partitioned applications, and are retrieved on demand by a user's RS 400 for interpretive execution and selective storage. All objects are interpreted by RS 400, thereby enabling applications to be developed independently of the personal computer brand used.

Objects may be nested within one another or referenced by an object identifier (object-id) from within their data structure. References to objects permit the size of objects to be minimized. Further, the time required to display a page is minimized when referenced objects are stored locally at RS 400 (which storage is determined by prior usage meeting certain retention criteria), or have been pre-fetched, or in fact, are already used for the current page.

Objects carry application program instructions and/or information for display at monitor screen 414 of RS 400. Application program objects, called pre-processors and post-processors, set up the environment for the user's interaction with network 10 and respond to events created when the user inputs information at keyboard 424 of RS 400. Such events typically trigger a program object to be processed, causing one of the following: sending of transactional information to the co-applications in one layer of the network 10; the receiving of information for use in programs or for presentation in application-dependent fields on monitor screen 414; or the requesting of a new objects b be processed by RS 400. Such objects may be part of the same application or a completely new application.

The RS 400 supports a protocol by which the user and the partitioned applications communicate. All partitioned applications are designed knowing that this protocol will be supported in RS 400. Hence, replication of the protocol in each partitioned application is avoided, thereby minimizing the size of the

partitioned application.

RS 400 includes a means to communicate with network 10 to retrieve objects in response to events occurring at RS 400 and to send and receive messages.

RS 400 includes a means to selectively store objects according to a predetermined storage criterion, thus enabling frequently used objects to be stored locally at the RS, and causing infrequently used objects to forfeit their local storage location. The currency of objects stored locally at the RS 400 is verified before use according to the object's storage control parameters and the storage criterion in use for version checking.

Selective storage tailors the contents of the RS 400 memory to contain objects representing all or significant parts of partitioned applications favored by the user. Because selective storage of objects is local, response time is reduced for those partitioned applications that the user accesses most frequently.

Since much of the application processing formerly done by a host computer in previously known time-sharing networks is now performed at the user's RS 400, the higher elements of network 10, particularly layer 200, have as their primary functions the routing of messages, serving of objects, and line concentration. The narrowed functional load of the higher network elements permits many more users to be serviced within the same bounds of computer power and I/O capability of conventional host-centered architectures.

Network 10 provides information on a wide variety of topics, including, but not limited to news, industry, financial needs,hobbies and cultural interests. Network 10 thus eliminates the need to consult multiple information sources, giving users an efficient and timesaving overview of subjects that interest them.

The transactional features of interactive network 10 saves the user time, money, and frustration by reducing time spent traveling, standing in line, and communicating with sales personnel. The user may, through RS 400, bank, send and receive messages, review advertising, place orders for merchandise, and perform other transactions.

In the preferred embodiment, network 10 provides information and transaction processing services for a large number of users simultaneously accessing the network via the public switched telephone network (PSTN), broadcast, and/or other media with their RS 400 units. Services available to the user include display of information such as movie reviews, the latest news, airlines reservations, the purchase of items such as retail merchandise and groceries, and quotes and buy/sell orders for stocks and bonds. Network 10 provides an environment in which a user, via RS 400 establishes a session with the network and accesses a large number of services. These services are specifically constructed applications which as noted are partitioned so they may be distributed without undue transmission time, and may be processed and selectively stored on a user's RS 400 unit.

SYSTEM CONFIGURATION

As shown in FIG. 1, in preferred form interactive computer network 10 includes four layers information layer 100, switch and file server layer 200, concentrator layer 300, and reception layer 401.

Information layer 100 handles: (1) the production, storage and dissemination of data and (2) the collection and off-line processing of such data from each RS session with the network 10 so as to permit the targeting of information to be presented to users and for traditional business support.

Switch and file server layer 200 and cache/concentrator layer 300 together constitute a delivery system 20 which delivers requested data to the RSs 400 of reception layer 401 and routes data entered by the user or collected at RSs 400 to the proper application in network 10. With reference to FIG. 2, the information used in a RS 400 either resides locally at the RS 400, or is available on demand from the cache/concentrator 300 or the file server 205, via the gateway 210, which may be coupled to external providers, or is available from information layer 100.

There are two types of information in the network 10 which are utilized by the RS 400: objects and messages.

Objects include the information requested and utilized by the RS 400 to permit a user to select specific parts of applications, control the flow of information relating to the applications, and to supply information to the network. Objects are self-describing structures organized in accordance with a specific data object architecture, described below. Objects are used to package presentation data and program instructions required to support the partitioned applications of a RS 400. Objects are distributed on demand throughout interactive network 10. Objects may contain: control information; program instructions to set up an application processing environment and to process user or network created events; information about what is to be displayed and how it is to be displayed; references to programs to be interpretively executed; and references to other objects, which may be called based upon certain conditions or the occurrence of certain events at the user's personal computer, resulting in the selection and retrieval of other partitioned applications packaged as objects.

Messages are information provided by the user or the network and are used in fields defined within the constructs of an object, and are seen on the user's RS monitor 412, or are used for data processing at RS 400. Additionally, and as more fully described hereafter, messages are the primary means for communication within and without the network. The format of messages is application dependent. If the message is input by the user, it is formatted by the partitioned application currently being processed on RS 400. Likewise, and with reference to FIG. 2, if the data are provided from a co-application database residing in delivery system 20, or accessed via gateway 210 or high function system 110 within the information layer 100, the partitioned application currently being processed on RS 400 causes the message data to be displayed in fields on the user's display monitor as defined by the particular partitioned application.

All active objects reside in file server 205. Inactive objects or objects in preparation reside in producer system 120. Objects recently introduced into delivery system 20 from the producer system 120 will be available from file server 205, but may not be available on cache/concentrator 302 to which the user's RS 400 has dialed. If such objects are requested by the RS 400, the cache/concentrator 302 automatically requests the object from file server 205. The requested object is routed back to the requesting cache/concentrator 302, which automatically routes it to the communications line on which the request was originally made, from which it is received by the RS 400.

The RS 400 is the point of application session control because it has the ability to select and randomly access objects representing all or part of partitioned applications and their data. RS 400 processes objects according to information contained therein and events created by the user on personal computer 405.

Applications on network 10 act in concert with the distributed partitioned applications running on RS 400. Partitioned applications constructed as groups of objects and are distributed on demand to a user's RS 400. An application partition represents the minimum amount of information and program logic needed to present a page or window, i.e. portion of a page presented to the user, perform transactions with the interactive network 10, and perform traditional data processing operations, as required,

including selecting another partitioned application to be processed upon a user generated completion event for the current partitioned application.

Objects representing all or part of partitioned applications may be stored in a user's RS 400 if the objects meet certain criteria, such as being non-volatile, non-critical to network integrity, or if they are critical to ensuring reasonable response time. Such objects are either provided on diskettes 426 together with RS 400 system software used during the installation procedure or they are automatically requested by RS 400 when the user makes selections requiring objects not present in RS 400. In the latter case, RS 400 requests from cache/concentrator layer 300 only the objects necessary to execute the desired partitioned application.

Reception system application software 426 in preferred form is provided for IBM and IBM-compatible brands of personal computers 405, and all partitioned applications are constructed according to a single architecture which each such RS 400 supports. With reference to FIG. 2, to access network 10, a user preferably has a personal computer 405 with at least 512K RAM and a single disk drive 416. The user typically accesses network 10 using a 1,200 or 2,400 bps modem (not shown). To initiate a session with network 10, objects representing the logon application are retrieved from the user's personal diskette, including the R.S. application software, which was previously set up during standard installation and enrollment procedures with network 10. Once communication between RS 400 and cache/concentrator layer 300 has been established, the user begins a standard logon procedure by inputting a personal entry code. Once the logon procedure is complete, the user can begin to access various desired services (i.e., partitioned applications) which provide display of requested information and/or transaction operations.

APPLICATIONS AND PAGES

Applications, i.e. information events, are composed of a sequence of one or more pages opened at screen 414 of monitor 412. This is better seen with reference to FIGS. 3a and 3b were a page 255 is illustrated as might appear at screen 414 of monitor 412. With reference to FIG. 3a, each page 255 is formatted with a service interface having page partitions 250, 260, 280, and 290 (not to be confused with application partitions). Window page partitions 275, well known in the art, are also available and are opened and closed conditionally on page 255 upon the occurrence of an event specified in the application being run. Each page partition 250, 260, 280, and 290 and window 275 is made up of a page element which define the content of the partition or window.

Each page 255 includes: a header page partition 250, which has a page element associated with it and which typically conveys information on the page's topic or sponsor; one or more body page partitions 260 and window page partitions 275, each of which is associated with a page element which as noted gives the informational and transactional content of the page. For example, a page element may contain presentation data selected as a menu option in the previous page, and/or may contain prompts to which a user responds in pre-defined fields to execute transactions. As illustrated in FIG. 3b, the page element associated with body page partition 260 includes display fields 270, 271, 272. A window page partition 275 seen in FIG. 3a represents the same informational and transactional capability as a body partition, except greater flexibility is provided for its location and size.

Continuing with reference to FIG. 3a, advertising 280 provided over network 10, like page elements, also include information for display on page 255, and may be included in any partition of a page. Advertising 280 may be presented to the user on an individualized basis from queues of advertising that are constructed off-line by business system 130, and sent to file server 205 where they are accessible to each RS 400.

Individualized queues of advertising are constructed based upon data collected on the partitioned

applications that were accessed by a user, and upon events the user generated in response to applications. The data are collected and reported by RS 400 to a data collection co-application in file server 205 for later transmission to business system 130. In addition to application access and use characteristics, a variety of other parameters, such as user demographics or postal ZIP code, may be used as targeting criteria. From such data, queues of advertising are constructed that are targeted to either individual users or to sets of users who fall into certain groups according such parameters. Stated otherwise, the advertising presented is individualized to the respective users based on characterizations of the respective users as defined by the interaction history with the service and such other information as user demographics and locale. As will be appreciated by those skilled in the art, conventional marketing analysis techniques can be employed to establish the user characterizations based on the collected application usage data above noted and other information.

Also with reference to FIG. 3b, the service interface is seen to include a command region 285 which enables the user to interact with the network RS 400 and other elements of network 10, so as to cause such operations as navigating from page to page, performing a transaction, or obtaining more information about other applications. As shown in FIG. 3b, command region 285 includes a command bar 290 having a number of commands 291-298 which the user can execute. The functions of commands 291-298 are discussed in greater detail below.

NETWORK OBJECTS

As noted above, in conventional time-sharing computer networks, the data and program instructions necessary to support user sessions are maintained at a central host computer. However, that approach has been found to create processing bottlenecks as greater numbers of users are connected to the network; bottlenecks which require increases in processing power and complexity; e.g., multiple hosts of greater computing capability, if the network is to meet demand. Further, such bottlenecks have been found to also slow response time as more users are connected to the network and seek to have their requests for data processing answered.

The consequences of the host processing bottle necking is to either compel capital expenditures to expand host processing capability, or accept longer response times; i.e., a slower network, and risk user dissatisfaction.

However, even in the case where additional computing power is added, and where response time is allowed to increase, eventually the host becomes user saturated as more and more users are sought to be served by the network. The method and apparatus of this invention are directed at alleviating the effects of host-centered limitations, and extending the network saturation point. In accordance with the invention, this is achieved by reducing the demand on the host for processing resources by structuring the network so that the higher network levels act primarily to maintain and supply data and programs to the lower levels of the network, particularly RS 400, which acts to manage and sustain the user screen displays.

More particularly, the method aspect of the invention features procedures for parsing the network data and program instructions required to support the interactive user sessions into packets, referred to as objects, and distributing them into the network where they can be processed at lower levels, particularly, reception system 400.

In accordance with the invention, the screens presented at the user's monitor are each divided into addressable partitions shown in FIG. 3a, and the display text and graphics necessary to make up the partitions, as well as the program instructions and control data necessary to deliver and sustain the screens and partitions, are formulated from pre-created objects. Further, the object; are structured in

accordance with an architecture that permits the displayed data to be relocatable on the screen, and to be reusable to make up other screens and other sessions, either as pre-created and stored sessions or interactive sessions, dynamically created in response to the user's requests.

In accordance with the method aspect of the invention and as shown in FIG. 4c, the network objects are organized as a family of objects each of which perform a specific function in support of the interactive session. More particularly, the network object family is seen to include 6 members: page format objects 502, page element objects 504, window objects 506, program objects 508, advertising objects 510 and page template objects 500. Within this family, page format objects 502 are designed to define the partitioning 250 to 290 of the monitor screen shown in FIG. 3a. The page format objects 502 provide a means for pre-defining screen partitions and for ensuring a uniform look to the page presented on the reception system monitor. They provide the origin; i.e., drawing points, and dimensions of each page partition and different values for presentation commands such as palette and background color.

Page format objects 502 are referenced whenever non-window data is to be displayed and as noted ensure a consistent presentation of the page. In addition page format objects 502 assures proper tessellation or "tiling" of the displayed partitions.

Page element objects 504, on the other hand, are structured to contain the display data; i.e., text and graphic, to be displayed which is mapped within screen partitions 250 to 290, and to further provide the associated control data and programs. More specifically, the display data is described within the object as NAPLPS data, and includes, PDI, ASCII, Incremental Point and other display encoding schemes. Page element object also control the functionality within the screen partition by means of field definition segments 516 and program call segments 532, as further described in connection with the description of such segments hereafter. Page element objects 504 are relocatable and may be reused by many pages. To enable the displayable data to be relocated, display data must be created by producers in the NAPLPS relative mode.

Continuing with reference to FIG. 4c, window objects 506 include the display and control data necessary to support window partitions 275 best seen in FIG. 3a. Windows contain display data which overlay the base page and control data which supersede the base page control data for the underlying screen during the duration of the window. Window objects 506 contain data which is to be displayed or otherwise presented to the viewer which is relatively independent from the rest of the page. Display data within windows overlay the base page until the window is closed. Logic associated with the window supersedes base page logic for the duration of the window. When a window is opened, the bitmap of the area covered by window is saved and most logic functions for the overlaid page are deactivated. When the window is closed, the saved bit map is swapped onto the screen, the logic functions associated with the window are disabled, and prior logic functions are reactivated.

Windows are opened by user or program control. They do not form part of the base page. Windows would typically be opened as a result of the completion of events specified in program call segments 532.

Window objects 506 are very similar in structure to page element objects 504. The critical difference is that window objects 506 specify their own size and absolute screen location by means of a partition definition segment 528.

Program objects 508 contain program instructions written in a high-level language called TRINTEX Basic Object Language, i.e., TBOL, described in greater detail hereafter, which may be executed on RS 400 to support the application. More particularly, program objects 508 include interpretable program code, executable machine code and parameters to be acted upon in conjunction with the presentation of

text and graphics to the reception system monitors.

Program objects 508 may be called for execution by means of program call segments 532, which specify when a program is to be executed (event), what program to execute (program pointer), and how programs should run (parameters).

Programs are treated as objects to conform to the open-ended design philosophy of the data object architecture (DOA), allowing the dissemination of newly developed programs to be easily and economically performed. As noted above, it is desirable to have as many of these program objects staged for execution at or as close to RS 400 as possible.

Still further, advertising objects 510 include the text and graphics that may be presented at ad partition 280 presented on the monitor screen as shown in FIG. 3b.

Finally, the object family includes page template objects 500. Page template objects 500 are designed to define the components of the full screen presented to the viewer. Particularly, page template objects 500 include the entry point to a screen, the name of the page format objects which specify the various partitions a screen will have and the page element object that contain the display data and partitioning parameters for the page.

Additionally, page template object 500 includes the specific program calls required to execute the screens associated with the application being presented to the user, and may serve as the means for the user to selectively move through; i.e., navigate the pages of interest which are associated with various applications. Thus, in effect, page template objects 500 constitute the "recipe" for making up the collection of text and graphic information required to make the screens to be presented to the user.

Also in accordance with the invention, object 500 to 510 shown in FIG. 4c are themselves made up of further sub-blocks of information that may be selectively collected to define the objects and resulting pages that ultimately constitute the application presented to the user in an interactive text and graphic session.

More specifically and as shown schematically in FIG. 4a, objects 500 to 510 are predefined, variable length records consisting of a fixed length header 551 and one or more self-defining record segments 552 a list of which is presented in FIG. 4c as segment types 512 to 540.

In accordance with the invention, and as shown in FIG. 4b, object header 551 in preferred form is 18 bytes in length and contains a prescribed sequence of information which provides data regarding the object's identification, its anticipated use, association to other objects, its length and its version and currency.

More particularly, each of the 18 bytes of object header 551 are conventional hexadecimal, 8 bit bytes and are arranged in a fixed pattern to facilitate interpretation by network 10. Particularly, and as shown in FIG. 4b, the first byte of header 551; i.e., byte 1, identifies the length of the object ID in hexadecimal. The next six bytes; i.e., bytes 2 to 7, are allocated for identifying access control to the object so as to allow creation of closed user groups to whom the object(s) is to be provided. As will be appreciated by those skilled in the art, the ability to earmark objects in anticipation of user requests enables the network anticipate requests and pre-collect objects from large numbers of them maintained to render the network more efficient and reduce response time. The following 4 bytes of header 551; bytes 8 to 11, are used to identify the set of objects to which the subject object belongs. In this regard, it will be appreciated that, again, for speed of access and efficiency of selection, the objects are arranged in groups or sets which are likely to be presented to user sequentially in presenting the page sets; i.e., screens that go to make up

a session.

Following identification of the object set, the next byte in header 551; i.e., byte 12, gives the location of the subject object in the set. As will be appreciated here also the identification is provided to facilitate ease of object location and access among the many thousands of objects that are maintained to, thereby, render their selection and presentation more efficient and speedy.

Thereafter, the following bytes of header 551; i.e., byte 13, designates the object type; e.g., page format, page template, page element, etc. Following identification of the object type, two bytes; i.e., bytes 14, 15, are allocated to define the length of the object, which may be of whatever length is necessary to supply the data necessary, and thereby provides great flexibility for creation of the screens. Thereafter, a single byte; i.e., byte 16, is allocated to identify the storage characteristic for the object; i.e., the criterion which establishes at what level in network 10 the object will be stored, and the basis upon which it will be updated. At least a portion of this byte; i.e, the higher order nibble (first 4 bits reading from left to right) is associated with the last byte; i.e., byte 18, in the header which identifies the version of the object, a control used in determining how often in a predetermined period of time the object will be updated by the network

Following storage characteristic byte 16, header 551 includes a byte; i.e., 17, which identifies the number of objects in the set to which the subject object belongs. Finally, and as noted above, header 551 includes a byte; i.e., 18, which identifies the version of the object. Particularly the object version is a number to establish the control for the update of the object that are resident at RS 400.

As shown in FIG. 4a, and as noted above, in addition to header 551, the object includes one more of the various segment types shown in FIG. 4c.

Segments 512 to 540 are the basic building blocks of the objects. And, as in the case of the object, the segments are also self-defining. As will be appreciated by those skilled in the art, by making the segments self-defining, changes in the objects and their use in the network can be made without changing pre-existing objects.

As in the case of objects, the segments have also been provided with a specific structure. Particularly, and as shown in FIG. 4a, segments 552 consists of a designation of segment type 553, identification of segment length 554, followed by the information necessary to implement the segment and its associated object 555; e.g., either, control data, display data or program code.

In this structure, segment type 553 is identified with a one-byte hexadecimal code which describes the general function of the segment. Thereafter, segment length 554 is identified as a fixed two-byte long field which carries the segment length as a hexadecimal number in INTEL format; i.e., least significant byte first. Finally, data within segments may be identified either by position or keyword, depending on the specific requirements of the segment.

In accordance with the invention, the specific structure for the objects and segments shown in FIG. 4c would be as described below. In that description the following notation convention is used:

< >	mandatory item
()	optional item
--	item may be repeated .vertline.item.vertline. .vertline.item.vertline.
< >( )	items in a column indicate either/or .vertline.item.vertline. .vertline.item.vertline.

The structure for objects is:

PAGE TEMPLATE OBJECT, ›<header> (compression descriptor) <page format call> (page element call) . . . (program call) . . . (page element selector) (system table call) . . . external reference) (keyword/navigation) . . . !;

As noted above, page format objects 502 are designed to define the partitioning 250 to 290 of monitor screen 414 shown in FIG. 3a.

PAGE FORMAT OBJECT, ›<header> (compression descriptor) (page defaults) <partition definition>!;

PAGE ELEMENT OBJECT, ›<header> (compression descriptor) (presentation data) . . . (program call) . . . (custom cursor) . . . (custom text) . . . (field definition) . . . (field-level program call) . . . (custom cursor type 2) . . . (custom graphic) . . . (field definition type 2) . . . (array definition) . . . (inventory control)!;

Page element objects, as explained, are structured to contain the display data; i.e., text and graphics, to be presented at screen partitions 250 to 290.

WINDOW OBJECT, ›<header> (compression description) <partition definition> (page element call) (presentation data) . . . (program call) . . . (custom cursor) . . . (custom text) . . . (custom cursor type 2) . . . (custom graphic) . . . (field definition) . . . (field level program call) . . . (field definition type 2) . . . (array definition) . . . (inventory control)!;

As noted, window objects include display and control data necessary to support window partition at screen 414.

PROGRAM OBJECTS, ›<header> (compression descriptor) <program data> . . . !.

Program objects, on the other hand, contain program instructions written in higher-level language which may be executed at RS 400 to support the application.

Advertising OBJECT, ›<header> (compression descriptor) (presentation data) . . . (program call) . . . (custom cursor) . . . (custom text) . . . (field definition) . . . (field-level program call) . . . (custom cursor type 2) . . . (custom graphic) . . . (field definition type 2) . . . (array definition) . . . (inventory control)!;

As can be seen, advertising objects are substantially the same as page element objects, with the difference being that, as their name implies, their subject matter is selected to concern advertising.

Continuing, in accordance with the invention, the structure for the object segments is as described hereafter.

PROGRAM CALL SEGMENT

Program call segments 532 are used to invoke programs. Program events will be specified in logical terms and will be mapped by the reception system native software 420 to specific physical triggers (e.g., the "logical" event end of page may map to the physical <ENTER> key). The logical event to be

completed to initiate the program is specified in a one-byte token within the segment. The structure of program call segment 532 is as follows: ##EQU1## where "st" is type; "sl" length; "event" is a one-byte token of the logical event to be completed to initiate the program; "prefix" is a one-byte prefix to an object id or displacement; "object id" is id of the program object 508; "displacement" is a pointer to an imbedded program call segment 532; and "parm" is the parameters specific to the program.

FIELD LEVEL PROGRAM CALL SEGMENTS

Some programs, such as edits, must be triggered at the field level. Field-level program call segments 518 relate program calls to specified field definition segments 516. The stricture of field-level program call segments is as follows: ##EQU2## where "st" is type; "sl" length; "event" is a one-byte token of the logical event to be completed to initiate the program; "field id" is the one-byte name of the field specified in a field definition segment 516 with which this call segment is associated; "prefix" is a one-byte prefix to an object id or displacement; "object id" is id of the program object 506; "displacement" is a pointer to an imbedded program call segment 532; and "parm" is the parameters specific to the program.

PROGRAM DATA SEGMENT

Program data segments 534 contain the actual program data to be processed by RS 400. Program data may include either source code, compiled machine code, macros, storage maps, and/or parameters. The structure of program data segments 536 is as follows:

›<st><sl><type><program data>!;

where "st" is type; "sl" length; "type" refers to the type of program data contained; i.e., (1=TBOL, 2=table data); and "program data" is the actual program to be executed.

COMPRESSION DESCRIPTOR SEGMENT

Compression descriptor segment contains information needed for the decompression of objects compressed in interactive network 10. The segment is a formalization of parameters to be used by a decompression routine residing at the RS 400, using; for example, Huffman encoding well known the art. The structure of compression descriptor segment 513 is:

›<st><sl><table number><length 1> (length 2)!;

where "st" is type; "sl" length; "table number" is a one-byte number corresponding to the "class" indicator in the table structure segment of the appropriate decompression system table object; "length 1" is a two-byte indicator of the length of the segment after compression (not including object header and length of compression descriptor); and "length 2" is a two-byte indicator of the length of the segment before compression (not including object header and length of compression descriptor).

PAGE DEFAULT SEGMENTS

Page default segments 540 specify defaults for the entire page using NAPLPS commands. The structure of page default segment 540 is:

›<st><sl><NAPLPS>!;

where "st" is type; "sl" length; and "NAPLPS" are the commands that may be used to specify default

characteristics of the page.

PARTITION DEFINITION SEGMENT

Partition definition segment 528 describes display screen areas into which data may be mapped. The structure of partition definition segment 528 is:

›<st><sl><partition id><origin><size> (NAPLPS)!;

where "st" is type; "sl" length; "partition id" is a one-byte partition id unique within the current page format object 502; "origin" is the partition origin point, a three-byte NAPLPS point set (absolute, invisible) operand contained the absolute coordinates of the lower left corner of the partition; and "size" refers to partition size, a three-byte NAPLPS point set (absolute,invisible) operand containing the absolute coordinates of the upper right corner of the partition.

PAGE FORMAT CALL SEGMENT

Page format call segment 526 is used by the page template object 500 to specify the particular page format object 502 to be used as the "blueprint" of the page. Page format call segment 526 structure is as follows:

›<st><sl><prefix><object id>!;

where "st" is type; "sl" length; "prefix" is a one-byte prefix to an object id or displacement; and "object id" is the object id of the page format object 502.

PAGE ELEMENT CALL SEGMENT

Page element call segment 522 specifies which data is to be present on the base page and in which page partition the data is to appear. The structure of page element call segment is as follows ##EQU3## where "st" is type; "sl" length; "partition id" is the partition id, as specified in the page format object 502 upon which this object will act; "priority" is a one-byte binary flag indicating priority (from 0-15 with 0 indicating no priority ›FIFO}) of object interpretation (high-order nibble) and of painting (low-order nibble); "prefix" is a one-byte object id or displacement; "object id" is the id of the page element object 504; and "displacement" is a pointer to an imbedded page element object 533.

PAGE ELEMENT SELECTOR SEGMENT

Page element selector segment 524 provides a mechanism by which page elements may be dynamically selected for presentation within a partition. The structure of page element selector segment 524 is: ##EQU4## where "st" is type; "sl" length; "part. id" is the partition id as specified within the page format object 502 upon which the object will act; "priority" is a one-byte binary flag indicating priority (from 0-15 with 0 indicating no priority ›FIFO}) of object interpretation (high-order nibble) and of painting (low-order nibble); "prefix" is a one-byte object id or displacement; "pgm.obj.id" is the object id of the program object 508 used to dynamically select an element object "displacement" is a pointer to an imbedded program object 508, and "parm" is parameters which are used by the program object 508.

SYSTEM TABLE CALL SEGMENT

System table call segments 537 call system table segments for use by the RS 400. Each table entry in a system table segment contains an index-addressable segment (e.g., a set of custom text segments 514).

System table call segments operate in a "locked-shift" mode, meaning that each system table of a particular class will remain operative until a new table is requested for that class of table. System table call segment 542 structure is as follows: ##EQU5## where "st" is type; "sl" length; "prefix" is a one-byte prefix to an object id or displacement; "object id" is the id of a system table segment; and "displacement" is a pointer to an imbedded system table segment.

TABLE STRUCTURE SEGMENT

Table structure segments 531 describe the basic class and composition of system table objects. The structure of table structure segment 531 is:

›<st><sl><class><number of entries><maximum entry length>!;

where "st" is type; "sl" length; "class" is a one-byte identifier indicating the class of the current table (as follows:

x`00`=custom text table

x`01`=custom cursor table

x`02`=custom graphic table

x`03`=custom cursor type 2 table

x`30` thru x`39`=decompression table);

"number of entries is a two-byte field specifying the total number of entries contained in the current table; and "maximum entry length" is a two-byte field specifying the length of the largest entry in the current table.

TABLE ENTRY SEGMENT

Table entry segment 535 contains the actual data that has been placed in tabular form. The meaning of the data is derived from the class indicator in the table structure segment 554. They will be treated as functional equivalent of certain other segments such as custom text segment 514 or custom cursor segment 512. Table entry segment structure is:

›<st><sl><data>!;

where "st" is type; "sl" length; and "data" is the data contained in the entry (text character attributes if table belongs to the custom text class; NAPLPS if the table belongs to the custom cursor class).

EXTERNAL REFERENCE SEGMENT

External reference segment 523 is provided to improve run-time performance by providing the RS 400 with a list of objects that are candidates for pre-fetching. External reference segments 523 contain a list of object-ids which are used within the current page. Each object indicated within this list is called explicitly from the current frame. Object ids specified within the external reference segment 523 will take advantage of the notion of "inheritance." If multiple object ids are contained within the segment, they may inherit high-order bytes from previously specified ids, thus avoiding repetition of information that is inherited (e.g. To specify objects ABC12, ABC22 and ABC37 in this segment, one encodes them

as ABC12, 22, 37). External reference segments 523 operate in a "locked-shift" mode, meaning that each external reference list will be active until the next external reference list is encountered. In the best mode, there should be no more than one external reference segment per page. External reference segment structure is as follows:

›<st><sl><# of ids><priority><prefix><object id>!;

where "st" is type; "sl" length; "# of ids" is a one-byte field specifying the total number of object ids contained in the current segment; "priority" is a one-byte priority value specifying priority of pre-fetch (priorities may be duplicated, in which case they will be processed from left to right); "prefix" is a one-byte prefix to an object id or displacement; and "object id" is the id of an externally referenced object.

KEYWORD/NAVIGATION SEGMENT

Keyword/navigation segments 520 may contain two types of information: (1) references to other page template objects 500 that are either logically higher than the current page template (e.g., a "parent" menu) or references to page template objects 500 outside the current "world" (a logically cohesive group of pages having a single entry point, such as a general map of the interactive service); or (2) a character string to be associated with the current page template object 500, which may be displayed to the user to indicate an alternative path or keyword which could be used to access the current page template. The structure of keyword/navigation segment is as follows:

›<st><sl><#ids> (<prefix><object id>) . . . (character string)!;

where "st" is type; "sl" length; "#ids" is the number of object ids in this segment; "pre-fix" is a one-byte object id prefix; "object id" is an object id associate with the current page as either an upward hierarchical reference or a non-hierarchical reference; and "character string" is the character string to be associated with the current page. (See also, discussion of Jump word navigation, below).

PRESENTATION DATA SEGMENT

Presentation data segments 530 contain the actual data to be displayed or otherwise presented to the user. Presentation data may contain NAPLPS codes, ASCII, and other codes for visual display. Presentation data may in the future contain codes for the presentation of audio signals. The structure of presentation data segment is:

›<st><sl><type><size><presentation data>!;

where "st" is type; "sl" length; "type" is the type of presentation data included in this segment (1=NAPLPS, 2=ASCII); "size" is a NAPLPS operand that defines the upper right portion of the display data; and "presentation data" is the actual data to be presented to the user.

FIELD DEFINITION SEGMENT

Field definition segments 516 define the location of a field, name the field, and specify how data will be acted on within the named field. Field definition segment 516 structure is as follows: ›<st><sl><attributes><origin><size><name><text id> (cursor id) (cursor origin)!; where "st" is type; "sl" length; and the structure is defined as below. "Attributes" of a field define ways in which the user interacts with RS 400 at a rudimentary level. Three basic field types are supported: (1) unprotected fields into which users may enter data; (2) protected fields into which users may position the cursor, function and enter keys, but may not enter data; and (3) skip fields which are inaccessible to the user

keyboard. Additional attributes which may be specified for a field include: numeric input only (unprotected); alphabetic input only (unprotected); foreground color; and background color. Attributes are encoded in two bytes. The first nibble of the first byte is a hexadecimal number (O-F) that represents the foreground color selection from the in-use palette. The second nibble of the first byte is a hexadecimal number (O-F) that represents the background color selection from the in-use palette. The first nibble of the second byte consists of a set of bit flags which, from left to right, indicate:

bit 0 if `1`: protect on;

bit 1 if `1`: automatic skip on;

bit 2 if `1`: numeric input only; and

bit 3 if `1`: alphabetic input only.

The second nibble of the second byte is reserved to accommodate for expansion of network 10.

Continuing, "Origin" is a three-byte NAPLPS point set (relative, invisible) operand that defines the lower left corner of the field; "Size" is a three-byte NAPLPS point set (relative, invisible) operand that defines the upper right corner of the field; "Name" is a one-byte name assigned to the field so that it may be accessible to programs; "Text id" is a ore-byte id of the text characteristics to be associated with the field (e.g., size, gaping, proportional spacing, etc.); "Cursor id" is a one-byte id of the cursor type to be associated with the field; "Cursor origin" is a three-byte NAPLPS operand specifying relative draw point to the cursor, if this operand is not present, the cursor origin point will be assumed to be the same as the field origin point.

FIELD DEFINITION TYPE 2 SEGMENT

Field definition type 2 segments 517 are provided to enhance run-time flexibility of fields. Field definition type 2 segment structure is as follows: ›<st><sl><attributes><origin><size><name><text id><cc ll> (<cursor id> (cursor origin))<# hot spots> (<hs ll><hssize> (hsorigin)) . . . (<cg ll><cgraphic id><cgmode> (cgorigin)) . . . !; where structure is defined below. As with the other segments, "st" describes segment type, and "sl" segment length. Further, "Attributes" describe how the user and RS 400 interact at a rudimentary level. Attributes for field definition type 2 segments 517 are contained in four bytes:

```
Byte 1        Field type
       bit 0   TBOL interpreter indicator:
               no fire; or
               fire
       bits 1-7
               Interaction type
               input (unprotected);
               action (protected);
               display (askip); and
               hidden (dark)
Byte 2       Text Attributes (bit flags)
       bits 0-7  left justify;
                 right justify; and
                 word wrap
Byte 3       Data Type:
       bits 0-7
               alphabetic;
```

```
                numeric;
                password;
Byte 4        Color:
       bits 0-3
                foreground color;
       bits 4-7
                background color.
```

"Origin" is a three-byte NAPLPS point set (relative, invisible) operand that defines the lower left corner of the field. "Size" is a three-byte NAPLPS point set (relative, invisible) operand that defines the upper right corner of the field. "Name" is a one-byte name assigned to the field so that it maybe accessible to the program. "Text id" is a one-byte id of the text characteristics to be associated with the field, such as size, gaping, proportional, etc. "cc ll" is the cursor length; a one-byte field describing the combined length of the cursor id field and the cursor origin field. If the length contains a 1, then the cursor origin operand is not present, in which case, the cursor origin defaults to the field origin point. "Cursor id" is a one-byte id of the cursor type to be associated with the field. "Cursor origin" is a three-byte NAPLPS operand specifying the relative draw point of the cursor. If this operand is not present, the cursor origin point will be assumed to be the same as the field origin point. "# hot spots" is the number of hot spots used by this field. "Hot spots" refers to a set of coordinates that will be selectable by a pointing device, such as a mouse. If the contents of this field are zero, the hot spot for the field will be assumed to be the coordinates that are covered by the custom cursor. "Hot spot sets" facilitate assigning a variable number of hot spots to a field. Each hot spot is described by a set of operand consisting of hot spot length, origin, and size. Each set of such operand describes one hot spot. When using multiple hot spots, multiple sets of operand must be present. "hs ll" or hot spot length is a one-byte binary field describing the length of the hot spot coordinates for a hot spot "instance." If this byte contains zero, the hot spot origin and size default to the coordinates described by the custom cursor. If this byte contains 3, then the hot spot origin point will not follow, but will default to the custom cursor origin point. If this byte contains 6, then both the hot spot origin and size are present. "Hot spot size" is a three-byte NAPLPS x,y coordinate describing the top right corner of the hot spot. "Hot spot origin" is a three-byte NAPLPS x,y coordinate describing the lower left corner of the hot spot. If the hot spot length is equal to 3, this field is not present. In that case, the hot spot origin point defaults to the origin point of the custom cursor (which may have also defaulted to the field origin point). If the hot spot length is equal to 6, then this field is present. A custom graphic operand set contains four operand each of which is given in the Field Definition Segment as shown. Particularly: "cg ll" is the custom graphic set length, which, if 2, then no custom graphic origin is present. In that case,the origin point of the custom graphic defaults to the field origin point; "cg id" is the custom graphic id, a one-byte identifier of a custom graphic string; "cgmode" is the custom graphic mode, which is one byte used to describe variable conditions that apply to the graphic. Defined values include: x'01 : blink; x'02: dynamic; x'03: permanent; and "cgorigin" is the custom graphic origin, a three-byte NAPLPS x,y coordinate indicating the lower left corner of the custom graphic. If this operand is not present, the lower left corner will default to the field origin point.

ARRAY DEFINITION SEGMENT

Array definition segments 515 define the names and relative locations of fields in a row that makes up an array or table. The first row of fields must have been defined using field definition segments 516. The array definition provides a short hand for specifying the replication of selected fields from the initial page. The structure of the array definition segment 515 is as follows:

›<st><sl><#occurrences><vertical gap><field name> . . . !;

where "st" is type; "sl" length; "#occurrences" is a one-byte field describing the number of rows to be generated to create the array (the first row is assumed to be generated from field definition segments 516); "vertical gap" is a NAPLPS point set operand (relative, invisible) containing the DY of inter-row spacing; and "field name" is a one-byte name (from the field definition) of the fields in a row of the array.

CUSTOM GRAPHICS SEGMENT

Custom graphics segment 521 provides a means to package graphics commands. These graphics commands may be related to a field and initiated based on run-time conditions. The structure of custom graphics segment 521 is as follows:

›<st><sl><id><size><NAPLPS>!;

where "st" is type; "sl" length; "id" is a one-byte identifier for this custom graphic; "size" is a three-byte NAPLPS operand specifying upper right corner of the graphic area in a relative mode; and "NAPLPS" are NAPLPS commands to paint the custom image.

CUSTOM CURSOR SEGMENT

Custom cursor segment 512 allows fancy graphics to be associated with cursor positioning in a field. Using this segment, cursor may be defined to any size or shape and may be placed at any desired location relative to their associated fields. The structure of custom cursor segment 512 is as follows:

›<st><sl><id><size><NAPLPS>!;

where "st" is type; "sl" length; "id" is a one-byte identifier for this custom cursor; "size" is a three-byte NAPLPS operand specifying upper right corner of the cursor area in a relative mode; and "NAPLPS" are NAPLPS commands to paint the custom image.

CUSTOM CURSOR TYPE 2

Custom cursor type 2 segment 519 allows cursor to be defined to any size or shape and may be placed at any desired location relative to their associated fields. The structure of custom cursor type 2 segment 519 is as follows:

›<st><sl><id><size> (<ll><NAPLPS>) . . . !;

where "st" is type; "sl" length; "id" is a one-byte identifier for this custom cursor; "size" is a three-byte NAPLPS operand specifying upper right corner of the cursor area in a relative mode; "ll" is the length of the following NAPLPS data; and "NAPLPS" are NAPLPS commands to paint the custom image.

CUSTOM TEXT SEGMENT

Custom text segments 514 allow the definition of custom display of text within a field when non-standard character field size is used (20.times.40 display characters is standard) or custom spacing movement, or rotation of characters is desired. The structure of custom text segments 514 is as follows:

<st><sl><id><NAPLPS>!;

where "st" is type; "sl" length; "id" is a one-byte identifier for this TXT command; and "NAPLPS" are NAPLPS commands specifying character field size, rotation, movement, inter-row and inter-character text gaps.

INVENTORY CONTROL SEGMENT

Inventory control segment 527 is provided to facilitate management of objects. The inventory segment is structured:

›<st><sl><type><inventory number> (sub-number)!;

where "st" is type; "sl" length; "type" is a one-byte indicator showing object usage a follows: 0=no defined use; 1=leader ad; 2=ad campaign completion; 3=leader ad completion; 4-255=reserved for future use); "inventory number" is a unique two-byte number to be used for inventory control and statistics; and "sub-number is the same as inventory number.

As shown in FIG. 4c, the family of object segments also includes imbedded objects and elements; i.e., segments 533 and 525, which represent objects and elements nested; i.e., imbedded within objects. As will be appreciated, the formulation of imbedded objects and elements would be as described above for objects and elements generally and, further, would be consistent with the described structure for segments.

NETWORK MESSAGES

In addition to the network objects, and the display data, control data, and the program instructions they contain as previously described, network 10 also exchanges information regarding the support of user sessions and the maintenance of the network as "messenger". Specifically, messages typically relate to the exchange of information associated with initial logon of a reception system 400 to network 10, dialogue between RS 400 and other elements and communications by the other network elements amongst themselves.

In accordance with the invention, to facilitate message exchange internally, and through gateway 210 to entities externally to network 10, a protocol termed the "Data Interchange Architecture" (DIA) is used to support the transport and interpretation of information. More particularly, DIA enables: communications between RS 400 units, separation of functions between network layers 100, 200, 300 and 401; consistent parsing of data; an "open" architecture for network 10; downward compatibility within the network; compatibility with standard industry protocols such as the IBM System Network Architecture; Open Systems Interconnections standard; support of network utility sessions; and standardization of common network and application return codes.

Thus DIA binds the various components of network 10 into a coherent entity by providing a common data stream for communications management purposes. DIA provides the ability to route messages between applications based in IBM System Network Architecture (SNA), (well known in the art, and more fully described in Data and Computer Communications, by W. Stallings, Chapter 12, McMillian Publishing, Inc. (1985)) and non-SNA reception system applications; e.g. home computer applications. Further, DIA provides common data structure between applications run at RS 400 units and applications that may be run on external computer networks; e.g. Dow Jones Services, accessed through gateway 210 As well, DIA provides support for utility sessions between backbone applications run within network 10 as described hereafter.

In make up, DIA is a blend of SNA and non-SNA based modes, and thus provides a means for