PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L 29/06

A2

(11) International Publication Number: **WO 97/44942**

(43) International Publication Date: 27 November 1997 (27.11.97)

(21) International Application Number: PCT/US97/08679

(22) International Filing Date: 22 May 1997 (22.05.97)

(30) Priority Data:
60/018,256 24 May 1996 (24.05.96) US

(71) Applicant: NARRATIVE COMMUNICATIONS CORP. [US/US]; 204 Second Avenue, Waltham, MA 02154 (US).

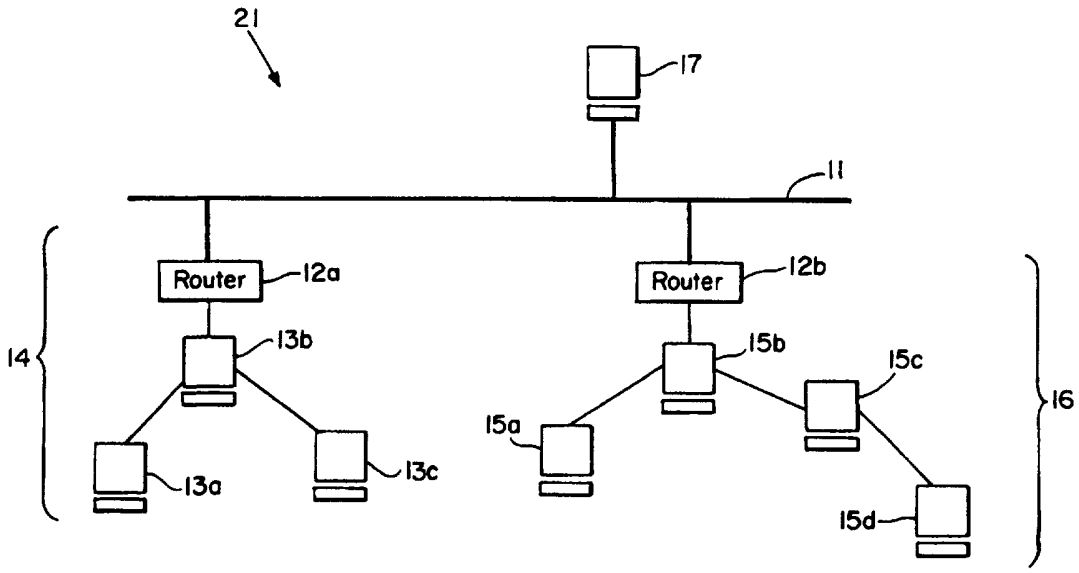
(72) Inventors: KLIGER, Scott, A.; 18 Jacob Amsden Road, Westborough, MA 02581 (US). MIDDLETON, Thomas, M., III; 25 Burditt Avenue, Hingham, MA 02043 (US). WHITE, Gregory, T.; 31 Old Billerica Road, Bedford, MA 01730 (US).

(74) Agents: WAKIMURA, Mary, Lou et al.; Hamilton, Brook, Smith & Reynolds, P.C., Two Militia Drive, Lexington, MA 02173 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: COMPUTER METHOD AND APPARATUS FOR OBJECT STREAMING



(57) Abstract

In a distributed computing environment, a data stream is formed of a sequence of requested objects. The defined order of the sequence of objects is determined from a client request for data. The order may be a default order, or, alternatively, the server may track client criteria to determine the order. For example, the server (17) may track objects previously transmitted in the stream to the client (13) such that there is no duplication of objects. In other instances, the server may select an object from a class of objects, depending upon object quality, bandwidth, client location, and other client-specific criteria. The server compiles and transmits the object data stream in real-time (on-the-fly) based on the criteria. Buffering of data with pausing to rectify buffer debt is provided by the client.

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.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

COMPUTER METHOD AND APPARATUS FOR OBJECT STREAMING

REFERENCE TO CO-PENDING APPLICATION

This application claims the benefit of a United States Provisional Application Serial No. 60/018,256 filed May 24, 1996.

BACKGROUND

"Distributed computing" makes use of a computer network formed out of one or more computers loosely coupled together to allow processes on different computers to communicate with each other and to provide services for each other. One of the most common paradigms of distributed computing is known as the "client-server model", in which consumers of services are called "clients", and make requests of service providers, called "servers".

In object oriented distributed computing, there is a notion of computer entities called "objects". Each object comprises a particular state and a set of defined behaviors. The state is represented by data maintained by the object. The behavior is specified in terms of operations that the object can perform with the operations, typically realized by executable code. Conceptually, the data and the code are inextricably bound together in the object. Objects may be "persistent", that is, they may continue to exist even though they are inactive or the computer on which they exist has failed or has been turned off. Further, objects may issue requests for services to other objects as well as supply services.

Typically, data is held in linear files on a server. When a client requests that data or a part thereof, a connection is formed between the data source (server) and delivery (client) point.

In the prior art there are in general two different types of servers. The first, known as a web server, typically stores data files of a number of different types. Web servers typically communicate with clients over a network such as the Internet using the well known TCP/IP protocol. The second type of server, known as a streaming media server, stores and transmits media files of various types.

More particularly, the web servers presently in use typically store data files in a format known as Hyper Text Markup Language(HTML). HTML permits the web servers to handle container files which reference other files of varying formats. Using HTML, a given web document may include content information in various formats and may also refer to other files by including reference information known as a Uniform Reference Locator (URL). URL's specify the location of remote servers at which files referenced in the HTML file may be located.

Upon receipt of an HTML file from the original web server, a client then must access each document referenced from its source. Each such request typically requires a full cycle of communication with a remote server, including opening a connection socket with the remote server, requesting that the file be transferred, waiting for the file to download, closing the connection, and then, finally parsing the file. To render a given web page may therefore require many such cycles.

The other type of server, known as a streaming media server, has been developed to be particularly suited for multimedia of various types. Such servers may handle single data types, such as a RealAudio™ file, or may

include mixed media types, in formats such as NetShow™ (RealAudio™ is a trademark of Progressive Networks, Inc., and NetShow™ is a trademark of Microsoft Corporation). In any event, media files are typically laid out in a linear fashion in a single file. Thus, when the client requests a file from a streaming server, a socket is simply opened and delivery of data is begun.

The client may perform a caching or buffering operation prior to actual play back of the media file. This ensures that the media file is played back to the user of the client computer in a continuous stream. In particular, the client may calculate in advance an amount of data that it must have on hand prior to actually beginning to render the media file, so that the user has an impression of continuous delivery of the media.

In such a linear streaming server, files may be formatted in advance with a specific communication transfer bandwidth in mind. For example, a Real Audio file may have been compressed for receipt at a baud rate such as 14.4 kilo bits per second (kbps). Another file would be made available for optimum playback at 28.8 kbps. These different file formats provide for allowances in playing back data such that it is rendered in a continuous fashion at the respective rates.

In streaming media server, the connection remains open with the server during the full duration of the play back of the file. Thus, for example, even on a high speed network connection such as a T1 line, if the media file is a ten minute audio file, then the connection will remain open for ten minutes, even though the available information transfer rate on a T1 line is much greater than the audio bandwidth.

In addition, one other disadvantage of streaming media servers is that they typically implement a lossy type of compression algorithm. Thus, if network traffic increases

after file download has begun, bits may be dropped and the quality of the presentation is adversely affected.

Therefore, with streaming media files, the content must typically be specific to each type of client at the targeted bit rate. In addition, the streaming media server is occupied for the real time duration of the media clip, and the presentation may experience degradation based upon the amount of latency in the network.

SUMMARY OF THE INVENTION

Briefly, in the present invention, rather than interpreting container objects that contain references to other objects that must be retrieved by the client opening multiple connections with various servers, and rather than specifying the streaming of data in a single access request, the present invention provides for transmission of data as a stream of objects. In particular, in response to a user or application request for a file, the client issues a request for the file in the form of a sequence of desired objects. The request is presented to the server as a single request that includes a list of multiple objects to be returned to the client.

On the server side, the request is pulled together according to what the client has requested. The requested objects are then sent together in a single stream to the client. To accomplish this, the server analyzes the request to locate the particular objects.

Objects that are available locally to the server are simply added to the outgoing stream, however, the server may also need to query back end file servers and other sources for objects that may be located at other computers in the network. The server then assembles these objects together and provides them to the client in a single object stream.

In one implementation of the invention, typically the default implementation, the server may analyze the reqeust and assemble objects based upon a predefined order, such as specified by an object map file located at the server.

In another implementation of the invention, the server may assemble objects "on-the-fly", based upon client-specific criteria. The present invention thus also permits the mixture of different objects in an object stream, depending upon the particular client or client request. The assembly of object streams may thus occur dynamically based upon any number of client-specific criteria, such as the objects already available to the client, the communication channel bandwidth, the desired presentation quality, client buffer capability, or other parameters associated with the client or the communications channel.

For example, the server may maintain a log of objects already sent to the client, and send only those objects which are not already available at the client computer.

The client may also specify an object class together with information that enables the server to determine which member object of the class is to be placed in the stream.

Such information may include the communication bandwidth, graphical resolution, physical location, or other information which varies from client to client.

The server may also send objects of a particular quality targeted to particular clients. The selection of objects may depend upon client parameters such as observed network latency in real time or desired object quality.

The system may also include buffering, so that the rendering of the set of objects may be delayed until a sufficient amount of data is received at the client.

BRIEF DESCRIPTIONS OF THE DRAWINGS

The foregoing and other objects, features and advantages of the invention will be apparent from the

following more particular description of preferred embodiments and the drawings in which like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention.

Fig. 1 is a block diagram of a computer network employing the present invention.

Fig. 2 is a schematic flow diagram of one embodiment of the present invention.

Fig. 3 illustrates how a server dynamically assembles an object stream in response to a client request.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Illustrated in Fig. 1 is a block diagram of a general computer network 21. A plurality of computers 13, 15, 17 are coupled across a communication channel 11 for communication amongst each other. Various subsets of the computers may themselves form a local area network (LAN) or other local network 13, 15. Each of the various local networks are coupled through a respective router 12a, 12b to channel 11. This enables communication from one local network to another across the channel 11 to form what is known as an "internet". In the preferred embodiment, the present invention is employed on what has become known as the Internet (an international computer network linking an estimated 35 million people to approximately 4.8 million host computers or information sources).

In the preferred embodiment the computers 13, 15, 17 or digital processors employing the present invention are of the PC or mini computer type, or the like, having

processing capabilities of the Intel XX386 processing chip or better. The communication channel 11 is a typical telephone line or other transmission/communication cable handling a 28,800 baud data rate or the like.

A sequence of steps are undertaken by the computers 13, 15, 17 to transfer data in the form of a stream of objects according to the present invention. For example, a given client computer 13a may issue a request for computer data to another computer 17, which acts as a server.

Referring to Fig. 2, the client 13a is a computer that executes an application (or other user interaction) for which certain data needs to be made present. In steps 100 and 102, the client receives an object stream request from the application that includes a global identification of an object map which indicates certain objects existing in the network 21 that the client application program seeks. In particular, the object stream request includes a list, or linear sequence of objects identified a global object identification number. The client 13a transmits the initial request to the server 17, which enters state 200 to transfer back a global identification of the object map listing subject objects.

The client next determines whether the object map is stored locally. Thus, the client 13a checks multiple local memories such as hardware caches, working memory, CD-Rom's and local network memories for the object map in state 108.

If the object map is found locally, then the client analyzes the object map. If the object map is not found locally, then the client requests the object map from the server in state 106.

In response to this request, the server enters states 202, 203 and 204 to (i) initialize a log of object identifications for this client, (ii) initialize a steady

state connection with the client, and (iii) transmit the object map identified by the client 13a.

The client 13a then analyzes the object map in state 108. This is accomplished by the client processing each object referred in on the object map, as indicated by the analyze loop of states 110,112 and 114.

For each such object, the client first determines, in state 110, whether the object is in local cache. If not, then the client 13a adds the identification of the object to a request block. As long as the block is not full, this loop continues with the client adding an identification of each object not found in local cache in state 112.

When the block is full in state 114, the client transmits a request to the server for the block of objects as a list of object identifications.

Upon receipt of this request (i.e., a block of object indentifications), the server 17 then initiates the assembly of a data stream and compiles the stream of objects based on (i) the sequence of requested objects and/or (ii) quality of the objects.

To accomplish this the server constructs blocks of objects to form the data stream, in states 206 through 214. In constructing the blocks, the server 17 maintains a log of objects being transmitted to fulfill the client's request. Specifically, in states 206 and 208, for each object in the block, the server 17 first determines from the log whether the object has previously been transmitted to the particular client 13a. If so, the server 17 enters state 214 to prevent that object from being placed in the current block. Therefore, states 110 and 212 are entered only for the objects that the client is actually in need of. In state 210, then, the server only fetches objects from remote servers which are actually needed by the client 13a. This provides a significant performance advantage, especially where the objects must be obtained elsewhere in

the network 21. As such, the server 17 constructs blocks of data and hence compiles the stream of data for the client 13a in real time, i.e., on the fly, without duplication of any one object throughout the total data stream.

On the receiving end, the client 13a receives the object stream in state 116. More accurately, the data of the object is then delivered to the requesting application of the client for further processing and use.

In summary, for each such object request by the client 13a, an object streaming type communication from the server 17 to client 13a is performed. In particular, a request for a sequence of non-local objects is made by the client 13a and transmitted to the server 17. In turn, the server 17 initializes a log of objects according to object identifications, and then fulfills the request for objects by transmitting the requested objects, in sequence order, which have not previously been transmitted to that client 13a as indicated by the log.

It should be understood that the objects may be computer data structures of various types and formats. The objects may, for example, be compiled in any number of ways within the object oriented computing model well known in the art. For example, objects may include text, graphics, audio, video, and other types of digitized information. Furthermore, objects may be complete data files or only portions of such files. In addition, the objects themselves may be classes that consist of a number of objects grouped together. The object request must then typically include information to specify which member of the object class is desired by the client 13a.

For example, an object class may consist of various versions of a particular graphic object. The selected version may depend upon the desired quality of the final graphical rendering of the object desired by the client

computer 13a. Quality may also have other meaning such as bandwidth, graphical resolution, number of colors, available client memory cache size, and other class criteria that depend upon the type of client computer 13a.

In addition, object classes may depend upon various other client-specific information such as domains, for example. In this scenario, when the client computer 13a is located in one area of the country, such as Massachusetts, a different object may be returned than when the client 13b is located in California, although each of the clients 13a, 13b actually requested the same global object.

Fig. 3 is a diagram illustrating how an example object stream 300 may be assembled by the streaming servers 17, and in particular showing how the object stream does not have to originate from a single source or a single file.

Here the client 13a requests and receives a particular object map 301 consisting of a list of object identifications such as the list (ID1,ID4,ID7,ID6,ID2, ID3), where each object identification IDx indicates a global address for a particular object. According to the process already described in connection with Fig. 2, the client 13a first creates a request block 302 taking into account any local objects 303 which it may already have available. In the particular illustrated example, the client 13a already has local objects (O1, O6) available locally.

After compiling the request block 302, the request block 302 is sent to the streaming server 17. Streaming server 17 receives the request block 302 and then assembles a stream block 310a for the client 13a. It should be understood that other stream blocks 310b may also be constructed for other clients 13b at the same time as the block being constructed for client 13a.

For example, assuming a first time interaction between the client 13a and server 17 in a given stream, the server

17 typically has a default data stream order regardless of user interaction on the client 18 side.

In a first modification, the server 17 may have available certain information concerning the client. Because the server 17 may compose the data stream on-the-fly, the default stream order may be changed in accordance with prior history of requests made on the server 17. That is, on the server processor, an analysis of multiple prior usage of server objects by other clients is made. The server 17 changes the default objects stream 300 order to the closest substitution (i.e., object map) based on demographics of clients 13a having prior server data/object usage. To accomplish such an analysis, a neural network may be employed.

In yet another modification, the streaming server 17 may create the stream block 310a from information that is known about the specific client 13a.

For example, the server 17 may maintain a log 311a of objects that have already been provided to client 13a as well as any available local objects 312 local to the server 17. In the illustrated example the log 311a indicates objects (ID1,ID6) as already having been provided to client 13a. In addition, the available local objects 312 include (O4,O12).

It should be understood, as previously described, that a particular object such as object O4 may actually consist of a class definition, as illustrated, wherein a number of objects comprise the class. For example, object O4 here is actually an object class (O4a,Ob4,...O4x). The streaming server 17 thus also receives together with the object identification request block 302 information as to which particular member of class O4 is appropriately provided to the client 13a.

The streaming server 17 then assembles the stream block 310a as illustrated, which includes all of the

objects that the client 13a has requested. In the illustrated example this includes objects (O4a, O7, O2, O12, O3). After assembly of the stream block 310a the streaming server 17 then provides the stream block 310a as a single object stream 300 to the client 13a.

During assembly of the stream block 310a the streaming server 17 may not have all the necessary objects available in the local object list 312. In such instances the streaming server 17 must query other remote server computers 15a, 15b connected to the network 21 in order to locate the objects. This is done in a manner which is well known in the art by the streaming server 17 making requests of the remote computers 15a, 15b to provide their respective objects that they have made available. In Fig. 3, objects (O7, O2, O4) are located at computer 15a and object (O3) at computer 15b.

An identical object map may be operated on in a different manner by server 17 for client 13b. In particular, although the object map 301b is the same as the objects map 301a, because a different list of object 303b is available to client 13b, the request block 302b created by client 13b will have different object identification numbers (ID4, ID6, ID3). Furthermore, the client parameter(s) provided with the request block by client 13b may very well be different for that provided by client 13a. Thus, an object of a particular quality may be targeted to client 13b which is different than as that supplied to client 13a. For example, when clients 13a and 13b request that object O4 be provided to them, client 13a may actually receive object O4a and client 13b may receive object O4b, despite the fact that the reference O4 was a common global object identification.

The object classes may also be defined as bandwith selectable objects. In particular, the resulting data stream 300 may be assembled based upon observed client

bandwidth availability. Therefore, depending on a periodically calculated data transfer rate, the client 13a or server 17 may choose to request or transmit a data stream of different objects. That is, for a given requested object 04, a specific version 04a, 04b...04x may be selected for transfer from the server 17 to the client 13a as a function of available bandwidth. This function is termed "bandwidth scalability" of the object stream.

In another possible implementation, "object specific compression" is provided by the server, such that on an object by object basis, the object data stream 300 may be compressed one object at a time depending upon client criteria. In the preferred embodiment, the server 17 determines which compressed object version to include in the object data stream 300 at the time of compilation.

The client 13a also manages the amount of data being delivered, i.e., throughput to a client 13a. In particular, this is useful to determine whether there is enough data being timely transferred; that is, whether data is being consumed on the client 13a end faster than the server 17 is delivering the requested objects.

In the preferred embodiment, the client 13a builds a map of uncompressed data consumption. The map tells how fast data is being consumed (used by the client 13a application). The client then measures the throughput (client receipt) of data in real time and contrasts that with the formulated map. Based on the comparison, the client 13a is able to determine how much of the object stream data 300 should be read by a buffer 320a at a time. Thus, the client 13a effectively maps the physical compressed object data stream 300 and logical consumption of data at each point in time.

The client 13a continually monitors the real data throughput versus data consumption. The client 13a wants the delivery-to-consumption ratio to be greater than one so

that the throughput (supply) is keeping up with the consumption (demand). When the delivery-to-consumption ratio is less than one, there is more data being consumed than the amount of data being delivered, such that there is a so-called "buffer debt" on the client 13a side. In that case, the transmission of data needs to take advantage of pause points in the object data stream 300, so that for a period of time, data is not being transmitted over the communication channel 11. In turn, the buffer 320a is allowed to fill up with the requested data and thus decrease or solve the "buffer debt", to assist with proper real time delivery of objects.

The pause points may be pre-defined by the author of the object content, or the pause points may be determined on the fly in accordance with the client's ability to consume data.

In the preferred embodiment this latter implementation is accomplished as follows. The client 13a at each time point, t, computes object data consumption. A running average of throughput such as number of bytes received divided by total time in seconds is employed. An adaptive running average or weighted average or the like may also be used to compute the data consumption. This is also known as the physical throughput at a given time, i, (i.e., (pt_i).

The total logical data (tld) optimally needed at a point in time t is calculated as follows:

$$tld = \sum_{i=o}^{t} rate\ map(i)$$

Buffer debt is then calculated as follows:

$$buffer\ debt = \sum_{i=t}^{end} (tld(i) - pt_i)$$

The client 13a then calculates the buffer debt from time to time. For a buffer debt greater than 0 the client calculates a maximum wait time

maxwait = buffer debt ÷ physical throughput

which equals the number of seconds of pausing needed to rectify the buffer debt. At each wait point or pause point in the object data stream 300, the client 13a will then wait a minimum of the maxwait time or the maximum time allowed at that wait point. For a buffer debt of less than zero, the server 17 transmission of data is ahead of the consumption and thus no pausing during the transmission of the object data stream 300 is warranted.

EQUIVALENTS

While the invention has been particularly shown and described with reference to a preferred embodiment thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the invention as defined by the appended claims.

For example, the log maintained by the server 17 for preventing duplication of objects in the transmitted object data stream 300 to the requesting client may be implemented with tables, cache memory and the like. In the case of caching, a sparse representation of the most recent transmitted objects is maintained in the log. Other caching or log implementations are suitable and are understood to be in the purview of one skilled in the art.

Another example is the term "local to the client". "Local" means in various memories including hard drives, cache memories, CD's and other working memories in the local network involving the client 13a.

CLAIMS

1. In a computer network having a plurality of digital processors loosely coupled to a communication channel for communication among the digital processors, a method of transmitting data from one digital processor to a second digital processor comprising the steps of:

providing a server processor;

providing a requesting processor;

coupling a communication channel between the server processor and requesting processor to enable communication between the server processor and requesting processor;

in the requesting processor, forming a request for a desired sequence of objects;

transmitting the formed request across the communication channel from the requesting processor to the server processor;

in the server processor, in response to receipt of the request, assembling and transmitting, across the communication channel to the requesting processor, a data stream based on the desired sequence of objects, such that a set of objects in a defined order is transmitted in the data stream from the server processor to the requesting processor.

2. A method as claimed in Claim 1 wherein the step of assembling and transmitting a data stream includes:

in the server processor, recording an indication of objects being transmitted to the requesting processor in response to the request; and

based on the recording, preventing plural and subsequent transmission of an object indicated on the

recording by deleting from the data stream objects indicated on the recording such that the transmitted set of objects is formed only of objects which have not been previously transmitted to the requesting processor in response to the request as indicated in the recording.

3. A method as claimed in Claim 1 wherein the step of assembling and transmitting a data stream includes:

in the server processor for each of certain objects, providing multiple versions of the object;

for each of the certain objects, in one of the server processor and requesting processor, determining one version of the object to be optimal for transmission in terms of available bandwidth of the communication channel; and

using the determined version of the object in the set of objects transmitted in the data stream.

4. A method as claimed in Claim 1 wherein the step of forming a request in the requesting processor includes:

for each object in the desired sequence, determining whether the object is locally stored; and

omitting from the request those objects determined to be locally stored but maintaining sequence ordering of the objects in the request.

5. A method as claimed in Claim 1 further including the step of monitoring a rate at which the requesting processor uses requested objects such that rate at which the requesting processor receives requested objects transmitted from the server processor is sufficiently fast to prevent the data stream from

lagging behind the requesting processor use of requested objects.

6. A method as claimed in Claim 1 wherein the step of assembling and transmitting a data stream includes:

in the server processor for each of certain objects, providing multiple versions of the object;

for each of the certain objects, in one of the server processor and requesting processor, determining one version of the object to be optimal for transmission to the client in terms of available client criteria,; and

using the determined version of the object in the set of objects transmitted in the data stream.

7. A method as claimed in claim 6 wherein the client criteria is one of object quality, bandwidth, graphic resolution, or client location.

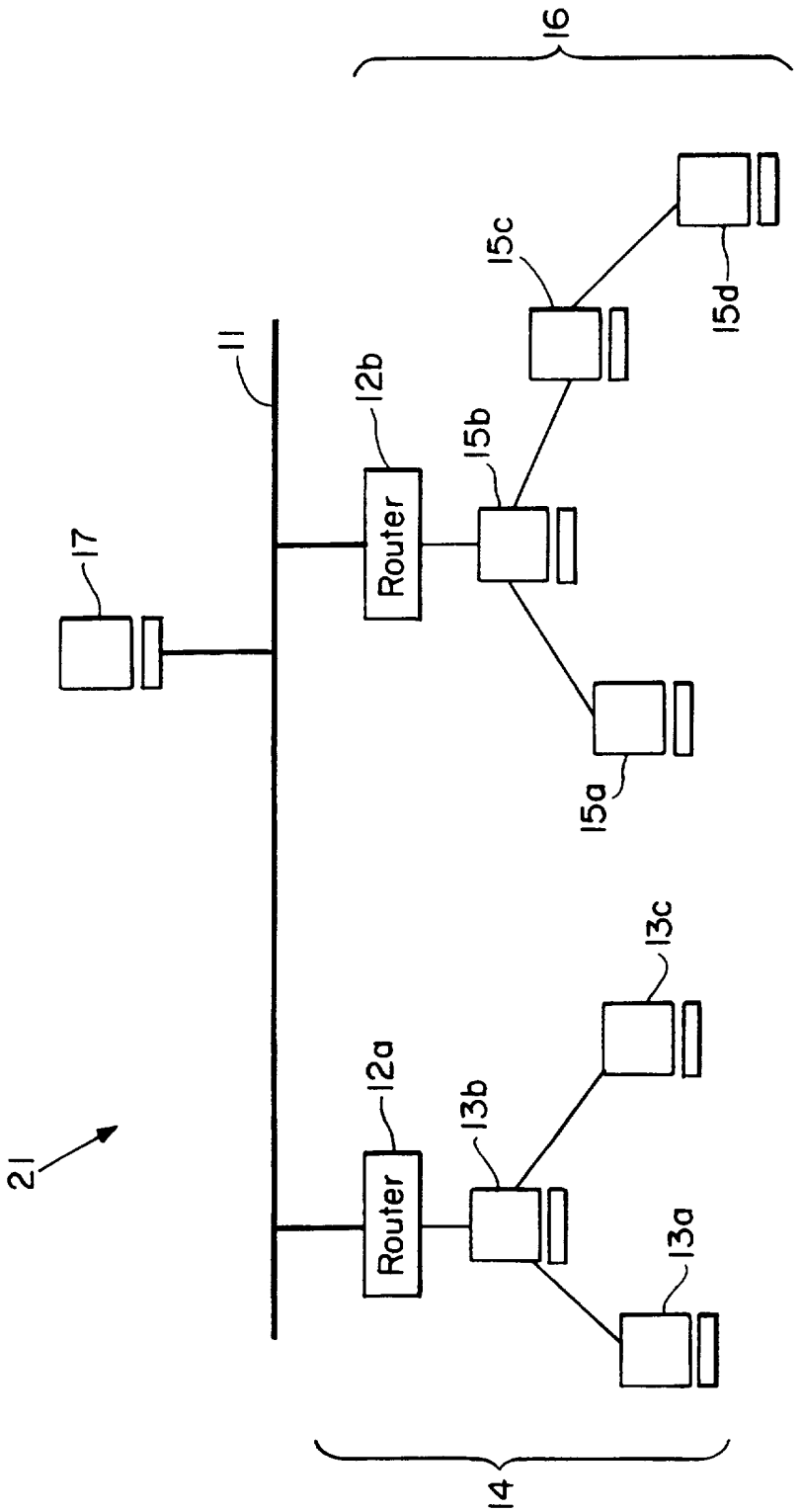


FIG. 1

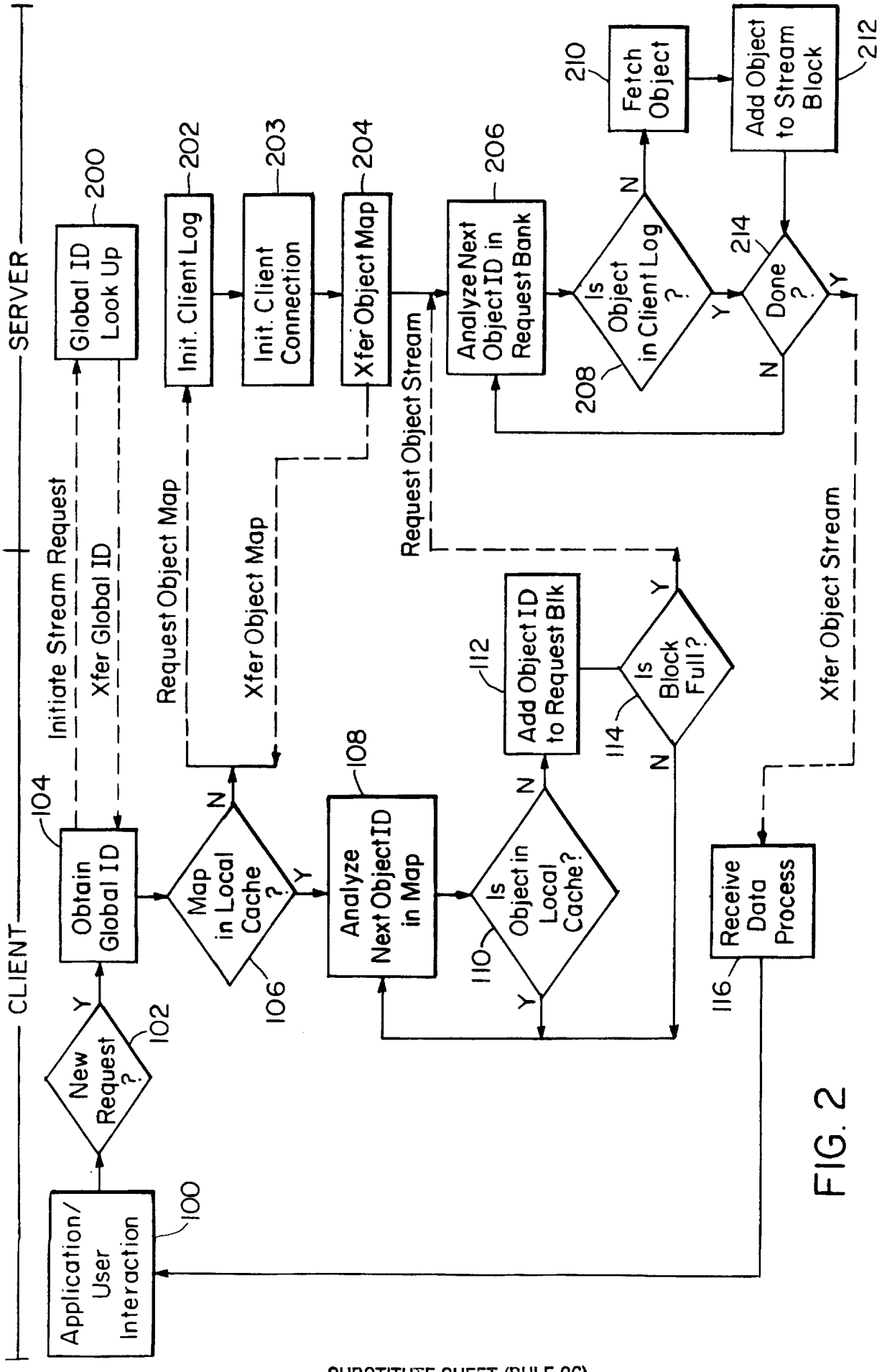

SERVER
CLIENT
Application/ User Interaction
100
New Request ?
102
Y
Obtain Global ID
104
Initiate Stream Request
Xfer Global ID
Global ID Look Up
200
Map in Local Cache ?
106
N
Y
Request Object Map
Init. Client Log
202
Xfer Object Map
Init. Client Connection
203
Xfer Object Map
204
Request Object Stream
Analyze Next Object ID in Request Bank
206
Is Object in Client Log ?
208
N
Y
Fetch Object
210
Done ?
214
N
Y
Add Object to Stream Block
212
Xfer Object Stream
Analyze Next Object ID in Map
108
Is Object in Local Cache?
110
N
Y
Add Object ID to Request Blk
112
Is Block Full?
114
Y
N
Receive Data Process
116

FIG. 2

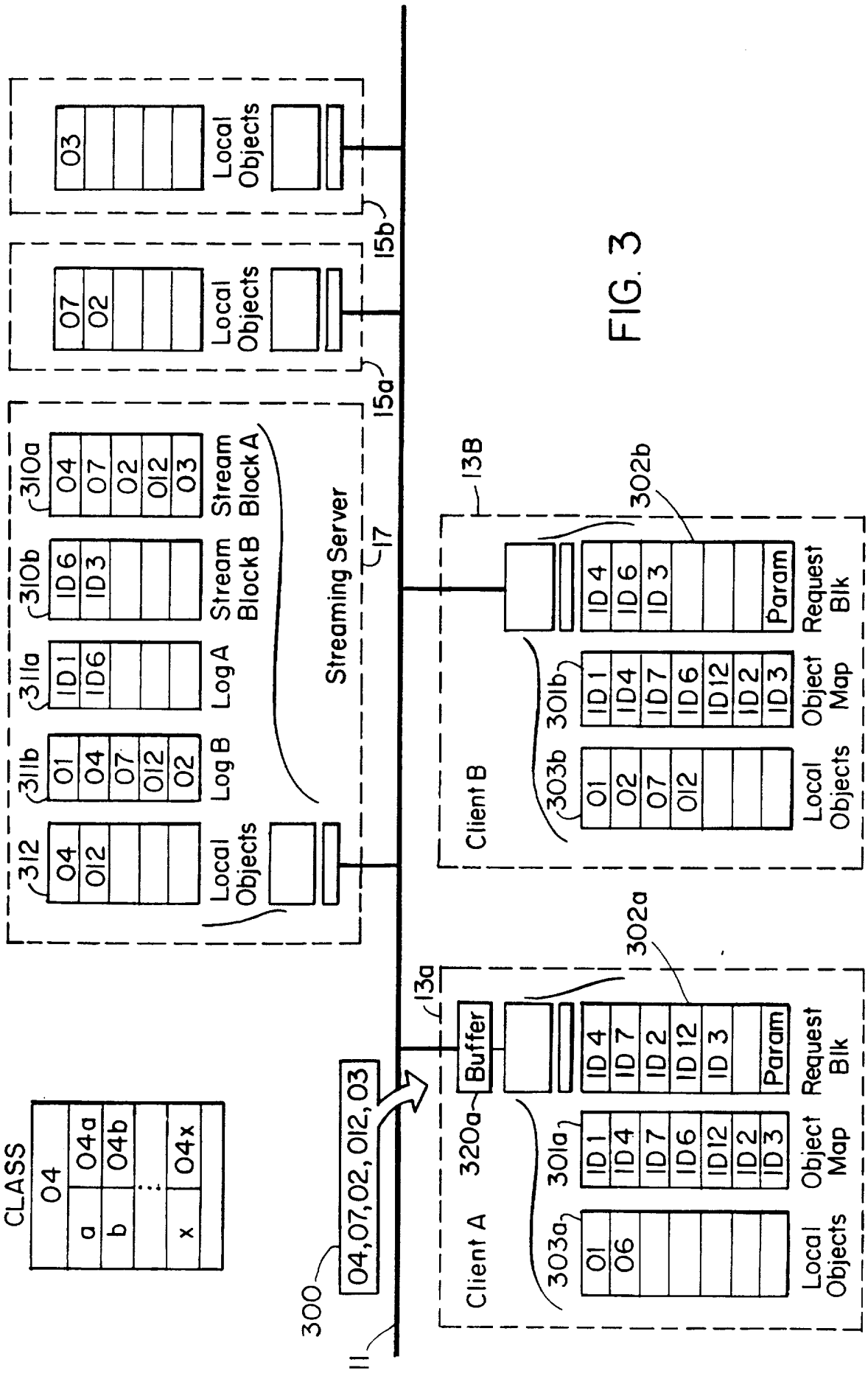

Local Objects
03
15b
Local Objects
07
02
15a
Streaming Server
17
Stream Block A
310a
04
07
02
012
03
Stream Block B
310b
ID 6
ID 3
Log A
311a
ID 1
ID 6
Log B
311b
01
04
07
012
02
312
04
012
Local Objects
13B
302b
Client B
ID 4
ID 6
ID 3
Param
Request Blk
301b
ID 1
ID 4
ID 7
ID 6
ID 12
ID 2
ID 3
Object Map
303b
01
02
07
012
Local Objects
302a
13a
Buffer
320a
Client A
ID 4
ID 7
ID 2
ID 12
ID 3
Param
Request Blk
301a
ID 1
ID 4
ID 7
ID 6
ID 12
ID 2
ID 3
Object Map
303a
01
06
Local Objects
300
04,07,02,012,03
11
CLASS
04
a
04a
b
04b
x
04x

FIG. 3