(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 September 2004 (10.09.2004)

PCT

(10) International Publication Number
WO 2004/077220 A2

(51) International Patent Classification[7]: G06F

(21) International Application Number:
PCT/IN2004/000031

(22) International Filing Date: 29 January 2004 (29.01.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
126/MUM/2003 30 January 2003 (30.01.2003) IN

(71) Applicant *(for all designated States except US)*: **VAMAN TECHNOLOGIES (R & D) LIMITED** [IN/IN]; Pawani Plot, Near Vipul Apartment, Bhakti Marg, Mulund (West), Mumbai 400 080, Maharashtra (IN).

(72) Inventor; and

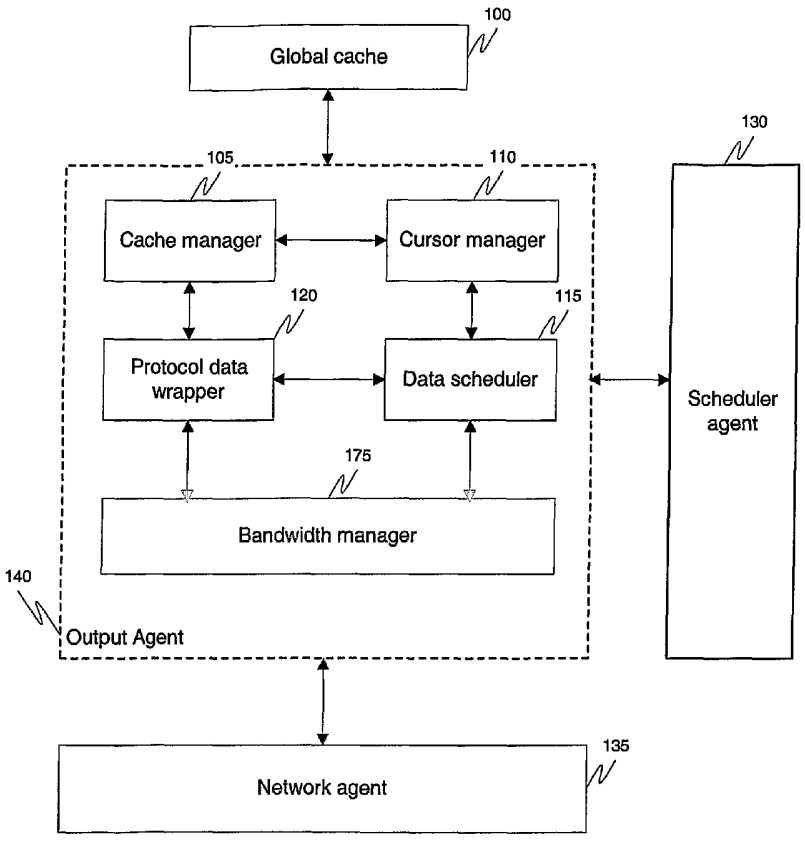
(75) Inventor/Applicant *(for US only)*: **RAO, Vinayak, K.** [IN/IN]; A/4 Vishwakarma Jyoti, Subhash Lane, Malad (East), Mumbai 400 097, Maharashtra (IN).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK,

[Continued on next page]

(54) Title: SYSTEM AND METHOD OF MANAGING AND CACHING DATA IRRESPECTIVE OF SERVER FUNCTIONALITY



(57) Abstract: The present invention relates to the field of managing response data outputs or buffers for various client requests in a server, irrespective of the nature of client requests and server functionality. More particularly the present invention relates to the field of balancing server resources across client requests, irrespective of the connecting protocol, nature of the request and the volume of data expected and provides seamless monitoring and audit capabilities.

WO 2004/077220 A2

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for all designations*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for all designations*

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

TITLE OF INVENTION
System and Method of Managing and Caching Data Irrespective of Server Functionality

BACKGROUND OF THE INVENTION
Typically one classifies machines in a network as clients requesting a service and a server servicing these client requests. The server is supposed to schedule and deliver responses to clients for their requests before their timeout, irrespective of the frequency of requests or the concurrent client count. Based on the functionality of servers, the types of clients vary and also the nature and amount of data varies. For example Browsers work with web servers communicating via the HTTP protocol, FTP clients also work with web servers, but via the FTP protocol. Similarly ODBC clients can be connected to their respective Database servers via Named Pipes, IPX/SPX, Netbeui, Netbios, TCPIP or similar such protocols.

Hence, as observed each of these protocols have their own restriction of data carrying capacity, guarantying content delivery within their respective timeouts. Also, as the concurrency increases the caching mechanism and content delivery without causing clients to starve for data varies largely.
With the Internet becoming a basic communication necessity, the demand and scheduling of bandwidth management across clients has become a significant issue and currently there exists no standard mechanisms to handle this. Further, the demand of web clients for more intelligence and database interactivity has become increasing difficult with major client– server technologies standardizing to TCP/IP as their basic communication protocol.

In most of these servers such as either a database server, a web server or a mail servers with an increase in concurrency the repetitive demand of similar data increased hence each server had developed its own caching mechanism to improve response and reduce disk Input/Output. Generally, for a web server the demand of content remains largely 'read only' with less chances of a demand for 'content write' while in the state of update. Further, web server technology was never designed to handle concurrent read and write requests like a database as the nature of data request relied largely on the Operating System (OS) file system to deliver the read and write demands. In addition, none of

the existing OS' had a mechanism built in their native file system to support concurrent security, sharing, reading and writing on a single file. Hence basic file caching technologies like Smartdrive and Findfast were enhanced to serve the purpose of concurrent file data demands.

There existed a need to standardize data caching and delivery mechanism across various nature of clients irrespective of protocol, pattern or volume of data demand, balancing currently available server resources across clients so as achieve best possible bandwidth distribution preventing client timeouts because of unreasonable spurts or volume of data demands by some clients. There existed a further a need to prepare and format the data to be sent, synchronize request data queries, manage protocol time-outs for every protocol in use when the application is running, controlling and creating bandwidth and initializing other managers in the system.

SUMMARY OF THE INVENTION

To meet the foregoing needs, the present invention provides a software-implemented process, system and method for use in a computing environment with a Dispatcher Agent or an Output Agent, where the Dispatcher Agent serves as a conduit between the client request query and response data the two primary interface agents communicating with the dispatcher are used to balance the bandwidth required across requests.

The agent implementation is purely finite machine based hence unlike procedural programming the functioning of the Output Agent is purely event driven dictated by messages that trigger these event chains but controlled and prioritized by Scheduler.

Further, the Output Agent comprises of a Cache Manager, a Cursor Manager, a Protocol Data Wrapper, a Data Scheduler and a Bandwidth Manager, which interacts with a Scheduler Agent and a Network Agent.

These blocks of the Dispatcher Agent serve different functionality such as the resource management, data age as per pre-configured parameters or server resource constraints, schedules the buffering of the data blocks, monitoring the burst of data, prepares protocol related pre/post amble wrappers for data packets, bandwidth allocation as per the requirement and capable of auditing and archiving parameters, which can be used for billing.

The Network Agent is responsible for transmitting data. The Scheduler Agent does the scheduling and synchronization across requesting processes.

The Output Agent analyzes these incoming source request protocol requirements such as to derive its buffer formats, packet size and protocol timeout limitations and further analysis is carried out to identify the nature of client request and classify the exact request command requirements. The Output Agent checks if the request is from an ODBC or OLEDB Client. Also according to the nature of the request classified, Large Object (LOB) Data can also be handled by the current invention. After the bandwidth is reserved, the Output Agent proceeds to analyze the volume of data. In the event that the request is from a web client, it is further checked for data content, to ascertain if there is larger data content present. Further proceeds to checked to ascertain if the request requires caching. The cursor (implied or explicitly classified) of the client request type notifies caching requirements and corresponding resource allocation for caching which the cache manager manages. The network agent transmits this data chunks and upon client data acknowledgement save the data in cache if demanded so, or reuses the resource buffers for further pending transmissions. Upon final completion of all packet transmissions resources are discarded if requests does not demand caching.

Many of the clients may demand data at a consistent rate and in specific volumes. These may typically be clients playing audio or video data. For such clients a certain portion of the bandwidth needs to be reserved and their demand for data prioritized over other client requests. These requests are similar to FTP clients uploading or downloading data, or any ODBC/OLEDB client demanding LOB data but without a time reference.

5

BRIEF DESCRIPTION OF THE DRAWINGS

The various objects and advantages of the present invention will become apparent to those of ordinary skill in the relevant art after reviewing the following detailed description and accompanying drawings, wherein:

10

Fig 1 is a block diagram illustrating the functional blocks of the preferred embodiment of the invention.

Fig 2 is a flow diagram illustrating the process by which the system manages the output of various processes using the Output Agent.

15

DETAILED DESCRIPTION OF THE INVENTION

While the present invention is susceptible to embodiment in various forms, there is shown in the drawings and will hereinafter be described a presently preferred embodiment with the understanding that the present disclosure is to be considered an exemplification of the invention and is not intended to

20 limit the invention to the specific embodiment illustrated.

In the present disclosure, the words "a" or "an" are to be taken to include both the singular and the plural. Conversely, any reference to plural items shall, where appropriate, include the singular.

25 Referring now to the drawings particularly in Fig 1 is a block diagram illustrating the functional blocks of the preferred embodiment of the invention. As depicted in the block diagram, there is shown a Global Dimensioned Cache 100, wherein each dimension serves a specific purpose and the algorithmic implementation varies as per the nature of data cached from various client sources. There further exists an Output Agent 140 comprising of a Cache Manager 105, a Cursor Manager 110, a Protocol data

30 wrapper 120, a Data Scheduler 115 and a Bandwidth Manager 125, which interfaces with a Scheduler Agent 130 and a Network Agent 135.

The Cache Manager 105 controls the resource management and age of data in cache considering pre-configured parameters or server resource constraints. It decides the cumulative resource across each dimension of the cache dictated by the current count of various clients and the frequency of data demand. It also s dynamically decides the search algorithm to be used for caching based on volume of data to be cached, nature of data type and nature of source request.

The Cursor Manager 110 schedules the buffering of data blocks between the scheduler, sourcing these blocks and the network, thereby successfully sinking it to the client's acknowledgement. The Cursor Manager 110 dictates whether these data blocks need to be cached or the same buffers recycled for subsequent data transport.

The Data Scheduler 115 monitors the burst of data generation from the scheduler and the Network Agent 135. In other words, it queues the sourcing and sinking of data blocks per request operation and delays the scheduler from sourcing blocks till the Network Agent 135 successfully acknowledges it. This process helps controls any network transmission retries or unexpected resource blockage in case the client connectivity dies.

The Protocol Data Wrapper 120 is responsible for preparing protocol related pre/post amble wrappers for data packets. It also acts as a presentation logic manager where the data translation to html or XML is undertaken, based on nature of source client.

The Bandwidth Manager 125 has certain unique tasks. Many of the clients may demand data at a consistent rate and in specific volumes. These may typically be clients playing audio or video data. For such clients a certain portion of the bandwidth needs to be reserved and their demand for data prioritized over other client requests. These requests are similar to FTP clients uploading or downloading data, or any ODBC/OLEDB client demanding LOB data but without a time reference. Also as a configuration requirement a certain bandwidth may be allocated to a particular user or session,

which needs to be tunneled through. Hence, t there exists a need to control and monitor unauthorized resource access. The Bandwidth Manager 125 takes care of such a burst of client requirements and is capable of auditing and archiving parameters, which can be used for billing.

The Scheduler Agent 130 takes care of the scheduling functions and helps to achieve synchronicity in the working of the current invention, while the

The Network Agent 135 is responsible for transmitting data.

Fig 2 is a flow diagram illustrating the process by which the preferred embodiment of the present invention manages the output of various processes using the Output Agent 140.

As soon as a client request for a fresh query is received 200, the first task of the Output Agent 140 is to classify the client source 202. After the classification of the client source the Output Agent 140 proceeds to analyze the source request protocol requirements such as derive its buffer formats, packet size and protocol timeout limitations 204. It then proceeds to analyze the nature of client request and classify the exact request command requirements.

The Output Agent 140 first checks if the request is from an ODBC or OLEDB Client 206. In the event that the request is identified from an ODBC or OLEDB Client then the Output Agent 140 further examines the request to check for Large Object (LOB) Data 208. In the event no LOB data is required then the volume of the data is analyzed 210. In the event a LOB Data is required, the request is checked to ascertain whether it requires caching 212.

In the event the request requires caching the cache analysis is carried out 214. Further, after the cache analysis is carried out, the cache is updated 216. After the cache is updated the protocol wrapper for the packet of data is created (that is the padding header as per source protocols) 218. The Output Agent 140 then proceeds to put this packet in queue for the Network Agent 135 to transmit 220..

The packet is then checked for successful transmission 222. In the event of unsuccessful transmission, an error is reported 224. In the event of successful transmission the request is checked for the need to cache 226. In the event the request requires caching the cache analysis is carried out 214. Further, after the cache analysis is carried out, the cache is updated 216

In the event of no requirement to cache request, the request is handed to reuse buffer 228, after which the count is decremented 230 Further an examination is carried out to check if the count is equal to zero 232. In the event that the count is equal to zero, the Output Agent 140 further proceeds to discard resources 234. In the event of the count is not equal to zero, then the protocol wrapper for the packet of data is created 218.

If after checking whether the request requires caching 212, it is found that the request does not require caching, then the verification for required stream of data is carried out 236. In the event that the required data is a stream, then the bandwidth is reserved as per the data stream 238. After the bandwidth is reserved, the Output Agent 140 proceeds to analyze the volume of data 210. After the analysis of data volume is carried out, the Output Agent 140 proceeds to prepare chunks as per volume and bandwidth constraints 240. After the chunks are prepared as per volume and bandwidth constraints, the packet count is derived 242. and the protocol wrapper for the packet of data is created 218.

On checking if the request is from an ODBC or OLEDB Client 206, in the event the request is non-ODBC or non-OLEDB compliant then the request is checked to ascertain whether it is from an ftp client 244. In the event the request if from an ftp client it is further checked for downloads. In the event of downloads, it is further checked to ascertain if the request requires caching 212.

In the event that the request has no downloads, then the volume of the data is analyzed 210

If on checking if the request is from an ftp client 244, it is found that it is not, then the request is checked to ascertain if it is from a web client 248 such as a browser. In the event that the request is from a web

client, it is further checked for data content, to ascertain if there is a larger data content present 250 then it is further checked to ascertain if the request requires caching 212. . In the event of the request not having large content then the volume of the data is analyzed 210.

If on checking whether the request if from a web client, it is found that it is not, then it is classified as being from a Mail client 252. In the event whether the request if from a mail server, it is found that it is not, then it is classified as being from an Unknown Client 253. In the event the request is for a Mail client 252 it is then checked for attachments 254. In the event of an attachment present then it is further checked to ascertain if the request requires caching 212. However, if it does not have an attachment, then the volume of the data is analyzed 210.

If any of this downloadable data is an audio / video stream which requires data at a consistent time and rate a portion of the bandwidth is reserved with all permissible resource and configuration constraints.

The cursor (implied or explicitly classified) of the client request type notifies caching requirements and corresponding resource allocation for caching which the cache manager manages. Further the scheduler before the beginning of transmission process also provides an estimation of data size. This helps to derive upon the number of packets to prepare as per the source client protocol packet size. Upon knowing the exact count of packets to prepare and queue for transmission the dispatcher agent prepares protocol related buffer wrappers for data blocks. The network agent transmits this data chunks and upon client data acknowledgement save the data in cache if demanded so, or reuses the resource buffers for further pending transmissions. Upon final completion of all packet transmissions resources are discarded if requests does not demand caching.

What is claimed is:

1. A computing system for managing and caching data irrespective of functional servers, comprising:

means for receiving data and instructions to send said data in a predetermined format;

a memory means to queue outgoing said data for said instructions received; and

bandwidth management means to control the rate of transfer of said data across a plurality of clients based on predetermined conditions;

whereby said computing system balances and manages responses and bandwidth to said plurality of clients is managed efficiently

2. The computing system as recited in claim 1 wherein the rate of transfer of said data to each individual client is dependent on the subscription of service of said client.

3. The computing system as recited in claim 1 wherein the presentation of said response to said plurality of clients is based on a user defined format.

4. A method for managing and caching data irrespective of functional servers, comprising:

receiving data and instructions to send said data in a predetermined format;

queuing and storing outgoing said data for said instructions received; and

managing bandwidth to control the rate of transfer of said data across a plurality of clients based on predetermined conditions;

whereby said computing system balances and manages responses and bandwidth to said plurality of clients is managed efficiently

5. The method as recited in claim 1 wherein the rate of transfer of said data to each individual client is dependent on the subscription of service of said client.

6. The method as recited in claim 1 wherein the presentation of said response to said plurality of clients is based on a user defined format.

Fig. 1

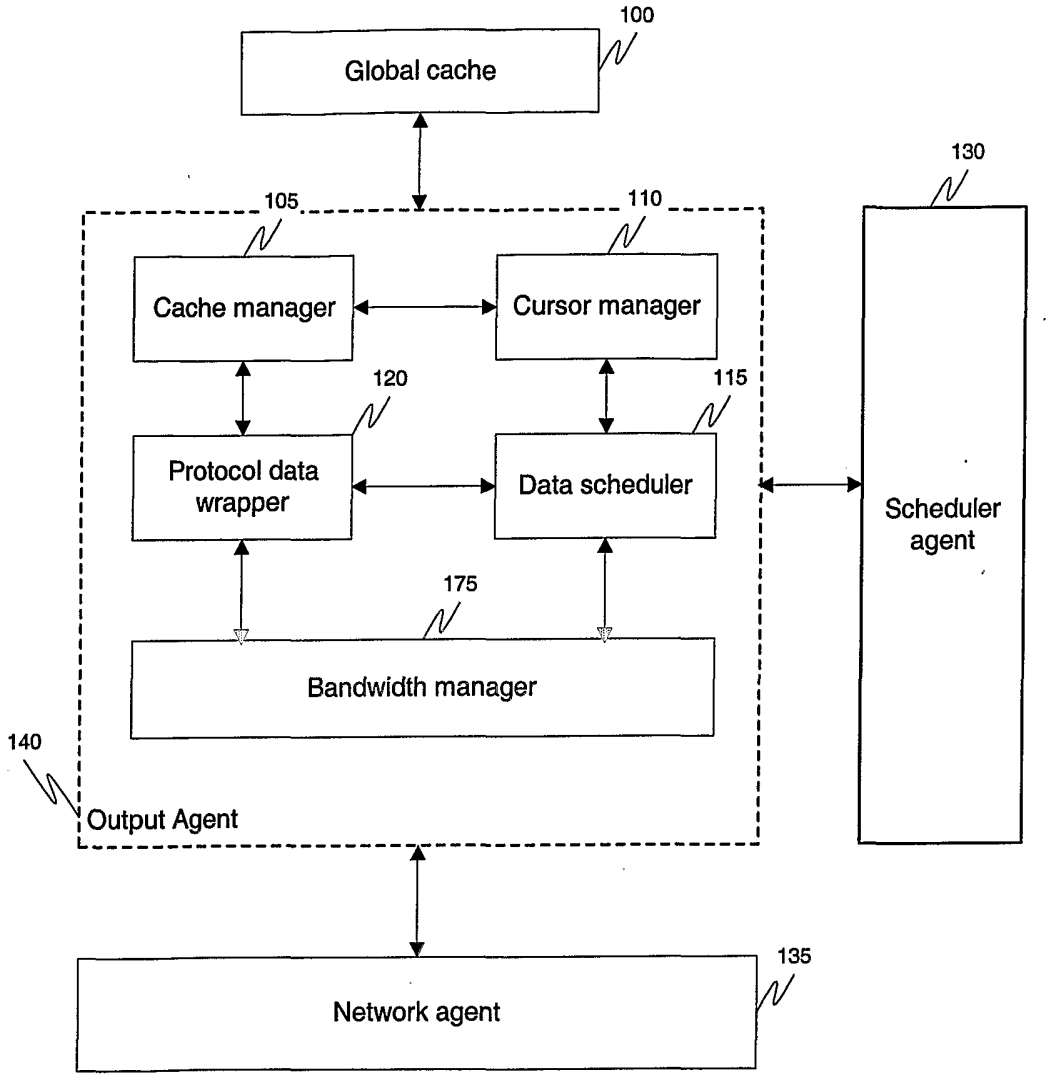
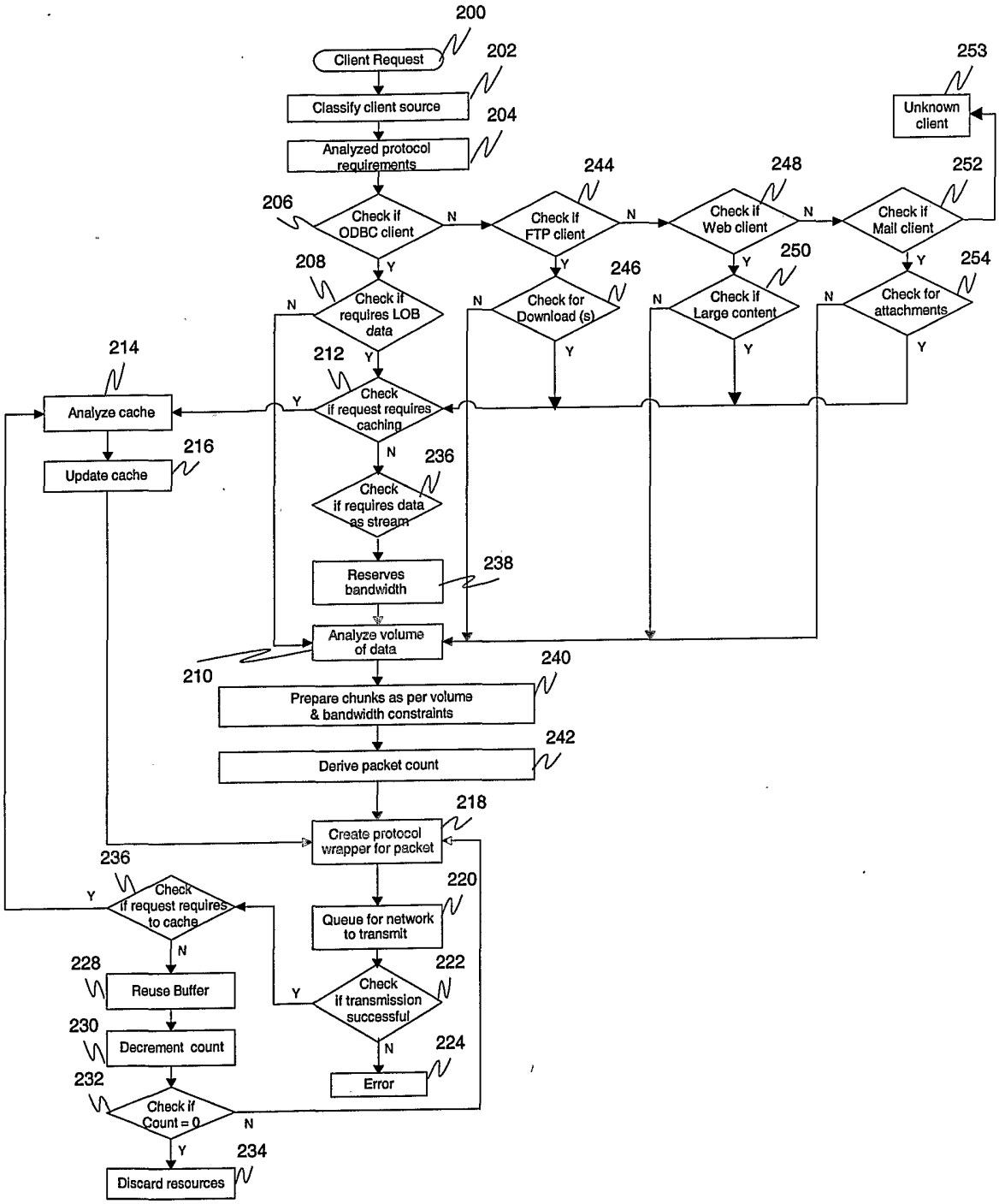

100
Global cache
130
105
110
Cache manager
Cursor manager
120
115
Protocol data wrapper
Data scheduler
Scheduler agent
175
Bandwidth manager
140
Output Agent
135
Network agent

Fig. 2


200
Client Request
202
Classify client source
204
Analyzed protocol requirements
206
Check if ODBC client
N
244
Check if FTP client
N
248
Check if Web client
N
252
Check if Mail client
N
253
Unknown client
Y
Y
Y
Y
208
Check if requires LOB data
N
Y
246
Check for Download (s)
N
Y
250
Check if Large content
N
Y
254
Check for attachments
N
Y
214
Analyze cache
212
Check if request requires caching
Y
N
216
Update cache
236
Check if requires data as stream
238
Reserves bandwidth
Analyze volume of data
210
240
Prepare chunks as per volume & bandwidth constraints
242
Derive packet count
218
Create protocol wrapper for packet
236
Check if request requires to cache
Y
N
220
Queue for network to transmit
228
Reuse Buffer
222
Check if transmission successful
Y
N
230
Decrement count
224
Error
232
Check if Count = 0
N
Y
234
Discard resources