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

CORRECTED VERSION

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 September 2000 (14.09.2000)

PCT

(10) International Publication Number
WO 00/54161 A1

(51) International Patent Classification[7]: G06F 13/14, 13/372, 3/00, H04N 7/10

(21) International Application Number: PCT/US00/06093

(22) International Filing Date: 9 March 2000 (09.03.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/268,512 12 March 1999 (12.03.1999) US

(71) Applicant: DIVA SYSTEMS CORPORATION [US/US]; 800 Saginaw Drive, Redwood City, CA 94063 (US).

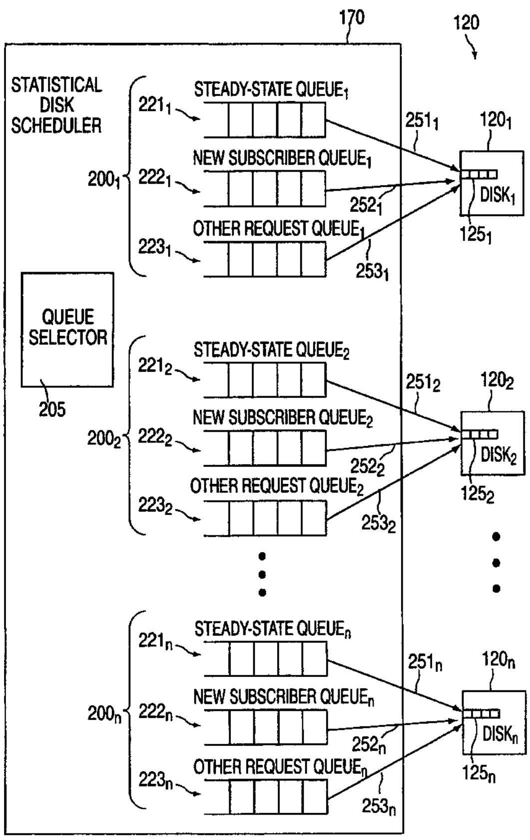
(72) Inventors: CHIN, Danny; 4 Strathmore Place, Princeton Junction, NJ 08550 (US). LERMAN, Jesse, S.; 191 Finnegans Lane, Kendall Park, NJ 08824 (US). TAYLOR, Clement, G.; 215 Hampshire Drive, Plainsboro, NJ 08536 (US). FREDERICKSON, James; 36 Vandeventer Avenue, Princeton, NJ 08542 (US).

(74) Agents: MOSER, Raymond, R. et al.; Thomason, Moser and Patterson LLP, Suite 100, 595 Shrewsbury Avenue, Shrewsbury, NJ 07702 (US).

(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, 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, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD,

[Continued on next page]

(54) Title: QUEUING ARCHITECTURE WITH MULTIPLE QUEUES AND METHOD FOR STATISTICAL DISK SCHEDULING FOR VIDEO SERVERS



(57) Abstract: A queuing architecture and method for scheduling disk drive access requests in a video server. The queuing architecture employs at least two access request queues (221; 222, 223) for each disk drive (120) within a disk drive array. A first queue (221) is for disk access requests by steady-state users currently viewing a program. A second queue, which may include multiple queues (222, 223), is for all other types of access requests including requests by users who wish to begin viewing a program, disk maintenance, meta data synchronizing and the like. A queue selector (205) gives highest priority to requests in the first queue to maintain time deadlines for steady state disk access requests, which are serviced in order of ascending deadlines. Requests from the second queue are serviced only if all of the steady-state requests in the first queue will meet their time deadlines in the worst case.

WO 00/54161 A1

SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(48) Date of publication of this corrected version:
8 November 2001

(15) Information about Correction:
see PCT Gazette No. 45/2001 of 8 November 2001, Section II

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.

QUEUING ARCHITECTURE WITH MULTIPLE QUEUES AND METHOD FOR STATISTICAL DISK SCHEDULING FOR VIDEO SERVERS

The invention relates to methods of scheduling disk access requests in a video server, and, more particularly, to statistical scheduling methods that improve the effective disk bandwidth provided by video servers.

BACKGROUND OF THE DISCLOSURE

Video-on-demand systems allow subscribers to request video programs from a video library at any time for immediate viewing in their homes. Subscribers submit requests to a video service provider via a communication channel (e.g., telephone lines or a back channel through the distribution network that carries the video to the subscriber's home), and the requested video program is routed to the subscriber's home via telephone or coaxial television lines. In order to provide such movie-on-demand services, video service providers use a video server to process subscriber requests, retrieve the requested programs from storage, and distribute the programs to the appropriate subscriber(s). One exemplary system for providing video-on-demand services is described in commonly assigned U.S. patent application serial number 08/984,710, filed December 3, 1997, which is incorporated herein by reference.

In order for video servers to provide good performance, it is crucial to schedule video storage (disk) access requests such that disk bandwidth is maximized. Also, once a subscriber is watching a program, it is imperative to continuously deliver program content to the subscriber without interruption. In addition to distributing content to subscribers, disk bandwidth in a video server is typically also required for operations such as loading content, disk maintenance and file system meta-data syncing. Disk bandwidth may also be reserved for reducing latency in data transfer to subscribers. The number of subscribers that can be properly served concurrently by a video server therefore depends on effective disk bandwidth, which in turn depends on how disk access requests are scheduled.

One of the problems facing current disk scheduling methods is the potential variation in time required to service disk accesses. For example, the internal transfer rate of a Seagate Cheetah disk varies from 152 Mbps on inner tracks to 231 Mbps on outer tracks, and the seek time can vary from 0ms to 13ms depending on how far apart the segments of data are from one another. Given these variations in seek and transfer times and the fact that the server may contain sixteen or more disk drives, it is difficult to determine the effective disk bandwidth of a video server. As a result, current disk scheduling methods allocate a fixed amount of time for every disk access request, regardless of whether the access finishes early. This results in a deterministic system in which the available disk bandwidth is known, but since the fixed amount of time must be large enough to accommodate a worst-case disk access, disk bandwidth is wasted.

Therefore, there is a need in the art for a method and apparatus for scheduling disk access requests in a video server without allocating worst-case access times, thus improving disk bandwidth utilization.

SUMMARY OF THE INVENTION

The disadvantages associated with the prior art are overcome by a method of the present invention, called Statistical Disk Scheduling (SDS), which exploits the fact that disk access times are on average significantly less than the worst case access time. The SDS finds use in improving video server functionality by increasing the bandwidth utilization of the storage medium in the following manner: worst case performance is used for priority operations (e.g., user read operations) but the bandwith created by better than worst case performance is used for non-priority operations such as loading content onto the disk drives and disk maintenance. As a result, bandwidth for loading content and disk maintenance, or file system meta-data syncing does not have to be specifically reserved, thus increasing the number of users that can be served simultaneously by the video server.

SDS maintains at least two queues and a queue selector. The first queue is an access request queue for access requests from a current user that are presently viewing a program and the second queue is for all other forms of access requests. The second queue may comprise multiple queues to provide a queuing hierarchy. The requests are ordered in each of the queues to optimize the bandwidth and ensure that the data to the current users is not interrupted such that a display anomaly occurs. The queue selector identifies the queue that will supply the next access request to a disk queue. The selected requests are sent to the disk queues for execution. The disk queues are generally located on the disk drives and are generally not accessible except to place a request in the queue for each disk drive. The requests are then executed on a first-in, first-out manner. In effect, the invention defers disk use to the latest possible moment because once the request is in the disk queue it is more difficult to change. The inventive queue structure provides opportunities to alter the disk access requests and their execution order prior to sending the requests to the disk queue. If a disk queue is not used, i.e., the disk drive does not have an internal queue, then the access requests are sent one at a time from the SDS to the disk drive for execution.

More specifically, the preferred embodiment of the SDS maintains three queues for each disk based on the type and priority of disk access requests, and a queue selector for managing queue selection. Selected requests are forwarded from the three queues to the disk such that bandwidth utilization is maximized, while giving highest priority to subscribers currently viewing a program so that their program streams are generally not interrupted. (Subscribers currently viewing a program are referred to as "steady-state" subscribers.) SDS dynamically monitors bandwidth utilization to determine when lower-priority requests can be scheduled without affecting on-time completion of the higher priority steady-state subscriber requests. In order to keep the disks busy and maximize disk bandwidth utilization, disk command queuing may be employed to ensure that the disk can begin seeking for the next access immediately after it finishes the data transfer for the current disk access.

Furthermore, popular content is migrated to the faster (outer) tracks of the disk drives to reduce the average access time and improve performance.

BRIEF DESCRIPTION OF THE DRAWINGS

The teachings of the present invention can be readily understood by considering the following detailed description in conjunction with the accompanying drawings, in which:

FIG. 1 depicts a high-level block diagram of a video-on-demand system that includes a generic video server incorporating the present invention;

FIG. 2 depicts the queuing architecture of the Statistical Disk Scheduler used to perform the method of the present invention;

FIG. 3 depicts a flowchart specification of the SDS Selection Procedure;

FIG. 4 depicts a flowchart specification of the Scheduling Interval Procedure;

FIG. 5 depicts a round-robin version of the Scheduling Interval Procedure;

FIG. 6 depicts a flowchart specification of the Command Completion Procedure;

FIG. 7 depicts a flowchart specification of the method of the present invention; and

FIG. 8 shows the software process architecture for a preferred multi-threaded implementation of the method of the present invention.

To facilitate understanding, identical reference numerals have been used, where possible, to designate identical elements that are common to the figures.

DETAILED DESCRIPTION

FIG. 1 depicts a video-on-demand system that utilizes a generic video server incorporating the teachings of the present invention. Specifically, video-on-demand system 100 contains a video server 110 that communicates with a plurality of disks 120 via a Statistical Disk Scheduler (SDS) 170. In addition to the SDS 170, video server 110 contains a CPU 114 and memory element 117. SDS 170 is coupled to disks 120 by paths 130 (e.g., fiber channel), and memory 117 by data path 177. The video server sends access requests along paths 130 to disks 120, and each disk 120 has its own internal queue 125 for buffering access requests. Data read from the disks are transmitted back to the video server along paths 130_n (where n is an integer greater than zero). The paths 130_n are "daisy chained" to form a data transfer loop 131, e.g., a fiber channel loop. Although one loop is depicted, multiple loops may be employed to interconnect subsets of the disk drives such that the data transfer rate amongst the disk drives and the video server is increased over that of a single loop system. The video server contains a Distribution Manager 180 that receives the data transmitted along paths 130_n and loop 131 and distributes this data to subscribers 160 via a transport network 140. Additionally, disks 120 send messages called *command completion messages* (to be discussed later) to the SDS 170 along paths 130.

The transport network 140 is typically, but not exclusively, a conventional bi-directional hybrid fiber-coaxial cable network. Subscribers 160 are coupled to the transport network 140 by paths 150 (e.g., coaxial cable). Additionally, transport network 140 forwards subscriber access requests along path 175 to the SDS 170, and receives video data from Distribution Manager 180 via path 185.

Commonly assigned U.S. patent application serial number 08/984,710, filed December 3, 1997, which is incorporated herein by reference, describes an information distribution system, known as the OnSet™ system, that uses a video server that may benefit from the present invention. Additionally, the video server of the OnSet system is described in U.S. patents 5,671,377 and 5,581,778 which are both herein incorporated by reference.

The SDS 170 performs the method of the present invention. A logical representation of the SDS data architecture is shown in FIG. 2. In a physical representation, the outputs of each queue are connected to the data loop (131 of FIG. 1). In the depicted embodiment, the SDS queuing architecture contains three queues for each disk 120 and a queue selector 205 for managing queue selection, i.e., the queue selector determines which queue is to transfer the next access request to a disk drive. For simplicity, the logical representation is more easily understandable. Although FIG. 2 depicts three queues for each disk drive, a greater or lesser number of queues may be used to fulfill the invention, i.e., at least two queues should be used; one for the "steady-state" requests and one for all other requests.

In the three queue embodiment of the SDS 170, a *steady-state subscriber queue (SSQ)* 221 is used for "steady-state" subscriber disk reads for active streams (i.e., continuous content retrieval for distribution to subscribers currently watching a program.) Disk access requests in SSQ 221 are assigned the highest priority. A *new subscriber queue (NSQ)* 222 is for subscriber requests to begin viewing a program or perform other program related commands, i.e., non-steady state commands such as fast forward or rewind that in essence are a request for a new data stream. Disk access requests in NSQ 222 are assigned medium priority. The *other request queue (ORQ)* 223 is for all non-subscriber operations, such as loading content, disk maintenance, and file system meta-data syncing. Disk access requests in ORQ 223 are assigned the lowest priority.

Queues 221_n, 222_n, and 223_n are collectively called the *SDS queues* 200_n, where n is an integer greater than zero that represents a disk drive 120_n in an array of disk drives 120. For each disk 120_n, the queue selector 205 selects requests from the three SDS queues 221_n, 222_n, and 223_n and forwards the requests to the corresponding disk queue 125_n. Each request has an associated worst-case access time based on the type of request and data transfer size. The worst-case access time can be fixed, or dynamically computed based on prior access time statistics. Additionally, each steady-state subscriber request has a time deadline for when the request must complete in order to guarantee continuous video for that subscriber. Disk requests in the NSQ and ORQ generally do not have time deadlines.

Requests in the SSQ 221_n are ordered by time deadline so that the request at the front of the queue has the earliest deadline. Consecutive SSQ requests with the same time deadline are ordered by logical disk block address according to an *elevator algorithm*. The elevator algorithm is a disk scheduling algorithm well-known in the art in which the disk head travels in one direction over the disk cylinders until there are no more requests that can be serviced by continuing in that direction. At this point, the disk head changes direction and repeats the process, thus traveling back and forth over the disk cylinders as it services requests. Since requests in the NSQ and ORQ do not generally have deadlines, they may be ordered on a first come first serve basis, or according to some other desired priority scheme.

In order to keep the disks 120 busy and maximize disk bandwidth utilization, disk command queuing may be employed to ensure that the disk can begin the seek for the next access immediately after it finishes the data transfer for the current disk access. When a steady-state request needs to access a sequence of multiple disks, the request is initially added to the SSQ 221_1 of the first disk 120_1. After this request is selected for servicing by the first disk 120_1, the request is added to the second disk's SSQ 221_2 as soon the video server begins sending the data that was recalled from the first disk 120_n to the subscriber. Steady-state requests are similarly added to the SSQ 221_n of each successive disk 120_n.

The queue selector 205 employs an *SDS Selection Procedure* to select requests from the three SDS queues 200_n and forward the requests to an associated disk queue 125_n located within each of the disk drives 120_n. The SDS Selection Procedure uses worst-case access times, request priorities, and time deadlines in determining which request to forward to the disk queue. The general strategy of the SDS Selection Procedure is to select a non-SSQ request only when such a selection will not cause any of the SSQ 221_n requests to miss their time deadlines, even if the non-SSQ request and all requests in the SSQ 221_n were to take their worst-case access times. If such a guarantee cannot be made, then the first request in the SSQ is always selected. As an optional step, once a request is selected, the SDS Selection Procedure checks whether the data for the selected read request is already in cache (if caching is used). If this is the case, the disk access can be discarded and the Selection Procedure is repeated. Otherwise, the selected request is removed from the SDS queue 221_n and forwarded to an associated disk queue 125_n.

FIG. 3 depicts a flow diagram of the SDS Selection Procedure 300. First, the Selection Procedure checks whether the first entry in the NSQ can be selected while guaranteeing that all SSQ requests will meet their time deadlines in the worst case (step 320), where worst case is defined by the system. Generally, the worst case value is the access value having a per user error rate that is acceptable.

Each queue maintains "a sum of the worst case values" selector that performs a worst case analysis and selects the queue that will be used (i.e., steps 320 and 330) to send the next command to the disk drive. The following pseudocode represents the operation of such a selector.

```
1) perform worst case analysis
     returns remaining time (the amount of time left
     on the SSQ if all commands take worst case time
     to execute, if the SSQ is empty, the remaining
     time is infinity)

2) if NSQ is !empty && NSQ.head.worstcase < remaining
   time
      take request off NRQ
   else if NSQ is empty && ORQ is !empty &&
      ORQ.head.worstcase < remaining time
      take request off ORQ
   else if SSQ is !empty
      take request off SSQ
      if request.deadline - request.worstcase >
      current time
          request missed deadline, terminate request,
          try selector again
else
     no requests pending
```

Preference is given to the NRQ over the ORQ, only take things off the ORQ if the NSQ is empty.

The ORQ.head.worstcase and NSQ.head.worstcase are the respective worstcase access times to fulfill the next request in the ORQ and NSQ. The "remaining time" value is computed as follows:

remaining time = disk Q Remaining Time (SSQ_n) - disk Q worst case (PQ_n)

```
disk Q Remaining Time (Q, now) {
     sum = 0
     min = MAX
     for each entry in Q {
     sum + = entry → worstcase
     left = entry → deadline + sum - now;
     if (left < = 0 || entry → deadline > now) { /*
     out of time */
     min = 0;
     break;
     }
     if (min > left)
     min = left;  /* there is now less time remaining
     */
```

```
        }
        return min;
    }
```

The worst case time value may be dynamically computed or empirically measured to be a cut off time that defines a period in which accesses have an acceptable error rate. If the first entry fulfills the requirement, then this first entry is selected (step 340); otherwise, the Selection Procedure checks whether the first entry in the ORQ can be selected while guaranteeing that all SSQ requests will meet their time deadlines in the worst case (step 330). If so, then this first entry is selected (step 350); otherwise, the procedure proceeds to step 315, wherein the procedure queries whether the first entry in the SSQ can be executed within its time deadline assuming the worst case access. If the request cannot be executed in time, the request is discarded at step 325 and the procedure returns to step 320.

If, however, the request can be executed in the allotted time, the first entry of the SSQ is selected at step 360. The selected request is then removed from its queue (step 370). Alternatively, if caching is used, the Selection Procedure checks whether data for the selected request is already in cache (step 380) (the caching step 380 is shown in phantom to represent that it is an optional step). If the request is cached, the selected request is discarded and the Selection Procedure is repeated. Otherwise, the selected request is forwarded to the associated disk queue (step 390).

The SDS executes the Selection Procedure during two scheduling events, called the *scheduling interval* and the *command completion event*. The scheduling interval is a fixed, periodic interval, while a command completion event occurs every time one of the disks completes a command.

(Note that it is possible, although highly unlikely, that multiple disks complete a command simultaneously at a command completion event.) At each scheduling interval, a procedure called the *Scheduling Interval Procedure* is executed, and at each command completion event, a procedure called *the Command Completion Procedure* is executed. In the case that a scheduling interval and a command completion coincide, the Command Completion Procedure is executed first (i.e., the Command Completion Procedure is given priority over the scheduling Interval Procedure). Alternatively, if the disk queue has a depth that is greater than one, then the execution priority of these routines is reversed. Such reversal leaves more time available to do other operations.

In the Scheduling Interval Procedure, steady-state requests are added to the next SSQ, if possible. (Recall that a steady-state request can be added to the next SSQ as soon as the data is output from the video server to the subscriber), and all SSQs are reordered to maintain correct time deadline order. The first entries in each of the SSQs are then sorted based on time deadlines, which determines the order with which the disks are serviced. For each disk, the Selection Procedure 300 is repeatedly executed as long as the associated disk queue is not full, at least one of the three SDS queues (SSQ, NSQ, ORQ) is not empty, and there is a request in one of the three SDS queues that satisfies the Selection Procedure criteria. For example, if in a three-Disk system when the disk queues are not full the first entry in Disk 1's SSQ has a time deadline of 35, the first entry in Disk 2's SSQ has a time deadline of 28, and the first entry in Disk 3's SSQ has a time deadline of 39, then the disks would be serviced in the following order: Disk 2, Disk 1, Disk 3. Once the disk order has been established, then the SDS

Selection Procedure is performed for each disk in that order.

Generally, in a video server application, the extents for the data are very long (e.g., hundreds of kilobytes) such that the disk queues have a depth of one. In other applications using shorter data extents, the disk queues may have various depths, e.g., five requests could be stored and executed in a first-in, first-out (FIFO) manner. The extent size is inversely proportioned to disk queue depth where data delivery latency is the driving force that dictates the use of a large extent size for video server applications. For other applications where the extent size is relatively small, the disk queue depth is dictated by the desire to reduce disk drive idle time.

FIG. 4 shows a formal specification of the Scheduling Interval Procedure 400 in flowchart form. First, the Scheduling Interval Procedure adds steady-state requests to the appropriate SSQs, if possible (step 420), and reorders all the SSQs by time deadlines (step 430). The disk that has the earliest deadline for the first entry in its SSQ is then selected (step 450). The Selection Procedure is performed for the selected disk (step 300), and then the Scheduling Interval Procedure checks whether a request satisfying the Selection Procedure criteria was selected (step 460). If not, the disk with the next earliest deadline for the first entry in its SSQ is selected (steps 475, 480, 450) and the Selection Procedure is repeated for this disk (step 300). Otherwise, the Scheduling Interval Procedure checks whether the selected disk's queue is full, or if all three SDS queues for the selected disk are empty. If either of these conditions are true, then the disk with the next earliest deadline for the first entry in its SSQ is selected (steps 475, 480, 450) and the Selection Procedure is repeated for this disk (step 300). If, however, both conditions are false, the Selection Procedure is repeated for the same selected disk. Thus, the disks are processed sequentially, ordered by the corresponding SSQ's first deadline, where "processing" means that the Selection Procedure is invoked repeatedly until the disk queue is full or there are no more requests for that disk.

As disclosed in FIG. 4, the Scheduling Interval Procedure fills each of the disk queues one at a time, which is most efficient for small disk queues. In the preferred embodiment, a small disk queue is used, as it facilitates the latency reduction. In particular, as soon as the servicing of a request extends past its worst-case access time, the request is aborted by the SDS, i.e., the SDS "times-out" waiting for the request to be serviced and then moves on the next procedural step. To assist in error handling when using a disk queue with a depth that is greater than one such that the server may determine which request was not fulfilled within a predefined time period, the server maintains a disk mimic queue that mimics the content of the disk queue of each of the disk drives. As such, the server can poll the mimic queue to determine the nature of the errant request and send an "abort" command to the disk drive for that request. The disk drive will then process the next request in the disk queue and the server updates the mimic queue.

In the case of large disk queues, however, filling the disk queues in a round-robin fashion may be more efficient. A round-robin version of the Scheduling Interval Procedure for large disk queues is shown in FIG. 5. As in the previous embodiment of the Scheduling Interval Procedure, steady-state requests are first added to the appropriate SSQs (step 520), and disks are ordered by the deadlines of the first entry in each disk's SSQ (step 530). In this round-robin version, however, the Selection Procedure is executed only once for a disk, and then the next disk is selected. Once all disks have been selected, the round-robin Scheduling Interval Procedure goes through each of the disks once again in the same order, executing the Selection Procedure once per disk. This process is continued until no more requests can be added to any of the disk queues.

Specifically, a vector D is defined as an ordered list of all the disks, where the order is based on the time deadlines of the first entry in each disk's SSQ (step 530). A Boolean variable *SELECT* is initialized to *false*, and an integer variable i is initialized to 1 (step 540). The following condition is then tested: if $i = n+1$ and *SELECT* = *false* (step 550). As will be seen shortly, this condition will only be true when all of the disks have been selected and no requests could be added to any of the disk's queues. Next (555), if $i = n+1$ (i.e., the last disk had been selected in the previous iteration), then i is set to 1 (start again with the first disk). If disk D_i's disk queue is full (step 560), or all three of D_i's SDS queues are empty (step 570), then the next disk is selected (step 585). The Selection Procedure is performed for D_i (step 300), and if a request satisfying the Selection Procedure criteria was found, *SELECT* is set to true (step 580), and the next disk is selected (step 585). Thus the *SELECT* variable indicates whether a request was added to one of the disk queues during a pass over the vector of disks.

The Command Completion Procedure is executed, on a first-in, first-out basis, every time a disk completes a command. Thus, for each completed command, the Command Completion Procedure executes in the order in which the commands are completed, i.e., using the FIFO command handling step 605. As such, the Command Handling Procedure begins at step 610, proceeds to step 605 and ends at step 690.

Alternatively, the procedure can be adapted to handle simultaneous command events. In this procedure, it is first determined if multiple disks have completed a command simultaneously at the command completion event. (Most likely only one disk will have completed a command at the command completion event, but the multiple-disk situation is possible.) If more than one disk has completed a command, then the first entries in the SSQs of these disks are sorted based on time deadlines, determining the order in which the disks are serviced. Once the disk order has been established, the SDS Selection Procedure is performed for each disk in order in the same manner as the Scheduling Interval Procedure. That is, for each disk, the Selection Procedure is repeatedly executed as long as the associated disk queue is not full, at least one of the three SDS queues (SSQ, NSQ, ORQ) is not empty, and there is a request in one of the three SDS queues that satisfies the Selection Procedure criteria.

A formal specification of both forms of the Command Completion Procedure is shown in flowchart form in FIG. 6. Step 605 represents the standard FIFO command handling procedure, while the dashed box 615 represents an alternative procedure capable of handling simultaneous command occurrences. In this alternative version, the Command Completion Procedure determines which disks have just completed a command, and the disk that has the earliest deadline for the first entry in its SSQ is then selected (step 650). Just as in the Scheduling Interval Procedure, the Selection Procedure is performed for the selected disk (step 300), and then the Command Completion

Procedure checks whether a request satisfying the Selection Procedure criteria was selected (step 660). If not, the disk with the next earliest deadline for the first entry in its SSQ is selected (steps 675, 680, 650) and the Selection Procedure is repeated for this disk (step 300). Otherwise, the Command Completion Procedure checks whether the selected disk's queue is full, or if all three SDS queues for the selected disk are empty. If either of these conditions are true, then the disk with the next earliest deadline for the first entry in its SSQ is selected (steps 675, 680, 650) and the Selection Procedure is repeated for this disk (step 300). If, however, both conditions are false, the Selection Procedure is repeated for the same selected disk.

As disclosed in FIG. 6, the Command Completion Procedure fills each of the disk queues one at a time, i.e., the disk with a complete event is refilled. Note that since it is highly unlikely that more than one disk is serviced on a command completion event, the choice of whether to employ round-robin or sequential filling of the disk queues in the Command Completion Procedure has essentially no impact on performance.

In both the Scheduling Interval and Command Completion Procedures, the ordering of requests within the disk queues are managed by the video server CPU, and not the disks themselves. (Any reordering operations normally performed by the disk must be disabled.) While reordering by the disks would improve the average seek time, managing the disk queues by the CPU is required to preserve the time deadlines of the user requests.

A formal specification of the method of the present invention is shown in flowchart form in FIG. 7. Whenever a command completion event occurs (720), the Command Completion Procedure is invoked (600), and whenever a scheduling interval occurs (730), the Scheduling Interval Procedure is invoked (400). As shown in the figure, if both a scheduling interval and a command completion event occur simultaneously, the command completion is given priority and the Command Completion Procedure is executed first. Alternatively, as discussed above, when a disk queue having a depth that is greater than one is used, the execution priority for these procedures is reversed.

In a preferred embodiment, the method of the present invention is implemented as a multi-threaded process. FIG. 8 shows the software process architecture 800 for the preferred embodiment. The *media control thread* 810 receives new-subscriber request messages from the transport network 140 and path 175, and forwards these requests through message queues 815 to the *T_s thread* 820. The T_s thread 820 is a top level scheduler responsible for two primary functions: first, it maintains all state information necessary to communicate with the disk interfaces 835 and video server memory 840; second, it performs the Scheduling Interval Procedure using a period of, for example, 100 ms. The T_s Loop thread allocates the commands to the SDS queues 875, where each disk drive is associated with a set of queues (e.g., ssa, NSQ and other queues) generally shown as queues 825_0, 825_1, … 825_N. At the startup condition, when the disks are idle, the initial commands (startup commands) from the T_s loop thread 820 are sent from the SDS queues 825 directly to the disk interfaces 835. Under steady-state operation, a response thread 830 communicates the commands from the SDS queues 825 to the disk drive interfaces 835. Each interface 835 communicates to individual disk drives through a fiber channel loop. Response thread 330 also receives command completion messages from the disk interfaces 835. Upon receiving these messages the response thread performs the

Command Completion Procedure. Media control thread 810, T_s loop thread 820, and response thread 830 are all executed by video server CPU 114 of FIG. 1.

While this invention has been particularly shown and described with references 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.

What is claimed is:

1. A queuing architecture for scheduling disk drive access requests in an information server, comprising, for each disk drive in said information server:

a first queue (221) for disk access requests from users currently receiving information provided by the information server;

a second queue (222; 223) for all other disk access requests; and

a queue selector (205) for selecting requests from said first and second queues and forwarding said requests to said disk drive.

2. The queuing architecture of claim 1, where requests in said first and second queues are assigned worst-case access times.

3. The queuing architecture of claim 1, where requests in said first queue each have an associated time deadline.

4. The queuing architecture of claim 1, where requests in said first queue are ordered from front to back by ascending time deadlines.

5. The queuing architecture of claim 1, where said queue selector gives highest priority to requests in said first queue and lower priority to requests in said second queue.

6. The queuing architecture of claim 1, wherein said information server is a video server (110) and said information is a video program.

7. The queuing architecture of claim 1, where requests in said first queue are ordered from front to back by ascending time deadlines.

8. The queuing architecture of claim 1 wherein said queue selector establishes priority in response to the location of the data upon a disk in said disk drive.

9. The queuing architecture of claim 1 wherein data is stored in said disk drives based upon queuing priority.

10. A method of scheduling access requests for a disk drive in an information server, said method comprising the steps of:

(a) providing at least two queues, where a first queue contains steady state access requests from users being supplied information from the information server and a second queue for all other types of access requests;

(b) selecting an access request from said second queue, if such selection does not cause a steady-state request to miss a time deadline within which the steady-state request must be completed to ensure that the information being viewed is not interrupted;

(c) otherwise, selecting a steady-state; and

(d) forwarding the selected request to said disk drive.

11. The method of claim 10 further comprising the step of repeating steps (b), (c), and (d) performed repeatedly while an internal queue within the disk drive is not full, and there are outstanding access requests for said disk drive.

12. The method of claim 10 further comprising the steps of checking, before forwarding said selected request to said disk drive, if data for said selected request is in a cache, and discarding said selected request if said data is in said cache.

13. A method of scheduling access requests for a plurality of disk drives in a video server, said method comprising the steps of:

providing, for each of the disk drives in said plurality of disk drives, three queues, where a first queue contains steady state access requests from users being supplied a program from the video server, a second queue contains new programming access requests and a third queue for all other types of access requests;

determining which of said disk drives has completed a command;

ordering said disk drives that have completed a command by earliest time deadline of steady-state access requests for said disk drives; and

for each said disk that has completed a command,

selecting the next new programming request, if such selection does not cause any steady-state request to miss an associated time deadline within which the steady-state request must be completed to ensure that the program being viewed is not interrupted;

otherwise, selecting the next other-request, if such selection does not cause any steady-state request to miss said associated time deadline within which the steady-state request must be completed to ensure that the program being viewed is not interrupted;

otherwise, selecting the steady-state request with the earliest time deadline; and

forwarding the selected request to said disk drive.

14. The method of claim 13, where said method is performed every time one of said disk drives has completed a command.

15. The method of claim 13, where said method is performed within a scheduling interval.

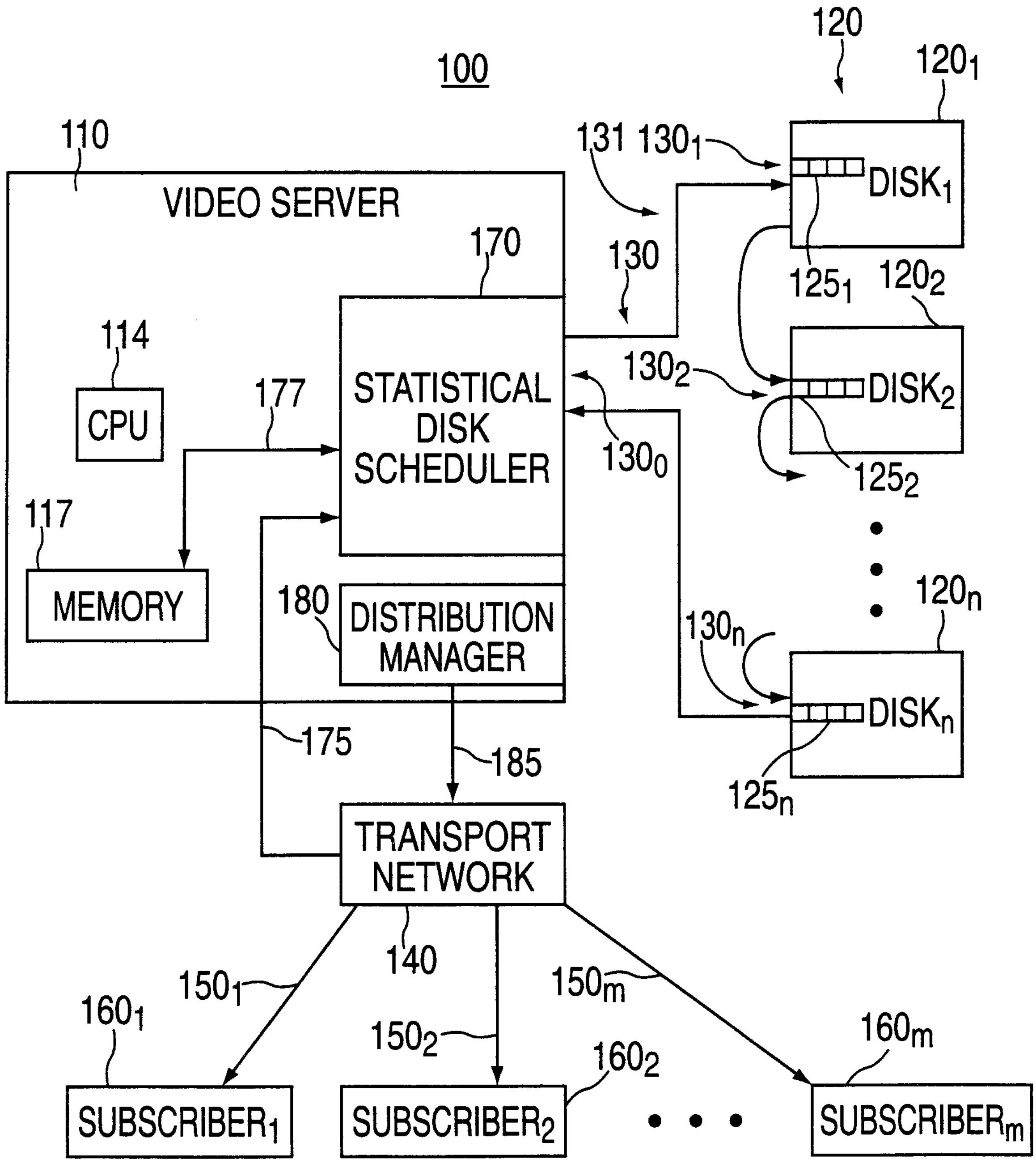

100
110
VIDEO SERVER
114
CPU
117
MEMORY
177
170
STATISTICAL DISK SCHEDULER
180
DISTRIBUTION MANAGER
175
185
TRANSPORT NETWORK
140
131
130
130₁
130₂
130₀
130ₙ
120
120₁
DISK₁
125₁
120₂
DISK₂
125₂
120ₙ
DISKₙ
125ₙ
150₁
150₂
150ₘ
160₁
SUBSCRIBER₁
160₂
SUBSCRIBER₂
160ₘ
SUBSCRIBERₘ

FIG. 1

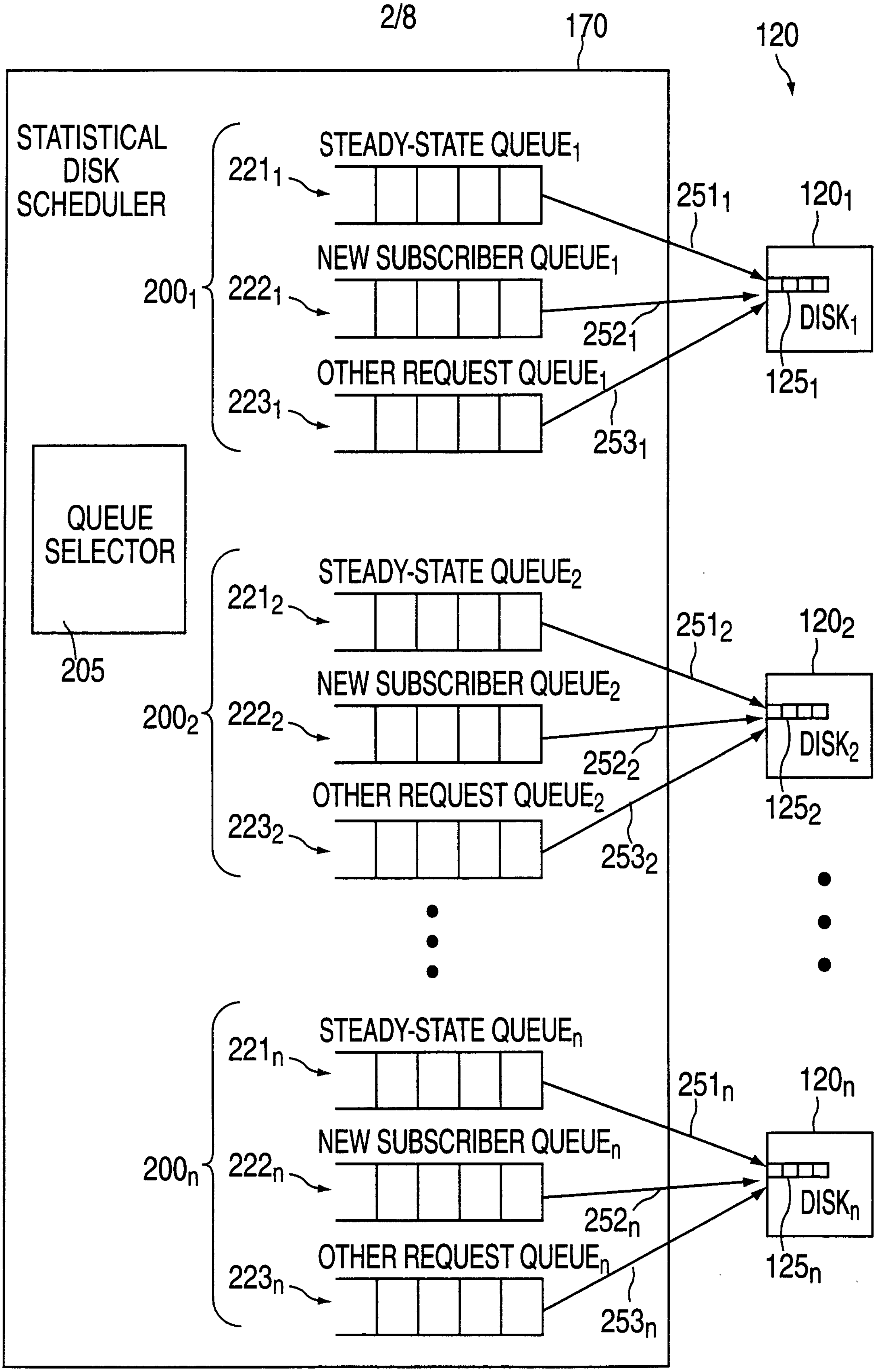
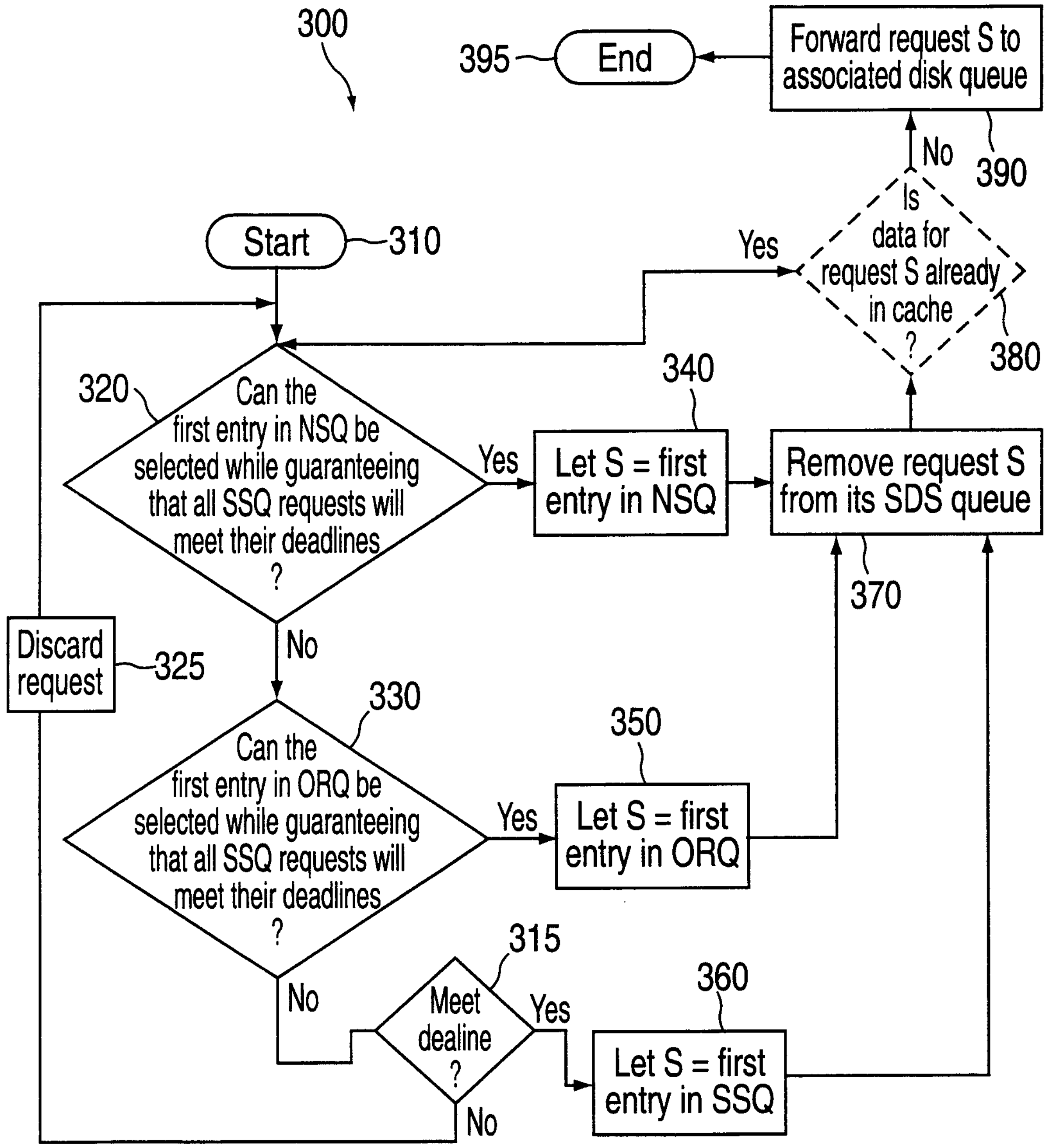

170
120
STATISTICAL DISK SCHEDULER
STEADY-STATE QUEUE1
2211
2511
1201
NEW SUBSCRIBER QUEUE1
2001
2221
DISK1
2521
OTHER REQUEST QUEUE1
1251
2231
2531
QUEUE SELECTOR
205
STEADY-STATE QUEUE2
2212
2512
1202
NEW SUBSCRIBER QUEUE2
2002
2222
DISK2
2522
OTHER REQUEST QUEUE2
1252
2232
2532
STEADY-STATE QUEUEn
221n
251n
120n
NEW SUBSCRIBER QUEUEn
200n
222n
DISKn
252n
OTHER REQUEST QUEUEn
125n
223n
253n

FIG. 2



300
395
End
Forward request S to associated disk queue
390
No
Start
310
Yes
Is data for request S already in cache ?
380
340
320
Can the first entry in NSQ be selected while guaranteeing that all SSQ requests will meet their deadlines ?
Yes
Let S = first entry in NSQ
Remove request S from its SDS queue
370
Discard request
325
No
330
350
Can the first entry in ORQ be selected while guaranteeing that all SSQ requests will meet their deadlines ?
Yes
Let S = first entry in ORQ
315
360
No
Meet dealine ?
Yes
Let S = first entry in SSQ
No

FIG. 3

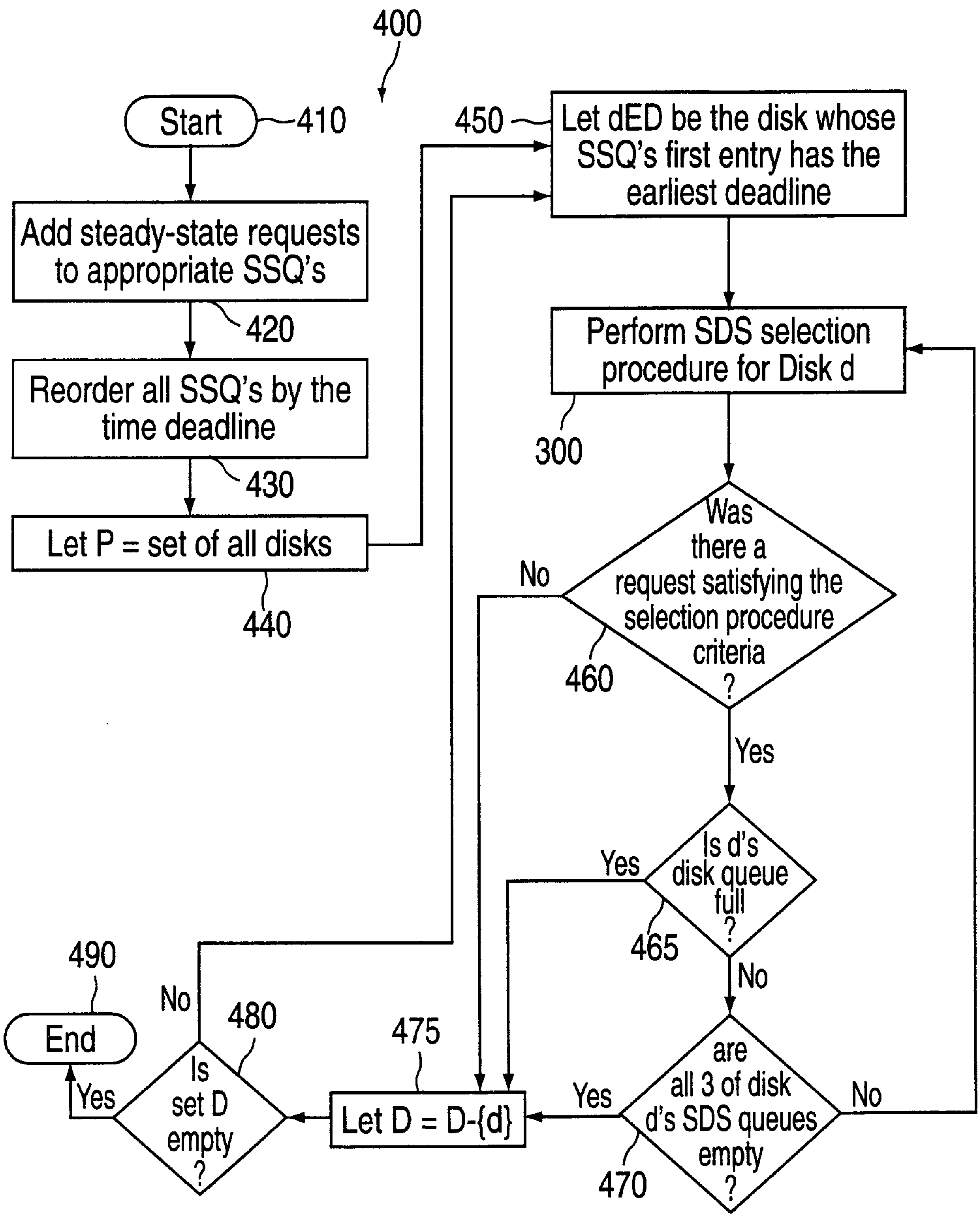

400
Start
410
Add steady-state requests to appropriate SSQ's
420
Reorder all SSQ's by the time deadline
430
Let P = set of all disks
440
450
Let dED be the disk whose SSQ's first entry has the earliest deadline
Perform SDS selection procedure for Disk d
300
Was there a request satisfying the selection procedure criteria ?
460
No
Yes
Is d's disk queue full ?
465
Yes
No
are all 3 of disk d's SDS queues empty ?
470
Yes
No
Let D = D-{d}
475
Is set D empty ?
480
No
Yes
End
490

FIG. 4

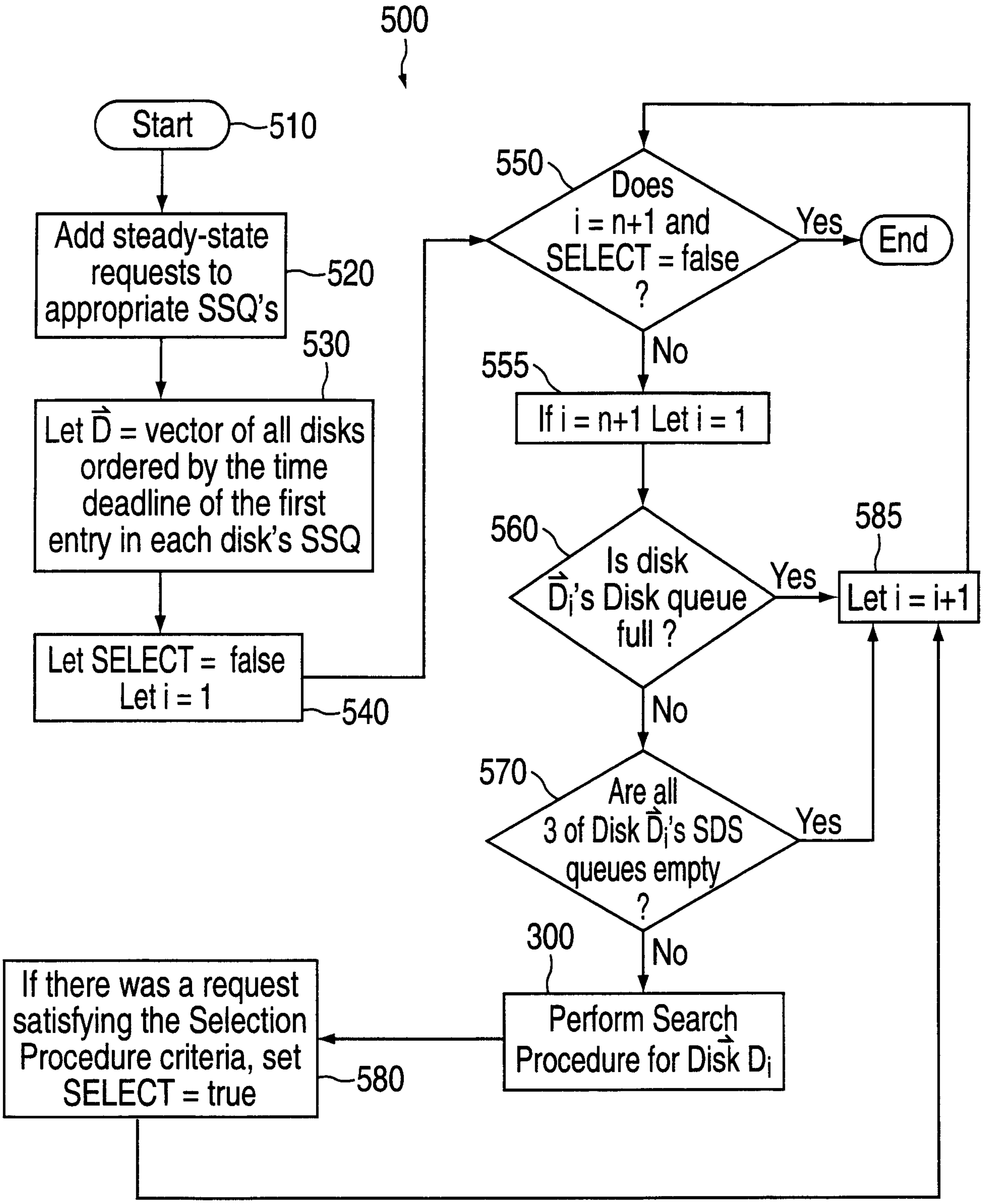

500
Start
510
Add steady-state requests to appropriate SSQ's
520
530
Let $\vec{D}$ = vector of all disks ordered by the time deadline of the first entry in each disk's SSQ
Let SELECT = false
Let i = 1
540
550
Does i = n+1 and SELECT = false ?
Yes
End
No
555
If i = n+1 Let i = 1
560
Is disk $\vec{D}_i$'s Disk queue full ?
Yes
585
Let i = i+1
No
570
Are all 3 of Disk $\vec{D}_i$'s SDS queues empty ?
Yes
No
300
Perform Search Procedure for Disk $\vec{D}_i$
If there was a request satisfying the Selection Procedure criteria, set SELECT = true
580

FIG. 5

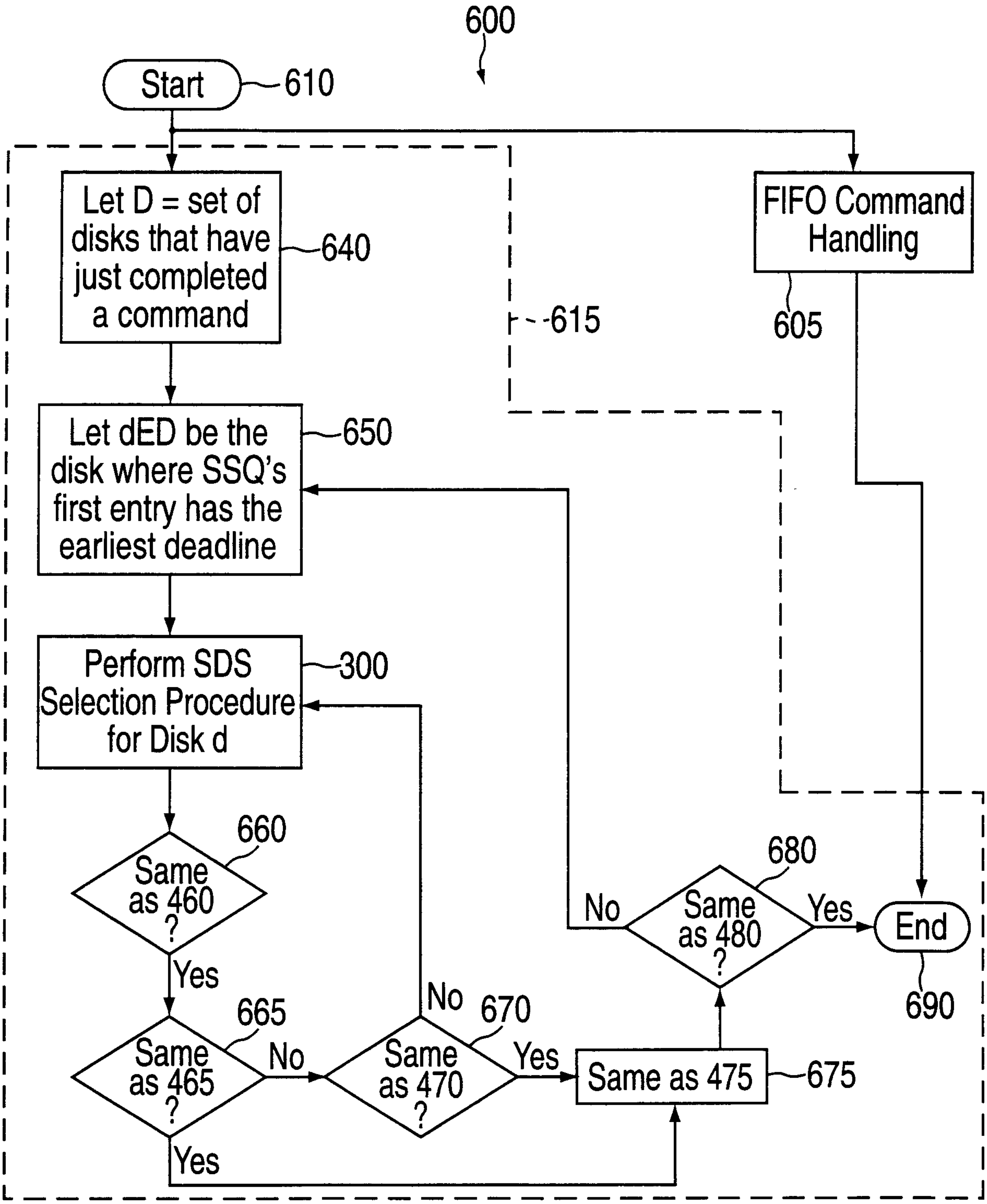

600
Start
610
Let D = set of disks that have just completed a command
640
615
FIFO Command Handling
605
Let dED be the disk where SSQ's first entry has the earliest deadline
650
Perform SDS Selection Procedure for Disk d
300
660
Same as 460 ?
Yes
665
Same as 465 ?
No
Yes
No
670
Same as 470 ?
Yes
Same as 475
675
680
No
Same as 480 ?
Yes
End
690

FIG. 6

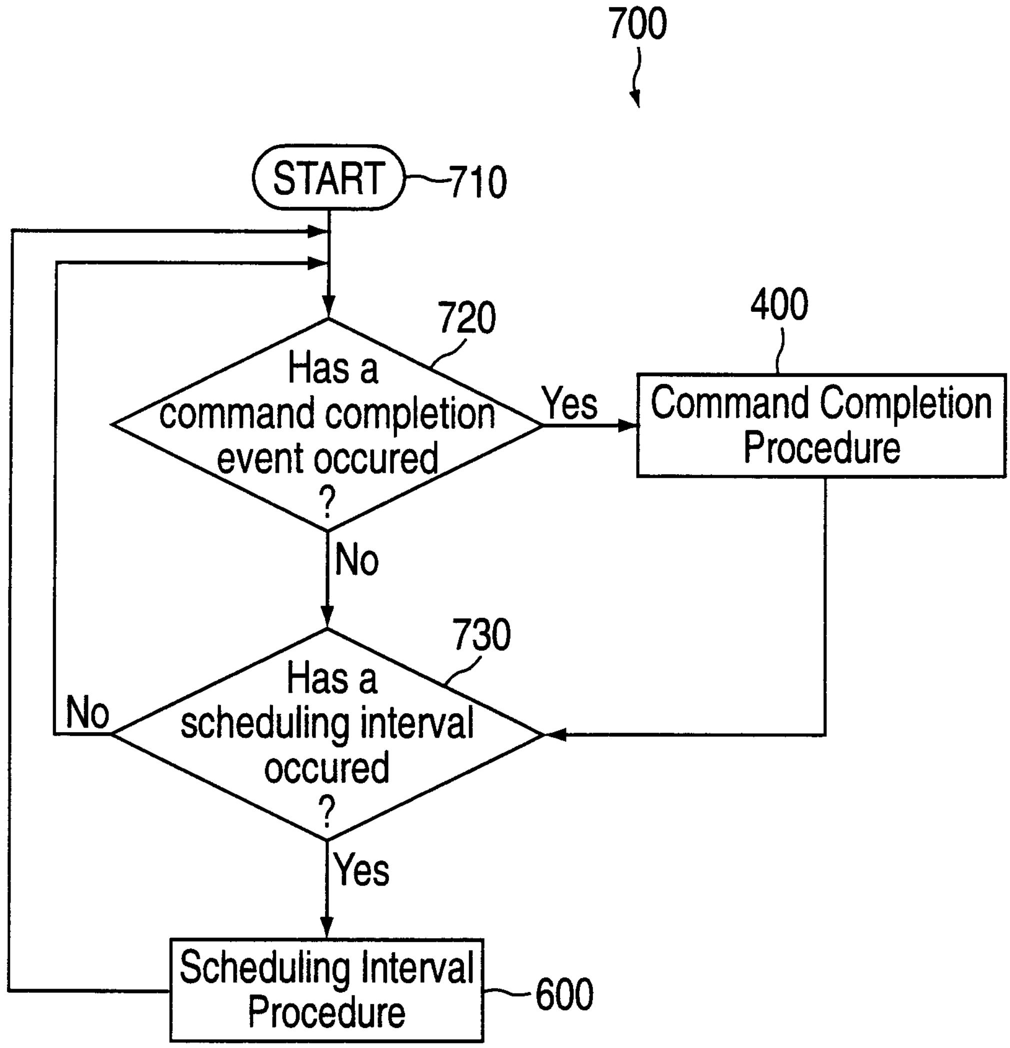

700
START
710
720
Has a command completion event occured ?
Yes
400
Command Completion Procedure
No
730
Has a scheduling interval occured ?
No
Yes
Scheduling Interval Procedure
600

FIG. 7

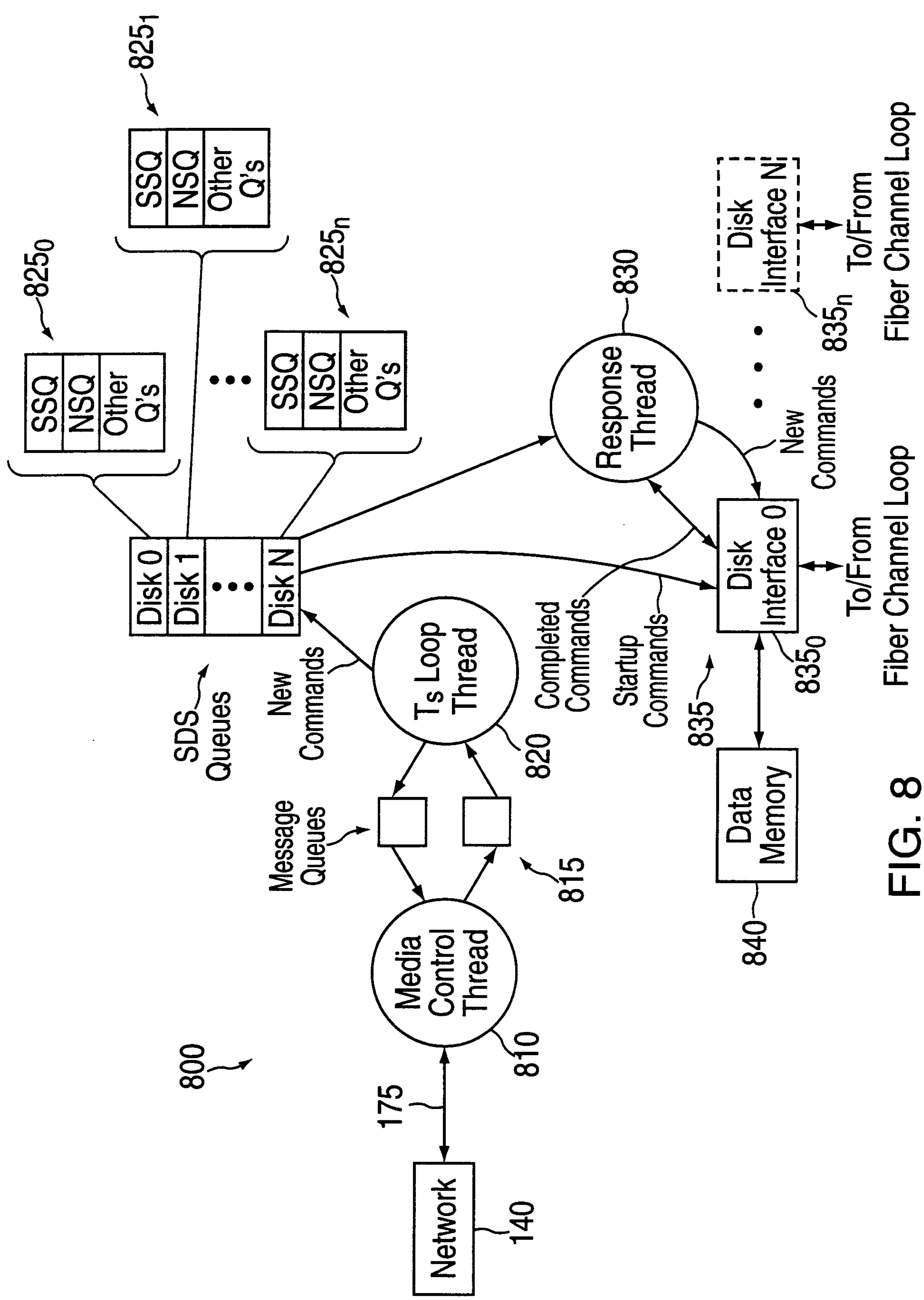

800
Network
140
175
Media Control Thread
810
Message Queues
815
Ts Loop Thread
820
New Commands
SDS Queues
Disk 0
Disk 1
Disk N
SSQ
NSQ
Other Q's
825₀
825₁
825ₙ
Response Thread
830
New Commands
Completed Commands
Startup Commands
Disk Interface 0
835₀
835
Disk Interface N
835ₙ
To/From Fiber Channel Loop
To/From Fiber Channel Loop
Data Memory
840

FIG. 8

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/06093

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G06F 13/14, 13/372, 3/00; H01N 7/10
US CL : 711/112, 114, 158; 710/39, 40, 54; 348/7
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED
Minimum documentation searched (classification system followed by classification symbols)
U.S. : 711/112, 113, 114, 151, 158, 167; 710/39, 40, 44, 52, 54, 57; 709/103; 348/7

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Please See Extra Sheet.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,787,482 A (CHEN ET AL) 28 July 1998, column 1, line 54 to column 2, line 6; column 5, lines 19-33; column 7, lines 10-15; column 9, lines 57-64; and column 10, lines 28-33.	1-15
Y, P	US 5,926,649 A (MA ET AL) 20 July 1999, column 3, line 63 to column 4, line 33.	1-15
Y	US 5,721,956 A (MARTIN ET AL) 24 February 1998, column 1, lines 22-50; column 3, lines 45-55; Figure 1.	12
A	US 5,802,394 A (BAIRD ET AL) 01 September 1998, column 28, lines 21-47; Figure 20.	1-15

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search
14 JUNE 2000

Date of mailing of the international search report
05 JUL 2000

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703) 305-3230

Authorized officer
For Glenn Gossage
GLENN GOSSAGE
Telephone No. (703) 305-3900

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/06093

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,561,456 A (YU) 01 October 1996, column 2, lines 42-56; column 3, lines 3-6; column 4, lines 15-60; column 6, line 33 to column 7, line 13.	1-15
A, P	US 6,023,720 A (AREF ET AL) 08 February 2000, column 2, line 46 to column 5, line 25; column 5, line 55 to column 6, line 15.	1-15
A, P	US 5,928,327 A (WANG ET AL) 27 July 1999, see entire document.	1-15
A	US 5,687,390 A (MCMILLAN, JR.) 11 November 1997, see entire document.	1-15
A	US 5,644,786 A (GALLAGHER ET AL) 01 July 1997, see entire document.	1-15
A	US 5,220,653 A (MIRO) 15 June 1993, see entire document.	1-15

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/06093

B. FIELDS SEARCHED
Electronic data bases consulted (Name of data base and where practicable terms used):

USPTO EAST search system (USPAT, JPO, EPO databases)

search terms: queue, buffer, disk, schedule, order, priority, time, deadline, broadcast, video, demand, server