file -
Multics proposals

Technical Memo

November 9, 1971

TO: Noel Morris

FROM: Lee Scheffler

SUBJECT: Proposed DSU-170 DIM Modification

cc: C. T. Clingen
F. J. Corbato
J. W. Gintell
J. H. Saltzer
S. H. Webber

Subject: Proposed Modification to the Multics DSU-170 DIM

Objective: To improve the DSU-170 subsystem performance under heavy load conditions by overlapping seek times on separate disk units and improving channel utilization.

Proposed General Strategy

1. Maintain a wait queue, Wp, of requests for each priority, p. Arriving requests are appended to the bottom of the appropriate priority wait queue.

2. Using the "off-line" seek capability of the DSU-170/IBM 2314, issue a seek for the highest request in the wait queues (highest priority wait queue first) for each unit that is not already executing a seek operation. Issue the next seek (or transfer, after all possible seeks have been issued) as soon as the seek-received status is returned from the GIOC (250 microseconds ±50); do not wait for, or be concerned with, the completion of the seek (special interrupt inhibited by switch).

3. Maintain a single FCFS (First Come First Served) service queue, S, of adjustable maximum length m, of requests that have already had seeks initiated. Not all requests with seeks initiated need be entered into S. Only 0 or 1 requests for each unit may appear in S at any one time; therefore, $0 < m \leq n$, where n is the number of disk units.

4. When a "slot" in S becomes free (data transfer is completed for the top request in S), a top-down linear search is made of the wait queues, highest priority first. If a request is encountered with its seek already initiated, it is unthreaded from the wait queue and added to the bottom of the service queue. If a request is encountered that has not already had its seek initiated, and the corresponding disk unit is not currently busy executing a seek operation for a previous request (or waiting for data transfer after completion of a seek; the two possibilities are indistinguishable to the software due to the inhibiting of the special interrupt on seek completion), then a seek is initiated for that request. If S is still not full, unthread the request from the wait queue and add it to the bottom of S. The search terminates when either the bottom of the lowest priority wait queue is reached, or when all units become "busy".

5. Issue a data transfer for the top request in the service queue. Return to the caller. Upon the interrupt at the completion of data transfer, pop the top request of the service queue, post it, and go to 4.

Pitfalls to be Avoided

1. Increasing the mini_gim interrupt load.

2. Prejudicing subsystem latency time on the request arrival distribution across all units.

3. Building seek, disk latency, and data transfer time expectations, number of disk units served by the channel, and request arrival distribution into the algorithm.

Expected Performance (when properly tuned)

Current time estimates (E = expectation value):

E(t_seek) = 75 ms

E(t_latency) + t_transmission = 35 ms

1. Subsystem capacity up from 9 requests/second to 28 requests/second.

2. No increase in subsystem latency time under any conditions (worst case for this algorithm is blocks of successive requests for the same unit).

3. Decrease in subsystem latency under heavy loads by a factor of about 3 to 1.

4. Negligible increase in CPU time required for DIM execution; small increase in data base and code size for the DIM.

Discussion

Wait Queues

Requests entering the subsystem see one wait queue for each priority. The algorithm issuing seeks and data transfers sees a single queue made up of the lower priority queue appended to the bottom of the higher priority queue.

Service Queue

There is a single service queue representing pre-decision of the order of units to/from which data transfers will take place. When the service queue length is limited to one, we have a degenerate case of no pre-decision, and consequent non-optimal use of the channel. There is room in the arrival discipline of the service queue (the service discipline of the wait queues) for optimization of things other than channel utilization (for example, seek arm movement); it is intended that the code for this arrival discipline be easily replaceable.

For the present, the discipline for filling a vacant slot in the service queue is "the next request in the wait queue(s) for a unit not already represented in the service queue."

The net result of these two levels of queueing is the simulation of n independent FCFS single-server queues, one for each of the n disk units, at most m of which are eligible to be served at any one time. The service process of these queues is the arrival process of the service queue. Figure 1 illustrates.

m is a tunable parameter of the algorithm. If $m = 1$, data transfer channel service occurs in precisely FCFS order, overriding channel optimization concerns (but not seek overlap). We define a block as a group of immediately successive requests in one of the wait queues (the top request of the low priority queue is immediately successive to the bottom request of the higher priority queue) for some one unit. The algorithm performance for blocks of requests reverts, for $m = 1$, to the lower (9 requests/second) channel utilization of the present DIM.

With $1 < m \leq k$, where

$$k = \text{integer}(E(t\_seek)/(E(t\_latency) + t\_transmission)) + 1$$

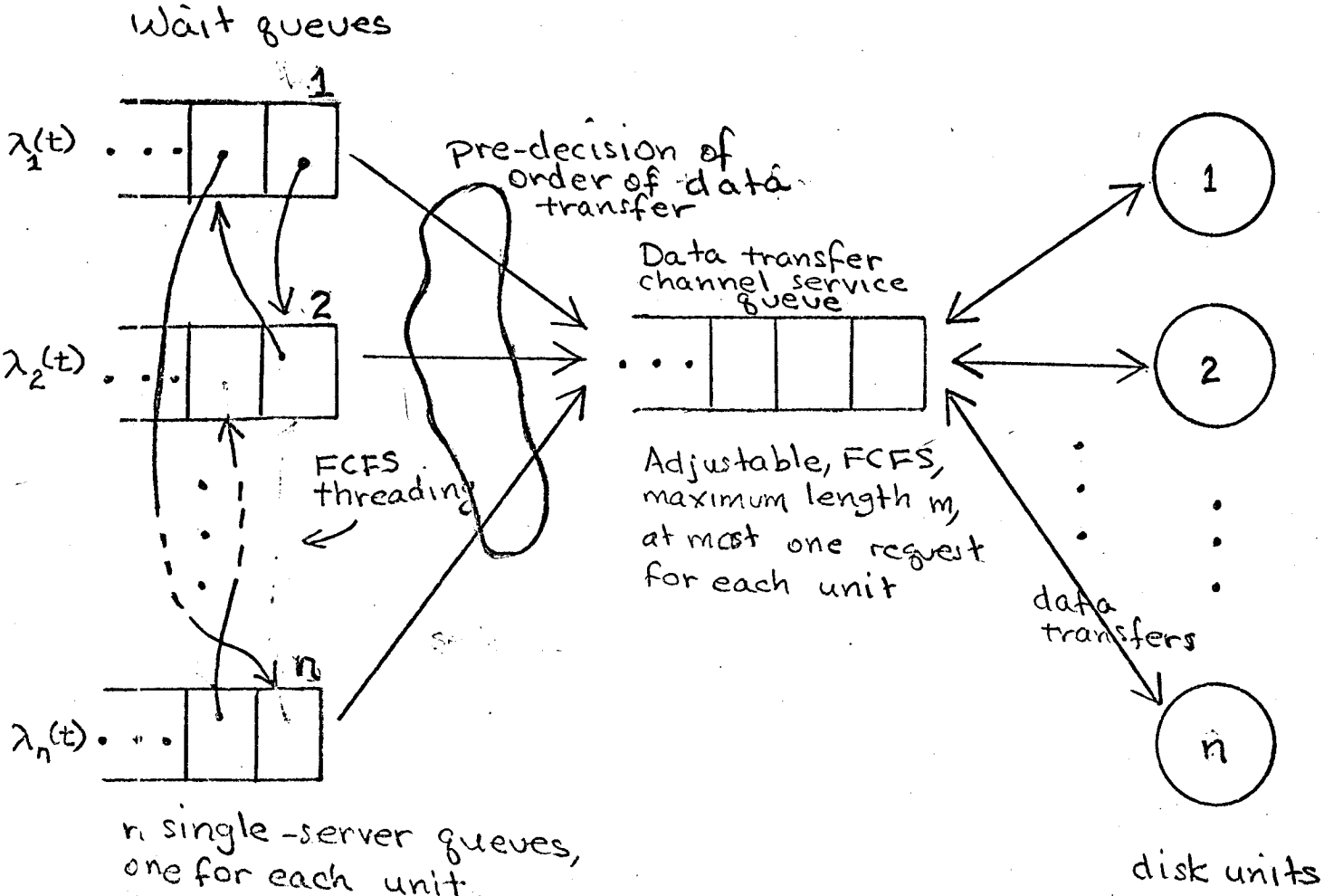


Figure 1. Two-level queuing system simulated by proposed algorithm. Pre-seeks are issued in FCFS order from the wait queues. Transfers are issued in FCFS order from the service queue. All optimization is in the wait-queue-service / service-queue-arrival discipline.

we expect increased channel utilization with little sacrifice of subsystem latency (as distinct from disk latency, t_latency) to blocks of requests. (Times between transmissions from the same unit are filled in with transmissions from other units.)

With n > k, we expect even higher channel optimization (increased probability that the seek for a particular request will be completed by the time that request has reached the head of the service queue, ready for transmission) under heavy loads, and therefore greater throughput, but at an ever increasing sacrifice of subsystem latency to later requests in blocks of requests for the same unit.

Figure 2 is a "desirability graph" depicting several aspects of performance as a very approximate function of service queue size. Notice that with a service queue size of less than k, the channel is often tied up waiting for the completion of the seek for the top request in S. With a service queue size greater than k, channel utilization is higher, but at the expense of forcing an ever-increasing minimum inter-transfer time between requests for the same unit. When service queue size reaches p, where

$$p * (t\_latency\_max + t\_transmission) > t\_seek\_max$$

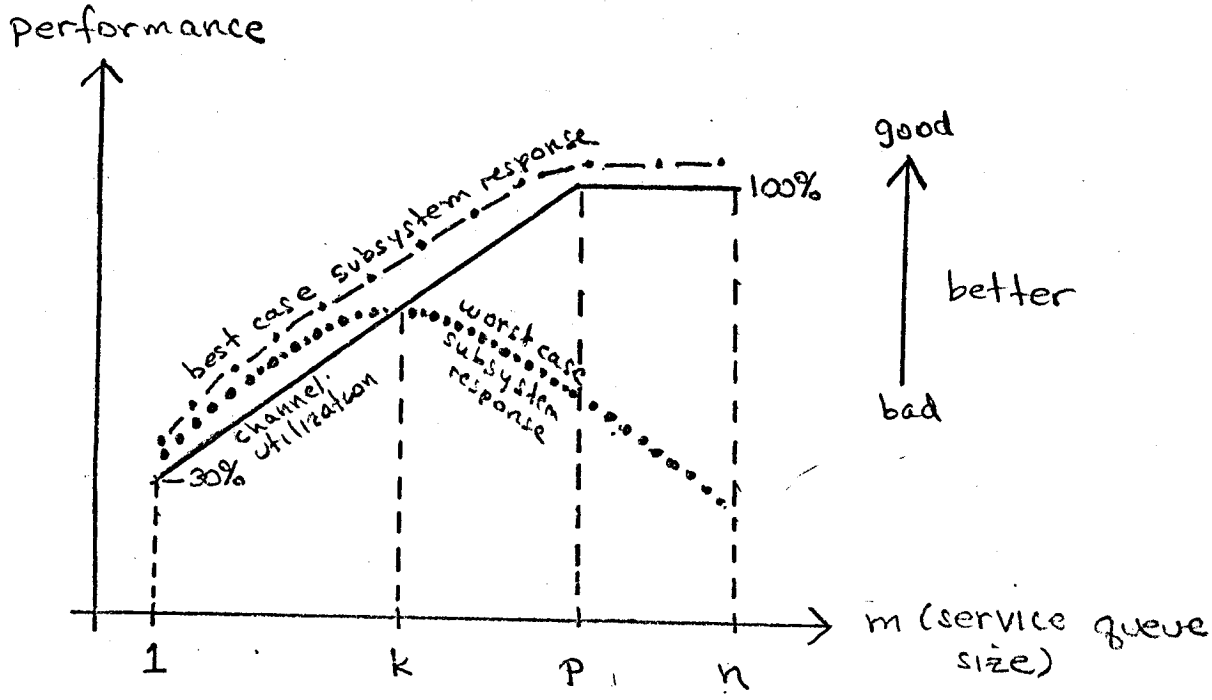
channel utilization approaches 100% of capacity, since there is a probability of 1 that by the time a request reaches the head of the service queue, its seek will have already been completed.

Pre-Seeks

At each point where the channel is not in use, a check is made for requests for units not currently busy seeking, and all possible new seeks (pre-seeks) are issued in FCFS order. A pre-seek uses a single channel connect (it need not, but it does in this implementation) with the following dcw list:

```
seek_dcw fixed bin(71), /* same as normal seek */
term_dcw fixed bin(71), /* normal termination  */
                        /* no interrupt */
```

The time for such a channel service is negligible, corresponding to receipt and initiation of an "off_line" seek operation on the unit indicated. The seek completion special interrupt is inhibited in this implementation by switch.



$$k = \left\lceil \frac{E(t_{seek})}{E(t_{latency}) + t_{transmission}} \right\rceil + 1$$

$$p * (t_{latency\text{-}max} + t_{transmission}) > t_{seek\text{-}max}$$

for Multics current system

$E(t_{seek}) = 75$ ms $t_{seek\text{-}max} = 150$ ms

$E(t_{latency}) \cong 13$ ms $t_{latency\text{-}max} \cong 26$

$t_{transmission} \cong 22$ ms

$k \cong 3$, $p \cong 4$

Figure 2 Approximate "desirability" curves for channel utilization, best case subsystem latency and worst case subsystem latency as a function of service queue size

Pseudo-Seeks

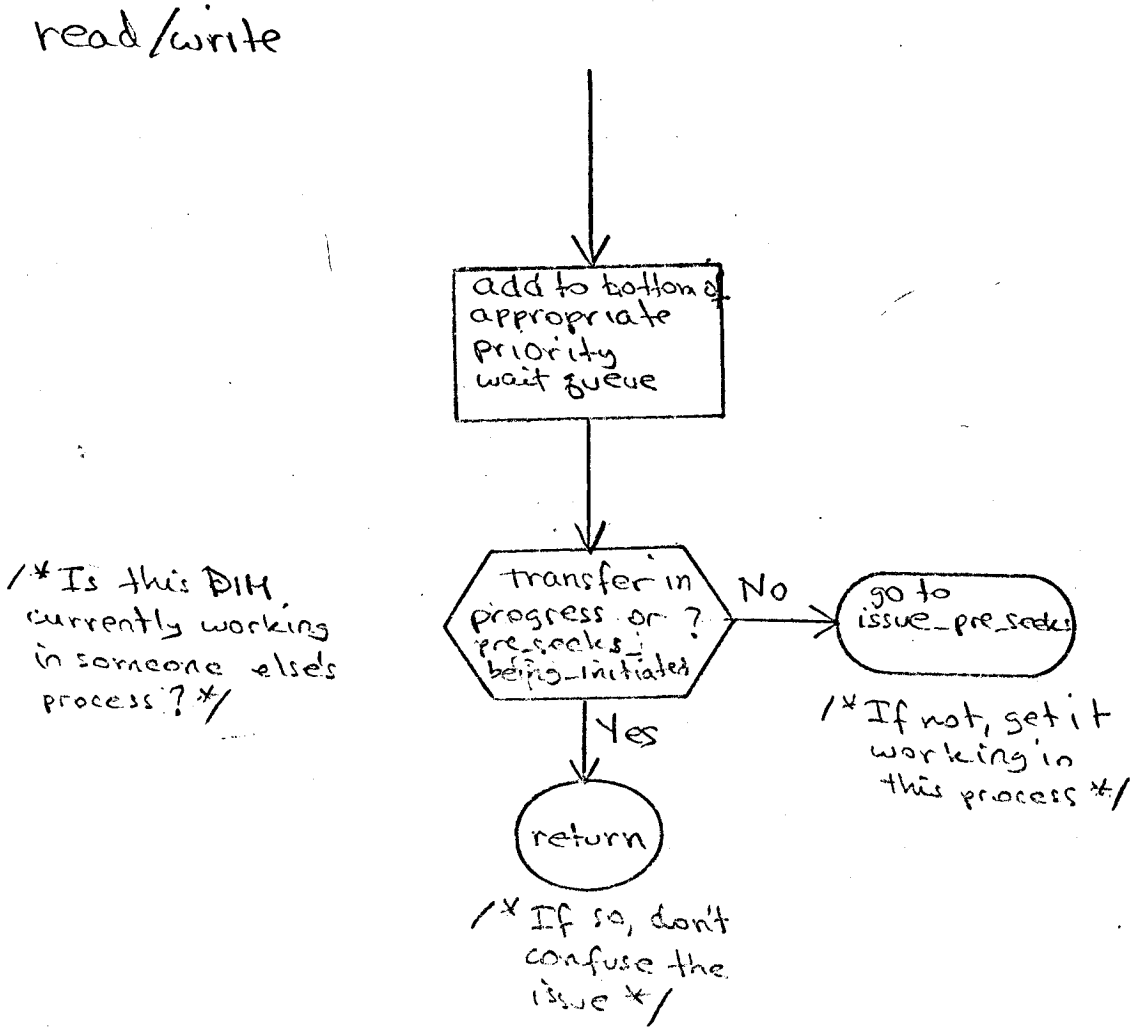
When data transfer for the top request in the service queue is to be undertaken, a second seek dcw (the pseudo-seek) is first issued to re-identify the unit under consideration to the 2314 controller, and to insure that the previously initiated seek operation (the pre-seek) is completed before the data transfer is attempted.

Metering

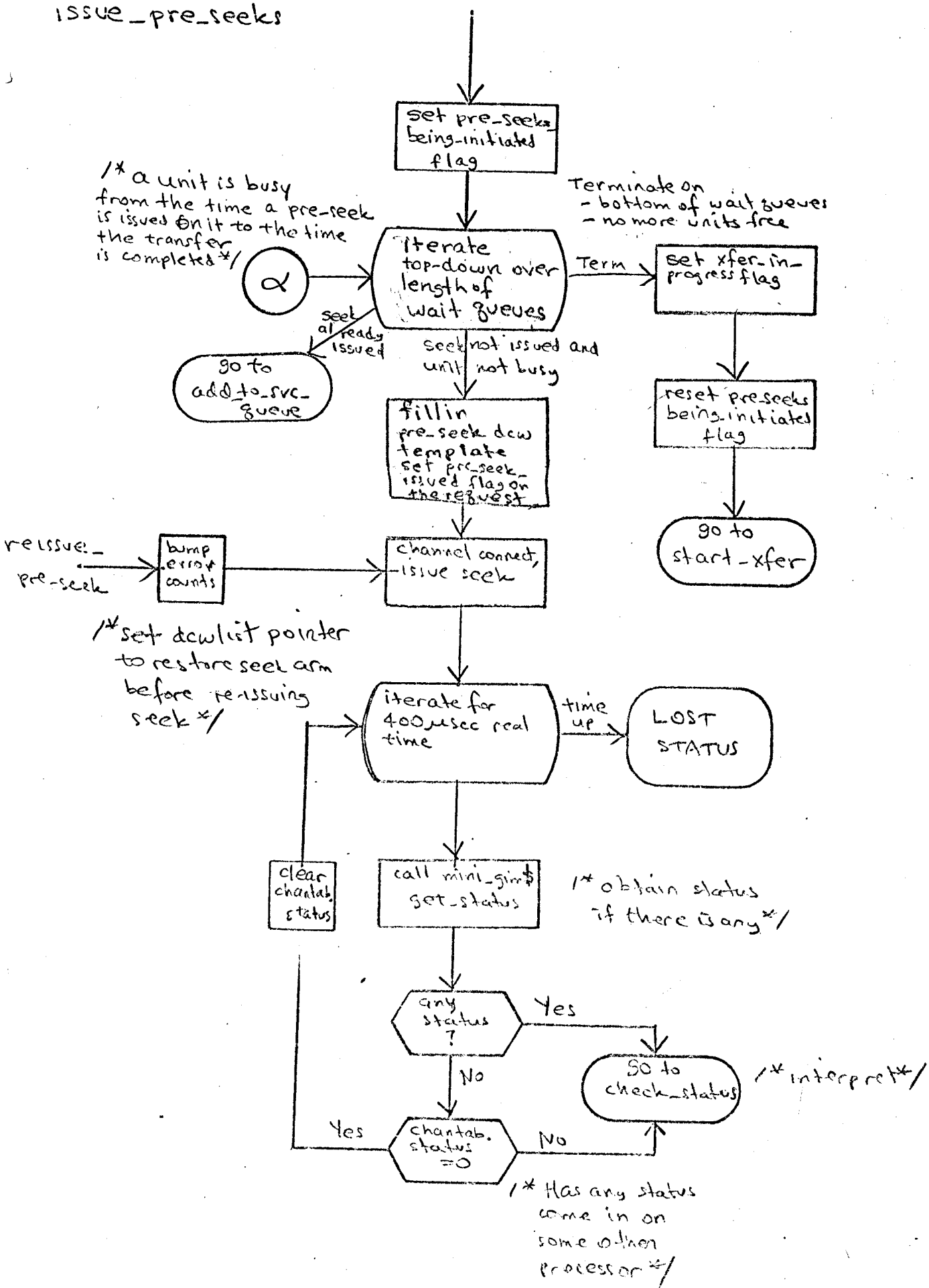
The statistics needed for proper evaluation of the performance of this algorithm are, for each value of m

Quantity	Statistics Desired	Statistics Acceptable
Request arriv rate	pdf(t)	mean and variance or % of time that rate is above certain thresholds
Wait queue size	pdf(t)	mean and variance
Subsystem latency	pdf(queue size)	mean and variance for each of a number of queue size ranges
Request distribution	pdf(unit id)	proportional distribution of page residence and activity
Channel utilization	idle time(t) idle time(queue size)	% idle time idle time mean for each of a number of queue size ranges

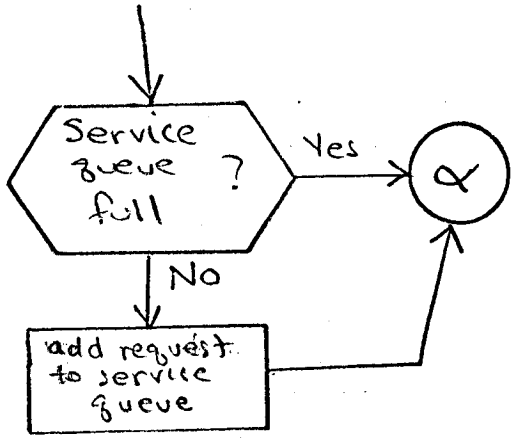
Figure 3. Flowchart of Modified Algorithm



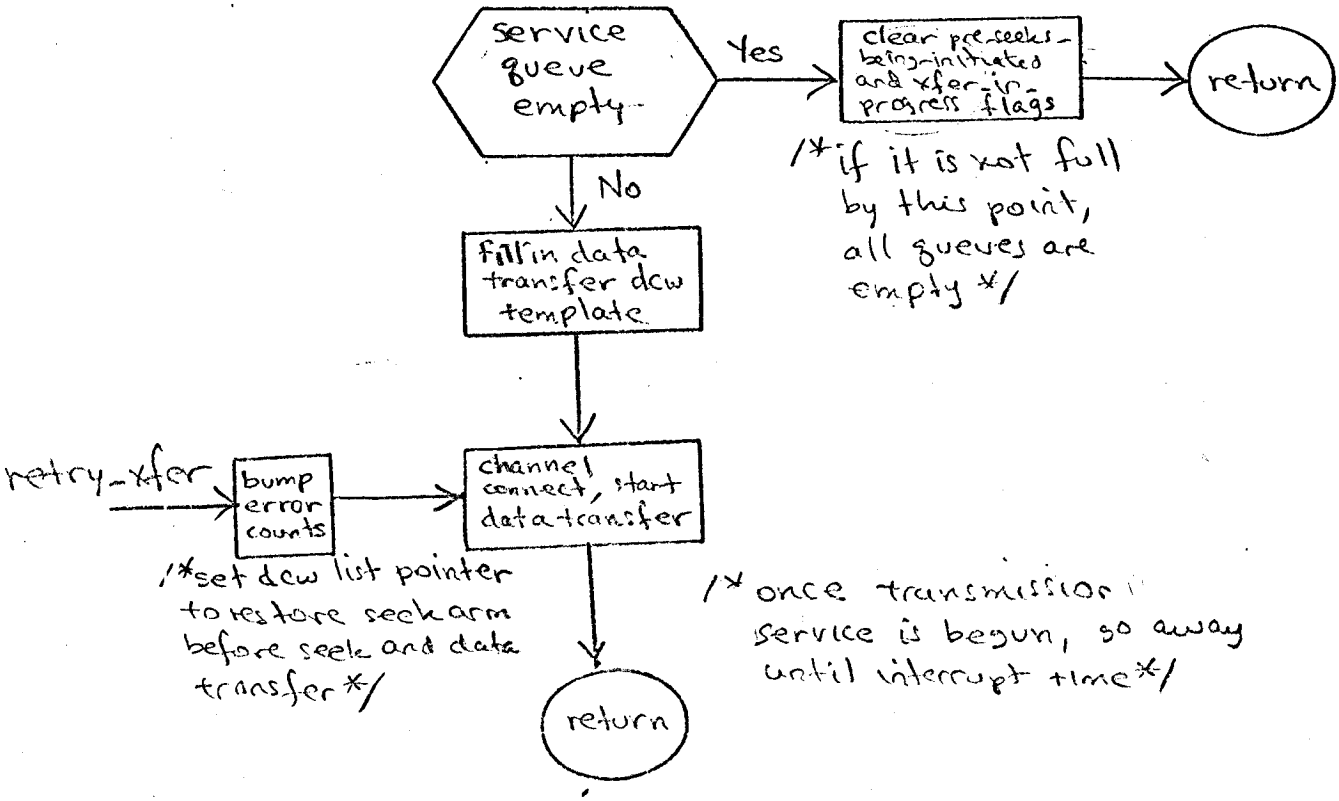
issue_pre_seeks

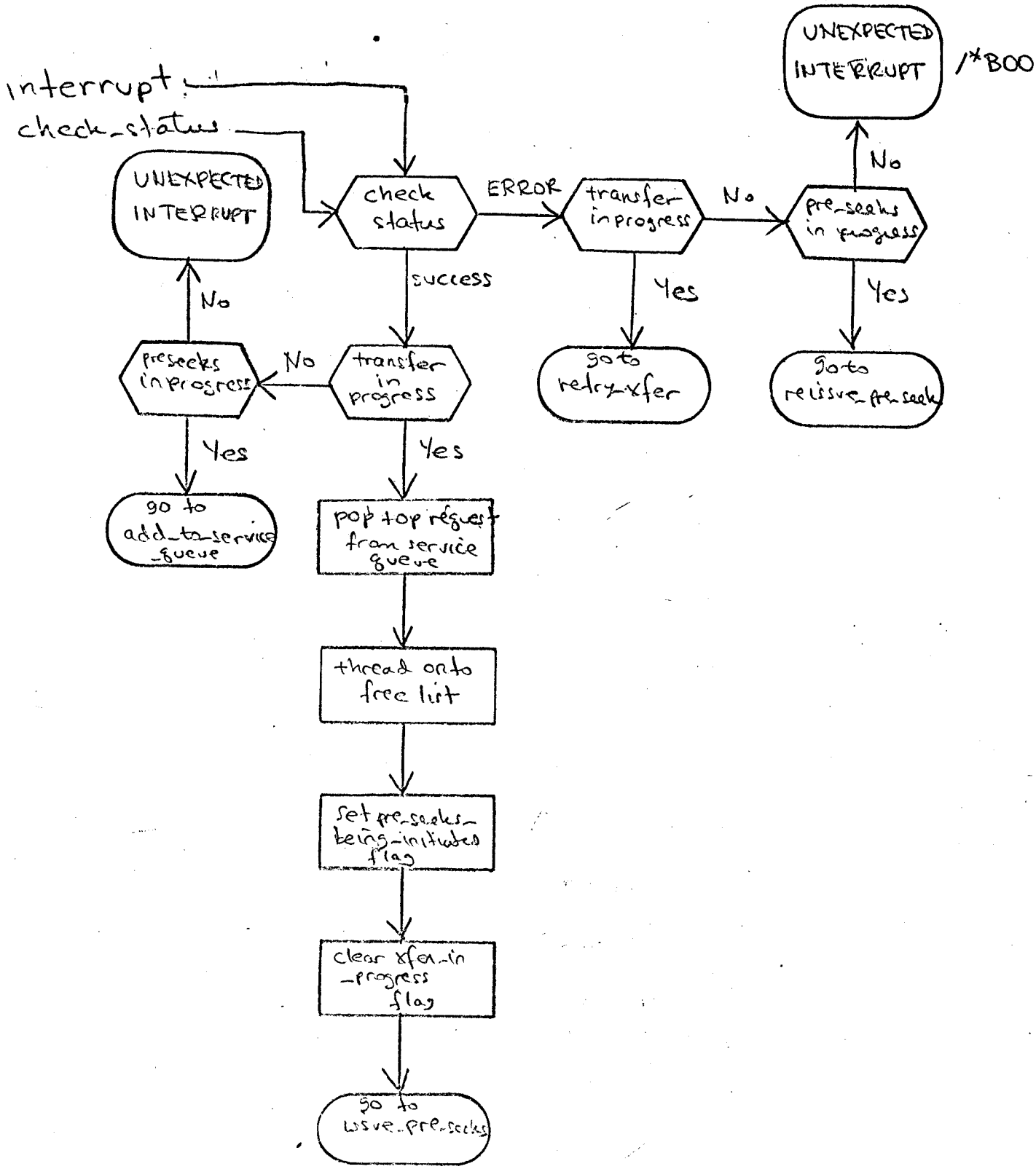


add_to_service_queue



start_xfer




interrupt
check_status
UNEXPECTED INTERRUPT
check status
ERROR
transfer in progress
No
pre_seeks in progress
UNEXPECTED INTERRUPT
/*BOO
No
Yes
success
Yes
No
presecks in progress
transfer in progress
go to retry_xfer
go to relase_pre_seek
No
Yes
Yes
go to add_to_service _queue
pop top request from service queue
thread onto free list
set pre_seeks_ being_initiated flag
clear xfer_in _progress flag
go to wsve_pre_seeks