University of
Waterloo Research Institute

Telidon Server Architecture

Final Report

December, 1983

Waterloo Research Institute

University of Waterloo

Project No. 207-03

Telidon Server Architecture

by

J.A. Field and H.C. Ratz

Sponsored by

Department of Communications

Canada

under

Department of Supplies and Services

Contract No. OST82-00068

Dr. M. Sablatash

Scientific Authority

Rec'd from DRMS Sept '84

December 15, 1983

P
91
C655
F52
1983
c. a

DD 4688798
DL 4735166

Telidon Server Architecture

J.A. Field and H.C. Ratz

Department of Electrical Engineering
and
Computer Communications Networks Group
University of Waterloo
Waterloo, Ontario

Executive Summary

A Telidon system consists of one or more host computers, with their associated disks, and a communications network for connecting user terminals to the host(s). Any component of the system can decrease total throughput and increase delay. This study investigated the performance limits on a Telidon server operating over the telephone or cable television networks. For purposes of evaluation it was assumed that the Telidon system would have 1000 active users, with a total request rate of 100 requests per second, and that the average Telidon page length was 1000 bytes. The study considered the communications network, the host computer, and the disk system with respect to performance and, to some extent, cost.

It was found that the telephone network was not practical for the postulated load, and that the page delivery system would require the use of cable television like facilities. User request transmission is much less critical and could use either the same cable television facility or one of several other methods.

Host computer performance is not a significant factor in the performance of a Telidon system.

If was found that disk service time was a severe system bottleneck, and that several disks would be required to provide the required throughput. Four alter-

nate Telidon server architectures are proposed that use either cache memory, or appropriate organization of the data on the disk, to provide faster response and hence reduce the number of disk units required. These schemes depend on increased knowledge of the data characteristics and the pattern of requests from the users.

An analysis of samples of Telidon traffic showed that the user requests had a highly structured nature, and that predicting of future requests is very practical. Based on the characteristics of the sample traffic, an analysis indicted that for a large number of active users a cache memory of one page per user, combined with predicting, and fetching, future requests, would allow a single disk system to perform as well as a conventional multiple disk system. This was verified by simulation. This system should be investigated further using other, and more extensive, samples of Telidon traffic.

The other Telidon architectures considered include a system with a cache, but without prediction of future requests, and a system that collects requests for an interval of time and then schedules the disk reads to minimize disk service time. This latter system requires that the pages with a high probability of use be grouped on the disk. Both these systems perform better than the conventional system but not as well as the cache plus prediction approach. The fourth architecture considered also uses fetching of anticipated pages but distributes the cache to the user terminals. It appears that providing storage for 6 pages at the terminal would effectively make the host computer and transmission delays transparent to the user. Since the memory requirements at the terminal are small this approach is very attractive. However, its performance is very dependent on the characteristics of the user request pattern, and more detailed information on the structure of data bases and the user request sequences is required before it can be evaluated fully.

Table of Contents

List of Figures

List of Tables

1. Introduction

The purpose of a Telidon System [1] is to distribute textual and graphical information rapidly and economically to many users. Obviously, by dedicating sufficient equipment to a single user, any criterion for timely response can be met. The problem is to configure a system so that for a given number of users reasonable response time criteria can be satisfied with a minimum of equipment, i.e. a minimum of expense. Alternatively, if it is discovered that some basic set of equipment is required, how can the system be organized to serve the maximum number of users?

We wish to distinguish two types of systems. The first is a small "private" system that serves only a few users. The other is a large "public" system serving many users. As shown in Figure 1.1, the first approach tends to have a fixed cost with no growth potential, while the latter has a larger initial cost plus a small incremental cost for each user. For a small number of users the first approach is more economical, while for a large number of users the latter approach yields the lowest cost per user. It is this latter case that we consider in this report.

In considering a large system it is necessary to identify the areas that could cause performance degradation, and develop strategies that reliably and economically minimize the problems. The performance factor with which we are most concerned is the response of the system to a user request. A Telidon system consists of the Telidon server (host computer(s)) and their associated disks), a communications network for connecting the user terminals to the Telidon server, and the user terminals. In this study we are interested primarily in systems that use either the telephone network or a cable television (CATV) network for communications.

All components in a Telidon system introduce delay and decrease total throughput. The possible problem areas can be identified by tracing the flow of

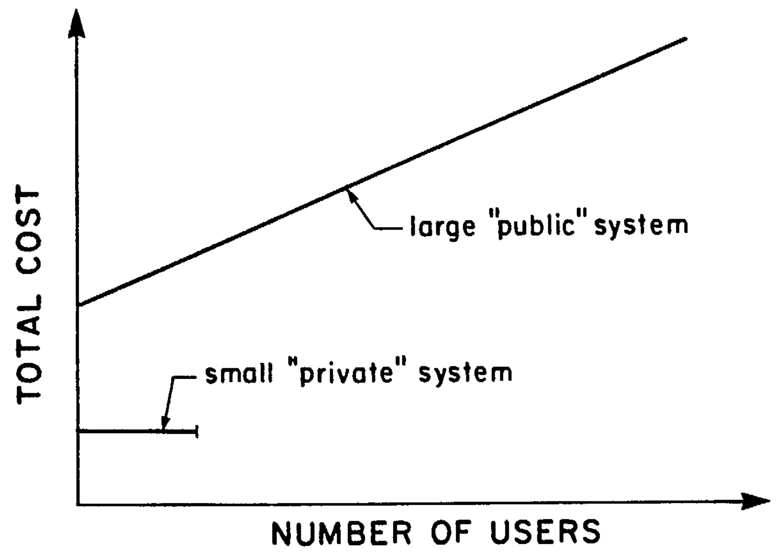

Figure 1.1: Telidon cost vs. number of users.

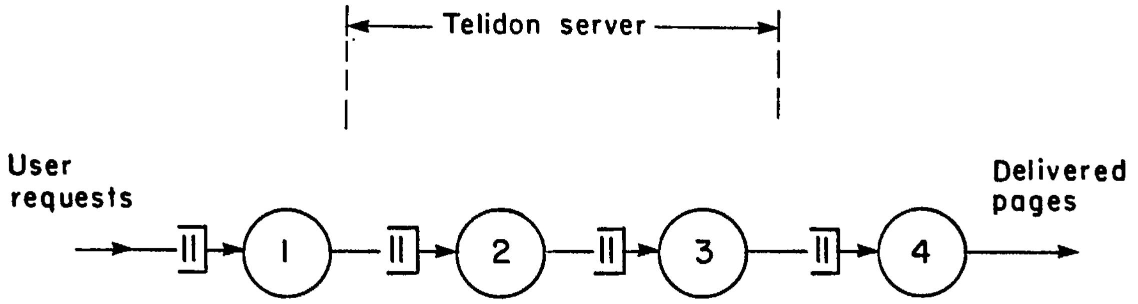

Figure 1.2: Queuing model of user request processing in Telidon.

a user's request and the server's response. Ignoring delays associated solely with the user and user terminal, four tandem delays can be identified: the user request is transmitted over the communications network, the request is analysed by the Telidon server host computer, the requested page is read from disk if not in memory, and the requested page is transmitted over the communications network. (See Figure 1.2). At each of these four points significant queuing delay can occur if the server is overloaded.

Section 2 considers the significance of each delay. Two of the delays are associated with the communications network, and the other two with the host server. In Section 2.2 it is shown that a distribution system based on CATV technology will give satisfactory performance for both communications servers. Of the two delays in the host server, it is shown in Section 2.3 that the delay associated with the analysis of the request is not significant. For large data bases it is necessary to hold the Telidon pages on disk. Thus the page access delay is likely to be the dominant factor in system performance, and multiple disk units appear to be required to reduce the delay to an acceptable value. This problem is considered in Section 2.4. It is shown there that single disk systems can give acceptable performance if certain information about the users' page request characteristics is available for use in the system design.

Section 3 considers the user characteristics that effect system performance. With a large data base on disk, the performance of a Telidon system, measured by the response time to requests, can be improved by holding anticipated requests in a high speed cache. The feasibility of such prefetching depends upon taking advantage of user behaviour. From empirical data, it is found in Section 3.2 that the distribution of the frequency of use of pages is skewed, and hence, the cache approach shows considerable promise even without prefetching of anticipated pages. In Section 3.4 it is found that the effective range of choices in sequential selection is very limited. Hence it is possible to develop a scheme for prefetching pages in anticipation of the next user request. Based on these results it is concluded that a cache size approximately equal to the number of users when that number is large, would be adequate to hold prefetched pages about 95% of the time. Problems in the management of the cache, and comparisons between centralized and decentralized cache are also discussed.

Section 4 discusses the merits of five Telidon architectures. The first is the conventional multidisk approach, while the others include some, or all, of cache memory, prefetching of anticipated requests, and data organization on the disk combined with disk access scheduling. These extensions either reduce disk traffic, reduce time waiting for a response, or reduce the mean time to transfer data from the disk. The major emphasis in Section 4 is on a system that combines the use of cache memory with the prefetching of anticipated requests. Based on the results of Section 2 and 3, and from simulation results, it is concluded that such a system is preferable to the conventional multidisk system. A system with a cache memory without prefetching, and a system using data organization on the disk combined with disk access scheduling are also discussed. These systems also appear to have some advantages over the multidisk system. Finally, a system using a distributed cache and prefetching of anticipated pages, combined with data organization on the disk and disk scheduling is discussed. This system has the potential to make the Telidon server and communications delays in the system transparent to the user.

2. Delays in Telidon Systems

2.1 Introduction

As indicated in Section 1 a Telidon system can be modelled as four tandem servers, each with a queue of requests waiting for service. These servers represent the transmission of the request, the analysis of the request by the host computer, drawing the requested page from the Telidon system memory, and finally the transmission of the page to the user. Obviously, if any of these operations introduces a large delay, the problem can be eliminated by the use of parallelism, i.e., multiple communications channels, or multiple processors, or multiple disk units. This approach, however, may be unreasonable from the economic viewpoint and less expensive methods are more desirable. In the following sections we will consider each of these delay sources. Where it appears that a significant delay may occur various schemes for improving performance are discussed.

To discuss the relative performance of various schemes it is necessary to have a model of a typical Telidon system. We will follow the approach of [2] and assume that a typical Telidon system consists of 1000 active users, each making a page request every 10 seconds. The page request message is in the order of 10 bytes, and the average page length is 1000 bytes. This produces a total traffic out of the Telidon system, after allowing for headers, error checking, etc., in the order of 1 Mbit/s. The total request traffic, on the other hand, is only in the order of 0.1 kbit/s. We will further assume that response characteristics similar to those of interactive computing are required, i.e., that the response time be in the order of one to three seconds. This model will be used throughout the balance of this section when comparing different alternatives.

2.2 Communications System

Two communications channels are required for each user-server connection. These channels may be either dedicated to a single user or shared among several, possibly all, users. The essential difference from the performance point of view is that on a dedicated channel the only delays are transmission time and signal propagation delay, whereas on a shared channel additional, and random, delays can be introduced by service to other traffic. These additional delays may cause a serious degradation in system performance.

The types of dedicated channels that are available are permanent circuits, dial telephone circuits, and switched digital circuits. Possible shared channels are broadcast, packet switching networks*, local area networks, and metropolitan area networks using CATV technology. In this investigation we do not consider local area networks, because of their limited geographic range, nor broadcast, since it is subject to bandwidth allocation regulations. However, most of the technical comments about CATV distribution are also relevant to broadcast.

Since in normal operation a user request is followed by a Telidon server response, the two channels can be provided on the same medium by a half- or full-duplex mode of operation. However, since the data flow is highly unbalanced, with the information delivery channel carrying in the order of 100 times as much traffic as the request channel, this may not be the most effective mode of operation. Indeed it is not clear that both channels should be provided by the same technique. Table 2.1 shows all possible combinations, and indicates three groups where all the members of a group have similar advantages-disadvantages. Each group will be discussed in detail below.

* It might be noted that while packet switched networks use shared channels, the sharing is done at the network level and is invisible to the user. Hence to the user they are very similar in use to switched digital circuits. However they exhibit the random delay of a shared facility.

Table 2.1

Communication Channel Alternatives

request channel	information channel				
	telephone	dedicated circuit	switched digital circuit	packet switched	CATV
telephone	A				C
dedicated circuit		A			C
switched digital circuit			A		C
packet switching				A	C
CATV					B

The entries in Table 2.1 that have not been assigned to a group represent combinations that have increased cost with no performance improvement over combinations with both channels the same, or represent combinations that use each medium for the operation for which it is least suited.

2.2.1 Group A

In this group both the request and information channels use the same transmission medium. The channels may be either dial telephone circuits, dedicated circuits, switched digital circuits or packet switched virtual circuits. All have the advantage of dedicated access, although packet switching does suffer random packet delays. Current Telidon systems, using modems and either dedicated circuits or the dial telephone system, fall into this class. However, such systems have a small number of users, and expansion to a large number, i.e. 1000, active users is not practical.

First we will consider the telephone system. Telephone systems are effectively limited at 1200 bit/s due to bandwidth and modem costs. This means that the transmission of a page of information takes in the order of 10 seconds. This exceeds the desired response times proposed in Section 2.1 without considering request processing and other delays in the system. In addition, supporting 1000 active users would require 1000 telephone lines, modems and interfaces at the computer with a cost in the order of one million dollars. While appropriate multiplexors, etc., could reduce the cost it would still remain prohibitively large.

We now consider dedicated lines. If the lines are short then limited distance modems may be used. These are higher speed and less expensive than those discussed for the telephone systems. Hence such a system could be practical for limited distances. However, for longer range the modem costs increase, and the line charges become high, yielding the same cost problem as for the telephone system.

Digital switched circuits do not have the severe bandwidth problem of the telephone system, but they are more expensive and not as readily available. The interface problem at the computer should not be as severe as for the telephone system but will still be a significant cost.

Packet switching networks simplify the interface problem at the server, since one interface can handle many calls. However, given the traffic requirements, the system would need in the order of one hundred 9600 kbit/s ports into the network. This would have an interface cost comparable to the telephone interface, and the expense of the interface at the terminals would increase. Care would have to be taken in the software to keep the load balanced over all the ports. This may not be easy. The time to deliver a page (including network delays) would be in the order of one to two seconds.

In summary, all the class A systems suffer from high cost, interfacing complexity at the Telidon server, and information delivery times that are on the high side.

2.2.2 Group B

This group has a single member. The CATV system is used both to carry requests to the Telidon server, and information pages to the user. The following discussion assumes separate CATV channels are available for the request and page delivery traffic. While in principle a single channel could be shared using local area network techniques, the sharing overhead is very high due to the distances involved and the highly biased traffic origination pattern. In addition, it is not good practice to have a high power transmitter and a sensitive receiver at the same frequency connected to the cable at the same point, i.e., the user taps. Hence only the two channel approach will be considered.

The idea of using the CATV distribution system for the delivery of the Telidon information pages has been widely proposed. The advantage is obvious: a 6 MHz channel on the CATV system could carry at least 5 Mbit/s of data. Hence, serving 100 user requests per second would use only about 20% of the channel capacity. This introduces only a few milliseconds of queuing delay waiting for transmission, leaving an ample time allowance for request processing, etc. There are two disadvantages to the system, compared to dedicated lines. First, the users must have address decoding equipment to select the correct page. This is not a significant problem since such equipment would not be expensive in large quantities. Second, the information is less secure against passive wire tapping since it is "seen" by all users. We conjecture that for most Telidon systems this is not a significant problem.

The CATV channel used as the request channel could be shared by the users in a random access mode in lightly loaded systems, or in a polling mode in more heavily loaded systems. The random access approach is simplest. In a random access scheme any terminal with a request transmits when it is ready. If two transmissions overlap (collide) then both requests are lost. The probability of losing a transmission is approximately $\lambda(2l/c)$ where λ is the number of transmissions per second, l is the message length, and c is the channel capacity. If the user requests are 100 bits long, with 100 requests per second, then on a 5 Mbit/s channel, the probability of a request being lost is 0.4%. Lost requests can be detected by a time out mechanism, or alternatively, by transmitting each request twice, with a random spacing, this loss can be reduced to 0.006%. This is sufficiently small that one may wish to ignore the loss and rely on the user resubmitting the request.

It should be noted that the above technique relies on the channel being lightly loaded. If higher request rates are desired, or if a full 6 MHz channel is not available, then some form of polling scheme that uses less bandwidth could be implemented. The disadvantage of such schemes is that they are more complex, and introduce extra delay.

2.2.3 Group C

This group contains communication systems that use a CATV channel for distribution of Telidon pages combined with a dedicated request channel using either dedicated lines, telephone, digital switched circuit or packet switching. This approach retains all the advantages of Group B with respect to page distribution, and that discussion will not be repeated here. The use of dedicated channels for request traffic, rather than CATV system as in Group B, may be desirable in some situations. The most obvious case is when an existing CATV system was not designed to allow traffic to originate at a customer's premise, or has limited bandwidth in that direction. Even if these restrictions do not exist, this approach may be attractive when all the costs are considered. It should be noted that since the request channel is low speed, the modem and multiplexing costs at the Telidon server will be reduced substantially compared to Group A. One very interesting possibility is the use of a packet switched network. Two 9600 bit/s ports at the server would be more than sufficient to carry the request traffic. The users could use either dedicated lines to the packet network if they were heavy users, or 300 bit/s dial telephone access if they were light users.

2.2.4 Summary

The use of a group A system appears feasible only where the geographical area is small and dedicated lines with low cost limited distance modems can be used. However, either group B or group C using shared distribution over a CATV system appears feasible for both local or city wide systems.

The method of collecting request traffic can be handled adequately by all suggested mechanisms. Given the low volume of request traffic there should be no difference in performance and the decision can be made purely on an economic basis. It would appear that if a CATV cable is being used for page distribution that the cable would also be the most convenient method of collecting the page requests. It might not, however, be the most economical.

In summary, while group A systems may have some specialized applications, most large systems will fall into groups B or C using a CATV cable system for page distribution. The balance of the report will be based on this assumption, and will assume that communication delays are negligible.

2.3 Processor Performance

The processing capabilities required in a Telidon server are not large. Typically a Telidon response either requests the next page in a sequence of pages, or specifies one of the items in the menu currently being displayed. To process these responses the system must maintain, for each active user, a short table containing the successor page identification for each menu item. Obviously where the page is one of a sequence of pages the table would reduce to a single item identifying the next page. When a user request is received, all that is required is a short search of the table entries to find the addressing information for the selected entry, and to initiate a disk read if the page is not in memory. When the page is available, either immediately, or after the disk read is complete, it must be queued for transmission to the requester. Assuming that all input/output is done by direct memory access (DMA) transfers, in the order of 1000 instructions per request seems adequate to perform the above operations, including the processing of interrupts at the end of the DMA transfers.

While in some cases it may be necessary to perform several disk accesses to obtain page directories, this will have to be a rare occurrence, or performance will drop due to disk delays. Hence this type of activity was ignored in the above estimate. If the use of labels is permitted, there will be some increase in processing time, but the use of efficient hash table techniques should keep this increase small.

Hence the processing power required is sufficiently small that it could be provided by any 16-bit microprocessor, and by some of the 8-bit microprocessors. For example, 1000 users with a request rate of 1 page every 10 seconds per user on a computer with a 3 μs average instruction execution time requires only 100 request/s $\times$ 1000 instruction/request $\times$ 3 μs/instruction = 0.3 of the processing capacity of the computer. To estimate the interference from the DMA transfers we note that memory speeds are seldom worse than 1 μs per byte transferred. In the worst case of no duplicate requests, and no pages already in memory, all pages would have to be transferred from disk to memory, and then from memory to the output channel. For 1000 byte pages this would require 100 page/s × 1000 byte/page × 2 × 1 μs/byte = 0.2 of the memory bandwidth. Hence only if the processor-DMA overlap were zero would the computer utilization reach 0.5. This leaves ample time for statistical fluctuation in arrival rates and processing time. A more powerful machine would be less heavily loaded, but not significantly, because this type of operation primarily uses the basic "bookkeeping" instructions such as compare, load, store, increment, etc. whose time on large and small processors is comparable (unlike floating point arithmetic for example).

2.4 Disk Performance

The large amount of data supporting a Telidon system requires a disk for mass storage. The rate of random retrieval of page information from a disk is significantly lower than the anticipated page request rate; hence, disk performance could well be the limiting factor on system performance. For example, consider the IBM 3350 disk unit. This device has the characteristics

average seek time	25 ms
maximum capacity per track	19000 bytes
time per revolution	16.7 ms

If the pages being read are stored consecutively on the disk, then for 1000 byte pages, a transfer rate of 1000 pages per second is possible. However, if the pages are scattered the average access delay consists of a 25 ms seek plus 16.7 ms of rotational delay, which yields a transfer rate of only 25 pages per second. Other disks have similar characteristics. The important point is that the rate for random retrieval of page information is significantly lower than the anticipated page request rate of 100 pages per second suggested in Section 2.1.

The initial reaction to the fact that a single disk drive cannot meet the anticipated page request rate may be that it is not a serious problem since multiple disks will be required purely for system reliability, and that the load can be shared across many disks. This is a dangerous assumption. The repair time for disks can be long, and during this period the system must still meet performance requirements. The correct viewpoint is that the system will need n disks to meet performance requirements, and a further m disks to meet reliability requirements. Obviously one wishes to minimize n and m in order to reduce cost. In this study we are concerned only with n.

Given that a single disk unit using random page retrieval cannot support the page request rate, three strategies are possible: replicate the disk unit to reduce the access rate, provide a cache memory for frequently accessed pages to reduce the number of disk accesses, or organize the page requests so that sequential access rather than random access is made to the disk.

2.4.1 Disk Replication

Replicating the disk system until the combined service rates exceeds the user request rate is the easiest and technically safest of the three strategies. The distribution of service times can be obtained by simulation without any specific knowledge of the data base organization. Thus it is easy to verify that performance criteria will be met before a commitment to a specific system is made. The only precaution required during operation is that the load on the disks remains balanced. This is not difficult if all the disk units are attached to the same computers. However, if multiple computers are used, then there must be a facility to pass user requests between computers to maintain the desired load balance. In this case it would probably be best for one computer to process all requests, and distribute them to the other computers.

The disadvantage of disk replication is, of course, the cost of the additional disks. For the typical system parameters given in Section 2.1, four disk units would be required to match the request rate. Additional capacity would be required to control the queuing delay caused by fluctuations in load.

2.4.2 Cache Memory

As will be discussed in Section 3, some pages of a Telidon data base are accessed much more frequently than others. If these commonly used pages were held in a cache memory, then the random access request rate to the disk(s) could be reduced. If the cost of the cache is less than the cost of the disk(s) it eliminates, then the system is economically superior to that discussed in Section 2.4.1. One would also expect that it has superior performance since many of the requests will be served from the cache without queuing delay. To obtain some performance bounds we will analyse a system with a cache.

Assume that the cache has a size C so that a fraction f of the requests is served instantly from the cache. The balance of the requests are placed in a queue for disk service. Let the mean arrival rate for user requests be λ, and the mean disk access time (seek and read) be $\bar{x}$.

If the number of disks is n, then the mean arrival rate of user requests to a disk queue is $(1-f)\lambda/n$. (This assumes a separate queue for each disk. If the pages are replicated on all the disks a single queue can be used which yields better performance; thus for $n>1$ the following results will be pessimistic.) If we assume that the user requests can be modelled as a Poisson process, then disk access is described by an M/G/1 queuing model. Hence the average time in the system for the user requests is

$$T_D = \bar{x} + \frac{\rho\bar{x}(1+C_b^2)}{2(1-\rho)} \qquad (2.4.2.1)$$

where $\rho = \bar{x}(1-f)\lambda/n$ is the disk utilization and C_b^2 is the squared coefficient of variation for the access time. For a disk system one would expect $C_b^2 < 1$. Now

$$T_D = \bar{x}\left\{1 + \frac{\bar{x}(1-f)\lambda(1+C_b^2)/n}{2\{1-(1-f)\bar{x}\lambda/n\}}\right\} \tag{2.4.2.2}$$

and the overall mean time in system, including the requests effectively serviced instantly from the cache is

$$T = (1-f)T_D \tag{2.4.2.3}$$

It should be noted that both T and T_D are significant. T gives the average response time to a request; however it must be remembered that $(1-f)$ of the requests have a mean response time of T_D. This may be significant if the system has a performance criterion of the form that a certain percentage of the requests must be serviced in less than a specified time. To obtain an exact verification that such a criterion is being met will require a simulation of the system.

Provided that f is large both T_D and T are quite satisfactory for typical values of $\bar{x}$ and λ. However, we note that

$$\rho = \bar{x}\lambda(1-f)/n < 1 \tag{2.4.2.4}$$

or

$$f > 1 - \frac{n}{\bar{x}\lambda} \tag{2.4.2.5}$$

is required for satisfactory response times. For $\bar{x}$ = 40ms, λ = 100 requests/s, and $n=1$ this implies $f > 0.75$ is required. Intuitively, this appears a severe requirement. The question of how the "hit ratio" f varies with cache size C, and how large a cache size is required is unknown. Section 3 discusses this problem. Unless a method is found to insure that f is large, then $n > 1$, i.e., multiple disks, is required and the cache method will not give a significant economic improvement over the multiple disk approach of Section 2.4.1.

2.4.3 Disk Access Scheduling

If all the page requests are to be provided from a single disk without using a cache memory it is clearly impossible to service the page requests on a one-by-one basis; they must be grouped and ordered to minimize the seek time overhead. Obviously if all the requested pages were adjacent, then the transfer rate would approach the maximum disk transfer rate which is considerably greater than the page request rate. The problem is determining the amount of grouping required to achieve mean transfer rates greater than the page request rate and establishing the associated delay in page delivery.

For the purpose of analysis, we will assume that the Telidon pages can be arranged on the disk such that some fraction f of the requests can be served by scanning a "common" set of pages at the maximum transfer rate r and reading the required pages. This would be typically several pages per disk rotation with a seek being required only when all tracks in a cylinder have been read. If the number of pages in the common set is C, and the total pages being read is m, with fm from the common set, then the average transfer time per page for pages from the common set is $C/(rfm)$. The balance $(1-f)m$ of the pages required are read individually with an average fetch time of $\bar{x}$. This yields an average page service time t_1 of

$$\begin{aligned} t_1 &= f\left[\frac{C}{rfm}\right] + (1-f)\bar{x} \\ &= \frac{C}{rm} + (1-f)\bar{x} \qquad (2.4.3.1) \end{aligned}$$

It should be noted that this model is not valid when m is sufficiently small that it is faster to seek all the pages in the common set rather than scan over the intervening pages. This occurs when $\bar{x}<C/(rfm)$, and $\bar{x}$ is, therefore the upper bound on the average page service time. Note also that the number of pages scanned in the common set per page transferred to memory is

$$k = C/(fm) \tag{2.4.3.2}$$

Obviously $k<1$ is invalid. At $k = 1$ all the pages in the common set are being read. This implies that $m \le C/f$.

The time to transfer all m pages is

$$t_m = \begin{cases} t_1 m = \frac{C}{r} + (1-f)\bar{x}m, & \bar{x} \ge C/(rfm) \\ \bar{x}\, m, & \bar{x} < C/(rfm) \end{cases} \tag{2.4.3.3}$$

and during the time t_m the average number of new requests received is

$$b = \lambda t_m \tag{2.4.3.4}$$

Figure 2.1 shows the relationship between b and m for a possible set of parameters. From Figure 2.1 it is noted that a value of m exists such that $b = m$. Call this value m'. Now, let the system operating policy set m equal to the number of requests awaiting service. If m is less than m' then the probable number of arrivals during the service cycle is greater than m. On the other hand if m is greater than m' the probable number of arrivals during the service cycle is less than m. Hence the system will converge towards a stable operating point at $b = m'$.

We note that if $\lambda \bar{x} > 1$ and $f > 1 - \frac{1}{\bar{x}\lambda}$ the b vs. m relationship always has the form shown in Figure 2.1. Hence for these conditions, the stable operating point is given by

$$m' = \lambda t_{m'} = \lambda \left\{ \frac{C}{r} + (1-f)\bar{x}m' \right\} \tag{2.4.3.5}$$

which yields the average number of pages being read per service cycle as

$$m' = \frac{\lambda C}{r\{1-(1-f)\lambda \bar{x}\}} \tag{2.4.3.6}$$

and the average service cycle duration of

$$t_m' = \frac{C}{r\{1-(1-f)\lambda \bar{x}\}} \tag{2.4.3.7}$$

The time that a request waits for service can vary from almost zero, for a

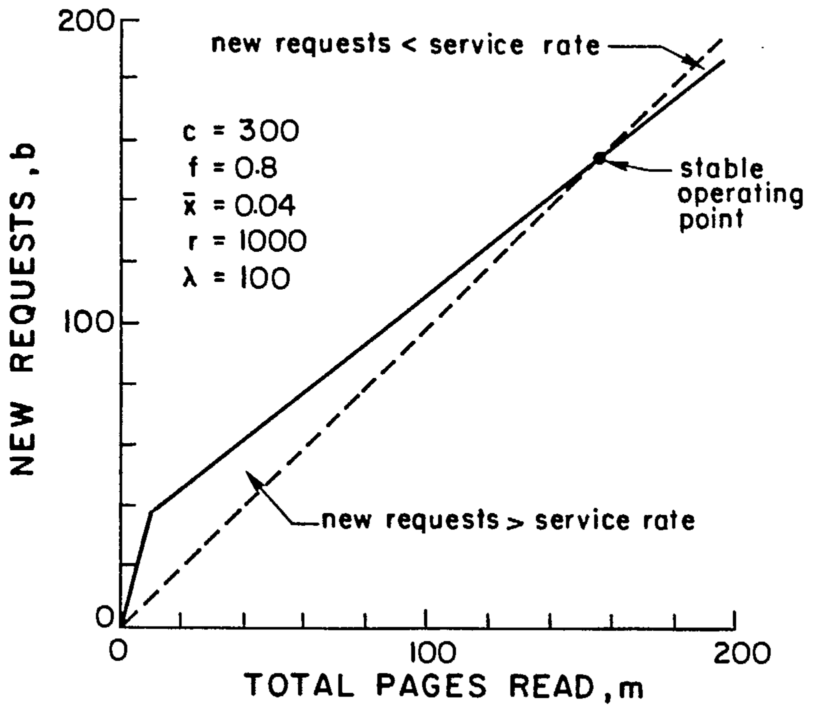

Figure 2.1: Relationship between the expected number of page requests, b, and the number of pages read, m, during a service cycle.

request that arrives just as the block of requests is being organized and is the first served, to $2t_{m'}$ for a request that just misses a service cycle and is the last served on the next cycle. While the distribution is not uniform across this interval, since the disk reads are closer together when the common page set is being scanned, $t_{m'}$ is a reasonable approximation for the mean time a request is in the system.

As an example, for the typical parameter values shown in Figure 2.1, $t_{m'} = 1.5$s. This is a feasible value for a Telidon server, but it must be noted that it is very sensitive to the value of f. To insure a specific value of $t_{m'}$ requires that

$$f \geq (1 - \frac{1}{\lambda \bar{x}}) + \frac{C}{\lambda \bar{x}\, r t_{m'}} \tag{2.4.3.8}$$

It is obvious that the system is practical only if $\lambda\bar{x}$ is small, and if a small value of C yields a high value of f. This problem is identical to that for the system with cache memory discussed in the Section 2.4.2. Again information is required on the relationship between a "common" set of pages and the "hit ratio" before the effectiveness of the system can be evaluated.

2.4.4 Summary

The time for random access of Telidon pages on a typical disk is too high for a single disk system to support a reasonable number of users. A reliable solution to the problem is to provide multiple disks. Alternative solutions that require fewer disks require that a collection of common pages, either in the computer memory or consecutive disk locations, provide a large fraction of the user requests. To evaluate these latter architectures requires a knowledge of the frequency of use of the pages in the Telidon data base. This problem is considered in Section 3.

3. Telidon User Behaviour

3.1 Introduction

The response time of a Telidon system to the user is a critical measure of its performance and usefulness. For large data bases, it is not possible to hold all the pages in a high speed store or cache. The use of a disk or several disks to provide the large storage necessary presents great difficulties in controlling the access time when there are many users. Suppose a cache supplies the data to the user communications network, then the critical area is the communications between the cache and the disks which hold the data base. The ultimate objective would be somehow to have all the necessary pages in cache when the requests arrive so that the disk access time is not part of the response time to the user.

We can be quite sure that all pages will not be equally used even in a network of many users. If it were possible to hold a substantial proportion of the pages which are more likely to be used in cache, then the disk access time would only be encountered by a user occasionally. Hence we wish to model Telidon user behaviour regarding the distribution of the relative frequency of use of pages in the data base. It will be found that a minority of pages account for essentially all the use. However, it is important to know from empirical data, the values of the parameters describing the distribution.

An appropriate model for the distribution can be inferred from other areas of human behaviour such as languages and economics, where a distribution proportional to a power of the rank plays a dominant part. Here the rank would be the rank of the Telidon pages listed in order of decreasing use. We will designate the anticipated frequency of use of the page at rank r in such a list as

$$A_0(r) \quad r = 1,2,3\ldots$$

where

$$A_0(\tau) \geq A_0(\tau+1)$$

If a list of such zero order anticipation numbers were available from the actual use of a particular data base, then it would be possible to calculate the performance as a function of cache size using this approach alone.

It is clear that once a user is in the data base system, he will follow a stream of selections with a very small number of effective branches relative to the size of the total data base available. Therefore if we could predict his next choice from among a very small set and have them ready in cache, the apparent performance could be enhanced. To do this, we must make use of the actual request made immediately preceding the next anticipated one. In other words, we need the first order conditional probabilities to predict his next request conditioned upon the last one known to have been made. We shall call these conditional probabilities the relative anticipation probabilities, A. These must be obtained by observations of user behaviour on a particular data base and stored in an extensive table. This approach holds some promise if indeed the set of anticipated next requests conditioned upon the known last request, is quite small.

We note that the conditional probabilities are invoked by the occurrence of an actual request at a particular instant. Therefore, as time passes these conditional probabilities lose their validity since it becomes increasingly likely that the user has made some other choice. Hence these relative anticipation numbers must become functions of time, $A(t)$, where time is measured from the instant of the last known request. A method is developed to have these anticipation probabilities decay in a form deduced from observations on the distribution of the pause interval between user requests (Section 3.3). We thus arrive at a situation where the cache contains those pages most likely to be used next, where "most likely" is based on data concerning all immediately preceding requests , as well as the relative frequency of use of pages.

3.2 Page Use Distribution

Suppose the pages in a Telidon data base are ranked according to their relative frequency of use in descending order. Then, there are excellent reasons to anticipate that such a frequency distribution will be of the form [3-5]

$$f(r) = f_1 / r^{\alpha} \qquad (3.2.1)$$

When the exponent α is unity, this is known as Zipf's Law. Eqn. (3.2.1) can be written in the form:

$$f(r) = f_1 \exp(-\ln(r)/B) \qquad (3.2.2)$$

which is part of an extensive analogy with thermodynamics in which the parameter B is the "temperature" of the distribution. If $B<1$, then the system is closed in the sense that the distribution can be summed over all ranks to infinity, while if $B>1$ the distribution is open and one expects that at some high rank, the distribution will fail or fall off drastically, or that there is a maximum rank (finite set).

The cumulative distribution function is more convenient for the reduction of observed data. It is defined:

$$S(r) = \sum_{i=1}^{r} f(i) \qquad (3.2.3)$$

From the integral of the probability mass function in eqn. (3.2.1) we can expect the cumulative distribution function also to be a power of the rank as follows:

$$S(r) = S_1 \, r^m = e^b \, r^m \qquad (3.2.4)$$

which leads directly to the linear equation

$$\ln S(r) = b + m \cdot \ln(r) \qquad (3.2.5)$$

In order to relate this to the probability mass function, we take the definition

$$\begin{aligned} f(r) &= S(r) - S(r-1) \\ &= e^b \left(r^m - (r-1)^m\right) \end{aligned} \tag{3.2.6}$$

and expand the second term using the binomial theorem. This gives us, approximately (which improves with increasing rank):

$$f(r) \approx e^b \, m r^{m-1} \tag{3.2.7}$$

so that the parameters that control the shape of the distribution are related by:

$$a = 1 - m = 1/B \tag{3.2.8}$$

Three sets of data were acquired over seven months at the University of Waterloo. The cumulative page use distributions followed eqn. (3.2.5) very closely up to the point where the cumulative sum approaches unity. This is shown in Figure 3.1 and Table 3.1 where the three data sets are identified by the codes 823, 827, and 210. Data set 823 used 820 pages, and we had 41 000 user requests recorded; while data set 827 used 659 pages with over 14 000 user requests recorded. Later, data set 827 was expanded to 1986 pages, and had over 70 000 user requests recorded and is called set 210. The exponent parameters for the data sets differ by 15%. The "effective number of pages" is obtained by extrapolating the linear relationship, eqn. (3.2.5), until the cumulative distribution is unity. This gives:

$$v = \exp(-b/m) \tag{3.2.9}$$

Actually, the distribution does not end there, but deviates from the assumed form and approaches unity for the last page. The measure of the accuracy of the fit of the observed data to the model is given by the maximum error between the data and the point where the two asymptotes meet. This error is about 10% to 15%. Note that the fraction of the available pages defined as "effective" is between 18% and 40% of the total number available.

Figure 3.1 answers the first question about predicting page use; 18 to 40% of the pages available account for 85% to 90% of the use. While it can be expected that the proportion effective will decrease further with larger data bases, the

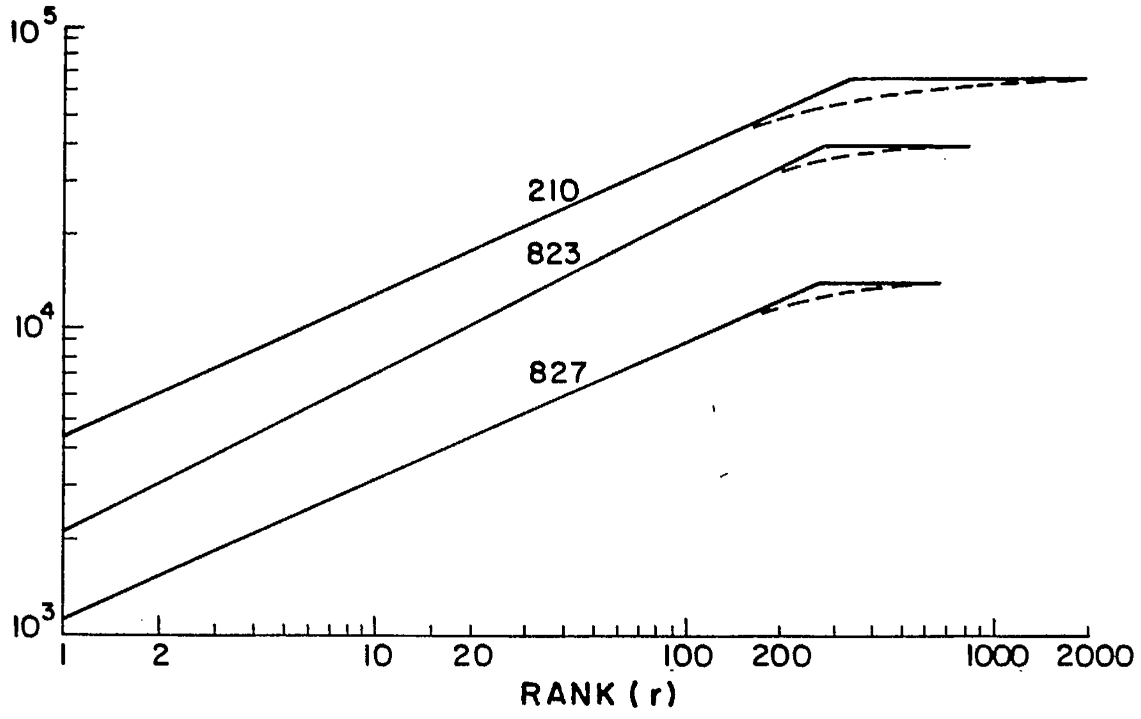

Figure 3.1: Cumulative frequency of page use.

Table 3.1

Parameters of the Page Use Distribution

Data Set ID		823	827	210
Sample Size	$S(V)$	41 001	14 400	70 464
Number of Pages	V	820	659	1986
Cdf slope	m	0.5434	0.4624	0.4601
"Temperature"	B	2.19	1.86	1.85
Effective pages	v	283	265	357
Proportion effective	v / V	34.5%	40.2%	18.0%
Maximum frequency	$S(1)$	4.7%	7.8%	6.4%
Error asymptote at v		10%	9.7%	15.5%

Note: Data Set 827 is a subset of Data Set 210.

number may remain too high to retain in cache. Horspool has reported similar results [2].

This very unequal frequency of use can be emphasized by calculating the entropy of the distribution, using data set 210 as an example. In this set the break point or "effective" number of pages is:

$$v = 357$$

The entropy of the asymptotic distribution of eqn. (3.2.1) from rank 1 to 357 is 8.103 bits (which is equivalent to 275 equally likely alternatives). If we now take into account the ranks from 358 to 1986 with a rank distribution over that range, we have two straight lines (on a log-log diagram) for the complete range and the entropy is increased to 8.472 bits (which is equivalent to 355 equally likely alternatives). Thus in this case, the average uncertainty in predicted page use as measured by the entropy, is approximately the same as that for the effective number of pages from eqn. (3.2.9) considering these to be used independently and equally likely.

3.3 User Pause Time

The user pauses between requests and it is during this time that we can ensure that his anticipated next request is available in cache. However, if an anticipated request should not occur, its validity in cache expires and it can be replaced. Therefore, it is important to know the distribution of the pause time both for the prediction of the next request, and to indicate how long a predicted page should remain in cache.

Figure 3.2 shows the cumulative distribution function for the user pause time. Note that 50% of the requests are made within 5 seconds of the previous request, while 3% may take as long as 30 seconds. The complementary cumulative distribution function (Ccdf) is the probability that a predicted page in cache will yet be used, treating all such individual predictions as independent. If the

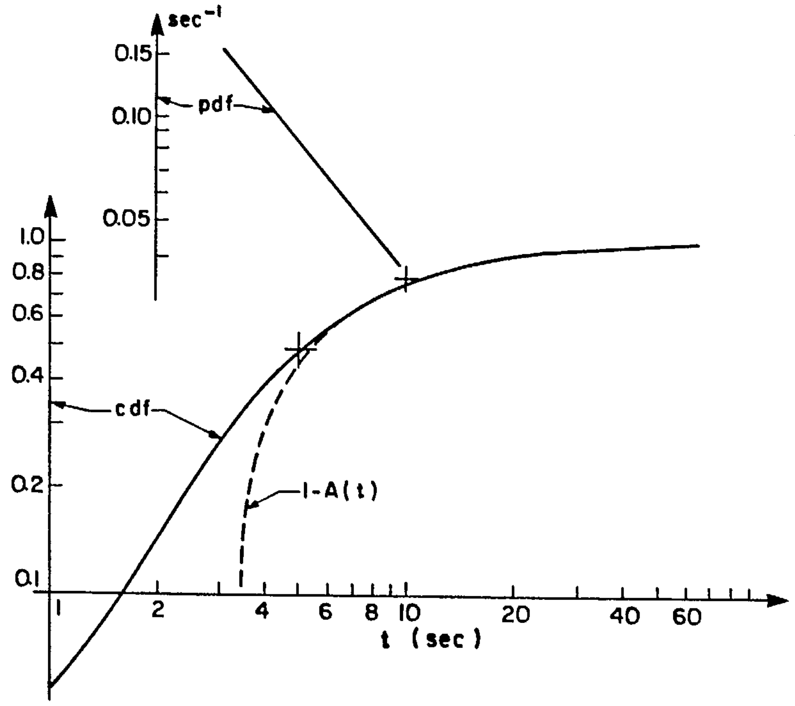

Figure 3.2: Pause time distribution.

Ccdf is exponential, then the probability density function is also exponential. From Figure 3.2, the probability density function for times over 3 seconds is approximately exponential, at least until 10 seconds after which there are few data with considerable scatter. Hence the anticipated use of a page in cache decreases exponentially after about 3 seconds.

There will be competition for space in cache, and the items that are to be discarded are those with the least anticipation of use. These will be the ones with the least values for the parameter $A(t)$. Two kinds of errors can be made: one is when an item is kept and never used which occurs with probability $(1-A)$, and the second kind is when an item is discarded and then called for which occurs with probability A. We are treating each item in a multiple prediction as independent, when in fact if one item is called the others could be discarded. This latter procedure could become quite complex since it would be necessary to retain and review the sources of each prediction.

Figure 3.2 suggests that each page in cache be retained for about 3.2 seconds, after which the value of the parameter $A(t)$ be decreased exponentially until it is discarded. If this is done, then for a given discard threshold level for the parameter A, the fraction discarded will be $A(threshold)$, and the fraction retained is:

$$1 - A(threshold)$$

which bounds the cumulative distribution function from below as shown in Figure 3.2. Since the cumulative distribution function is the proportion of predicted items in cache which have been used by a certain time, the probability of holding onto a page in cache is greater than the probability that it will be used, but approaches the latter for pause times greater than 5 seconds.

It is proposed to implement the exponential decay of the anticipation parameter by periodically multiplying all values with a factor of

$$\frac{7}{8} = (1 - \frac{1}{8}) = \exp(-1/7.5).$$

This can be done in binary arithmetic by a shift of 3 positions and a subtraction. To obtain the time constant observed in the data of Figure 3.2 these decay multiplications should be done every 0.8 seconds. If the implementation of the exponential decay is delayed for four such periods, then the policy described above of holding pages for 3.2 seconds will also be implemented.

The procedure for processing the cache directory every 0.8 seconds is outlined in Figure 3.3. A 3 bit register is used to count four intervals or 3.2 seconds, after which the register containing the anticipation parameter, A, is reduced exponentially. Step 5 in Figure 3.3 allows for the case of several users where predicted requests may overlap items already in cache. Finally, in the case of a light load on the system it is desirable to refresh the cache from the set of default pages rather than retain unused and very stale predictions. Hence step 6 in Figure 3.3.

Under the above decay scheme for the anticipation parameter, the actual length of time for which this parameter remains non-zero will depend upon the number of bits in the A register. The number of processing intervals before the value of A is less than one least significant digit and therefore truncates to zero, is given approximately by

$$steps = \frac{\log(2^b - 1)}{\log(8/7)} = 5.2\, b \qquad (3.3.1)$$

where b is the number of bits in the register storing A. Using 30 seconds which includes 97% of the pause intervals, and an interval processing time of 0.8 seconds, we get the following lower bound on the number of register bits:

$$b \geq \frac{30}{(0.8)(5.2)} = 7.2 \text{ bits} \qquad (3.3.2)$$

We may conclude, therefore, that an 8 bit register should be adequate for this purpose.

Under heavy load conditions, competition for space in the cache can be based upon the values of the anticipation parameter. More space can be made available by increasing a threshold for discarding items. On the other hand, under light load conditions stale predictions in terms of items with zero value for A may remain for a long time in the cache.

It is proposed at each processing interval to sweep out those items for which the anticipation parameter has already reached zero. This will be done by replacing such items by pages from the zero-order default list which are not already in the cache. Should a default list page already be in cache, its A value will be updated using the value A_0 from the default list. Such a procedure also takes care of initialization at a time when the cache is empty.

1 When an item is brought into cache in anticipation of a future request, the cache directory (CD) will include two registers: A (minimum, 8 bits) and B (3 bits; B_3, B_2, B_1). These are initiated with:

A = (Conditional Probability from the Prediction Table)

$B = 0$

2 Every item in the cache directory is to be processed every 0.8 seconds with the following algorithm:

```
IF ($A=0$) THEN STOP
ELSE IF ($B_3=0$) THEN
            $B' = B + 1$
     ELSE
            $A' = A - A/8$
```

3 To bring a new item into cache, enter its I.D. etc. in the cache directory buffer. Then remove the item with minimum value for A (including the new item), and transfer the new item into the vacant space in cache.

4 When a request is received, whether or not the item is already in cache, the prediction table is consulted and either disk requests issued for anticipated items which are brought into cache (see 1 and 3), or the A value is adjusted according to (5) if the item is already in cache.

5 If a prediction requires an item to be brought into cache with "anticipation" A2, and it is already there with anticipation number A1, then adjust A1 to:

$A1' = (A1 + A2 - A1 \cdot A2)$

6 If at a processing interval (see step 2) one or more items have values $A = 0$ then the list of default pages is to be consulted. Starting with the highest value of A_0, if the item is already in cache then employ step 5, otherwise enter the item in place of one of those for which $A = 0$.

Figure 3.3: Procedure for cache directory registers.

3.4 Distribution of Page Sequences

It was seen in Section 3.2 that as little as 18% of the pages could account for 84% of the use. Now we wish to extend our model one step by following the user and analyzing his choice conditioned upon the fact that the page he currently has displayed is known. We suppose that once a user is in the system, he will generally follow a chain of selections with a very small number of branches at each selection event relative to the complete data base available. Indeed, if a table of the possible links of successive pairs of pages actually chosen were available, it could be used to predict the next request in a particular situation. In this way the next request could be anticipated and brought into cache in preparation for delivery. If the number of possible links or branches from a given page is small, this procedure could greatly enhance the performance with a relatively modest cost in storage. The practicality of exploiting such predicted characteristics requires a knowledge of the joint behaviour of successive pairs of requests, each pair consisting of an "antecedent" and a "consequent". Such an analysis has been carried out on data set number 210 which contains the largest sample size available to us. This record is sufficiently long that it can be used to estimate the average error rate for a prediction procedure.

One might suppose that each page could be the antecedent, and have every page possible as a consequent. Then there would be V^2 possible links or branches if there are V pages in the system. However, the actual number of consequents per page averages only a very few, and the first thing which can be deduced from our observations is the amount of choice or the number of branches from each antecedent. Let the index a enumerate the antecedents; then because all pages are possible antecedents and we have

$$a = 1,2,3,\ldots\ldots V$$

Next, for each a, let the index c enumerate the consequents; therefore,

$$c = 1,2,3,\ldots\ldots V_a$$

This indicates that page a as an antecedent has V_a possible consequents. Knowledge of the set of V_a's over the V pages is important in defining the size of the problem and the amount of storage a prediction strategy would require.

We regard each link or successive pair of requests that occur as an event, and from the given data set we can count the number of times such distinct events occur. Let $N(a,c)$ be the number of times a particular antecedent has a particular consequent; that is, the pair (a,c) has occurred $N(a,c)$ times. Note that the range of the index c as well as the actual identity of the pages designated by it, are dependent upon the antecedent page. The total of such events is:

$$\sum_{a=1}^{V} \sum_{c=1}^{V_a} N(a,c) = N_T \tag{3.4.1}$$

The actual number of pair events that occur in the data (of the V^2 possible) is given by:

$$\sum_{a=1}^{V} V_a = V_T \tag{3.4.2}$$

Then the average number of branches at each user selection is given by:

$$\text{avg}(\,V_a\,) = V_T\,/\,V \tag{3.4.3}$$

Next we take the V_T events and determine their distribution in order to give a more detailed description of user behaviour. To do this we can employ the same analytic model as used in Section 3.2, in which the cumulative distribution function is a power of the reciprocal rank where the events have been ranked in descending order of their frequency of occurrence. From eqn. (3.2.4) we have:

$$S(r) = \frac{S_1\, r}{r^{1/B}} = e^b\, r^m \tag{3.4.4}$$

Again, an "effective" number of events is obtained by extrapolating this linear relationship until

$$S(v) = 1$$

so that

$$v = e^{-b/m} \qquad (3.4.5)$$

Using data set 210 which was the largest one available for analysis, it was found that with a data base of 1986 pages (see Table 3.1), the number of distinct successive pairs of requests was 3682. This is less than 0.1% of the possible number of pairs, and the cumulative distribution function for the frequency of occurrence of the different pairs is shown in Figure 3.4. The parameters of this distribution are tabulated in Table 3.2.

From Figure 3.4 it is seen that only 114 or 3.1% of all the distinct pairs actually observed (namely 3682) account for one-half of the total number of joint events observed. Also, this small group which accounts for half of the activity obeys the reciprocal power law extremely well. Of all the events observed, the most likely pair occurred 2% of the time, while the 114th had a probability that was already down to 0.3%. Hence 97% of the distinct pairs observed had probabilities less than this, although they accounted for about half the activity. The picture that emerges from Figure 3.4 regarding the pairs of successive requests, is of two types each accounting for about half the activity each. The one type consists of a relatively small number of links which occur often, while the other type consists of a very large fraction of the distinct pairs observed, but which happen relatively rarely as distinct events.

The number of consequents that occur for each antecedent will determine the number of lines required in a prediction table and hence the total size of such a table. Using the data in Table 3.2 and eqn. (3.4.3) we find that:

$$\text{avg.}(V_a) = 1.9$$

It was observed that very nearly 2/3 of the pages (as antecedents), had only a single consequent. The other 1/3 had 2 or more consequents with an average of 3.7. It was interesting to note that to a very close degree of approximation the

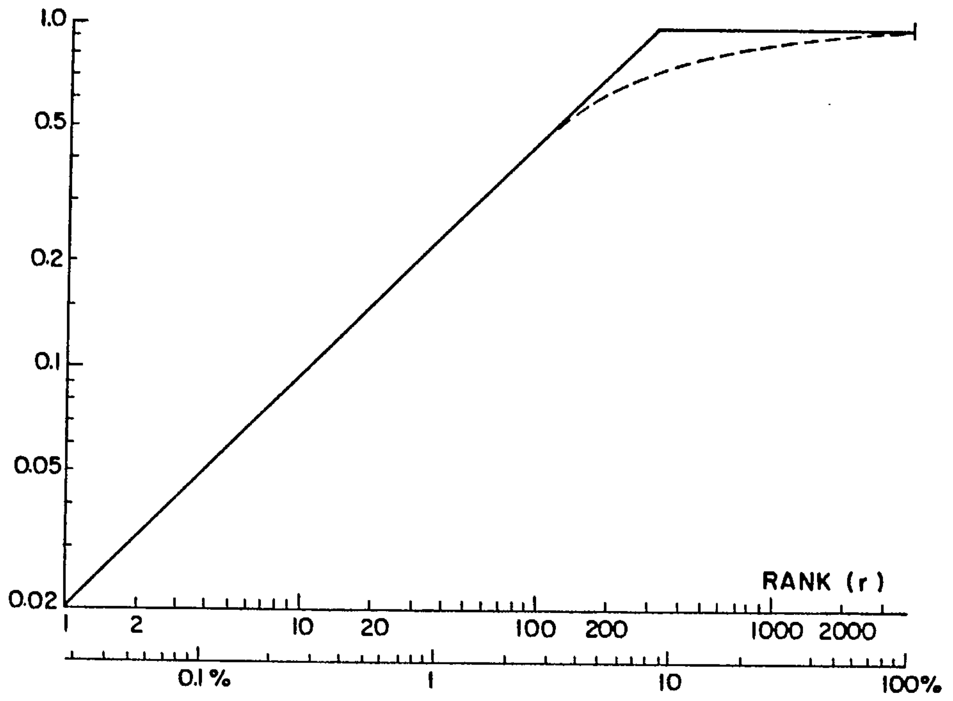

Figure 3.4: Cumulative distribution of sequential page requests.

Table 3.2

Parameters of the Page-Pair Joint Cdf

Data Set I.D.		210
Sample Size	N_T	70 463
Possible page-pairs	V^2	3 944 196
Observed page-pairs	V_T	3682
Cdf. slope	m	0.684
"Temperature"	B	3.16
Effective number of pairs	v	317
Proportion effective	v / V_T	8.6%
Maximum frequency	$S(1)$	2%
Error at asymptote v		27%

number of antecedents which had exactly V_a consequents varied as:

$$\frac{1}{V_a^{\ 2}} \qquad (3.4.6)$$

As a check we see that:

$$\frac{2}{3}\sum_{1}^{10} V_a^{\ -2} = 1.03$$

Figure 3.4 can be used to relate the size of the stored table required for prediction purposes to the errors resulting from the inability to make predictions because of omissions from the table. Thus we see that if the number of lines in the prediction table is equal to 114, or about 6% of the number of pages in the data base (V) then the error rate, or the inability to make predictions, will be approximately 50%. If we increase the size of the prediction table to 317 which is the asymptotic value from Table 3.2 then the error rate will be about 27%. Such a prediction table is about 16% of the number of pages in the data base. Figure 3.4 would suggest that if the number of lines in the prediction table were equal to the number of pages in the data base, then predictions would be possible in 95% of the cases required. Beyond that it becomes increasingly expensive of storage to reduce the error rate further.

Before drawing any further conclusions, we note that the picture changes somewhat if we look closely at the sample size. It has already been observed that one group of events (or sequential pairs) have a frequency distribution that obeys the reciprocal power law, while the other group of events are individually rather rare although they form a large group. The quantity $N(a,c)$ was the count of the number of times a particular pair, (a,c) occurred. However, in many cases we have

$$N(a,c) = 1$$

which corresponds to a probability of 0.0014%. From Figure 3.4 we see that the most likely event (or joint sequential pair) had a probability of about 2% so that the range is almost 2000:1.

Using the Poisson distribution as a model for the occurrence of rare events and a confidence limit of 90%, we may say that the observance of a single event implies an expected probability in the range

$$0.00015\% \text{ to } 0.0055\%$$

while for a particular event which did not occur at all the same range would be

$$0\% \text{ to } 0.0033\%$$

For this reason, sequential pairs which only occurred once in the observed data could be neglected from the prediction table. If this is done, the number of observed events tabulated in Table 3.2 reduces to

$$V_T = 2334$$

However, the lost events account for only 2% of the total activity. It is also interesting that now 23% of the pages do not appear as antecedents at all.

We now consider the subset of the data which includes all events which occurred two or more times. Of this subset, we find that 80% of the antecedents have a single follower while the other 20% have, on the average, 3.6 consequents. The average number of branching links per antecedent is thus 1.52. However, this applies to only 77% of the pages in the data set, so that the number of lines in the prediction table is approximately 1.2 times the number of pages in the data set.

3.5 Cache Size

The purpose in predicting user selections is to have the likely selections available in a cache memory for quick response. To judge the cost effectiveness of such a system, it is necessary to estimate the size of cache required. We have already seen that the average number of pages predicted (consequents) is quite small. We can identify three main factors that will influence the size of cache required, and then offer some comments on them. These factors are:

(a) As the number of users in a system increases, the probability of new selections, different from those already chosen, decreases. This means that the required size of cache increases more slowly than the number of users, since additional users in a crowded system are likely to repeat requests already made. Moreover, while the number of different requests that are likely to occur with a given number of users is of interest, it will not directly add to the cache size estimate since one expects in a functioning prediction system that each user request will be in cache, and not require a new fetch from disk and an additional page in cache.

(b) Once we have identified the number of different requests or antecedents expected with a set of users, then the cache size required is that to hold the consequents predicted from these. We will assume that these consequents are all different in identity. However, we can bound the total required by taking into account the distribution of the number of consequents per independent antecedent given in eqn. (3.4.6).

(c) If a number of consequents are brought in cache because of their expected request by a user, and only one (or perhaps rarely, none) is actually used, and the others are treated according to Figure 3.3, they may linger in the cache for some time, depending upon the pause time distribution of Figure 3.2, and the threshold for the parameter A. This could cause much unnecessary occupancy of the cache.

First of all, we deal with (a); namely, the increase in new (i.e. different) requests with an increase in the number of users. Supposing the Nth user chooses the page of rank r; then the probability that none of the $(N-1)$ other users requested that page is:

$$P(r;N) = [1 - f(r)]^{N-1} \tag{3.5.1}$$

where $f(r)$ is given by eqn. (3.2.1). Averaged over the set available, the probability that the Nth user requests a different page is:

$$P(N) = \sum_{r=1}^{v} f(r)[1 - f(r)]^{N-1} \tag{3.5.2}$$

This is shown in Figure 3.5 using the parameters of data set 210 (Table 3.1) that describe the asymptotic distribution (Figure 3.1) in eqn. (3.2.1) for $f(r)$. The graph is approximated by:

$$P(N) \approx 0.72e^{-N/463} + 0.29e^{-N/78} \leq e^{-\frac{N-1}{400}} \tag{3.5.3}$$

Next, we attempt to use $P(N)$ to estimate the number of distinct antecedents, A, that will occur with U users. Each additional user adds one antecedent if his request is different from the $(N-1)$ others, and none if it is the same. On the average, the Nth user adds $P(N)$ antecedents. Hence

$$\bar{A} = E[A \mid U] = \sum_{N=1}^{U} P(N) \tag{3.5.4}$$

In Figure 3.5, the solid line shows the compression of $\bar{A} / U$ that results from overlapping requests as the number of users increases.

Let us define the probability that U users will request exactly A distinct pages, as $Q(A \mid U)$. Then we have:

$$Q(U \mid U) = \prod_{N=1}^{U} P(N)$$

and

$$Q(1 \mid U) = \prod_{N=2}^{U} [1 - P(N)]$$

and an iterative formula can be developed; viz.,

$$Q(A \mid U) = P(U)Q(A-1 \mid U-1) + [1 - P(U)]Q(A \mid U-1)$$

which is facilitated by using the exponential bound of eqn. (3.5.3). However, rather than pursue that line, we wish to indicate how a confidence bound can be

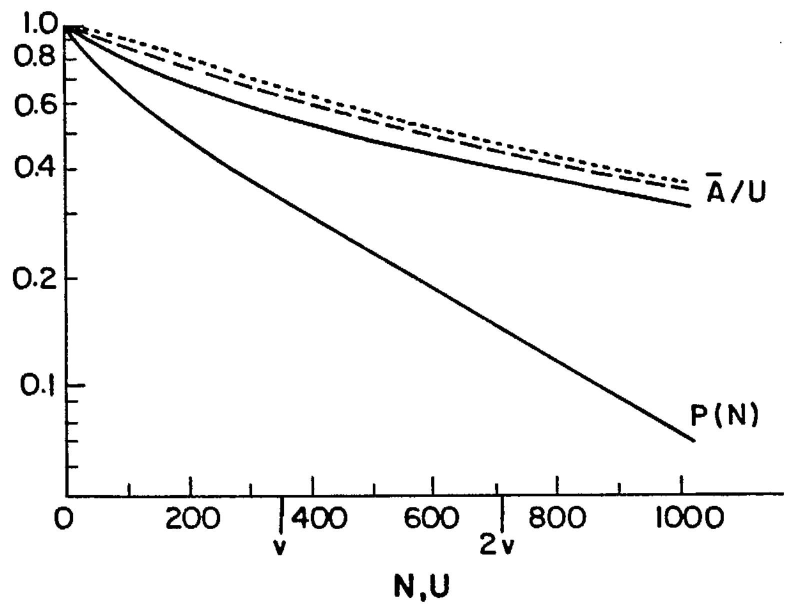

Figure 3.5: Average probability of the Nth user selecting a different page, $P(N)$, and number of distinct requests per user, A_α / U (v =357, α=5%).

derived. Supposing we wish to assert that allowing for A distinct antecedents will be adequate $(1-\alpha)$ of the time; that is, the probability of requiring more than A_α is α. Then, this limit is obtained from

$$\alpha = \sum_{i=0}^{U-1-A_\alpha} Q(U-i \mid U) \tag{3.5.5}$$

Another approach is to use results from the statistics of cell occupancy problems. Entropy considerations suggest that the complete skew distribution behaves like that of v equally likely alternatives. Then the probability that U users will request A distinct pages from v equally probable alternatives is asymptotic to

$$Q(A \mid U) \approx \frac{e^{-\lambda}\lambda^{v-A}}{(v-A)!} \tag{3.5.6}$$

where the expected value of $(v - A)$ is:

$$\lambda = ve^{-U/v} = v - \bar{A} \tag{3.5.7}$$

In Figure 3.5, the broken line shows the compression of $(\bar{A}/U)$ under these assumptions. Of course, eqn. (3.5.6) is the Poisson distribution which may be approximated by the Gaussian for large expected values. An equation for confidence limits analogous to eqn. (3.5.5) then results; namely,

$$\alpha = \sum_{A=A_\alpha}^{\min(U,v)} e^{-\lambda} \lambda^{v-A} / (v-A)!$$

$$\approx \frac{1}{2}\operatorname{erfc}\left\{\frac{\lambda - v + A_\alpha}{\sqrt{2\lambda}}\right\} - \frac{1}{2}\operatorname{erfc}\left\{\frac{\lambda - v + \min(U,v)}{\sqrt{2\lambda}}\right\} \qquad (3.5.8)$$

For example, the probability that the number of different antecedents exceeds A_α, is α, where α and A_α are solutions of eqns. (3.5.7) and (3.5.8) for given values of v, and U. Such a confidence line for $\alpha = 5\%$ is shown dotted in Figure 3.5 for the same parameters and assumptions as the broken line, for illustrative purposes.

As noted above under (b), given the number of different antecedents, A, then the cache required is that needed for the consequents of these. From Section 3.4, the *average* requirement would be $1.52A$. However, the requirement for individual users follows the distribution given by eqn. (3.4.6) which is very skew. For example, a 4 page cache would be adequate for one user 95% of the time, but 70% of the time, 3 pages of the 4 would be empty. Again 6 pages per user would be adequate over 98% of the time, but a cache of $6A$ size would be highly redundant. In the case of two independent users, if instead of providing 12 (6 each) pages only 6 pages are provided, then they are jointly covered 94% of the time. The point is, the requirements are not simply additive because of the very skewed distribution, in which a requirement of more than one is unlikely.

To estimate the extent of this "compression" effect we note that the cmf (cumulative mass function) for the distribution of eqn. (3.4.6) is lower bounded by:

$$1 - e^{-x/1.8}$$

Using this for a cumulative distribution function of a continuous density, we obtain an associated probability density function:

$$f(x) = x_0 e^{-x_0 x} ; x_0 = 1/1.8 \qquad (3.5.9)$$

Using this continuous approximation to the lower bound, we can now set up the integral expression for the probability of having enough cache $(1-\alpha).100\%$ of the time, if the cache is of size Λ pages and there are A different antecedents. This is:

$$1-\alpha = \int_0^{\Lambda} f(x_1)dx_1 \int_0^{\Lambda-x_1} f(x_2)\,dx_2 \;\ldots \int_0^{\Lambda-x_1 \ldots -x_{A-1}} f(x_A)dx_A \qquad (3.5.10)$$

where $f(x_i)$ in eqn. (3.5.10) is given by eqn. (3.5.9). The integration in eqn. (3.5.10) yields:

$$\alpha = e^{-x_0 \Lambda} \left\{ \sum_{k=0}^{A-1} \frac{x_0{}^k \Lambda^k}{k!} \right\} \qquad (3.5.11)$$

Eqn. (3.5.11) provides the solution for the cache size, Λ, given the confidence level, α, and the number of different user requests, A. Let the solution be C_α; that is, the space required for consequents with probability $(1-\alpha)$.

If we take 6 pages as the maximum requirement per user, then

$$C_0 = 6A$$

The behaviour of the ratios C_5 / A and C_{50} / A as functions of A are shown in Figure 3.6. This results from solving eqn. (3.5.11) using the bound of eqn. (3.5.9), and therefore is an upper bound on C_α. It would indicate that with a large number of users, a cache size of twice the number of distinct requests would be adequate.

The third factor noted above as (c) is more difficult to analyze without specific data. When several predicted consequents are brought into cache, in general we can expect one to be required and the others not. Using an algorithm such as that in Figure 3.3, the others will eventually be discarded when their A-values drop below the threshold. There would seem to be three

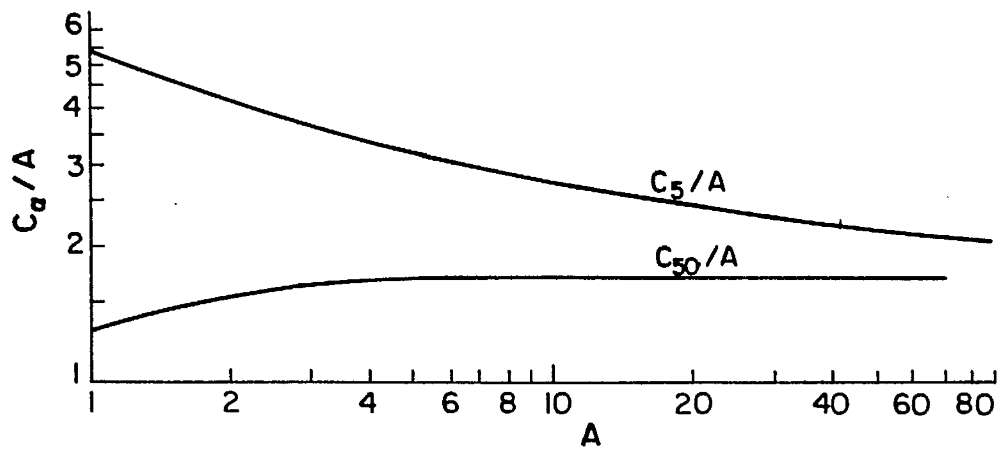

Figure 3.6: Upper bound on cache requirements per request, C_α / A (α=5%).

approaches to this problem:

(i) The unused consequents can be left to be discarded by the threshold method. This could congest the cache unnecessarily. To the extent that the requests overlap with a large number of users, it may actually save disk transfers.

(ii) Each predicted page brought into cache could be tagged with the user's identity; then, when that user made a request, all items tagged with his identity only, would be discarded in favour of new predictions. With large numbers of users, multiple tags can occur and the item would be discarded when the last one is cleared. For a large number of users, say

$$U > 1.5v$$

then Figure 3.5 indicates that only about half the requests will be distinct; i.e.,

$$A / U \approx 1/2 \qquad (3.5.12)$$

However Figure 3.6 shows that for large A,

$$C/A \approx 2; \qquad (3.5.13)$$

hence a cache size approximately equal to the number of users is indicated, if the unused predictions are systematically cleared out.

(iii) The cache could be decentralized to each user terminal where both the current and predicted requests would be held. This would certainly facilitate the clearing out since tagging would be unnecessary. Collectively, the system would have to provide for 6 predicted pages per user, or six times the cache in (ii). This is because no advantage can be taken of duplicate requests by different users, nor of the compression in storage requirements that result from the very skewed distribution of the number of consequents (eqn. (3.5.10)). It also would require about 1.5 as much communication capacity on the average as centralized systems because of the transmission of predicted but unused material. (The average 1.8 from eqn. (3.5.9) is an upper bound). Nevertheless, the decentralized cache approach may have advantages from a system viewpoint. The organization at the central supplier is simpler and less congested; it must retain a prediction table (Section 3.4) and be advised of user requests even if they are already in the local cache.

3.6 Summary

The analysis of the data set on the basis of a predictive model offers some guidelines as to the expected behaviour of a prefetching method. It suggests that a conditional prediction table can be constructed by neglecting joint events with a probability of less than 0.002% and keeping those with a probability equal or greater to 0.003%. The number of lines that are required in the prediction table itself is not at all excessive, being of the same order as the number of pages in the data set. This results from the relatively few number of multiple branch points evidenced by the user data. We can also expect the prediction to be effective about 95% of the time or better.

The prediction table will include the anticipation number, A, along with each predicted subsequent page. This anticipation number is a conditional probability calculated from past observations and is given by

$$A(c\,;a) = \frac{N(a,c)}{\sum_c N(a,c)} \leq 1 \qquad (3.6.1)$$

As noted above, in 66% to 80% of the cases we will have

$$A = 1$$

because only one consequent will be recorded. The number of bits required to record A can be calculated from the minimum value for the numerator, and the maximum value for the denominator. Using data set 210, and keeping events with two or more occurrences, we have:

$$\min N(a,c) = 0.003\%$$

From Table 3.1 we have:

$$\max \sum_c N(a,c) = 6.4\%$$

Therefore, $\min(A) = 0.0004$. This implies that the minimum size of the words for recording anticipation numbers should be 12 bits.

We have seen that the number of predicted pages for a given user at any point in his chain of selections will average about 1.5 to 1.9.

Furthermore, if the cache is distributed, then the communications requirements would increase by the same factor (1.5). However, for a centralized cache and a large number of users, the number of different requests does not increase linearly with the number of users because of the increased chances of users making the same request. Such chances are much greater than might otherwise be expected because the effective number of pages is much smaller than the total available (1/4 or 1/3 in the data reported here), and these are used with a skewed frequency distribution. From an example based on data, when the number of users exceeds 1.5 times the effective number of pages, then the number of distinct requests is only half the number of users (Figure 3.5). The first order sequential chain model for a user searching a Telidon tree also results in a highly skewed distribution for the number of anticipated consequent pages. As a result, while one user may occasionally require 6 pages, the number required per user, when there are many, is much less than this. Figure 3.6 shows that two pages per distinct user request are enough when there are several hundred users. As a result of these two factors we can conclude that the size of a central cache is approximately that of the number of users in the system.

It is not known how much the structure of the data base used here for analysis, influenced the numerical conclusions. It seems highly probable that the skew distribution will describe the user's behaviour regardless of the organization of the material; however, the extent to which the interaction of the data base structure and the predictability of choices will affect the numerical values of the parameters in the statistical description is not known. Data set 210 was assembled from a number of sources and it is possible to see different patterns in the tree structure depending upon either the originator or the subject matter of the source. It would be valuable therefore to acquire a data base from other sources and with a different type of material.

4. Alternative Telidon Architectures

4.1 Introduction

In Section 2 and 3 two aspects of a Telidon system were discussed; first the aspects related to the performance of the communications and computer systems, and then the aspects related to how the user interacts with the data base. In this section the results from these two sections will be combined to evaluate the feasibility of the various schemes that were proposed in Sections 2 and 3.

In Section 2 it was shown that the use of a CATV network for distributing Telidon pages was the only feasible solution for a large Telidon system. Because of the low data rate required the choice of network for collecting the user requests is more flexible, and could be a shared CATV channel when technically and economically feasible, or could be dedicated lines, telephone, digital switched circuit, or a packet switched network. The choice has little impact on the Telidon server architecture. Also in Section 3 it was shown that the computing load on the Telidon server is low, and the processing time does not place any constraints on the Telidon server architecture. Hence the significant factors to be decided are the tradeoffs between the number of disks, page fetching policy, amount of cache memory and its location, and the organization of the pages on the disks.

In Section 2 three basic approaches were discussed. Based on the results of Section 3, two of these have variants based on whether or not prediction and prefetching of pages is employed. This gives a total of of five possible architectural alternatives. These are:

(i) a multidisk system without cache memory

(ii) a system with cache memory

(a) passive cache, i.e., no prediction

(b) a cache with successor page prediction

(iii) disk organization with disk access scheduling

(a) no cache memory

(b) distributed cache

Each of these alternatives will be discussed in the following sections. For purposes of comparison we will assume, as before, 1000 active users with a total request rate of 100 request/s, and disk(s) with sequential transfer rates of 1000 page/s and random access times of 40 ms. Where needed, these assumptions will be supplemented with the user characteristics developed in Section 3.

4.2 Multidisk System Without Cache Memory

The multidisk system without a cache memory is in many ways the simplest system. The only user characteristic required is the total request rate. The performance can be estimated from eqn. (2.4.2.2) (with $f=0$). For five disks the mean time to process a request is in the order of 200 ms which is probably acceptable, although a simulation would be required to find the actual response time distribution. The disadvantage of this system is, of course, the large number of disks required.

4.3 Passive Cache Memory

The systems with cache memory attempt to reduce the number of disks required by serving a large number of requests from the cache. We will consider first a system with a passive cache, i.e., there is no attempt made to predict future requests and the cache is updated only when a user request is not found in the cache. The performance of such a system is described by eqns. (2.4.2.2) and (2.4.2.3). In addition to the user request rate it is necessary to know the relationship between the size of the cache and the cache hit ratio f. One would expect that the pages in the cache tend towards the most frequently accessed pages. Hence, based on the information in Table 3.1, one would expect hit ratios in the range of 85 to 90% for systems with small data bases if the cache size was approximately the same as the number of effective pages v. Using $f = 0.85$ with one disk in eqn. (2.4.2.2) yields a value of approximately 100 ms as the mean response time for the 15% of the requests not served from the cache. As before this is probably satisfactory, although a simulation would be required to find the actual response time distribution. However there is a significant problem in extending this performance evaluation to large data bases, i.e., 50 000 pages. From Table 3.1 we note that tripling the size of the data base appears to increase the number of effective pages by about 1/3, and reduce the hit ratio by 5%. If these ratios hold as the data base size increases we would expect a required cache in the order of 850 pages with a hit ratio of 70% for a 50 000 page data base. The size of the cache would not be a problem, since a 1000 page cache would require only 1 Mbyte of memory, which is quite feasible. However, from eqn. (2.4.2.5) at least two disks would be required. This is of course speculative; it could be better or it could be worse. Until information on large data bases is available we are unable to predict the performance of this approach for large data base systems. For small data base systems, since only one disk is required with only a modest cache size, the passive cache approach is superior to the multidisk approach.

4.4 Cache Memory With Page Prefetching

The analysis of Sections 3.4 and 3.5 is relevant to a system with a centralized cache in which successor pages are predicted and fetched in anticipation of a user request. It was shown that to have a 95% probability of finding the page in the cache the cache size should be about 2 pages per user for a large number of users when the users share the cache memory. However when the number of users is greater than the effective number of pages in the data base, we saw that about half the requests will be duplicates. As a result of these two factors we can conclude that the size of a central cache should be approximately equal to the number of active users in the system. Hence, a system with 1000 active users would require a 1000 page cache (1 Mbyte of memory). For such a system approximately 95% of the requests would be served from the cache. Hence, if the average total request rate was 100 request/s about 5 request/s would require disk service. From eqn. (2.4.2.2), and the typical disk parameters given earlier, the mean response time for these direct disk requests would be about 50 ms for a single disk system assuming they were given priority over the prefetching of anticipated pages. This response time is satisfactory, but there is still the question of the disk access capacity required to handle the prefetching. With 100 user requests per second, and an average of 1.9 successors per request, and approximately 0.5 of the requests distinct, one would expect an average of 95 prefetch requests to be generated per second. For a large cache memory most of these would be satisfied from the existing cache contents, and only a small fraction would cause disk activity. For data set 210 we note from Figure 3.1 that if the cache contained the 1000 most frequently used pages, then only about 5% of the prefetch requests would not be resolved in the cache. Hence for this data set, and a 1000 page cache, one would expect about 5 pages per second being prefetched from the disk. This level of prefetch activity would not cause any disk performance problems, and one disk would be sufficient. If due to a larger data base, or a higher request rate, the prefetching traffic approached 20 pages per second it would be necessary to supply a second disk unit. We note, however, that it might be possible to delay the addition of the second disk by appropriate organization of the data on the first disk. This would involve placing the successor pages to a given antecedent sequentially on the disk. If necessary some pages could be replicated. This entire group could then be fetched in a time not significantly greater than fetching a single page. This would reduce the effective disk traffic.

To investigate further the relationship between the number of users, cache size, and the disk access time a simulation study was performed. The main objective of the study was to check the relationship between the number of users and the cache size required for a 95% cache hit ratio. The simulated user page requests were based on the page request distribution, and the antecedent-consequent relationships, of data set 210. To give seek times that would be realistic for larger data bases the 1986 pages of the data set were assumed to be randomly distributed over the disk. The simulation considered only a single disk system. Two queuing disciplines were studied: a simple first in, first out (FIFO) discipline, and a scan discipline.

In a FIFO discipline pages are read from the disk in the same order that the requests are received, i.e., the cache misses are not given priority over the prefetching requests. In the scan discipline the disk access arm is swept across the disk, always from the cylinder last involved in data transfer to the closest cylinder, in the current direction of motion, for which a data transfer request exists. When no further data requests exist in the direction of motion, the direction of motion is reversed. The scan discipline performs significantly better than FIFO for heavy disk utilization [6-9]. For the simulation using the FIFO discipline a mean disk access time of 50 ms was used. For the scan discipline simulation a slightly faster disk was assumed, with a mean access time that varied from 39 ms at light load to 20 ms at heavy load.

The user page requests in the simulation had a generating function whose interarrival time approximated that shown in Figure 3.2 with a mean interarrival time of 3.25 s. The simulation results are shown in Tables 4.1 and 4.2. The values given are the number of users, the cache size, the total number of requests simulated, the percentage of requests satisfied from the cache (hit ratio), the percentage of requests that had to be queued for disk service, the mean disk service time for these latter requests, and the overall mean response time. Where the disk and cache requests do not sum to 100%, the difference represents items already scheduled for disk access when the request arrived.

The simulation results confirm the calculations in eqns. (3.5.12) and (3.5.13) which indicated that for a large number of users one cache location per user would be sufficient for a 95% cache hit ratio. In fact, this estimate is conservative since the actual number of cache locations required to yield a 95% hit ratio for a specified number of users, as found by simulation, was about 15% less than the product of $C/A=2$ and the value of A/U given in Figure 3.5 for $U>1.5v$. It is further observed that the cache size does not change significantly between the two queuing disciplines even though the the scan discipline reduces the disk access time by 50% under heavy load conditions.

It will be noted in Tables 4.1 and 4.2 that the mean disk service time remains relatively constant over a wide range of number of users. This is, of course, due to the page prefetch traffic. This traffic was not recorded during the simulation, but it may be estimated. For the FIFO discipline simulation the mean disk access time $\bar{x} \approx 50$ ms and $C_b^2 \approx 0$. If we assume that the total disk traffic is Poisson then eqn. (2.4.2.2), with $f = 0$, applies and we can use the measured mean disk service time from the simulation to calculate the disk utilization and disk access request rate. The results are shown in Table 4.3. The

Table 4.1

Cache Size Simulation Results (FIFO Disk Scheduling)

Active Users	Cache Size	Total Requests	In Cache	On Disk	Service Time	Overall Response
150	260	40003	95.1%	4.8%	123.3ms	5.9ms
200	300	40000	94.8%	5.1%	134.1ms	6.8ms
200	310	40002	95.4%	4.4%	124.7ms	5.5ms
250	340	20001	94.8%	5.1%	131.1ms	6.7ms
250	345	20004	95.4%	4.4%	120.1ms	5.3ms
300	360	40004	95.1%	4.8%	135.4ms	6.4ms
300	370	40003	95.1%	4.8%	133.0ms	6.4ms
400	420	40004	95.0%	4.9%	139.6ms	6.8ms
500	450	40012	93.8%	6.0%	193.4ms	11.7ms
500	460	40001	95.6%	4.3%	149.2ms	6.4ms
600	475	40006	94.9%	5.0%	226.8ms	11.3ms
600	485	40001	95.7%	4.2%	167.1ms	7.0ms
700	525	40004	95.1%	4.9%	232.9ms	11.3ms
800	550	20007	95.0%	4.9%	362.2ms	17.7ms
900	575	20009	94.7%	5.2%	531.6ms	27.6ms
900	600	20013	96.1%	3.9%	409.4ms	15.9ms

prefetch activity is expressed as a percentage of the user request rate. As a check we note that for 300 users the cache size of 360 is approximately the effective number of pages in the data set. Hence we would expect about 84% of the prefetch requests to be resolved in the cache. Hence the number of prefetch requests passed to the disk would be $1.9 \times 0.5 \times 16\% \approx 15\%$ of the user requests. This compares very well with the value of 12% in Table 4.3. Hence we conclude that the prefetch concept functions well over a wide range of loads. The limit on the number of users appears to be primarily a function of the cache miss traffic approaching the disk throughput limit.

With the page prefetching disk accesses transparent to the user, as they will be since the disk is not overloaded for a small data bases similar to data set 210, the system becomes equivalent to the system with a passive cache from the user viewpoint. However the performance will be better since the user will see a

Table 4.2

Cache Size Simulation Results (Scan Disk Scheduling)

Active Users	Cache Size	Total Requests	In Cache	On Disk	Service Time	Overall Response
150	240	20004	94.9%	5.0%	100.8ms	5.0ms
200	275	20001	95.0%	4.9%	94.5ms	4.6ms
250	300	20001	95.1%	4.8%	102.3ms	4.9ms
250	310	20003	95.1%	4.8%	95.3ms	4.6ms
300	335	20003	94.5%	5.4%	103.7ms	5.6ms
300	340	20001	96.0%	3.9%	91.3ms	3.6ms
400	370	20000	94.8%	5.1%	99.9ms	5.1ms
400	375	20000	95.4%	4.5%	98.5ms	4.5ms
500	400	20004	94.6%	5.3%	104.4ms	5.5ms
500	420	20002	95.9%	4.0%	93.3ms	3.8ms
550	420	20002	94.5%	5.4%	124.1ms	6.7ms
550	430	20003	95.4%	4.5%	108.3ms	4.9ms
600	440	20000	95.1%	4.8%	104.2ms	5.0ms
650	450	20002	95.0%	4.9%	106.2ms	5.2ms
700	440	20009	93.5%	6.3%	160.3ms	10.1ms
700	445	20000	95.7%	4.3%	102.5ms	4.4ms
750	450	20000	94.3%	5.6%	115.9ms	6.4ms
750	465	20011	94.7%	5.2%	121.6ms	6.4ms
800	485	20001	94.5%	5.4%	130.2ms	7.0ms
800	490	20005	95.5%	4.4%	105.1ms	4.6ms
900	510	20000	94.0%	5.9%	221.9ms	13.1ms
900	520	20004	95.6%	4.3%	118.2ms	5.1ms
1000	525	20024	95.1%	4.8%	156.3ms	7.5ms

95% hit ratio rather than 85%. We also conjecture, that because the performance estimates are based on antecedent-successor relationships which are unlikely to change significantly with data base size, this performance should also be found for large data base systems. As before information on user characteristics for large data base systems is required before a definitive evaluation is possible.

4.5 Disk Access Scheduling, No Cache Memory

When we consider the approach of organizing the data on the disk and using

Table 4.3

Page Prefetch Activity (FIFO Disk Scheduling)

Active Users	Cache Size	Cache Miss	Service Time	Disk Utilization	Prefetch Activity
150	260	4.8%	123.3ms	0.75	27.5%
200	300	5.1%	134.1ms	0.77	20.0%
250	340	5.1%	131.1ms	0.76	14.8%
300	360	4.8%	135.4ms	0.77	12.0%
400	420	4.9%	139.6ms	0.78	7.8%
500	450	6.0%	193.4ms	0.85	5.1%
600	475	5.0%	226.8ms	0.88	4.5%
700	525	4.9%	232.9ms	0.88	3.3%
800	550	4.9%	362.2ms	0.93	2.6%
900	575	5.2%	531.6ms	0.95	1.7%

disk scheduling, the results given in Table 3.1 clearly indicate it is possible for small data base systems. For data set 210 we found that 357 pages accounted for 84% of the page requests made. If these pages were stored sequentially on the disk, than eqn. (2.4.3.7) indicates that for a total request rate of 100 request/s the technique of disk access scheduling would be effective for this data base using only a single disk. It must be noted however that there is an service time of approximately one second if the effect of duplicate requests is ignored. If the number of users is large enough that the number of unique requests is 0.5 of the total requests, the service time is reduced to about 0.5 s. In both cases the system is inferior to the cache methods with respect to response time but is slightly better economically since the cache memory is not required. It should also be noted that if there were no duplicate requests the system would fail completely if the common set of pages only gave a hit ratio 0.7, as we speculated earlier might be the case for a 50 000 page data base. If the number of users was large enough that only half the requests were unique the system could operate with a hit ratio as low as 0.5. Before further evaluation of this alternative can be performed more information on the data characteristics is required.

4.6 Disk Access Scheduling, Distributed Cache

In previous sections the possibility of having the cache of predicted page requests distributed among the users was mentioned. This would require memory for 6 pages per user to give equivalent performance on the storage medium alone compared to the centralized cache. However, the centralized cache has a problem which a distributed cache does not have; namely, when multiple predictions have been made and the pages entered in cache, some mechanism must be included in the centralized cache management which disposes of the unused predictions. In the case of the distributed cache system, these unused predictions can be discarded at the local terminal. Thus, the decentralized cache approach may have advantages from a system viewpoint since the organization at the central server is simpler and less congested, and to the extent that prediction is effective, the user will not be aware of any system delays whatever. It should be noted that the distributed cache would effectively increase the the page request rate (by 1.9 for the data base studied) by fetching all the successors. However, this increase should not be significant if the disk scheduling approach is used. While the service time would increase it would be transparent to the user. Further, by storing successors sequentially, the disk activity could be reduced to effectively one disk access per request, regardless of the number of successors. We also note that if the number of users is large enough, the reduction in the number of distinct request would compensate for the increased page request rate. The key to successful performance is the size of the common set of pages.

4.7 Summary

Five possible architectures for the Telidon server have been considered. The essential differences are the disk scheduling policy, and the presence or absence of cache memory. It was shown that a system with a central cache memory and prefetching of anticipated page requests appears to significantly reduce the number of disks required. This system should be investigated further. The concept of distributing the cache memory to the user terminals also seems promising, but lack of information on Telidon data structures prevented more detailed investigation. Further study of the characteristics of Telidon data structures, and user characteristics, are required before the question of the best architecture can be resolved.

References

[1] D. Godfrey and E. Chang (ed.), The Telidon Book, Press Porcépic , Toronto, 1981.

[2] R.N. Horspool, G.V. Bochmann, and G.E. Saunders, "Memory Structures for Videotex," Publication #454, Département d'informatique et de recherche opérationnelle, Université de Montréal, Oct. 1982.

[3] G.U. Yule, "A mathematical theory of evolution, based on the conclusions of Dr. J.C. Willis, F.R.S.," Philosophical Transactions B, Vol. 213, p. 21, 1924.

[4] G.K. Zipf, Human Behavior and the Principle of Least Effort, Addison-Wesley, Reading, 1949.

[5] B. Mandelbrot, "A Note on a Class of Skew Distibution Functions: Analysis and Critique of a Paper by H.A. Simon," Information and Control, Vol. 2, p. 90, 1959.

[6] T.J. Teorey and T.B. Pinkerton, "A Comparative Analysis of Disk Scheduling Policies," CACM, Vol. 15, pp. 177-184, 1972.

[7] C.C. Gotlieb and G.H. MacEwen, "Performance of Moveable-head Disk Storage Devices," JACM, Vol. 20, pp. 604-623, 1973.

[8] E.G. Coffman, L.A. Klimko and B. Ryan, "Analysis of Scanning Policies for Reducing Seek Times," JACM, Vol. 22, pp. 602-620, 1975.

[9] N.C. Wilhelm, "A General Model for the Performance of Disk Systems," JACM, Vol. 24, pp. 14-31, 1977.