

A Study of File Sizes and Functional Lifetimes

M. Satyanarayanan

DEPARTMENT OF COMPUTER SCIENCE
CARNEGIE-MELLON UNIVERSITY

1. Introduction

The performance of a file system depends strongly on the characteristics of the files stored in it. This paper discusses the collection, analysis and interpretation of data pertaining to files in the computing environment of the Computer Science Department at Carnegie-Mellon University (CMU-CSD). The information gathered from this work will be used in a variety of ways:

1. As a data point in the body of information available on file systems.
2. As input to a simulation or analytic model of a file system for a local network, being designed and implemented at CMU-CSD [1].
3. As the basis of implementation decisions and parameters for the file system just mentioned.
4. As a step toward understanding how a user community creates, maintains and uses files.

2. Data Collection

2.1. The Environment

The data used in this paper was obtained on a Digital Equipment Corp. PDP-10 Model KL-10 processor [7] with 1 Mword of primary memory and eight 200 Mbyte disk drives, running the TOPS-10 operating system [13]. This machine has been the main computational resource of the CMU-CSD for the past five years. Towards the end of this period, a number of other machines were added to this environment. Though the machine used for this study is now off-loaded by those machines, it continues to play a very important role and is still

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

heavily used. Consequently, it is expected that the data presented here is a good reflection of the file usage characteristics of this community.

2.2. The File System

In the TOPS-10 operating system, every file has a 6-character *file name* and a 3-character *file extension*, and is a member of exactly one *directory*. The file extension indicates the nature of the contents of a file. For example, a Pascal program source would have the extension PAS, while its relocatable object module would have the extension REL. An installation-dependent number of extensions are regarded as "standard" extensions. Though system and user programs often make assumptions about a file based on its extension, there is no mechanism for validating or guaranteeing these assumptions. In practice, it is extremely rare that a standard extension is used for non-standard purposes. A quarter of the files examined had non-standard extensions; such files were ignored for those parts of this study that discriminated on the basis of file type. File names, unlike extensions, have no system-wide significance and were not examined.

A file consists of a sequence of fixed-length *blocks*, which are the units of addressability on the disks. Each block consists of 128 36-bit words. The last block in a file may be only partially written; such blocks were regarded, in this study, as whole blocks. The size of a file is limited only by the amount of secondary storage available. Unlike some file systems, such as OS/VS2 for the IBM 370 [5], a user does not have to estimate the size of a file at the time of its creation.

The operating system maintains, for each file, information regarding its size, its owner, the date it was last written, the date it was last accessed and its physical storage map. This information may be obtained by queries from user programs to the operating system.

In the environment in which this study was done, a manual file migration scheme is used to relieve the paucity of disk space. Every month, the operations staff runs a program which copies onto magnetic tape, and deletes from disks, those files which have neither been written

nor read in the preceeding three months. Files so migrated may be restored to disk at the request of their owners; in practice, very few such requests are received. Each user has a file named MIGRAT.DIR to which the migration program appends details of every file of that user it migrates. The union of a user's current directory and his MIGRAT.DIR entries constitutes the set of all files created, but not deleted, by that user. Users who wish to edit their MIGRAT.DIR files may do so; in practice this is quite rare. Files may also be migrated at the explicit request of users; it is observed that few files are migrated for this reason.

2.3. The Collection Technique

The files in this study fall into two classes: *current files* and *migrated files*. Data for both classes were obtained without any modifications to the operating system. A vendor-supplied utility program which creates a file containing details of every other file in the system was used to obtain data on current files. Data on migrated files was obtained by examining the MIGRAT.DIR file of every user in the system. For both classes the data extracted was organized as a 3-dimensional array with logarithmic age histogram buckets on one dimension, logarithmic size histogram buckets on another dimension, and the set of standard file extensions on the third. This array was created once each for current and migrated files, recorded in a file, and used as a database for software written to answer questions such as "What is the distribution of file sizes for current files with ages in *a given range* and with *a given set of extensions*." Table 6-1 shows an example of the output for one such query.

It should be noted that the data gathered by this method is a snapshot of the file system at one point in time. To examine the temporal behavior of the file properties described here, one would have to take snapshots spaced apart in time and compare the data from each.

2.4. The Quantities Measured

Probably the three most common questions asked about any file are:

1. "What does it contain?"
2. "How big is it?"
3. "How old is it?"

To the designer of a file system, the first question is probably only of marginal relevance. In any case, a precise answer to it requires a complete specification of the contents of a file! Specifying the extension of a file answers this question at one level of granularity. One outcome of this study is, therefore, a histogram of file extensions for any cross-section of the set of files examined. Figure 6-1 shows such a

histogram. The integers on the abscissa are mappings from the set of extensions to integers; Table 6-2 gives some of these mappings.

The size distribution of files is a crucial factor in deciding many of the file system parameters. The size of a file, measured in blocks, is one of the two quantities of primary interest in this study.

The other important quantity is the age of a file. "Age" is usually understood to mean the interval between the creation of a file and the instant of data collection. However, the original date of creation of a file is not maintained by TOPS-10; only the dates of last modification and last access are available. The difference between these two dates is a measure of the usefulness of the current data in the file. This quantity, the *functional lifetime* of a file, is the second item of interest in this study. For brevity, the term "f-lifetime" will mean "functional lifetime" in the rest of this paper. Fortunately, it is the f-lifetime of a file, not its chronological age, which is important in the design of file migration algorithms. Further, the file system design described in [1] and referred to in Section 1 is predicated on the assumption that the f-lifetime of files is short — this study was conducted, in part, to verify this assumption.

3. Data Interpretation

3.1. General Observations

A total of about 36,000 current files and 50,000 migrated files were examined in this study. About 99% of the files examined had sizes less than 1000 blocks and f-lifetimes less than 2000 days.¹ Both size and f-lifetime are discrete variables, with minimum values of 1 block and 1 day respectively. However, for ease of data interpretation and analytical approximation, both variables are treated as continuous variables.

Even a cursory examination of the data reveals some interesting facts. As Figure 6-2 indicates, the size distribution is skewed towards small sizes: 50% of the files are less than 5 blocks long and 95% of them are less than 100 blocks long. Figure 6-3 shows that the f-lifetime distribution is also skewed towards the low end, though not as sharply as the size distribution. Nearly 30% of the files have f-lifetimes of one day and 50% of them have f-lifetimes less than 30 days. Tables 3-1 and 3-2 present a subset of the data points used to generate Figures 6-2 and 6-3.

¹There are some files which are older than the system — they were obtained from other, older, PDP-10 systems.

Size	Cum. Fraction
1 block	0.245
5 blocks	0.518
10 blocks	0.665
100 blocks	0.952
1000 blocks	1.000

Table 3-1: Cum. Dist. Fn. of File Sizes

F-Lifetime	Cum. Fraction
1 day	0.320
10 days	0.410
100 days	0.651
1000 days	0.947
2000 days	0.986

Table 3-2: Cum. Dist. Fn. of File F-Lifetimes

The preponderance of very small files indicates that the TOPS-10 file allocation algorithm, which allocates disk storage in units of 5 blocks called *clusters*, tends to waste a significant amount of storage. However, the size of allocation tables increases as the unit of allocation decreases. Further, sequential access to a file is likely to be faster if successive blocks of the file are close to each other; this is more likely with larger allocation units. The TOPS-10 choice of 5 blocks as the unit of allocation probably represents a reasonable tradeoff between minimizing storage fragmentation and improving performance. Powell [8] discusses these tradeoffs and the use of adaptive disk allocation strategies in the context of the Demos file system. The minimum unit of allocation in that system is one disk sector, 4K bytes, which is of the same order of magnitude as the unit of allocation, 2.5K bytes, in TOPS-10.

The rest of the data analysis discusses three questions:

1. Are the properties of migrated files different from those of current ones?
2. Does the type of a file affect its properties?
3. Does the size of a file influence its f-lifetime?

3.2. Effect of Migration

Figure 6-4 compares the size distributions of current and migrated files. Except at the very low end, there is virtually no difference between the curves. At the low end, there are fewer migrated files than current files. The following explanation may explain this phenomenon: a large number of very short files are created by system programs. Text editors and mail servers are two examples of programs which create short auxiliary files which are used only once. These files are automatically deleted by the programs which created them when they are run a second time, or by users when they run out of disk quotas.

Such files are unlikely to remain both unaltered and undeleted for a period of time long enough to qualify them for migration. Consequently, small files are likely to form a smaller fraction of the migrated population than the current population.

Figure 6-5 shows that migrated files tend to have shorter f-lifetimes than current files. To see why this is so, consider how a long-f-lifetime file gets migrated. It would have to get created, then read (but not written) frequently for a long time and then all accesses to it would have to stop for a period long enough for it to qualify for migration. The only obvious files that meet these criteria are the successive versions of commonly used system or user programs. The infrequency of generation of such files leads to the fact that there are fewer long-f-lifetime files in the migrated population than in the current population.

The rest of this paper discusses only current files. Unless otherwise specified, the comments about current files also hold for migrated files with, perhaps, slightly different absolute numbers.

3.3. Effect of File Type

Since it is located in a research-oriented, academic environment, the machine on which this study was conducted is used primarily for two activities: document preparation and program development. Nearly half the files examined were created in conjunction with one of these two activities: program source files, program object files, document processor input files, and document processor output files. This section examines the characteristics of these four classes. The remaining half of the files was highly fragmented, with no clearly identifiable, large classes. Detailed study, discriminating on the basis of file type, of that set of files is unlikely to yield any fresh insights.

Figure 6-6 shows the effect of file type on file size. Object files and document processor output files tend to have much larger sizes than source files and document processor input files. The size characteristics of the entire population resembles that of source and document processor input files.

Figure 6-7 shows the effect of file type on file f-lifetimes. Document processor files tend to have much shorter f-lifetimes than program files. It is possible that this is due to the fact that once a document is complete, people tend to read the hard copy rather than the machine-readable copy. Important programs, on the other hand, tend to be used many times after they are debugged. Certain program source files are read long after they are debugged; for example, useful macro definitions are often included in other programs.

Table 3-3 summarizes the important characteristics of different file types. Probably the most important lesson to be learned in this section

is that the type of activities engaged in by a user community strongly influences the size and f-lifetime properties of the files created by it. Files in a commercial data processing environment or a fusion research center can be expected to exhibit markedly different characteristics from those reported here.

Type of File	Number	Size		F-Lifetime	
		Mean	Std Dev	Mean	Std Dev
Program Sources	4010	21.84	47.63	363.6	731.3
Object Files	3474	53.99	116.3	414.6	681.4
Doc. Proc. Input	7085	29.28	70.95	137.5	322.7
Doc. Proc. Output	872	61.6	111.04	45.2	207.9
Entire Population	35652	23.89	66.83	238.9	531.9

Table 3-3: Effect of File Type on File Sizes and F-Lifetimes

3.4. Size/F-Lifetime Correlation

How does the size of a file affect its f-lifetime? Since the environment contains no large, frequently-modified databases, the most likely type of large files are infrequently-modified databases, or frequently-used and rarely-modified system programs such as compilers and editors. Small files, on the other hand, are likely to be temporary files of various sorts, or files associated with use-once-and-throw-away programs. Intuitively, therefore, one would expect large files to exhibit longer f-lifetimes than small files.

Figure 6-8 shows the f-lifetime distribution of files, with size as a parameter². Surprisingly, the curves indicate that large files tend to have shorter, not longer, f-lifetimes than small files. The largest average f-lifetime is, in fact, that of 1-block files! Table 3-4 summarizes the information in Figure 6-8.

Size	Number	F-Lifetime	
		Mean	Std Dev
1 block	8745	264.8	633.1
10 blocks	762	231.4	512.8
91-100 blocks	207	170.5	308.8
401-500 blocks	101	123.1	344.5
901-1000 blocks	13	120.2	240.6

Table 3-4: Effect of Size on F-Lifetime

The data was closely examined to see if any particular file type, with markedly different characteristics from the total population, was causing this anomaly. Perhaps document processor output files, whose f-lifetimes tend to be short, perturb the data.³ However, the data fails to support this hypothesis — document processor output files tend to have

²There are too few files of size 500 blocks or more to obtain a smooth cumulative distribution function; a discrete function is therefore shown for such files.

³This suggestion was offered by one of the reviewers of this paper.

short f-lifetimes independent of size, and they do not form more than 9% of the total population larger than 100 blocks. Removing document processor output files from the total population does not change the size/f-lifetime correlation. One file type, with extension MSG, exhibits an average size nearly four times and an average f-lifetime one-tenth that of the total population. Files with this extension are used as repositories of electronic mail messages which have been received and read by users, but not yet deleted by them. Table 3-5 presents the size/f-lifetime correlation for MSG files. However, excluding such files does not change the size/f-lifetime behaviour. In fact, no single file type, constituting 5% or more of the total population, is responsible for this characteristic of files. For example, Table 3-6 shows the effect of excluding MSG files and document processor output files — there is still a falloff of f-lifetime with size. One can therefore take this to be a characteristic of files in general. If one were to assume that every bit of information stored in a file system is equally likely to be modified, the probability of a file being modified is directly proportional to the size of the file. Under this assumption, large files are indeed likely to have shorter f-lifetimes than small files.

Size	Number	F-Lifetime	
		Mean	Std Dev
1 block	7	414.0	581.0
10 blocks	16	50.9	115.0
91-100 blocks	13	25.8	71.5
401-500 blocks	11	29.5	76.5
901-1000 blocks	3	1.0	0.0

Table 3-5: Size/F-Lifetime Correlation for MSG Files

Size	Number	F-Lifetime	
		Mean	Std Dev
1 block	8725	266.5	640.3
10 blocks	717	243.0	528.2
91-100 blocks	182	189.3	322.7
401-500 blocks	80	142.4	375.1
901-1000 blocks	9	173.1	272.1

Table 3-6: Effect of excluding MSG and Doc. Proc. Output Files

4. Analytic Approximation

4.1. General Discussion

Analytic approximations to the size and f-lifetime distributions are investigated here for two reasons:

- To obtain a simple and computationally efficient means of generating random size and f-lifetime variables.
- To see if a model useful in analytic performance evaluations can be postulated for file sizes and f-lifetimes.

The models examined in this paper were motivated by the following observations:

- At least to a first approximation, the size and f-lifetime of a file one creates is independent of the files one has created in the past.
- Both size and f-lifetime distributions are sharply skewed toward the low end.
- A Markovian model is analytically the most tractable [2, 3].

The simplest model meeting these criteria is an exponential distribution. If the size distribution is exponential with mean M , the probability that a random file has a size less than X is given by $1 - e^{-X/M}$. Both the mean and standard deviation of such a distribution are equal to M . Unfortunately, almost all the size and f-lifetime distributions observed have standard deviations between two and three times that of the corresponding means. This implies that a simple exponential model is certain to be unsuitable.

A hyperexponential model can exhibit coefficients of variation (i.e., ratio of standard deviation to mean) greater than unity. A k -stage hyperexponential consists of k simple exponentials with means M_1, M_2, \dots, M_k , weighted so that they have probabilities $\alpha_1, \alpha_2, \dots, \alpha_k$ of being chosen. Figure 4-1 shows such a model.

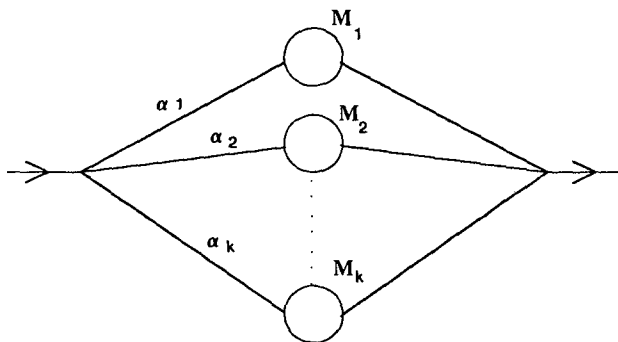


Figure 4-1: A k -Stage Hyperexponential Server

To generate a value for the random variable represented by this model, one proceeds in two steps:

1. With probability α_i , select one of the k stages.
2. Generate one value from an exponential distribution of mean M_i .

Each of the k stages can be viewed as being one population class with a simple exponential distribution. There is no guarantee, however, that such classes correspond to any clearly identifiable types of files. To fit a hyperexponential model to empirical data one needs to determine the number of stages, k , the means M_i , and the probabilities α_i . The next

two sections discuss two alternative approaches for estimating these parameters.

4.2. The Moment Matching Method

In this method we try to find a hyperexponential model whose first few moments match the corresponding moments of the empirical distribution. If the empirical distribution were truly hyperexponential, one could find a model with all its moments matching. Otherwise the model is only an approximation to the empirical distribution.

The p^{th} moment of a k -stage hyperexponential is related to its parameters (α 's and M 's) by the following relationship:

$$\alpha_1 M_1^p + \alpha_2 M_2^p + \dots + \alpha_k M_k^p = (p^{\text{th}} \text{ moment})/p!$$

This is easily derived using the moment generating function technique [3]. By using an iterative solution technique on $2k-1$ such equations and the constraint $\alpha_1 + \alpha_2 + \dots + \alpha_k = 1$, one can solve for the $2k$ unknowns, α_1 to α_k and M_1 to M_k .

Figure 6-9 compares the empirical size distribution of current files with a 2-stage hyperexponential fit.⁴ The first three moments of these two curves are identical. The two curves differ by no more than 5% at all points except at the very low end. Figure 6-10 shows the distribution of f-lifetimes of current files versus a 2-stage hyperexponential fit. Clearly the fit is not as good as for file sizes, especially at the low end, where the hyperexponential grossly underestimates the empirical distribution.

Adding more stages to the hyperexponential, thereby matching more moments, yielded negligible improvements in the fit (less than 0.5% except at the very low end, where the improvement was close to 1%.) The moment matching technique is thus only of limited usefulness in analytically approximating the empirical data presented here.

Fig. No.	α_i	M_i
6.9	0.089	164.9
	0.91	10.08
6.10	0.835	77.49
	0.165	990.8
6.11	0.518	2.0
	0.433	23.83
	0.048	252.1
6.12	0.407	1.66
	0.337	70.0
	0.256	795.0
6.13	0.32	1 (const)
	0.132	7.97
	0.385	132.1
	0.162	1082.8

Table 4-1: Derived Parameters for Fitted Curves

⁴Table 4-1 presents the derived parameters α_i and M_i for all the fitted curves discussed in this section and the next.

4.3. A Heuristic Approach

The basis of this technique is that a simple exponential has its mean equal to its standard deviation. Suppose the actual distribution consists of a hyperexponential with wide separation of means between stages. Then the abscissa may be partitioned into sections which are in one-to-one correspondence with the stages of the hyperexponential. The majority of data points which fall into a given section may be assumed to have been generated by the stage corresponding to that section. To fit such a hyperexponential distribution one has to obtain a suitable partitioning of the data points in an empirical distribution. Starting from the low end, one examines successively larger initial segments of the empirical distribution until one finds a segment with its mean close to its standard deviation. This segment is represented as one stage of a hyperexponential. Its mean, M , is the mean of the segment and its probability of selection, α , is the fraction of the total population in that segment. This initial segment is removed from the distribution, and the procedure is repeated on the rest of the distribution. The procedure terminates when the entire distribution has been approximated. The resulting distribution is thus an overall hyperexponential in which the parameters of each stage have been obtained by method described above.

The above procedure is a heuristic rather than an algorithm because judgement has to be used in deciding when the mean of a segment is close enough to its standard deviation. In practice, good results were obtained when the mean and standard deviation were within 25% of each other. Only 3 or 4 stages were necessary to fit the empirical data well. Closer examination of this procedure, to see if it can be used as the basis of an algorithm, is one possible extension of the work presented here.

Figure 6-11 shows the fit of a 3-stage hyperexponential obtained by this method to the empirical distribution of file sizes. The fit is indeed excellent throughout the range of the curves (within 2%).

Figure 6-12 shows the fit of a 3-stage hyperexponential to the f -lifetime distribution. The fit is not as good as the file size fit, but it is certainly better than that obtained by matching moments (within 4%, except at the very low end). At the very low end, near 1 day, the approximation grossly underestimates the actual curve. The only way to remove this anomaly was to treat files of f -lifetime 1 as a class by themselves and assign a constant probability to them. This constant distribution causes no problems in random variable generation, but it does violate the Markovian assumption thereby making analytical solutions harder. One could get over this difficulty by viewing the constant distribution as a distribution with mean 1 and extremely small standard deviation. Such a distribution could be realized using an Erlang distribution [2, 3], thus retaining the Markovian property of the

model. Figure 6-13 shows the effect of this modification — as expected, the fit at the very low end is much better.

Below about 500 days, Figure 6-13 shows that the hyperexponential f -lifetime curve is consistently above the empirical curve. Whether this is merely an outcome of the curve fitting procedure or an observation of significance is not clear. If it is significant, the curves tell us that the f -lifetime of files is longer than that predicted by the postulated model. There are, in fact, certain processes that might give rise to such a phenomenon. The most commonly used programming languages and document processor in this environment support separate compilation and inclusion of external sources files (called "require" files). During the course of debugging, attention is typically focussed on one such file and modifications are made to it. However, each compilation results in all the related files being accessed, thereby lengthening their f -lifetimes. The f -lifetime of such files is no longer independent of all other files, thus violating the one of the assumptions implicit in the model.

4.4. Goodness of Fit

The Kolmogorov-Smirnoff test [4] was used to quantitatively ascertain the quality of the curve-fitting procedures described in Sections 4.2 and 4.3. In this test, the statistic of interest is the maximum magnitude of the vertical displacement between the empirical and fitted curves. Formally, if an empirical distribution function $F_n(x)$ is obtained from n independent observations of a random variable X , and if $F(x)$ is a curve fitted to these observations, the statistic K_n is given by:

$$K_n = (\sqrt{n}) \text{ MAX} |F_n(x) - F(x)| \quad \text{for } -\infty < x < +\infty$$

K_n can be used to determine to any desired level of confidence, whether or not $F(x)$ really represents $F_n(x)$.

In all cases, the fitted curves of Sections 4.2 and 4.3 were rejected at the 99% confidence level. This implies that it is unlikely that the actual size and f -lifetime distributions and the calculated distributions are identical. The good graphical fits observed only indicate that for practical purposes, one may represent these distributions as hyperexponentials.

5. Related Work

Three previous studies of file systems have been reported, all examining file usage on IBM 360 or 370 computers and focussing on the time-dependent behaviour of various file properties. This section briefly summarises the three studies and examines how they differ from the one described in this paper.

Using data collected from an IBM 360 Model 195 at the IBM San Jose Research Laboratory, Revelle [9] examined the frequency of reference to each file as well as the total space occupied each day by

three disjoint file classes: files created that day, files deleted that day, and old files accessed that day. The data was collected over a period of 144 days, on a job mix consisting of scientific jobs, commercial jobs, and program development jobs. Revell's main observations were:

1. The rate of reference to a file decreases with its age.
2. For a constant user population size, the total space occupied by all files increases approximately linearly with time.
3. The total space occupied by files in active use remains approximately constant.
4. The file creation and file reference processes are stationary with respect to time, while the file deletion process is not.
5. A Poisson process is not adequate to describe the file creation and reference processes. Negative binomial distributions fit the distributions of these processes well.

Stritter [12] collected data over a period of one year from an ensemble of IBM 360 and 370 machines at the Stanford Linear Accelerator Center, where the job mix consisted of both large scientific jobs and small program development jobs. System files were excluded from this study. Stritter observed that:

1. The size distribution of files is strongly skewed to small sizes, and is well approximated by a Pareto distribution [6].
2. Exponential distributions are good approximations to the file age distribution and the file lifetime distribution.⁵
3. For a subset of the files examined (selection criteria unspecified), the file reference process is well approximated by a Poisson process.

Smith [10] conducted a more detailed and thorough statistical investigation of the data collected by Stritter, and applied it to the design of file migration algorithms [11]. A complete characterization of Smith's work is impossible to present in the space available here. Some highlights of his study, pertinent to this paper, are:

1. File sizes tend to be strongly skewed toward small sizes.
2. Most files are used very few times.
3. Large files tend to be used more frequently than small files.
4. Older files tend to be used less frequently than young files.
5. The lifetime of a file and its size are not strongly correlated.⁵
6. A Bernoulli process (the discrete counterpart of a Poisson process) does not adequately describe the file reference process.

Because different variables were measured, and because of differences in the nature of the file systems involved, it is difficult to quantitatively compare the results of those studies with the data presented here. About the only observation in this study that can be compared to the other studies is the file size distribution — in all cases, the distribution is strongly skewed to small sizes.

The studies mentioned above differ from the work described in this paper in a number of ways:

- There are obvious differences in the computer systems on which the measurements were made and in the kinds of activities engaged in by the corresponding user communities. This study deals with a PDP-10 system supporting an academic community. The other three examine IBM 360-370 systems supporting user populations whose primary activities are scientific or commercial data processing, or program development.
- File size measurements on IBM 360-370 systems are expressed in disk tracks, which are more than order of magnitude larger than PDP-10 disk blocks.
- There is no external indication (such as file extension) of the nature of the contents of a file in IBM 360-370 systems. Consequently, none of the studies of these systems examines file properties as a function of file type.⁶
- IBM 360-370 systems support *Partitioned Data Sets (PDSs)* while TOPS-10 does not. A PDS is a file composed of subfiles called *members*, and is analogous to a directory with subdirectories in other file systems. Members of a PDS may be read, written, created, or deleted individually. However, statistics on them were available, in all three studies, only for the PDSs as a whole and not for individual members. Thus the observed characteristics of a PDS are the cumulative characteristics of its members. The absence of PDSs in PDP-10 systems implies that, even if the quantities measured had been identical, one could expect significant differences in the properties of files on PDP-10 systems and IBM 360-370 systems.
- None of the studies cited distinguish between read and write accesses to files. In a network environment, the viability of maintaining a cached copy of a file depends on the relative frequencies with which the file is read and written. The f-lifetime data presented here will probably be of help in designing file caching strategies in such an environment.
- The studies on IBM 360-370 systems examine the lifetimes of files.⁵ This information, which will be useful in estimating secondary and archival storage requirements, is not available in the study described here. Obtaining this information would require modifications to the TOPS-10 operating system.

⁵Lifetime being defined as the interval between the dates of file creation and deletion.

⁶Smith [10] does distinguish between three classes of files, but the classification is at a much coarser level of granularity than that presented here.

- This study only examines a snapshot of a file system at one point in time. The other three studies examine file systems over a period of time and investigate the time-dependency of various file properties. An obvious extension of the work presented here is to study the behaviour of the file system as a function of time.

6. Caveat and Summary

In using the data and analysis presented in this paper, a file system designer should bear in mind the conditions under which they hold:

1. The data was gathered from one machine (the most heavily used one) in an environment with many machines. It would be instructive to examine the file usage patterns on the other machines and compare them with the one presented here. Such a comparison would have to take into account the fact that both the hardware and software on those machines is different from that on the machine discussed here.
2. The data is a profile of the file system at one point in time. The time-varying behavior of the system remains to be studied.
3. The data was obtained in an academic environment and may not be directly extendable to other environments.
4. In contrast to systems such as VMS on the VAX machines[15] and TOPS-20 on the PDP-20 machines[14], only the latest version of a file is kept. Consequently, the original state of a file is lost when it is modified. It is difficult to quantify the effects of this characteristic of TOPS-10 on the way in which people make use of the file system. File systems in which a different design decision has been made on this issue may exhibit different file characteristics.

Keeping in mind these constraints, the results of this study may be summarized as follows:

- Most files are very small. This seems true for other file systems too.
- Most files have short f-lifetimes. However, some files have significantly longer f-lifetimes.
- The type of a file class strongly affects the properties of that class.
- Larger files tend to have shorter f-lifetimes; i.e., the data within them tends to remain unaltered for shorter lengths of time.
- A hyperexponential is a very good model of the file size distribution and a reasonable model of the f-lifetime distribution.

Acknowledgements

I would like to thank George Robertson for his advice and encouragement, Louis Monier for helping me devise the iterative solution procedure mentioned in Section 4.2, and Ivor Durham, whose PLOT program enabled me to produce the graphs in this document with a minimum of effort. I would also like to thank Richard Snodgrass and Alan Smith for their critical comments on the paper and their suggestions for its improvement. Of course, I am also indebted to all the users, past and present, of the CMU-10A who unwittingly contributed to the data presented here!

References

- [1] Accetta, M., Robertson, G., Satyanarayanan, M. and Thompson, M.
The Design of a Network-Based Central File System.
Technical Report CMU-CS-80-134, Department of Computer Science, Carnegie-Mellon University, August, 1980.
- [2] Allen, A.O.
Probability, Statistics, and Queueing Theory.
Academic Press, 1978.
- [3] Kleinrock, L.
Queueing Theory Vol.1: Theory.
John Wiley & Sons, 1975.
- [4] Knuth, D.E.
The Art of Computer Programming.
Addison-Wesley, 1981.
Second Edition, pp. 45-52.
- [5] International Business Machines Corp.
OS/VS2 MVS JCL.
Order No. GC28-0692-4, May 1979.
- [6] Johnson, N.L. and Kotz, S.
Distributions in Statistics: Continuous Univariate Distributions—1.
Houghton Mifflin, Boston, MA, 1970.
pp. 233-248.
- [7] Digital Equipment Corp., Maynard, Mass.
DecSystem10 Hardware Reference Manual.
DEC-10-XSRMA-A-D.
- [8] Powell, M.L.
The DEMOS File System.
In *Proceedings of the Sixth Symposium on Operating Systems Principles*, pages 39-40. Association for Computing Machinery, November, 1977.
- [9] Revelle, R.
An Empirical Study of File Reference Patterns.
Technical Report RJ 1557, IBM Research Laboratory, San Jose, April, 1975.

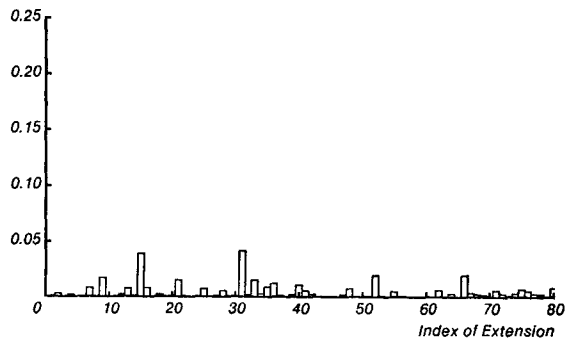
- [10] Smith, A.J.
Analysis of Long Term File Reference Patterns for Application
to File Migration Algorithms.
IEEE Transactions on Software Engineering SE-7(4):403-417,
July, 1981.
- [11] Smith, A.J.
Long Term File Migration: Development and Evaluation of
Algorithms.
Communications of the ACM 24(8):521-532, August, 1981.
- [12] Stritter, E.P.
File Migration.
PhD thesis, Department of Computer Science, Stanford University,
January, 1977.
- [13] Digital Equipment Corp., Maynard, Mass.
DecSystem10 Operating Systems Command Manual, DEC-10-
OSCMA-A-D.
May 1974.
- [14] Digital Equipment Corp.
TOPS-20 User's Guide.
Maynard, MA, January 1980.
Form No. AA-4179C-TM, pp. 4.1-4.12.
- [15] Digital Equipment Corp.
VAX11 Software Handbook.
Maynard, MA, 1978.

```

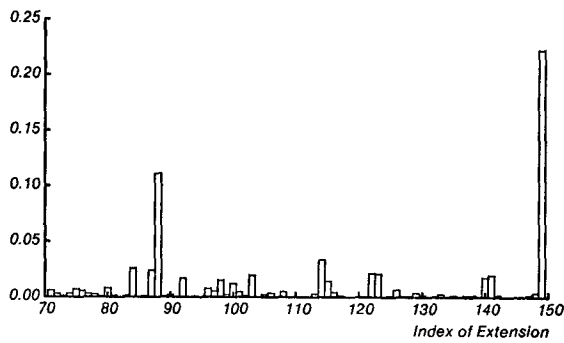
; Data Source Identification: "Combined histogram for
;   OSKB, DSKC and TEMP on CMUA on 14,15-JAN-81."
;
; The files contained no Migrated files
; Population Selection Parameters:
;   LoSizeValue =          1 HiSizeValue =          1000
;   LoSizeIndex =          1 HiSizeIndex =           28
;   LoAgeValue =           1 HiAgeValue =          5000
;   LoAgeIndex =           1 HiAgeIndex =           32
; Extension mode specified was ALLBUTUFD
; Total number of relevant files =          35652
;
; Moment 1 of Size = 2.3889931E+01
; Moment 2 of Size = 5.0366638E+03
; Moment 3 of Size = 2.4054326E+06
; Moment 4 of Size = 1.5510999E+09
; Moment 5 of Size = 1.6628644E+12
;
; Moment 1 of Age = 2.2819419E+02
; Moment 2 of Age = 3.3500745E+05
; Moment 3 of Age = 9.6468970E+08
; Moment 4 of Age = 3.6899178E+12
; Moment 5 of Age = 3.6545613E+16
;
;
; Data points: Size Limit, Cumulative Fraction
;
;   1, 2.4528778E-01
;   2, 3.5052731E-01
;   3, 4.2118254E-01
;   4, 4.7523280E-01
;   5, 5.1828789E-01
;   6, 5.5808930E-01
;   7, 5.9045776E-01
;   8, 6.1805789E-01
;   9, 6.4330192E-01
;  10, 6.6467519E-01
;  20, 7.8256478E-01
;  30, 8.3689553E-01
;  40, 8.7445304E-01
;  50, 8.9950071E-01
;  60, 9.1683496E-01
;  70, 9.2861549E-01
;  80, 9.3851677E-01
;  90, 9.4625827E-01
; 100, 9.5206439E-01
; 200, 9.7983281E-01
; 300, 9.8816335E-01
; 400, 9.9276337E-01
; 500, 9.9559632E-01
; 600, 9.9725120E-01
; 700, 9.9806462E-01
; 800, 9.9868170E-01
; 900, 9.9963537E-01
;1000, 1.0000000

```

Table 6-1: Sample Output: Size Distribution of Current Files



Histogram of File Extensions

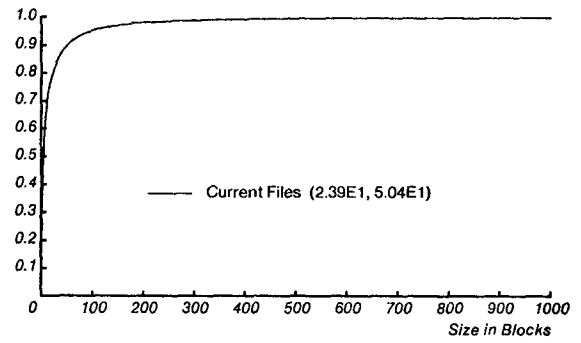


Histogram of File Extensions

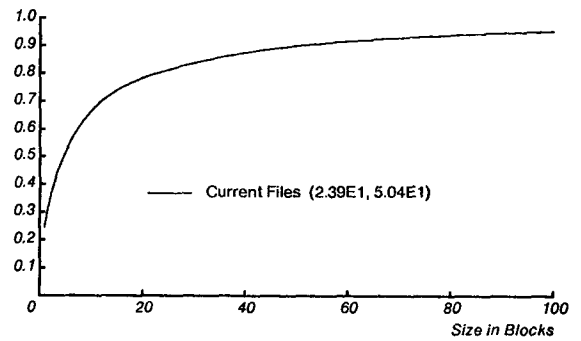
Figure 6-1: Histogram of Current File Extensions

Index	Ext.	Purpose
0	A68	Algol-68 source file
1	ABS	Nonrelocatable program
2	ADA	ADA source file
3	ALG	Algol-10 source file
4	APL	Saved APL workspace
	...	
	...	
	...	
	...	
146	USR	List of users of a system for SHEPHERD
147	XGO	XGP output file from PUB, Scribe, others
148	XXY	Uninteresting standard extensions
149	ZZZ	Non-standard extensions

Table 6-2: Examples of Mappings of Extensions



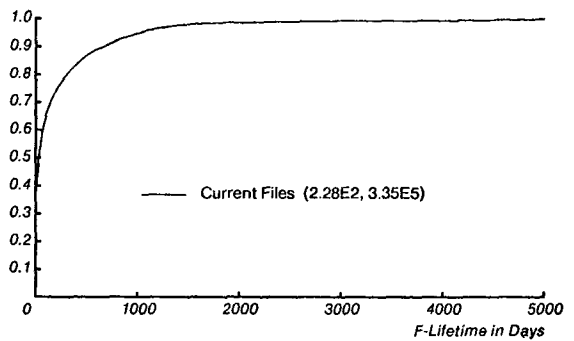
Cum. Dist. Fn. of Sizes



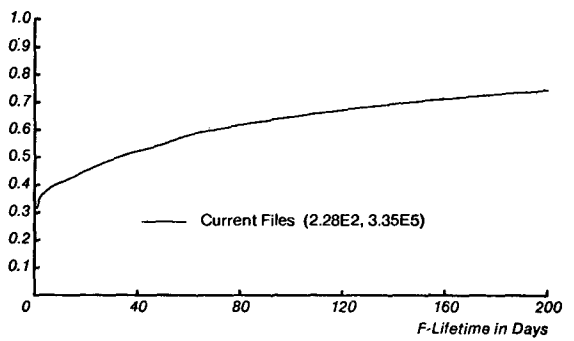
Cum. Dist. Fn. of Sizes

Figure 6-2: Size Distribution of Current Files

Note: Numbers in parenthesis are the first and second moments.



Cum. Dist. Fn. of F-Lifetimes



Cum. Dist. Fn. of F-Lifetimes

Figure 6-3: F-Lifetime Distribution of Current Files

Note: Numbers in parenthesis are the first and second moments.

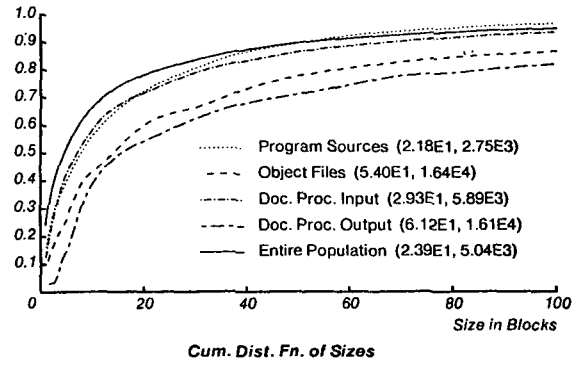
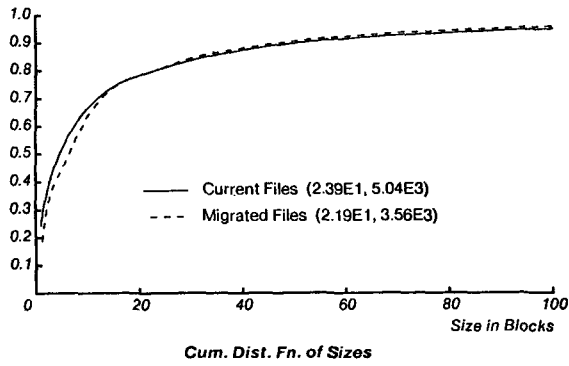
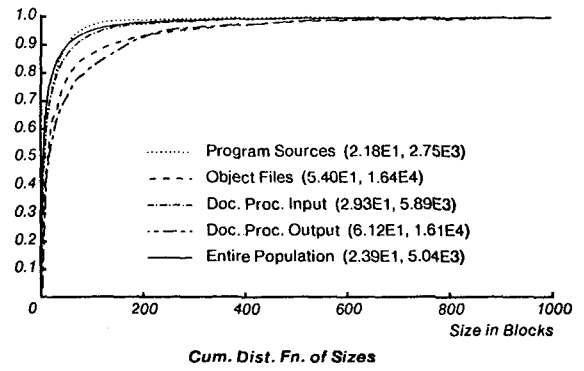
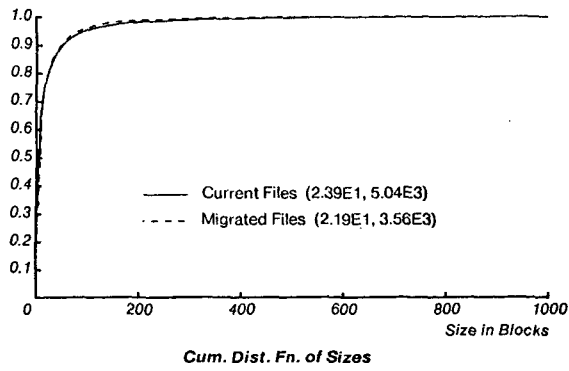


Figure 6-4: Effect of Migration on File Size

Note: Numbers in parenthesis are the first and second moments.

Figure 6-6: Effect of File Type on Size

Note: Numbers in parenthesis are the first and second moments.

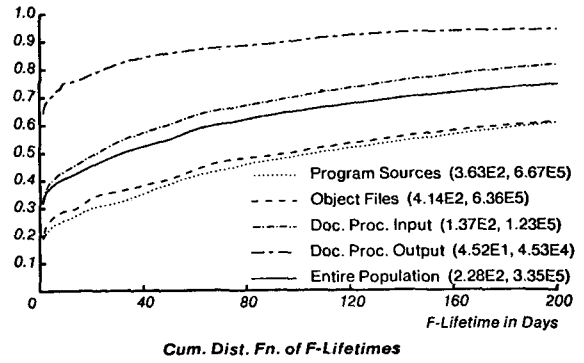
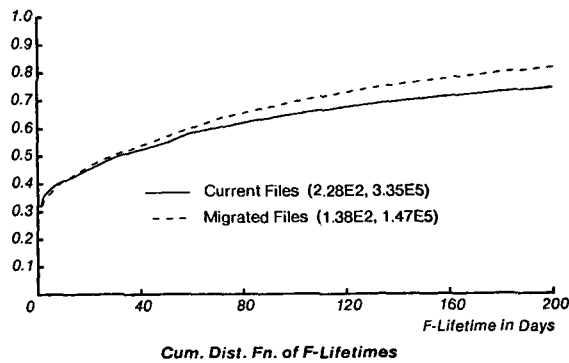
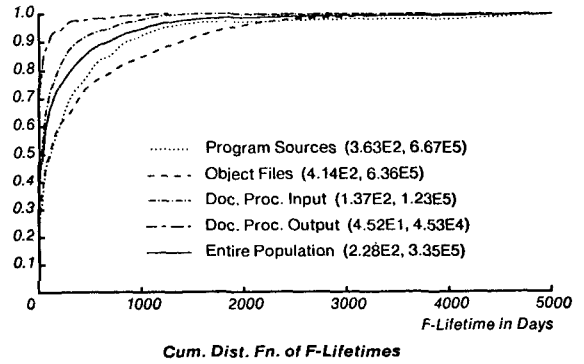
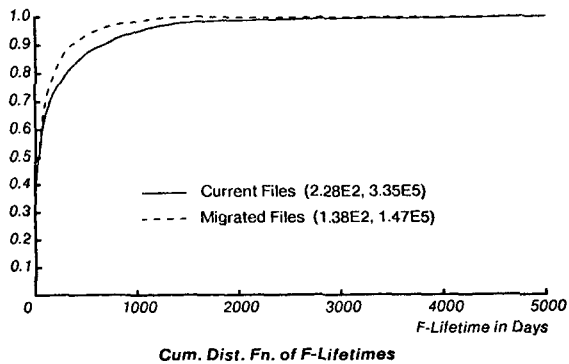


Figure 6-5: Effect of Migration on File F-Lifetime

Note: Numbers in parenthesis are the first and second moments.

Figure 6-7: Effect of File Type on F-Lifetime

Note: Numbers in parenthesis are the first and second moments.

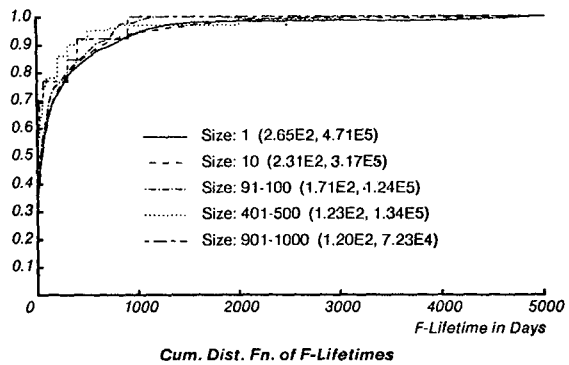


Figure 6-8: Effect of Size on F-Lifetime

Note: Numbers in parenthesis are the first and second moments.

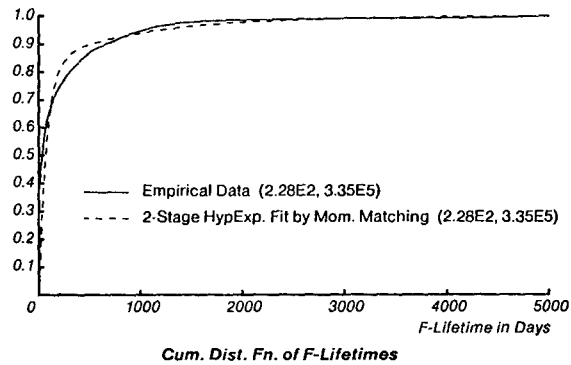


Figure 6-10: 2-Stage HyperExponential Fit for File F-Lifetime

Note: Numbers in parenthesis are the first and second moments.

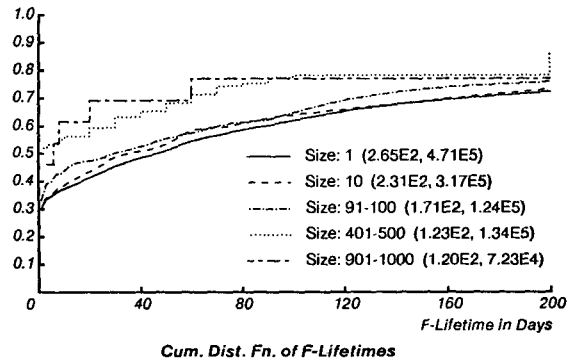


Figure 6-9: 2-Stage HyperExponential Fit for File Size

Note: Numbers in parenthesis are the first and second moments.

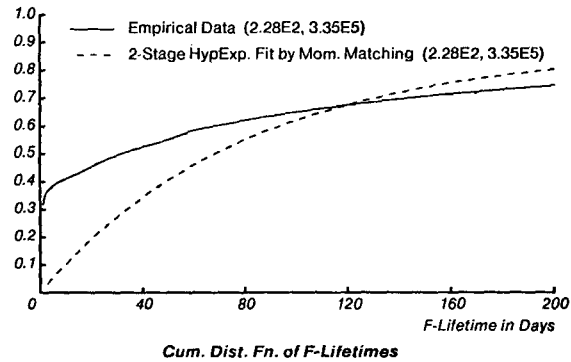
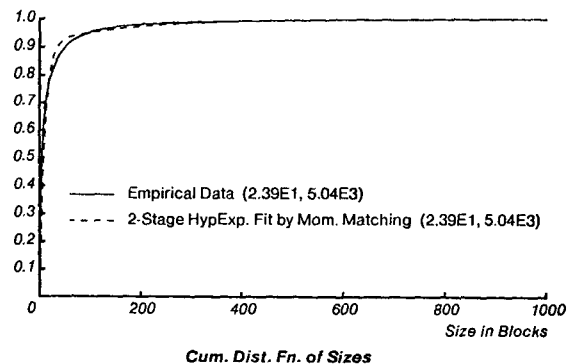


Figure 6-11: Heuristic 3-Stage HyperExponential Fit for File Size

Note: Numbers in parenthesis are the first and second moments.



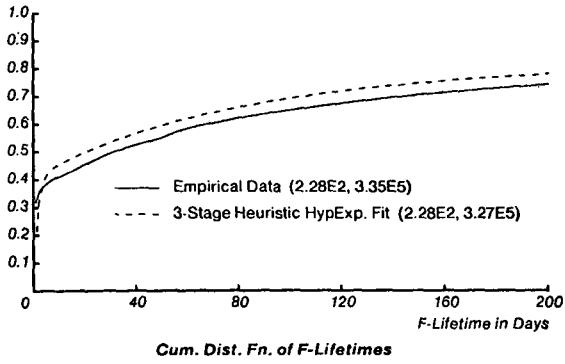
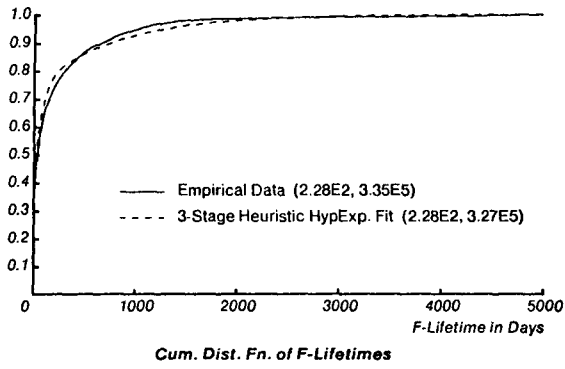


Figure 6-12: Heuristic 3-Stage HyperExponential Fit for F-Lifetime

Note: Numbers in parenthesis are the first and second moments.

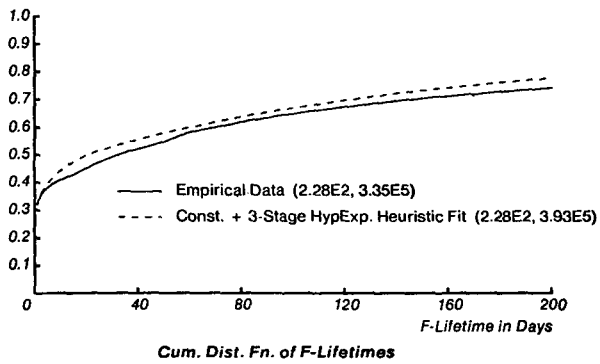
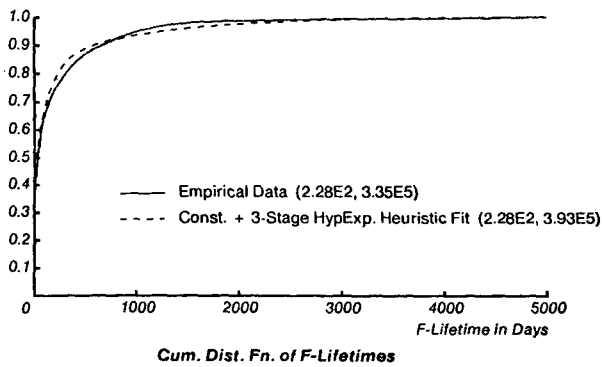


Figure 6-13: Const. + 3-Stage HyperExponential Fit for F-Lifetime

Note: Numbers in parenthesis are the first and second moments.