CHAPTER 3: **DATA ANALYSIS AND SIMULATION**

The widespread availability of inexpensive computing power is having a major impact on data analysis. In times to come, greater volumes of data be collected and analyzed than ever before, and the complexity of the data will mushroom. Computers allow us to carry out calculations and displays of data that were literally *unthinkable* only a decade ago. This will have a profound impact on the design of computer systems: an integral part of the design will be data gathering and analysis tools to determine system performance. As more and more data is gathered on each design, iterations will be carried out based on inferences from data.

Because so much data can be gathered (literally millions of items per day), it is essential that the data be stored and displayed in a manner that guides someone in drawing cause and effect inferences quickly. This means imposing *some* structure at the outset on the data gathering, storage and display. This structure is a *model.* All models imposed on the data should be checked against both *analytic* or *simulation* models. This could occur at any point in a product life cycle of a computer communication system: during initial conception, during development, during manufacturing, and on into installation, operations, maintenance, repair and salvage. Digital simulation provides a useful and effective adjunct to direct analytical evaluation of communication system performance. Indeed, there are many situations where explicit performance evaluation defies analysis and meaningful results can be obtained only through either actual prototype hardware and software evaluation or digital computer simulations. The former approach is generally cumbersome, expensive, time-consuming, and relatively inflexible. These are considerations which typically weigh heavily in favor of digital simulation.

Simulation also frees the analyst from a great deal of repetitive work involved in substituting numbers into formulae and tables and enables the analyst to concentrate on results. Another advantage is the insight into system performance provided, both by the modeling process itself and by the experience gained from simulation experiments. Again, computers can assist us: it may be quite difficult or expensive to gather data or measurements, so computer assisted analysis can quantify the importance of the data gathering and analysis procedures.

3.1 Methodology

There are two stages in this methodology. The first is a collection of techniques that together are called *exploratory data analysis* to determine if any patterns arise. For example, if we have one hundred items of data, and ninety nine of them are between zero and one, and one is at five hundred, what do we do: Do we assume the ninety nine are fine and discard the one outlying value? Do we assume none of the hundred data are valid because the one outlying value uncovered a fundamental flaw in the measurement technique? Our point is that a great deal of judgement is required at this stage, intuition built upon practice, hours and hours of checks and cross checks to ascertain that in fact the data gathered are valid, hold together, and tell a story. Put differently, exploratory data analysis is an art, and some people are better artists than others.

A second stage is to summarize the data using a small number of parameters that comprise a *model* which must be tested on the data and then used to *predict* wholly new operations. The model can be based on a mathematical *analysis* or it can be *algorithmic* in nature, a *simulation* model. In this chapter we focus on exploratory data analysis, and on simulation models, while the remainder of the book focuses on analytic models for the second stage.

This is an iterative process: gathering data, analyzing, and modeling, followed by more data gathering, analysis, and modeling. The trick is really knowing when to stop!

3.1.1 Additional Reading

[1] B.Efron, *Computers and the Theory of Statistics: Thinking the Unthinkable,* SIAM Review, **21,** 460-480 (1979).

[2] U. Grenander, R. F. Tsao, *Quantitative Methods for Evaluating Computer System Performance:A Review and Proposals,* in **Statistical Computer Performance Evaluation,** W. Freiberger (editor), Academic Press, NY, 1972.

[3] F.Hartwig, B.E.Dearing, **Exploratory Data Analysis,** Sage Publications, Beverly Hills, 1979.

[4] F.Mosteller, J.W.Tukey, **Data Analysis and Regression,** Addison-Wesley, Reading, Mass, 1977.

3.2 Statistics

Three types of statistics will concern us here:

- *First* order statistics such as averages or mean values
- *Second* order statistics that measure *fluctuations* about averages and *correlation* with time of the same thing or two different things
- *Distributions* or fractions of time that a given event is true

We now will examine each of these in more detail.

3.2.1 First Order Statistics Computer communication systems have *states.* This suggests we should *count* the number of times each state is entered or left, and measure the *time* the system is in each state.

Suppose that N observations are made. For example, the data might be the time intervals that a system is in a given state, say K, or the number of times a transition is made from state K to state J. Let $X_I, I=1,...,N$ denote the Ith observation. The mean of these observations, denoted $E(X)$, is given by

$$E(X) = \frac{1}{N} \sum_{I=1}^{N} X_I$$

For example, consider a job that goes through two steps: submission from a terminal by an operator and processing. This cycle is repeated again and again. Typical types of data that might be gathered would be

- The time spent reading and thinking and entering each job in
- The time an operator spends waiting for a response to a submission
- The time intervals the system is busy processing jobs which infers the fraction of time the system is busy at all, its *utilization,* or equivalently, the fraction of time the system is idle at all, its *idleness*
- The number of jobs submitted to the system from operators, which should be checked against the number of jobs actually executed by the system. This would infer the mean throughput rate: the number of jobs executed over the observation time interval.

EXERCISE: Can you think of any more?

3.2.2 Second Order Statistics Two types of measures are of interest here: the *fluctuations* about the mean or average, and the *correlation* between different events.

For our computer system model, we might look at the number of transitions from one state to another state, or we might measure the time the system spends in a pair of states. The measure of fluctuation is called the *variance,* denoted by $\sigma_{XX}{}^2$ and given by

$$\sigma_{XX}{}^2 = E\left[\sum_{I=1}^{N}(X_I - E(X))^2\right]$$

The subscript XX emphasizes that this is the average of the *square* of each value of X from the total mean. A related measure of interest is its *square root* which is denoted by σ_{XX} and called the *standard deviation.* Often it will be clear from the context what the underlying variable is, such as X here, and this subscript will be dropped.

A related measure of interest is the so called *squared coefficient of variation* which is defined as the variance over the mean squared:

$$squared\ coefficient\ of\ variation = \frac{\sigma_{XX}{}^2}{E^2(X)}$$

In words, this measures the standard deviation in units of the mean or average. If the squared coefficient of variation is much less than one, then there is little fluctuation about the mean. The squared coefficient of variation will be *zero* when there is no fluctuation. When the squared coefficient of variation is much greater than one, there is significant variability about the mean.

The measure of correlation called the *cross-variance* is denoted by σ_{XY}^2 and is given by

$$\sigma_{XY}^2 = E\left[\sum_{I=1}^{N}(X_I - E(X))(Y_I - E(Y))\right]$$

The subscript XY emphasizes that this is the product of two measures of distance from the mean. Because of our definition, the variance is also called the *covariance,* because it is a special type of cross variance, one that is crossed with itself.

3.2.3 Examples In a computer system, what type of data might be aggregated into these statistics?

- The fraction of time terminals four, seven, and thirteen are busy submitting work.
- The fraction of time terminals one, five, and eight are busy submitting work *and* the system is busy executing jobs, to get a quantitative measure of *concurrency.*
- The mean response time at terminal two
- The standard deviation and squared coefficient of variation of response times for terminal five

3.2.4 Distributions The final type of statistic we will examine is the fraction of time a given event takes place, which is also the *distribution* of an event. For example, we might ask what fraction of data are less than a given threshold:

$$PROB[X_I \le Y] = fraction\ of\ data\ less\ than\ or\ equal\ Y$$

One example is the datum for which half the data are bigger and half are smaller, called the *median.* This would be similar to an *average* or first order statistic.

A second example is the fraction of time X per cent of the data is below a given point, such as the twenty fifth percentile (also called the first quartile) and denoted by $P_{0.25}$, or seventy fifth percentile (also called the third quartile) and denoted by $P_{0.75}$. The difference between the twenty fifth and seventy fifth percentile would be a measure of spread or fluctuation about the *median.*

A third example would be the *minimum* or *maximum.* These show the *worst* or *most atypical* pieces of data. Often these uncover errors in underlying software and hardware, and should be in every measurement package.

A fourth example would be the fifth percentile, denoted $P_{0.05}$, or the ninety fifth percentile, denoted $P_{0.95}$. These also give a measure of the *extremes* or *atypical* patterns present in the data. On the other hand, because the final five per cent of the data is being rejected or trimmed (in this sense of asking where ninety five per cent of the data lies), this statistic can test the *sensitivity* or *robustness* to extreme values of the minimum and maximum. This is like an insurance policy, that checks to see if the minimum and maximum really are telling what we think they are. Similar statements hold for the tenth percentile, $P_{0.10}$, and the ninetieth percentile, $P_{0.90}$.

All of these measures are summarized in the figure below:

Figure 3.1.Statistical Measures

3.2.5 Sensitivity of Statistics to Assumptions Certain statistics are quite sensitive to two different phenomena:

- A small number of values may be quite different from the majority of the data, and distort the statistic
- Most of the data may be slightly off from the exact true measure, due to timing or rounding

We wish to have quantitative measures as well as heuristic measures to test to see if a statistic is in fact *robust* or insensitive to either phenomenon.

In our example in the previous chapter, ten jobs had to be printed, with one job requiring ten thousand lines to be printed, four jobs requiring two thousand lines each to be printed, and five jobs requiring one thousand lines each to be printed. The mean or average number of lines per job was

$$mean\ number\ of\ lines\ per\ job = \frac{1}{10}\sum_{K=1}^{10} L_K = 2{,}300\ lines$$

On the other hand, the median is simply

$$P_{0.50} = median = \tfrac{1}{2}(1{,}000 + 2{,}000)\ lines = 1{,}500\ lines$$

and nine of the ten jobs are within five hundred lines of the median.

A different way of seeing this is to deal with a *trimmed* mean, where we discard an equal number of extreme values. This tests the *sensitivity* of our analysis to extreme values. If we discard the highest and lowest data, we find

$$trimmed\ mean = \frac{1}{8}\sum_{K=1}^{8} L_K = 1{,}500\ lines$$

which is much smaller than the total mean. If we trim two more extreme values off, we find

$$trimmed\ mean = \frac{1}{6}\sum_{K=1}^{6} L_K = 1{,}500\ lines$$

and hence this is stable or robust to trimming thresholds.

What about fluctuations about the mean? The variance about this mean is given by

$$\sigma_L^2 = \frac{1}{10} \sum_{K=1}^{10} [L_K - E(L)]^2 = 6{,}810{,}000 \ lines^2 \quad \sigma_L = 2{,}610 \ lines$$

Note that the standard deviation, which is one measure of fluctuation about the mean, is *larger* than the data for all jobs but one. Again, this suggests the job that requires much more processing than the other jobs is causing these problems. This can be seen by using the difference between the twenty fifth and seventy fifth percentiles as a measure of spread about the mean:

$$P_{0.75} = 2{,}000 \ lines \quad P_{0.25} = 1{,}000 \ lines \quad P_{0.75} - P_{0.25} = 1{,}000 \ lines$$

EXERCISE: Can you think of any more tests for showing if data is anomalous?

EXERCISE: Can you construct a flowchart for automating this?

3.3 Measurement Criteria

Earlier we observed that there are two types of measurement criteria: first, user oriented criteria, such as delay statistics for different transaction types, with examples being

- Arrival time to start of service, or waiting time
- Arrival time to end of service, or queueing time
- Arrival time to end of execution of a given step of a job
- Completion time minus deadline, or lateness
- The larger of zero and lateness, or tardiness

Second, system oriented criteria, such as utilization of different system resources with examples being

- Fraction of time a given single resource is busy, e.g., the fraction of time a processor is busy, or the fraction of time a disk is busy
- Fraction of time two given resources are busy, e.g., the fraction of time both the processor and the disk are simultaneously busy
- Fraction of time three given resources are busy, e.g., the fraction of time the processor, the disk, and the terminal input/output handler are simultaneously busy
- Mean rate of execution of a given type of job, or mean throughput rate

Measurements should be designed to directly measure criteria such as these, so that a direct assessment can be made: is the system in fact meeting its goals? what goals are reasonable?

The majority of measurement tools in the field at the present time are system oriented; it is quite rare to directly measure delay statistics and throughput rates for different transaction types. Why is this so? In some systems, the load placed on the system software to gather data will be comparable (or greater than!) the load we wish to measure; this is dependent on the software and hardware technology employed, and may well change in the future.

3.3.1 Additional Reading

[1] D.Ferrari, G.Serazzi, A.Zeigner, **Measurement and Tuning of Computer Systems,** Prentice-Hall, Englewood Cliffs, NJ, 1983.

3.4 An Example: Report Generation on Two Printers

As an example, consider the statistics associated with printing a set of reports using two printers attached to a computer system. First, we summarize the reports and how long each will take to print, plus the desired time interval or *window* we can tolerate for printing each report:

All reports arrive at once, i.e., all reports are ready to be executed at time *zero.* The figure below shows the completion times for one schedule where the priority is chosen according to the job letter: job A

Table 3.1.Report Execution Time Summary

Report	*Print Time*	*Window Time*
A	3 minutes	5 minutes
B	2 minutes	3 minutes
C	10 minutes	15 minutes
D	2 minute	4 minutes
E	1 minutes	2 minutes

Figure 3.2.Printer Activity Using Alphabetic Priorities

has higher priority than job B, and so forth.

All jobs are completely executed at twelve minutes past zero. For each job, we can tabulate delay statistics, and find:

Table 3.2.Alphabetic Priority Delay Statistics

Report	*Completion*	*Waiting*	*Queueing*	*Lateness*	*Tardiness*
A	3	0	3	-2	0
B	2	0	2	-1	0
C	12	2	12	-3	0
D	5	3	5	1	1
E	6	5	6	4	4

A variety of companion statistical measures, averaged over all jobs, are summarized below:

Table 3.3.Alphabetic Priority Job Performance Measures

Statistic	*Completion*	*Waiting*	*Queueing*	*Lateness*	*Tardiness*
Average	5.6	2.0	5.6	-0.2	1.0
σ	3.5	1.9	3.5	2.5	1.6
$P_{0.75}-P_{0.25}$	2.0	3.0	2.0	3.0	1.0
Minimum	2	0	2	-3	0
Maximum	12	5	12	4	4

Here is one interpretation of this summary: The average time in system consists of an average waiting time of 2 plus an average service time of 3.6, equaling the average time in system of 5.6. The maximum tardiness is four, while the average tardiness is one. Inspection reveals that twice the standard deviation does *not* approximately equal the difference between the third and first quartile, which suggests that the one long job has significantly skewed the statistics.

What about system oriented performance criteria? These are summarized below:

Table 3.4.Alphabetic Priority System Performance Measures

Statistic	*Value*
Mean Throughput	5 jobs in 12 minutes
Fraction of Time Printer 1 Busy	50.0%
Fraction of Time Printer 2 Busy	100.0%
Fraction of Time Both Printers Busy	50.0%

The lack of balance in utilization between the two printers makes it evident that the one long job has significantly skewed the utilizations away from equality.

In contrast to this, we assign priorities such that the shorter the processing time of a job, the higher its priority. This attempts to let the short jobs run quickly at the expense of the long job that will take a long time anyway. The execution pattern for this schedule is shown in Figure 3.3.

Figure 3.3.Shortest Processing Time Priority Arbitration

As above, the same statistics are tabulated for each job in Table 3.5.

Table 3.5.Shortest Processing Time Priority Delay Statistics

Report	*Completion*	*Waiting*	*Queueing*	*Lateness*	*Tardiness*
A	5	2	5	0	0
B	2	0	2	-1	0
C	13	3	13	-2	0
D	3	1	3	-1	0
E	1	0	1	-1	0

As before, the same job oriented statistical performance measures, averaged over all jobs, are summarized below in Table 3.6.

Table 3.6.Shortest Processing Time Job Performance Statistics

Statistic	*Completion*	*Waiting*	*Queueing*	*Lateness*	*Tardiness*
Average	4.8	1.2	4.8	-1	0
σ	4.3	1.2	4.3	0.6	0.0
$P_{0.75}-P_{0.25}$	1.0	2.0	1.0	0.0	0.0
Minimum	1	0	1	-2	0
Maximum	13	3	13	0	0

Here is one interpretation of this data: Each job has a mean waiting time of 1.2 plus a mean service time of 3.6, resulting in a mean queueing time of 4.8. This is smaller than the first priority arbitration rule. As with the alphabetic schedule, the one long job has skewed the standard deviation significantly more than would be expected from the difference of the two quartiles.

What about system oriented performance measures? These are summarized below:

Table 3.7.Shortest Processing Time System Performance Statistics

Statistic	*Value*
Mean Throughput Rate	5 jobs in 13 minutes
Fraction of Time Printer 1 Busy	38.46%
Fraction of Time Printer 2 Busy	100.0%
Fraction of Time Both Printers Busy	38.46%

The one long job has skewed the utilization of the two printers far away from equal loading.

A second point to keep in mind is that *all* five jobs are being printed, but some are printed ahead of others. This means that *logically* all jobs are being printed concurrently (actually only one job at a time can use a printer), but the choice of priorities can control the *delay* or *responsiveness* (waiting time or completion time) of a job, with the amount dependent on the workload. For these two priority arbitration rules, these differences are summarized as follows:

Table 3.8.Job Delay Statistics Summary

Performance Measure	*Alphabetic Priority*	*Shortest Processing Time Priority*
Average Completion	5.6	4.8
Average Waiting	2.0	1.2
Average Queueing	5.6	4.8
Average Lateness	-0.2	-1.0
Average Tardiness	1	0

Judged on these measures alone, the shortest processing time arbitration rules offer superior performance to the alphabetic arbitration rules.

Finally, what about system oriented performance measures:

Table 3.9.System Utilization Statistics Summary

Performance Measure	*Alphabetic Priority*	*Shortest Processing Time Priority*
Mean Throughput Rate	5 jobs/12 min	5 jobs/13 min
Fraction of Time Printer 1 Busy	50.0%	38.46%
Fraction of Time Printer 2 Busy	100.0%	100.0%
Fraction of Time Both Printers Busy	50.0%	38.46%

>From the system operations point of view, it looks like a wash: neither priority scheme offers significant benefits over the other.

3.4.1 Summary Our intent here was to take two printers and five jobs and walk through a data analysis exercise of performance, from the point of view of each job and its delay, and from the point of view of the system and its operations. We did not even begin to exhaust the list of statistics that we could in fact generate for such a *simple* system. Our intent was not to bury you, the reader, with numbers, but rather to show that the key ingredients are understanding the workload (look at how the one long job impacted everything), understanding how the system executed jobs (for different schedules), and to develop techniques for making judgements. The job delay statistics carried the day for shortest processing time scheduling: the system performance statistics appeared to be comparable. Take from this the *methodology* of data analysis, and *not* the conclusions per se.

3.5 Modeling

We impose structure at the outset on the data gathered for two reasons: first, there is simply too much data so findings must be summarized in a small number of parameters; second, there are only a limited number of actions that a performance analyst can suggest. Remember that the possible changes dealt with application code, operating system, and hardware, with different costs associated with each type of change: why not focus at the outset on these three topics? what types of models or structures could we impose on the data at the outset that would suggest fruitful cause and effect changes in system performance?

Here is one such model: different types of transactions require different amounts of hardware resources (terminal processing time, data link transmission time, data base processing time, data retrieval access time, and so forth) and operating system resources (files, tables, messages, processes), and application program resources (input handler, data base manager, scheduler, communications controller). For each transaction we would log or measure the amount of each resource actually used, on a per process basis, i.e., a transaction is executed by a set of processes.

Here is a typical example of a model of a computer communication system. The system resources consist of

Table 3.10.Computer System Resources

Hardware	*Amount*	*Operating System*	*Amount*
Processors	3	Processes	298
Disks	7	Files	365
Links	15	Buffers	149
Disk Controllers	3	Virtual Circuits	487
Terminals	29	Semaphores	732
Terminal Controllers	5		
Tape Drive	1		

How do logical resources arise? They arise from attempting to *control* the flow of *data* through a computer communication system. Each physical and logical resource has a *name.* Each resource must be assigned to a given *location.* There is a policy for interconnecting or *routing* data and control information among the different resources: disks are connected to disk controllers, terminals to terminal controllers, and so forth. Conceptually, the logical resources are tables associated with these entities: these tables must be accessed and contention arbitrated just as with physical resources. The reason logical resources are often ignored is that a system administrator configures the system when it is initially powered up so that these resources *hopefully* will never be bottlenecks. This must be examined on a case by case basis.

Each job will be holding one or more resources at each step of execution. One possible representation of state of the system at any instant of time is what jobs hold what resources. A second possible state representation is what jobs are holding what resources and are queued waiting for other resources to become available. One system statistic related to this is the fraction of time each resource is busy and idle. One job statistic related to this is to determine what processes are busy and idle.

EXERCISE: What about the workload imposed on this system? How does this impact the measurements?

3.5.1 Additional Reading

[1] G.E.P.Box, W.G.Hunter, J.S.Hunter, **Statistics for Experimenters,** Wiley, NY, 1978.

[2] D.Freedman, R.Pisani, R.Purves, **Statistics,** Norton, New York, 1978.

3.6 An Illustrative Example: Directory Assistance Operator Work Times

When directory assistance operators handle queries, two steps are involved: listening to the query, typing it into a terminal, and perhaps the reading the terminal, followed by waiting for the response to the query. What is the distribution of time to listen and enter the query?

Two stages are involved in answering this question: examining a variety of nonparametric statistics which suggest naturally what distributions are reasonable candidates for what models, and a model fitting.

3.6.1 Description of the Data The times for a directory assistance operator to answer a query were measured. This was done by a stop watch and the results recorded on a paper by pencil. The 597 data were aggregated into classes or bins, with the total width of the bin being five seconds, beginning with zero. A visual inspection of the data gathered did not reveal any apparent outlying or spurious or erroneous values. The data are summarized below:

3.6.2 Nonparametric Exploratory Data Analysis The figure below shows an empirical distribution for the data, while the next figure shows a histogram. The table below summarizes a variety of statistics computed from the observations:

Table 3.11.Directory Assistance Operator Work Time Histogram

Number of Observations		*Total*	*Number of Observations*		*Total*
Greater Than	*But Less Than*	*Observed*	*Greater Than*	*But Less Than*	*Observed*
0 sec	5 sec	0	50 sec	55 sec	16
5 sec	10 sec	0	55 sec	60 sec	16
10 sec	15 sec	18	60 sec	65 sec	14
15 sec	20 sec	85	65 sec	70 sec	6
20 sec	25 sec	112	70 sec	75 sec	4
25 sec	30 sec	110	75 sec	80 sec	6
30 sec	35 sec	82	80 sec	85 sec	4
35 sec	40 sec	51	85 sec	90 sec	2
40 sec	45 sec	36	90 sec	95 sec	2
45 sec	50 sec	30	95 sec	---	3

Figure 3.4.Empirical Distribution Function

Figure 3.5.Histogram of Directory Assistance Operator Work Time Data

Table 3.12.Moment Estimates

Mean	32.77 sec
Variance	238.99 sec^2
Variance/(Mean^2)	0.2224

On the other hand, the percentile order statistics are summarized below:

Table 3.13.Order Statistics

1%	12.5 sec	*75%*	37.5 sec
5%	17.5 sec	*90%*	52.5 sec
10%	17.5 sec	*95%*	62.5 sec
25%	22.5 sec	*99%*	87.5 sec
50%	27.5 sec		

It is of interest to compare the square root of the variance, or the standard deviation, with the difference between the twenty fifth and seventy fifth percentiles:

$$standard\ \ deviation = 15.46 \text{ sec} \quad 75th\ \ to\ 25th\ \ percentile = 15 \text{ sec}$$

This shows that the standard deviation significantly *underestimates* the actual fluctuation about the mean compared with the differences in the third and first quartiles.

3.6.3 Model Fitting We choose to fit the data with a gamma distribution with integer parameter. These are also called *Erlang-K* distributions, so called because each distribution is the K fold convolution of identical exponential distributions. If we let $G(X)$ denote the fraction of time an observation is less than or equal to X, then its moment generating function is given by

$$\hat{G}(z) = \int_0^\infty e^{-zX} dG(X) = \left[\frac{1}{1+\tau z}\right]^K$$

The first two moments of this distribution are given by

$$\int_0^\infty X dG(X) = K\tau$$

$$\int_0^\infty X^2 dG(X) = K(K+1)\tau^2$$

We have already computed the first two moments of this distribution, so let's see what type of distribution might fit these moments. One way to investigate this is to examine the *ratio* of the variance to the mean squared. This is dimensionless, and in fact equals the reciprocal of the degree of the distribution:

$$\frac{variance}{(mean)^2} = \frac{1}{K}$$

The data suggests this ratio is 0.224, so an Erlang-4 or Erlang-5 might be an adequate fit.

A quantile quantile plot is used to fit the data to the model. On the horizontal axis sorted observations are plotted. For each observation, there is a corresponding quantile Q, i.e., that fraction of the data that is less than or equal to that observation. We now ask what quantile in the model distribution yields the same value of Q, and choose that for the vertical axis. If the model quantiles and observed quantiles are identical, the plot should be a straight line with slope unity. We will tune the model parameters until we have a visual straight line fit!

The figures below show different quantile quantile plots for an Erlang K distribution, where K=1,2,3,4,5.

Figure 3.6.Erlang 1 vs Data Q-Q Plot

As is evident, an Erlang 1 or exponential distribution does *not* match the data very well.

Figure 3.7.Erlang 2 vs Data Q-Q Plot

An Erlang 2 does much better than an Erlang 1 at fitting the data, but the match is still not very good.

Figure 3.8.Erlang 3 vs Data Q-Q Plot

An Erlang 3 appears to do an excellent job of fitting the data.

Figure 3.9.Erlang 4 vs Data Q-Q Plot

An Erlang 4 does not do as well as an Erlang 3 at fitting the data.

Figure 3.10.Erlang 5 vs Data Q-Q Plot

An Erlang 5 does worse than an Erlang 4 in fitting the data.

The Erlang 3 comes closest to fitting the data over the *entire* range.

One way to quantify goodness of fit of model to data besides eye ball is to check an error measure. We define *local* error as $e_K = | q_{model,K} - q_{data,K} |$ where $q_{model,K}$ is the Kth model quantile, and $q_{data,K}$ is the Kth data quantile, and K=1,...,N. The first measure we will use is the mean absolute deviation of data from the model, denoted E_1:

$$E_1 = \frac{1}{N} \sum_{K=1}^{N} e_K$$

The second measure is the square root of the mean square deviation, denoted E_2:

$$E_2 = \left[\frac{1}{N} \sum_{K=1}^{N} e_K^2 \right]^{1/2}$$

The third measure is the maximum deviation of the data from the model, denoted $E_{\max}$:

$$E_{\max} = \max_{K=1,\ldots,N} e_K$$

The table below summarizes the calculations for each measure of error:

Table 3.14.Error Criterion Summary

Model	E_1	E_2	$E_{\max}$
Erlang 1	29.5	38.1	81.0
Erlang 2	15.6	17.5	31.3
Erlang 3	4.0	5.2	11.1
Erlang 4	8.3	10.2	18.4
Erlang 5	13.1	16.5	30.3

This suggests the Erlang 3 is the best model, confirming the graphical test.

3.6.4 Additional Reading

[1] M.B.Wilk, R.Gnanadesikan, *Probability Plotting Methods for the Analysis of Data,* Biometrika, **55,** 1-17 (1968).

3.7 Measurement Tools

There are two types of tools for measuring system traffic handling characteristics, hardware monitors and software monitors.

3.7.1 Hardware Monitors A hardware monitor might consist of a set of high impedance probes (much as are found in high performance oscilloscopes), a logic plugboard, a set of counters, and a display or recorder of some sort. The probes are attached to memory locations, registers, bus connectors, and so forth, as wished, and interface to the logic plugboard. The logic plugboard is used to count the number of transitions of a given type that a probe measures: number of clock cycles that the processor or disk controller is busy, number of clock cycles that both the processor and disk controller are busy, and so forth. The display or recorder presents this information to a human in a useful manner, or records it for processing elsewhere. One trend today is to build the hardware performance monitoring capabilities directly into the computer communication system hardware, and to allow remote diagnostic capabilities to also be built in. The value of this in field service and support can be immense: a remote site can notify a central repair bureau site of a failure (either transient or permanent), the central site with trained personnel can carry out specialized tests, and take an appropriate action, often without intervention of the remote site personnel.

3.7.2 Software Monitors A software monitor makes use of physical resources, unlike the hardware monitor, and thus can place some load on the system. How much of a load depends upon a variety of factors: how frequently is the monitor invoked, how many resources are consumed per invocation, and so forth. In practice the load placed on any one device by the measurement tool should be negligible, but in point of fact for current systems it is difficult to drop the measurement load utilization below ten per cent. On the other hand, a great deal of flexibility is gained in software versus hardware monitoring, and the timeliness of the information can be invaluable. One approach to the problem of measurement load is to simply insert software modules at the outset of system design that consume resources but do no useful work: as the system software monitor is installed, more and more of these extraneous modules are removed, and no one knows the difference!

3.7.3 Putting It All Together How do we use these tools? First, we must start up the system with a controlled load, and then determine that the measurements being gathered have stabilized. Second, we must examine the data to determine if the system behavior is stationary: do certain events occur at five or ten or sixty minute time intervals that will impact performance? Put differently, when do we stop the measurements? The measurements can be either event driven, i.e., gathered only at those time instants when certain events have occurred, e.g., after one thousand transactions have been processed, or clock driven, i.e., gathered at evenly spaced time instants. The load placed on the system by event driven monitors is often much greater than clock driven monitors, but the event driven monitor may be more useful and flexible for interpreting data.

3.7.4 Example Here is an illustrative set of measurements that might be gathered on a computer communication system:

- Processors--Busy (application, monitor, system) or idle (blocked or true idle), system buffer hit rate, cache hit rate, time from when a process is ready to run until it runs
- Disks--Busy (seeking, transferring) or idle, number of requests per unit time, number of reads, number of writes, number of swaps, number of characters in/out
- Disk Controllers--Busy or idle, number of characters in/out, number of reads, number of writes
- Links--Busy or idle, number of characters in/out, number of reads, number of writes
- Terminals--Busy or idle, number of characters in/out, number of reads, number of writes, time from last character input until first character output
- Terminal controllers--Busy or idle, number of characters in/out, number of reads, number of writes
- Process--Busy or idle (blocked or true idle), number of context switches per unit time, number of characters in/out, number of page faults, number of active pages per unit time, number of messages

in/out, number of files opened/closed, number of reads, number of writes

- Files--Busy or idle (locked or true idle), number of reads, number of writes, number of changes, number of characters in/out

EXERCISE: Can you think of any more?

3.7.5 Additional Reading

[1] D.Ferrari, G.Serazzi, A.Zeigner, **Measurement and Tuning of Computer Systems,** Prentice-Hall, Englewood Cliffs, NJ, 1983.

[2] P.A.W.Lewis, G.S.Shedler, *Empirically Derived Micromodels for Sequences of Page Exceptions,* IBM J.Research and Development, **12,** 86-100 (1973).

[3] S.Sherman, F.Baskett III, J.C.Browne, *Trace-Driven Modeling and Analysis of CPU Scheduling in a Multiprogramming System* Communications of the ACM, **15** (12), 1063-1069 (1972).

[4] D.W.Clark, *Cache Performance in the VAX-11/780,* ACM Transactions on Computer Systems, **1** (1), 24-37 (1983).

3.8 Data Analysis and Interpretation

How do we interpret data? With great care! Data can be aggregated in a variety of ways: the utilization of a given hardware device can be found by simply measuring the total time it is busy and then dividing by the total measurement time interval. What good is this? Bottlenecks, hardware resources that are completely utilized can be spotted quickly, and a variety of secondary avenues can now be explored. Why is the hardware device busy? Is it intrinsic to the task it must do, i.e., more resources of that type are needed? Is it a software error, e.g., a print statement was inadvertently left in the code? Is it a structural problem of the operating system or application code using some logical resource, e.g., a file, again and again when in fact the application or operating system could be changed and this problem would simply not occur? Again, the consequences of each of these avenues must be quantified before choosing one.

How can we aid this process? By graphically plotting utilization of one or more devices versus time: this will show, first of all, which devices are bottlenecks, and second of all the correlation between the utilization of different devices. A picture here is worth everything! Next, we can repeat this procedure for different processes versus time. Does this picture of what is actually taking place match what should take place: is there an error in the software? This approach can help to uncover it.

How do we do this in practice? First, we develop a clear picture of how control and data interact for each step of each job. Second, we examine the utilization of each resource: which resources are close to complete utilization, which are not?

Third, we examine which processes and files are associated with those resources that are generating this load, and attempt to see which of these entities are generating the majority of the load on the bottleneck

Fourth, we ask ourselves if this is reasonable: maybe there is an error, maybe there is no reason for this to be so, yet it is! If this is reasonable, we go to the next step.

Fifth, the evaluation of alternatives, either hardware or operating system or application software. In any event, there is a bottleneck in the system: if it is not acceptable, the bottleneck should be moved to some other resource. If it cannot be moved to some other resource, then perhaps scheduling of the bottleneck to meet delay goals is possible.

3.9 Simulation

What are the stages in doing a simulation?

- Model Formulation--Gathering a precise description of the arrival statistics and resources consumed for each transaction type, along with the policies for ameliorating contention for resources

- Model Implementation--Programming the model in a language (e.g., assembler, BASIC, FORTRAN, GPSS, and so forth) to match the formulation
- Model Validation--Generating controlled loads with known behavior on subsystems or modules, then on aggregations of modules, and finally on the whole system, and matching that behavior against expectations with negligible discrepancy
- Experimental Design--Creating a variety of controlled loads with unknown behavior; often the time required to carry out all experiments can prove prohibitive, and systematic techniques must be employed
- Data Analysis--Displaying the measurements using the tools described earlier in order to gather cause and effect inferences concerning behavior

The figure below shows a time line of a simulation. We note three distinct items there:

- Event--Change in state of system entity
- Activity--Collection of operations that change state of entity
- Process--Sequence of events ordered in time

Figure 3.11.Representative Timing Diagram of a Simulation

In this figure the latter portion of activity I and the first portion of activity II can be executed concurrently, but activity III must be done serially.

The table below summarizes some simulation languages that are in widespread use and the method they use for timing control.

Table 3.15.Examples of Simulation Languages

Event Scheduling	*Activity Scanning*	*Process Interaction*
GASP	CSL	GPSS
SIMSCRIPT		SIMULA

Any simulation must present evidence of

- Stationarity--Are initial start up transients still in evidence? What about shut down transients? How repeatable is the experiment? Or are the transients really what is of interest?

- Variability--Are fluctuations severe? What types of confidence intervals or error brackets are present?
- Correlation--What types of time scales are evident? Do all events occur on the same time scale or are some short relative to others: what impact might this have? Is there coupling between events, activities, processes?

A wealth of information on these topics is found in the reading, and elsewhere.

3.9.1 Additional Reading

[1] G.S.Fishman, **Concepts and Methods in Discrete Event Digital Simulation,** Wiley, NY, 1973.

[2] S.H.Fuller, *Performance Evaluation,* in **Introduction to Computer Architecture,** H.Stone (editor), Science Research Associates, Chicago, Illinois, 1975.

[3] H.Kobayashi, **Modeling and Analysis: An Introduction to System Performance Evaluation Methodology,** Addison Wesley, Reading, Mass., 1978.

3.10 The Structure and Mechanics of Simulation

In order to provide a maximum flexibility, simulation software packages used to aid communication systems analysis and design should have a modular structure. Most of the software packages that are in use now are made up of four major components:

- model library
- system configuration tools
- simulation exercise tools
- post processor tools

The system configuration tools select a set of models of functional blocks from the model library and connect them in the desired topology as specified by the block diagram of the system being designed. Parameters of various functional blocks are specified either when the system is being configured or during the execution of the simulation which is supervised by the simulation exercise tools. Time histories of events at various points in the system are generated and stored by the simulation exercise tools. These time histories are examined by the post processing tools at the conclusion of the simulation run, and performance measures are computed from the time histories. At the end of a simulation, the design engineer uses the post processor output to verify if the performance requirements and design constraints are met. If the design objectives are not met, then several iterations are made until a suitable design is found.

3.10.1 Simulation and Programming Languages Software should be written in a higher level (than assembly language) *programming* language such as FORTRAN, PASCAL or C. Unfortunately, these languages do *not* allow input via block diagrams, which requires a preprocessor *simulation* language description of the system being analyzed or designed. This approach lends itself to portability and frees the user from having to know the details of the underlying operating system and hardware configuration of the simulation facility. For discrete event simulation, three widely used languages are GPSS, SIMSCRIPT and GASP. Each of these are preprocessors to a lower level language. For additional flexibility the simulation software may permit the intermixing of statements written in the simulation and programming languages. This will result in a minor restriction of the free format input of blocks, namely the models will have to appear in the order of their precedence in the description of the simulated system.

3.10.2 Topological Configuration The model configuration tools in the simulation package should permit the design engineer to connect the functional blocks in any desired topological interconnection. While this free topological requirement may complicate the simulation software structure, it provides the maximum flexibility.

3.10.3 Model Library The usefulness of a simulation package depends heavily on the availability of a model library that contains a large number of models of various functional blocks that make up transmission systems and networks. Computer routines that model functional blocks should be reentrant so that functional blocks may be used at multiple and arbitrary instances in the simulation of a transmission system. The model configuration tools should permit unlimited nesting of models such that subsystem models may be built using library models. It is also desirable to make provisions to enable user to write his own models, use it in the simulation directly, and/or enter it into the model library.

3.10.4 Time and Event Driven Simulation A simulator can be designed to be time driven where processing is initiated at every "tick" of the simulation clock, or event driven where processing takes place only when an event of interest (such as the arrival of a message) takes place. For maximum flexibility, provisions should be made for both modes of processing such that some blocks in the system are event driven whereas others could be time driven.

3.10.5 Testing for Stationarity The reason for checking the status of the simulation is to gather a variety of either transient or long term time averaged statistics. If long term time averaged statistics are of interest, then the question of how much data to gather at each sample point, and with what degree of confidence, will influence the simulation parameters. This monitoring feature can be very useful in long Monte-Carlo Simulations. As an aside, we note that just because one million separate pieces of data have been collected does *not* guarantee we have a statistically significant sample: only if the samples are uncorrelated from one another can we say that we have some confidence that the data set may be sufficient.

3.10.6 Post Processor The postprocessing routines are an important part of a simulation package since these routines are the ones that enable the design engineer to view the results of the simulation. The model configuration tools and the simulation exercise tools should be designed to allow a designer to draw direct cause and effect inferences about system operation. As a minimum, the postprocessor package should have routines that perform the functions of common laboratory test equipment, (load generators, profilers of resource utilization for each job step and for time delays of each job step). Statistical analysis routines as well as graphics display routines are also essential.

3.10.7 User Interface Finally, the whole simulation package should be made user friendly. This includes both online and offline documentation and help. Two currently popular approaches here are using menus to trace a tree of commands or actions, and using key words (which appears to offer certain advantages for sophisticated users) by allowing menus to be bypassed if need be. This is currently an area of active research, encompassing novel interactive graphics and mouse controls, among other approaches. The simulation software should also have the following provisions to aid the user: symbolic debugging, run-time diagnostics, parameter checking and online error control for inputting.

3.10.8 Additional Reading

[1] N. R. Adam (editor), *Special Issue on Simulation Modeling and Statistical Computing,* Communications ACM, **24** (4), 1981.

[2] G.W.Furnas, T.K.Landauer, L.M.Gomez, S.T.Dumais, *Statistical Semantics: Analysis of the Potential Performance of Key-Word Information Systems,* Bell System Technical Journal, **62** (6), 1753-1806(1983).

[3] U. Grenander, R. F. Tsao, *Quantitative Methods for Evaluating Computer System Performance:A Review and Proposals,* in **Statistical Computer Performance Evaluation,** W. Freiberger (editor), Academic Press, NY, 1972.

[4] L. Kleinrock, W. E. Naylor, *On Measured Behavior of the ARPA Network,* Processing AFIPS National Computer Conference, **43,** 767-780(1974).

3.11 Simulation of Link Level Flow Control

Here is a case study in doing a simulation study. A transmitter sends packets to a receiver over a channel and gets a positive acknowledgement. A hardware block diagram is shown in the figure below:

Figure 3.12.Communications Link Hardware Block Diagram

Here is a model of the operation of such a system: Each packet is processed by the transmitter, propagates through the channel to the receiver, is processed by the receiver, and leaves. An acknowledgement propagates back to the transmitter, and requires zero transmitter processing time. The receiver can only buffer a maximum of W packets. If the receiver buffer is full, the transmitter is shut down until space is available in the receiver. A queueing network block diagram of this system is shown in Figure 3.13.

Figure 3.13.Communications Link Queueing Network Block Diagram

An example of a simulation of such a system, done in GPSS, is shown below.

The sequence of transmitter and receiver packet processing times is constant. The channel propagation time has a given mean value; the sequence of channel propagation times is constant. The packet interarrival times are independent exponentially distributed random variables with a given mean interarrival time. The remaining parameters are the maximum number of unacknowledged packets transmitted, called the *window* size, denoted W, and the number of acknowledgements required to be received by the transmitter to start sending packets once shut down, called the *batch* size, denoted B. Flow control is used to make sure no messages are lost due to memory or buffers not being available in

```
*LOC   OPERATION   A,B,C,D,E,F,G
*
*  FULL DUPLEX LINK LEVEL WINDOW FLOW CONTROL
*  WITH CHANNEL PROPAGATION DELAYS
*
       SIMULATE
 1      FUNCTION     RN3,C24
0               0               0.1             0.1040.20.222
0.3             0.355           0.4             0.5090.50.69
0.6             0.915           0.7             1.20.751.38
0.8             1.6             0.84            1.830.882.12
0.9             2.3             0.92            2.520.942.81
0.95            2.99            0.96            3.20.973.5
0.98            3.9             0.99            4.60.9955.3
0.998           6.2             0.999           7.00.99978.0
*
* INTERARRIVAL TIMES IID EXPONENTIAL ARRIVALS
* WITH MEAN INTERARRIVAL TIME 280
*
 1              GENERATE        280,FN1
*
*  QUEUE 1--TRANSMITTER QUEUE
*
 2              QUEUE 1
*
* LOW WATER MARK=5
*
 3              TEST E          Q3,K7,5
 4              LOGICS          1
*
* HIGH WATER MARK=7
*
 5              TEST E          Q3,K6,7
 6              LOGICR          1
 7              GATE LR         1
 8      SEIZE   1
 9              DEPART          1
 10             ADVANCE         100
*
* TRANSMITTER PACKET PROCESSING TIME=100
*
 11             RELEASE         1
 12             TABULATE        4
*
*  QUEUE 2--FORWARD AND REVERSE CHANNEL
*
 13             QUEUE           2
 14             SEIZE           2
 15             DEPART          2
 16             ADVANCE         0
*
* CHANNEL PACKET DELAY=0
*
 17             RELEASE         2
 18          TABULATE    5
*
* QUEUE 3--RECEIVER
*
 19          QUEUE       3
 20          ADVANCE     100
*
* RECEIVER PACKET PROCESSING TIME=100
*
 21          SEIZE       3
 22          DEPART      3
 23          RELEASE     3
 24          TABULATE    6
 4           TABLE       M1,100,10,40
 5           TABLE       M1,100,10,10
 6           TABLE       M1,200,50,40
 25          TERMINATE   1
*
* GENERATE 10000 EVENTS
*
             START       1000
```

Figure 3.14.GPSS Link Flow Control Simulation

the receiver. This occurs for a variety of reasons, for example, when there is a mismatch in the speed of the two entities, or when the receiver must handle other work than just packet communication and is effectively slowed, and so forth.

Ideally, controlling or pacing the rate at which the transmitter sends packets should have no impact on performance. Our goal is to study not only mean throughput rate but also packet delay statistics, using this simulation as a tool to understand design tradeoffs.

3.11.1 Nonnegligible Channel Propagation Delay First, the channel propagation delay is set to fifty times that of the processing time required by the transmitter or receiver. This is representative of what might be found in a ground to space satellite in synchronous orbit to ground data link. The first case studied set the window size equal to fifty two packets, W=52. In this regime the transmitter and receiver are the bottlenecks here, not the channel. It was suspected that the larger the window size, the better the delay characteristics of packets. The two cases studied here were identical in their delay characteristics, to within statistical fluctuations! It may well be that the details of the simulation used here versus an actual field system may differ in critical ways that are not reflected here.

The simulation consisted of generating ten groups of packets, with each group comprising one thousand packets. The results were examined for homogeneity, to see if the system had reached statistical steady state. Statistics were gathered for ten groups of one thousand packets each; the observed variability was felt to be within statistical fluctuations, and the uniformity of the statistics across the samples suggested that long term time averaged statistics were in fact meaningful. An initial warmup of one thousand packets was used before gathering statistics. A statistical summary of results is tabulated below:

Table 3.16.Simulation Statistics--Finite Channel Propagation Delay(W=52)

$E(T_{trans})=E(T_{rec})=1$--$E(T_{trans-rec})=E(T_{rec-trans})=25$

Mean Inter-arrival Time	*Mean Ar-rival Rate*	*Mean Delay*	*Standard Deviation*	*90th Delay Percentile*	*95th Delay Percentile*	*Max Packets in Transmitter*
4.00	0.25	50.169	0.375	50.730	50.934	4.3
2.20	0.45	50.443	0.741	51.369	51.927	6.6
1.80	0.56	50.654	0.985	51.855	52.584	7.9
1.50	0.67	51.056	1.402	52.785	53.805	9.9
1.30	0.77	51.748	2.053	54.625	55.955	12.4
1.10	0.91	55.140	4.564	59.314	61.593	21.3

Effectively, the system performance is dominated by the transmitter which is holding more and more packets as the load increases; this is also clear from the simulation source code. On the other hand, this may be acceptable engineering practice.

3.11.2 Negligible Channel Propagation Delay Next, the time spent in propagation from the transmitter to the receiver and back again is assumed to be negligible compared to the transmitter and receiver packet processing times. For this case, since we only have two resources, a transmitter and a receiver, we only investigate two cases: W=1 so that one packet at a time is handled, and W=2 so that two packets at a time are handled, offering the hope for keeping both the transmitter and the receiver busy, and ideally doubling the mean throughput rate for the same or better delay statistics.

Table 3.17.First Two Moments of Packet Delay Statistics

Mean Interarrival Time	*Mean Arrival Rate*	*W=1*		*W=2*	
		Mean Delay	*Standard Deviation*	*Mean Delay*	*Standard Deviation*
6.67	0.15	3.366	4.085	2.088	0.257
4.00	0.25	3.378	3.453	2.169	0.375
2.80	0.36	4.248	3.194	2.292	0.546
2.20	0.45	4.983	3.316	2.442	0.741
1.70	0.59	--	--	2.745	1.080
1.30	0.77	--	--	3.748	2.052
1.10	0.91	--	--	7.141	4.562

Table 3.18.Percentiles of Packet Delay Statistics

Mean Interarrival Time	*Mean Arrival Rate*	*W=1*		*W=2*	
		90th Percentile	*95th Percentile*	*90th Percentile*	*95th Percentile*
6.67	0.15	6.395	10.610	2.393	2.722
4.00	0.25	8.145	10.635	2.710	2.930
2.80	0.36	8.670	10.675	2.915	3.400
2.20	0.45	9.615	11.295	3.365	3.955
1.70	0.59	--	--	4.025	4.820
1.30	0.77	--	--	6.585	7.965
1.10	0.91	--	--	11.300	13.557

The above tables suggest that, to within statistical fluctuations, the delay characteristics for W=2 and W=1 appear to differ greatly. Furthermore, the delay statistics for W=2 and W=7 (which are not presented here) appear to be virtually identical, to within fluctuations. The main difference in the delay characteristics for the double buffering W=2 and W=7 case was the duration of the flow control startup transient in each: as congestion rises, both systems will always be started up, and the impact of this transient is apparently negligible for the numbers investigated above.

3.11.3 Additional Reading

[1] B.Efron, *Biased Versus Unbiased Estimation,* Advances in Mathematics, **16,** 259-277 (1975).

[2] S.H.Fuller, *Performance Evaluation,* in **Introduction to Computer Architecture,** H.Stone (editor), Science Research Associates, Chicago, 1975.

[3] L.Kleinrock, *On Flow Control,* Proc.Int.Conf.Communications, Toronto, Canada, 27.2-1--27.2-5, June 1978.

[4] L.Kleinrock, W.E.Naylor, *On Measured Behavior of the ARPA Network,* Proceedings AFIPS National Computer Conference, **43,** 767-780(1974).

[5] D.E.Knuth, *Verification of Link-Level Protocols,* BIT, **21,** 31-36 (1981).

[6] G.W.R.Luderer, H.Che, W.T.Marshall, *A Virtual Circuit Switch as the Basis for Distributed Systems,* Journal of Telecommunication Networks, **1,** 147-160 (1982).

[7] G.W.R.Luderer H.Che, J.P.Haggerty, P.A.Kirslis, W.T.Marshall, *A Distributed UNIX System Based on a Virtual Circuit Switch,* ACM Operating Systems Review, **15** (5), 160-168 (8th Symposium on Operating Systems Principles, Asilomar, 14-16 December 1981), ACM 534810.

[8] C.A.Sunshine, *Factors in Interprocess Communication Protocol Efficiency for Computer Networks,* AFIP NCC, pp.571-576(1976).

[9] K.C.Traynham, R.F.Steen, *SDLC and BSC on Satellite Links: A Performance Comparison,* Computer Communication Review, 1977

Problems

1) The figure below is the cumulative empirical distribution function for the interarrival times of twenty jobs submitted to an online computer system. The data was measured in time units of milliseconds, and was sorted smallest to largest:

Table 3.19.Interarrival Time Sorted Data (x1000)

6	6	7	9	15
24	29	32	37	39
41	42	42	68	83
84	88	97	116	134

Figure 3.15.Interarrival Time Empirical Cumulative Distribution Function

A. What is the mean and median of this data?

B. What is the standard deviation and $P_{0.25}$ first quartile and $P_{0.75}$ third quartile? What is the ratio of the variance to the square of the mean?

C. What is the fifth percentile $P_{0.05}$? What is the tenth percentile $P_{0.10}$? What is the ninetieth percentile $P_{0.90}$? What is the ninety fifth percentile $P_{0.95}$?

D. What are the minimum and maximum data values?

E. **BONUS:** Construct a quantile quantile plot versus an exponential or Erlang-1 distribution. How good is the fit?

2) An online transaction processing system has the following hardware configuration:

Table 3.20.Hardware Configuration

Processor	1
Memory	4 Megabytes
Disks	2
Asynchronous Ports	32

Each disk has its own controller. The two disks are identical with one used for swapping, and the other used for retrieving text and data. Specification sheets for the disk state:

Table 3.21.Disk Specification

Velocity	3600 revolutions/minute
Transfer Rate	4 Megabytes/second
Time to Move Head	
One Adjacent Track	5 milliseconds
Time to Move Head	
Full Sweep	100 milliseconds
Average Seek Time	30 milliseconds

The operating system maintains a cache of buffers of fixed size (1024 bytes): programs first check the buffers to see if required data is there before making any physical disk accesses. All disk accesses involve transferring 1024 bytes of data per access.

The following measurements are carried out on the system in field operation during a peak busy hour:

[1] Processor measurements

- Nineteen per cent of the time spent executing application programs
- Fifty five per cent of the time spent executing operating system code
- Seventeen per cent of the time idle waiting for input/output to complete
- Nine per cent of the time idle with no work whatsoever

[2] Memory measurements

- Static text plus data requires one quarter of a megabyte
- With a desired peak busy hour load on the system, the amount of memory occupied dynamically is 1.86 megabytes

[3] File system measurements

- Two hundred twenty four logical reads per second, with a sixty nine per cent cache hit rate (i.e., sixty nine per cent of the time the logical read did *not* generate a physical read to secondary storage)
- Seventeen logical writes per second, with a sixty seven per cent cache hit rate (i.e., sixty seven per cent of the time the logical write did *not* require the entity to be retrieved from secondary storage first, because it already was resident in main memory in a buffer; this will still generate *one* physical write to secondary storage after the system buffer contents are modified)
- Ten directory blocks per second were accessed
- Eleven accesses to file system tables per second were made
- Three name queries to the file system per second were made

[4] Disk activity measurements

- The swap disk was busy either accessing data or transferring data to and from main memory seventy three per cent of the time; the text and data disk was busy sixty two per cent of the time
- The swap disk averaged forty accesses per second (reads and writes); the text and data disk averaged thirty nine accesses per second (reads and writes).

[5] Asynchronous terminal measurements

- Twenty four characters per second were received by all the different ports
- Two characters per second were transmitted by all the different ports

[6] Operating system measurements

- One hundred seventy four system calls per second were made
- Fifty one system read calls per second were made
- Four system write calls per second were made
- One process per second was created or killed
- Roughly one semaphore per second was set, unlocked, or tested
- The mean number of jobs in the run queue was three
- The fraction of time the run queue contained at least one job was seventy eight per cent
- The mean number of jobs in the swap queue was three
- The fraction of time the swap queue contained at least one job was forty six per cent
- The maximum size of the process table was eighty eight out of a maximum allowable number of two hundred
- The maximum number of simultaneously open files was one hundred thirty five out of a maximum of six hundred

[7] Swap statistics

- Roughly one and a half swaps into main memory occurred every second
- Roughly one and a half swaps out of main memory occurred every second
- One hundred and two process context switches occurred every second

Answer the following questions:

a. What are the system resources?

b. What is the state of the system at any given instant of time?

c. Which if any resource is near complete utilization?

d. What can be done to reduce congestion?

3) Additional measurements are carried out on the above system. The asynchronous ports are used to poll other computer systems. The number of ports used to poll was varied, and the duration of the polling process for a fixed number of computer systems was measured:

Table 3.22.Controlled Experiment Summary

Ports Polled	*Poll Duration (Hours)*	*Processor Utilization*	*Swap Disk Utilization*	*Data Disk Utilization*
4	5.5	0.58	0.35	0.15
6	3.6	0.74	0.45	0.20
8	2.9	0.82	0.55	0.25
10	2.9+	0.93	0.70	0.35
12	2.9+	0.97	0.90	0.35
14	2.9+	0.98	0.92	0.35
16	2.9+	0.99	0.92	0.37
18	2.9+	0.99	0.93	0.38
20	2.9+	0.99	0.93	0.38

Utilization refers to the fraction of time the processor and disks are busy doing anything whatsoever.

a. As the number of ports increases from four, the mean throughput rate goes up. Why?

b. What resource is reaching complete utilization?

c. What can be done to reduce congestion?

3) A batch processing system has the following hardware configuration:

Table 3.23.Hardware Configuration

Processor	1
Memory	2 Megabytes
Disks	4
Asynchronous Ports	32

Memory is sufficent to hold all application programs and operating system code with no swapping. The four disks are identical, with one storing system and application code, and the other three data. Specification sheets for the disk and controller state that the disk spins at 3600 revolutions per minute, can transfer data at a maximum rate of four million bytes per second, requires five milliseconds to move the head from one track to an adjacent track, and requires an average of thirty milliseconds to move the head from one point to any other point on the disk. The operating system maintains a cache of fixed size 512 byte buffers: programs first check the buffers to see if required data is there before making any physical disk accesses. All disk accesses involve transferring 512 bytes of data per access.

The following measurements are carried out on the system in field operation during a peak busy hour:

[1] Processor measurements

- Thirty seven per cent of the time spent executing application programs
- Fifty seven per cent of the time spent executing operating system code
- Six per cent of the time idle waiting for input/output to complete
- Zero per cent of the time idle with no work whatsoever

[2] File system measurements

- Thirty seven logical reads per second, with a seventy one per cent cache hit rate (i.e., seventy one per cent of the time the logical read did *not* generate a physical read to secondary storage)
- Six logical writes per second, with a sixty five per cent cache hit rate (i.e., sixty five per cent of the time the logical write did *not* require the entity to be retrieved from secondary storage first, because it already was resident in main memory in a buffer; this will still generate *one* physical write to secondary storage after the system buffer contents are modified)
- Forty two directory blocks per second were accessed
- Seventeen accesses to file system tables per second were made
- Seven name queries to the file system per second were made

[3] Disk activity measurements

Table 3.24.Disk Statistics Summary

Spindle	*Utilization*	*Accesses/Sec*	*Mean Waiting Time*
1	41%	56	26 msec
2	41%	30	27 msec
3	9%	6	27 msec
4	25%	11	28 msec

[4] Asynchronous terminal measurements

- Seven thousand, two hundred and forty nine characters per second were received by all the different ports
- One thousand, eight hundred and thirteen characters per second were transmitted by all the different ports

[5] Operating system measurements

- Seventy three system calls per second were made
- Forty one system read calls per second were made
- Five system write calls per second were made
- One process per second was created or killed
- Roughly one semaphore per second was set, unlocked, or tested
- The mean number of jobs in the run queue was twenty one
- The fraction of time the run queue contained at least one job was ninety nine per cent
- The maximum size of the process table was thirty two out of a maximum allowable number of sixty nine
- The maximum number of simultaneously open files was two hundred eighty one out of a maximum of four hundred and fourteen

Answer the following questions?

a. What are the system resources?

b. What is the state of the system at any given instant of time?

c. Which if any resource is near complete utilization?

d. What can be done to reduce congestion?

4) A *full* duplex link is to be simulated. This consists of two *half* duplex or one way noiseless links, connected to a transmitter and receiver, with a dedicated microprocessor and a quarter of a megabyte of memory for the transmitter and receiver at each end of the link. Each physical link can have a maximum of five hundred and twelve virtual circuits active at any one time. The control data associated with each virtual circuit requires thirty two bytes of storage. Each virtual circuit can have a maximum number of one, two, or seven data packets unacknowledged at any one time. Control packets are used to set up virtual circuits, to acknowledge successful reception of packets, and to take down virtual circuits. Data packets are used to handle data only. The processing time and storage for each type of packet is summarized below:

Table 3.25.Packet Resources

Type of Packet	*Processing*	*Storage*
Control Packet/VC Set Up	50 msec	32 bytes
Control Packet/ACK	5 msec	32 bytes
Control Packet/VC Take Down	25 msec	32 bytes
Data Packet	100 msec	4096 bytes

A. Draw a block diagram of the hardware of this system.

B. Draw a block diagram of the steps with no contention for resources required to set up a virtual circuit, transmit a message over the link in packets, and take down the virtual circuit.

C. Control packets must be processed within one control packet transmission time. Data packets must be processed within ten data packet transmission times. Compare the performance of a static priority preemptive resume schedule with a deadline schedule assuming that there is *always* a data message consisting of five packets waiting to be transmitted.

6) The performance of a local area network is to be simulated. The hardware consists of *S* stations connected to a common coaxial cable or bus. Each station can send and receive packets. The stations are polled in turn to see if a packet is ready for transmission: if it is, it is transmitted, and all stations receive it. The total length of the network is such that it takes twenty microseconds for a signal to propagate from one end to the other of the cable. The stations are physically equidistant from one another on the cable. The electronics at each station require five microseconds to process signals, irrespective of whether a packet is ready for transmission or not. Each packet requires one hundred microseconds to be transmitted.

A. Draw a hardware block diagram of the system

B. Draw a flowchart showing the operation of each station

C. Code a simulation that accurately models the physical propagation characteristics of the problem, and test it for two cases

- One station only always has a packet to transmit
- Every station always has a packet to transmit

D. Suppose the packet transmission time is reduced to ten microseconds: what changes?

7) You telephone a colleague, the colleague is not at the telephone, and a secretary takes your message asking your colleague to return your call. Later, your colleague gets your message and telephones, but now you are not in, and a secretary takes the message that your call has been returned. You call back, and the process repeats itself, until eventually you both talk to one another via telephone. This is called *telephone tag* because you and your colleague are tagging one another with telephone messages.

A. Draw a figure summarizing the flow of messages.

B. What are the resources held at each step?

C. Write a simulation for this problem. Test it using the following numbers: each secretary can take one message every two minutes; thirty minutes elapse from when you are called and you pick up your message; each telephone call is one quarter minute to leave a message with the secretary and ten minutes when you finally talk plus each telephone call generates two tags or messages.

D. What if a voice message service is installed, so there is no need for telephone tag. This means that your message is stored for retrieval at the convenience of the callee. How does the above simulation have to be changed?

7) A paging memory management subsystem of an operating system is to be simulated. Hardware memory addressing allows the physical location of each piece of a given program to be scattered throughout main memory in units called *pages.* Logically the program appears to be resident in one contiguous area of memory. A typical program first will initialize some data and then execute data processing activities in tightly coded chunks of code that can be contained in one page before generating output that is properly formatted. A typical program will stay within a page for one hundred machine instructions on the average before switching to another page. The process repeats itself until the program finishes execution. The degree of multiprogramming refers to the number of simultaneous active programs contending for main memory.

A. Construct a flowchart showing the memory reference patterns of a typical program

B. Suppose that all programs are identical and consist of P pages each. Code a simulation that will estimate the length of time it takes to completely execute one thousand programs, assuming there is sufficient memory to hold S programs, using the following numbers:

Table 3.26.Numerical Parameters

Case I	P=10,S=10
Case II	P=100,S=1
Case III	P=10,S=100

C. What is gained by using paging versus simply loading programs into a contiguous area of memory from the point of view of performance?

9) A single processor must execute a given workload of jobs. Each job has three steps: interrupt handling, job execution, and cleanup. Interrupt handling and cleanup make use of serially reusable operating system tables, and cannot be preempted. Job execution can be preempted by more urgent work. Two types of jobs are executed by this system, one urgent type that has an allowable execution time window of ten milliseconds, and one less urgent type that has an allowable execution time window of fifty milliseconds. The interrupt handling is identical for each job and is one millisecond. The cleanup is identical for each job and is one half millisecond. The urgent job has an execution time of two milliseconds, the less urgent has an execution time of twenty milliseconds. The less urgent job can be preempted by the more urgent job and by interrupts and cleanup.

A. Draw a flowchart showing the steps required by each type of job

B. Code a simulation for this problem with the following two scheduling policies

- A deadline scheduler
- A static priority preemptive resume scheduler

Compare the simulation results. Which scheduler appears to be superior? Why?

10) One approach to speeding up performance of a computer is to attach a small *scratchpad* or *cache* of memory that is much higher speed (typically a factor of four) than main memory. The processor will load text and data into the cache, and if the cache is well designed, most of the time the next instruction to be executed will be in the cache, effectively speeding up the processor.

A. Draw a flowchart showing the flow of control and data in a program.

B. What data must be gathered as input to the simulation?

C. What types of policies can be implemented for the cache operations?

11) The performance of a telephone switching system is to be simulated. The following information is available about telephone call processing: the hardware configuration consists of one hundred telephones connected fifty each to one of two local telephone switching systems with two trunks between the local telephone switching systems. Each telephone makes one call every ten minutes on the average, and the call offhook interarrival times are independent identically distributed exponential random variables. Each telephone caller is equally likely to call any other telephone user. The talking duration of each telephone call is a sequence of independent identically distributed exponential random variables with a mean of five minutes.

The steps involved in telephone call processing are shown in the figure below:

Figure 3.16.Offhook and Dial Tone Generation Steps

Assume that there is one terminal handler at each local switching system that requires ten milliseconds to detect offhook. Furthermore, assume that each local switching system has four processors for call processing, and one dial tone generator.

The steps involved in telephone call digit dialing are shown in Figure 3.17.

Figure 3.17.Digit Dialing and Billing

Assume that digit dialing requires one second for each of seven digits. The time to connect the loop to a trunk is one hundred milliseconds, the time for recording accounting data is one hundred milliseconds, and the time to generate an offhook signal on a trunk is ten milliseconds, with the dial tone generator used at the start of call processing generating the offhook signal.

The steps involved in sending digits over a trunk to the other local switching system are shown in the figure below:

Assume that the trunk handler detects an offhook in ten milliseconds, the time to connect an idle processor to the trunk is fifty milliseconds, the time to generate an offhook signal on the trunk is ten milliseconds, and the time to receive all digits is twenty milliseconds

Figure 3.18.Local Office to Local Office Digit Transmission

The figure below shows the steps involved in setting up a path from the receiving local switching system back to the originating local switching system:

Figure 3.19.Completion of Call Path from Receiver to Transmitter

Assume that all the steps take the same amount of time as cited above. The step of generating ringing takes two hundred milliseconds.

The final steps involved in call processing are to generate an audible ringing (taking fifty milliseconds), to stop all ringing (taking twenty milliseconds), and these are summarized in the figure below:

Answer all the questions:

A. Make a table showing each step of telephone call processing, the resources held at each step, and the time required with no contention for each step

Figure 3.20.Final Steps in Call Set Up Processing

B. Make a flowchart showing how control and voice information flows through this system from the time a call is originated by going offhook, to the end of a voice telephone conversation

C. What is the state of each trunk at each step of call set up processing?

D. Code a simulation program of this system. Adopt the following priority arbitration policy for contention for shared resources: step J has higher priority than step I if $J > I$. If two or more calls are contending for a shared resource at the same step of call processing, break the tie by flipping a fair coin. What is the fraction of time that dial tone delay exceeds one second for this load? What is the fraction of time that dial tone delay exceeds three seconds? What happens if time between call attempts is reduced to five minutes, and each call lasts for two minutes?

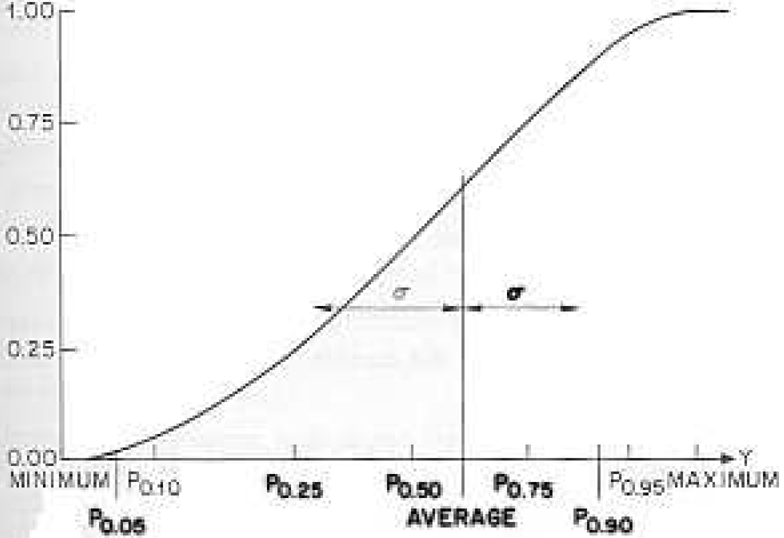

Figure 3.1.Statistical Measures

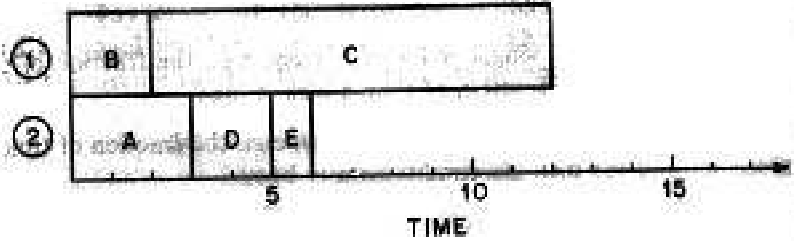

Figure 3.2. Printer Activity Using Alphabetic Priorities

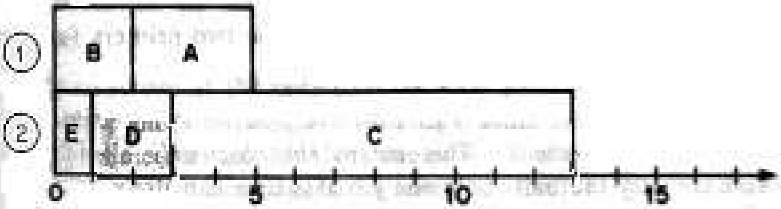

Figure 3.3.Shortest Processing Time Priority Arbitration

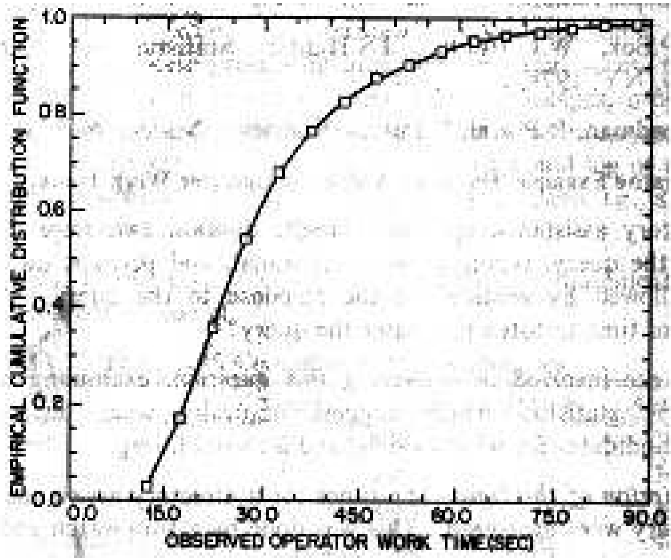

Figure 3.4.Empirical Distribution Function

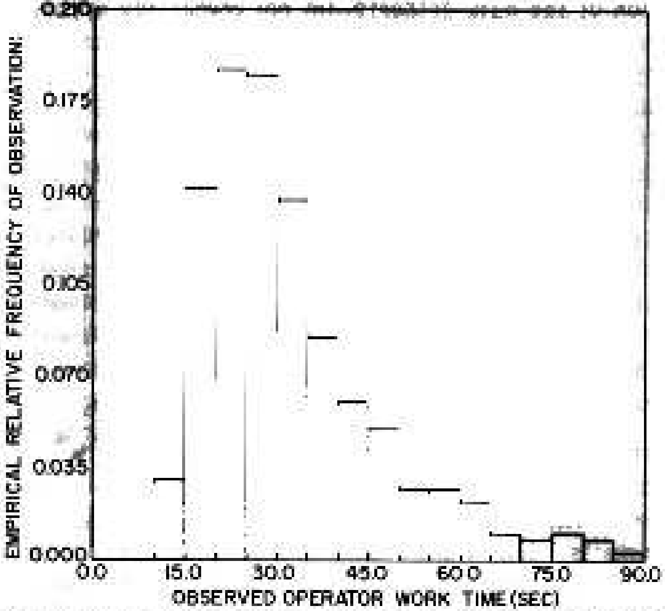

Figure 3.5. Histogram of Directory Assistance Operator Work Time Data

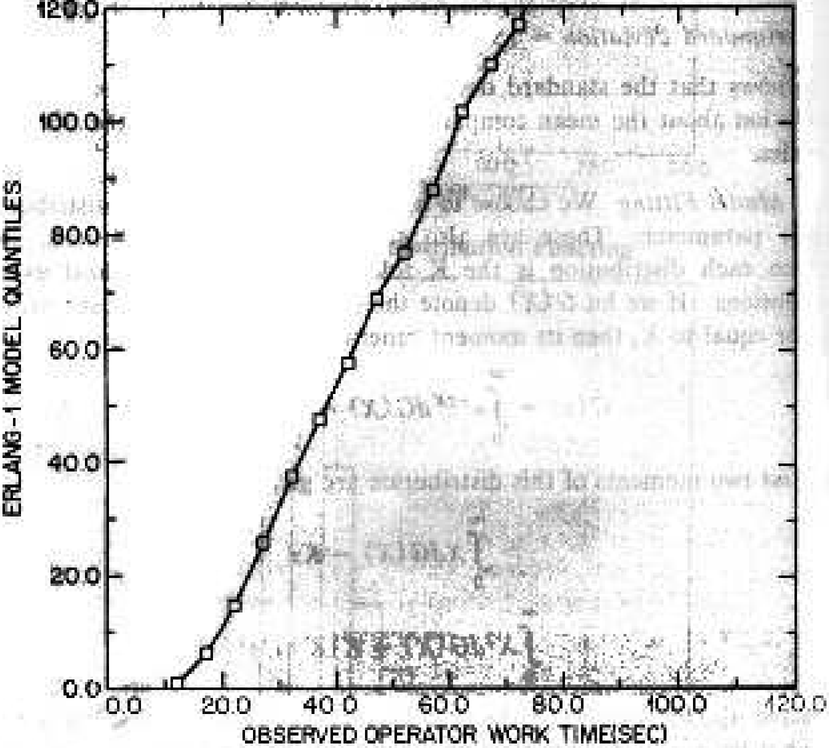

Figure 3.6.Erlang 1 vs Data Q-Q Plot

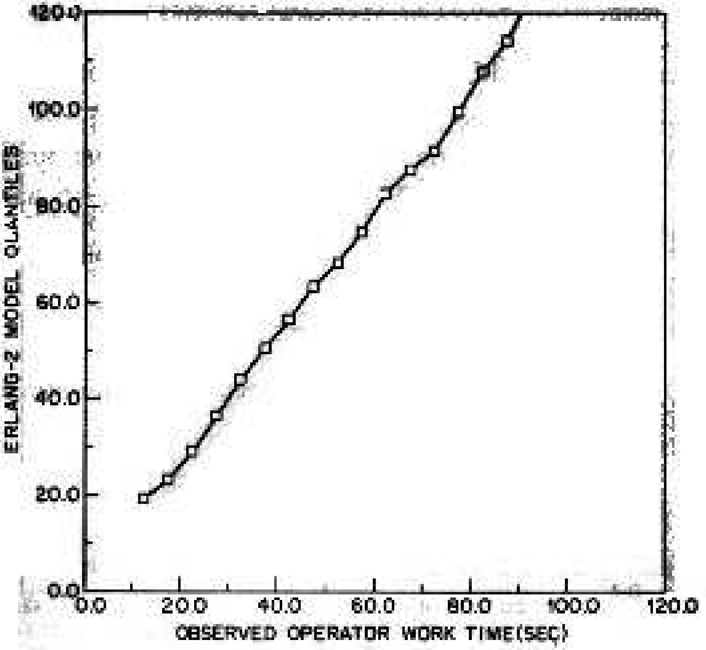

Figure 3.7.Erlang 2 vs Data Q-Q Plot

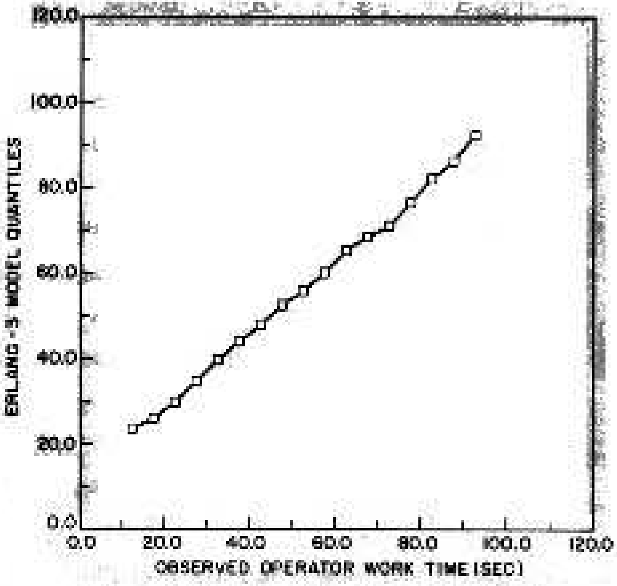

Figure 3.8.Erlang 3 vs Data Q-Q Plot

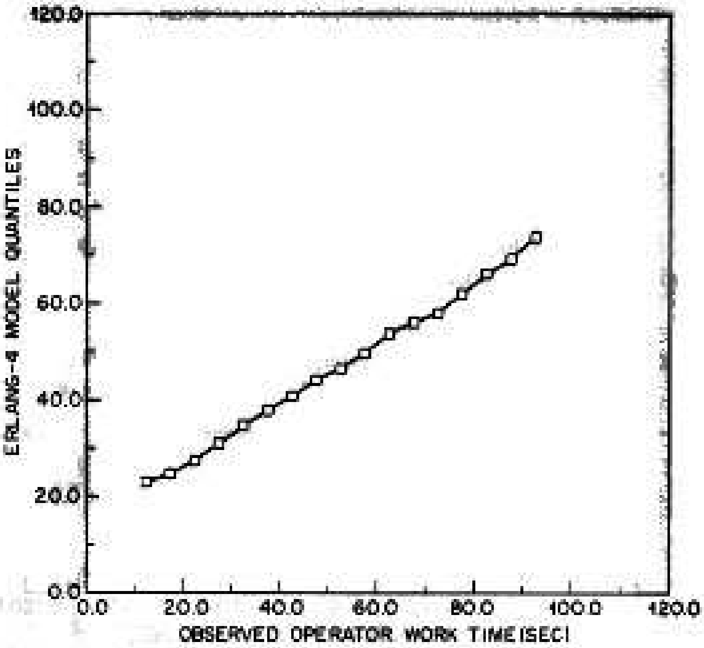

Figure 3.9.Erlang 4 vs Data Q-Q Plot

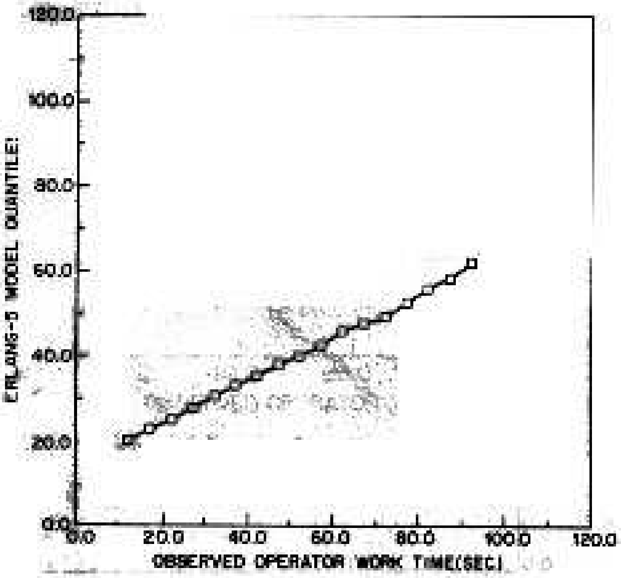

Figure 3.10.Erlang 5 vs Data Q-Q Plot

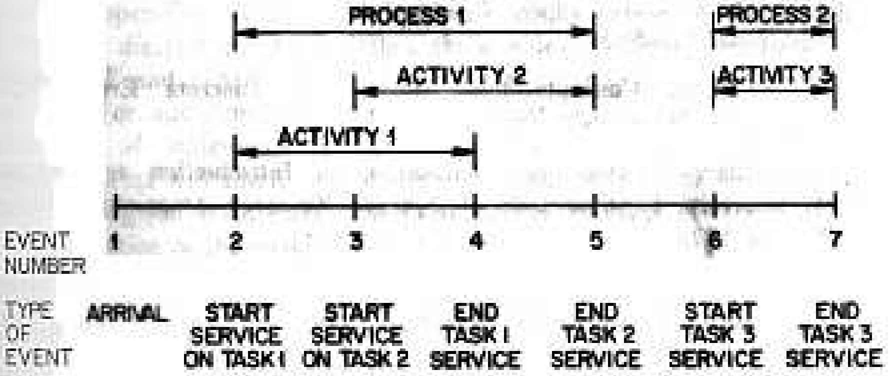

Figure 3.11.Representative Timing Diagram of a Simulation

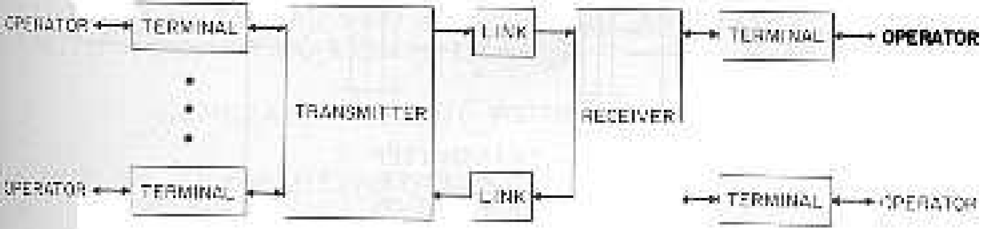

Figure 3.12.Communications Link Hardware Block Diagram

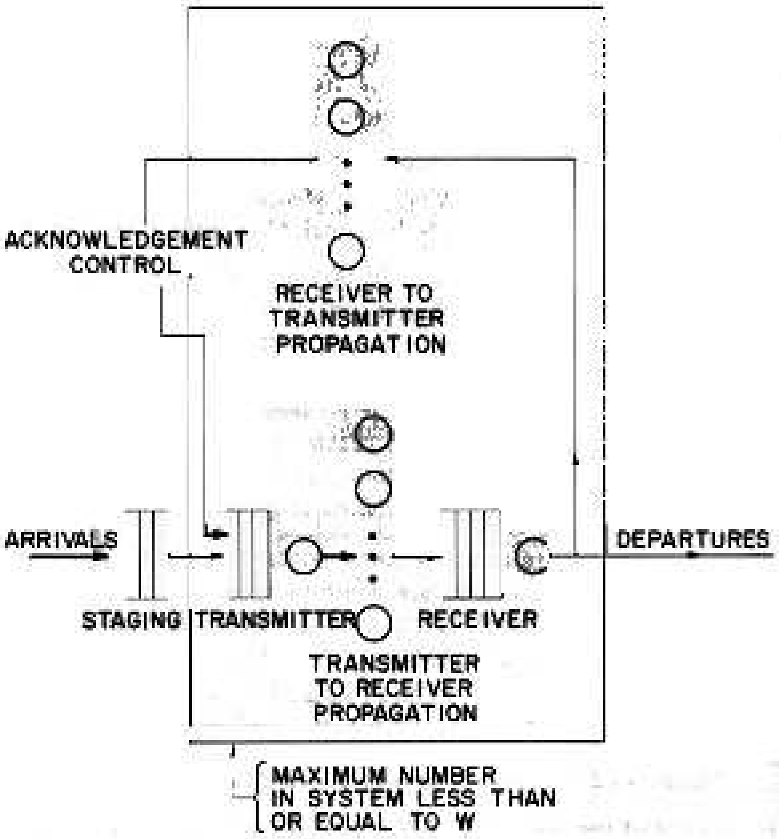
ACKNOWLEDGEMENT CONTROL
RECEIVER TO TRANSMITTER PROPAGATION
ARRIVALS
STAGING
TRANSMITTER
RECEIVER
DEPARTURES
TRANSMITTER TO RECEIVER PROPAGATION
MAXIMUM NUMBER IN SYSTEM LESS THAN OR EQUAL TO W

Figure 3.13.Communications Link Queueing Network Block Diagram

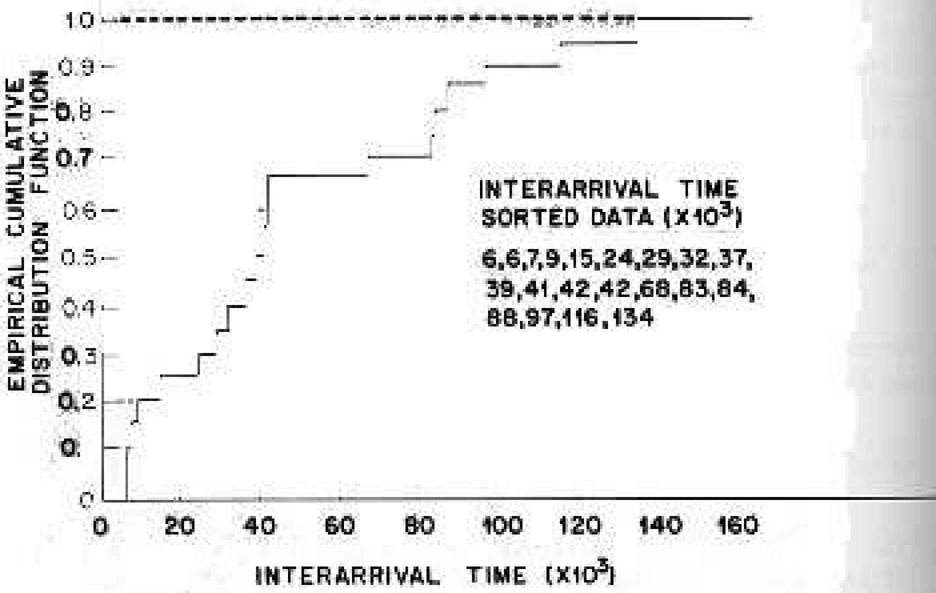

Figure 3.15. Interarrival Time Empirical Cumulative Distribution Function

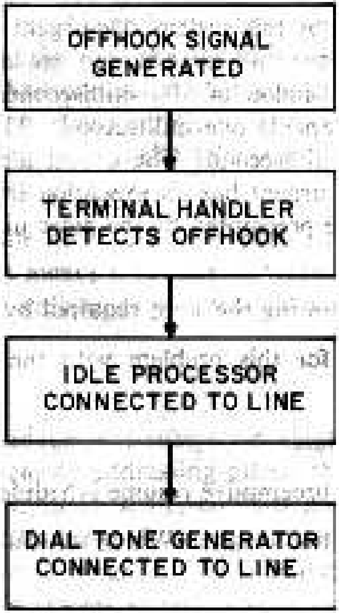
OFFHOOK SIGNAL GENERATED
TERMINAL HANDLER DETECTS OFFHOOK
IDLE PROCESSOR CONNECTED TO LINE
DIAL TONE GENERATOR CONNECTED TO LINE

Figure 3.16.Offhook and Dial Tone Generation Steps

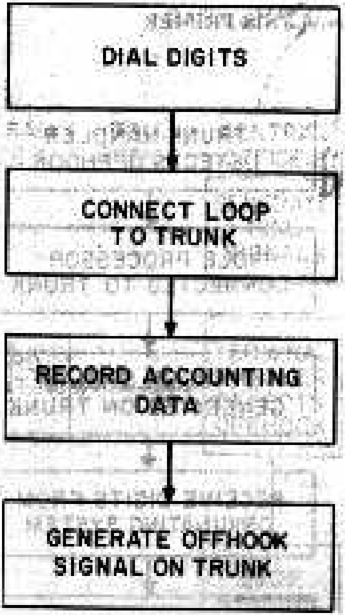

Figure 3.17.Digit Dialing and Billing

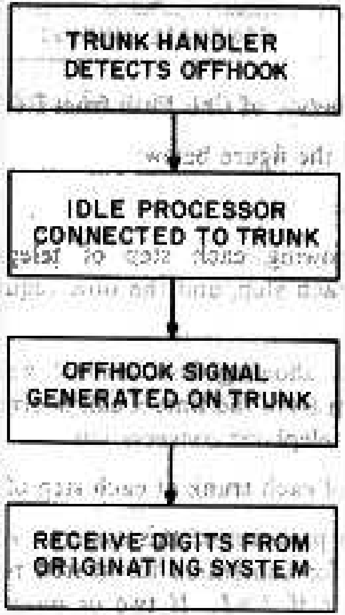
TRUNK HANDLER DETECTS OFFHOOK
IDLE PROCESSOR CONNECTED TO TRUNK
OFFHOOK SIGNAL GENERATED ON TRUNK
RECEIVE DIGITS FROM ORIGINATING SYSTEM

Figure 3.18.Local Office to Local Office Digit Transmission

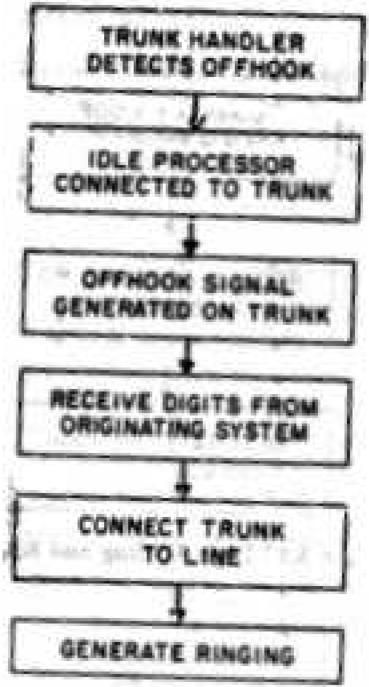
TRUNK HANDLER DETECTS OFFHOOK
IDLE PROCESSOR CONNECTED TO TRUNK
OFFHOOK SIGNAL GENERATED ON TRUNK
RECEIVE DIGITS FROM ORIGINATING SYSTEM
CONNECT TRUNK TO LINE
GENERATE RINGING

Figure 3.19.Completion of Call Path from Receiver to Transmitter

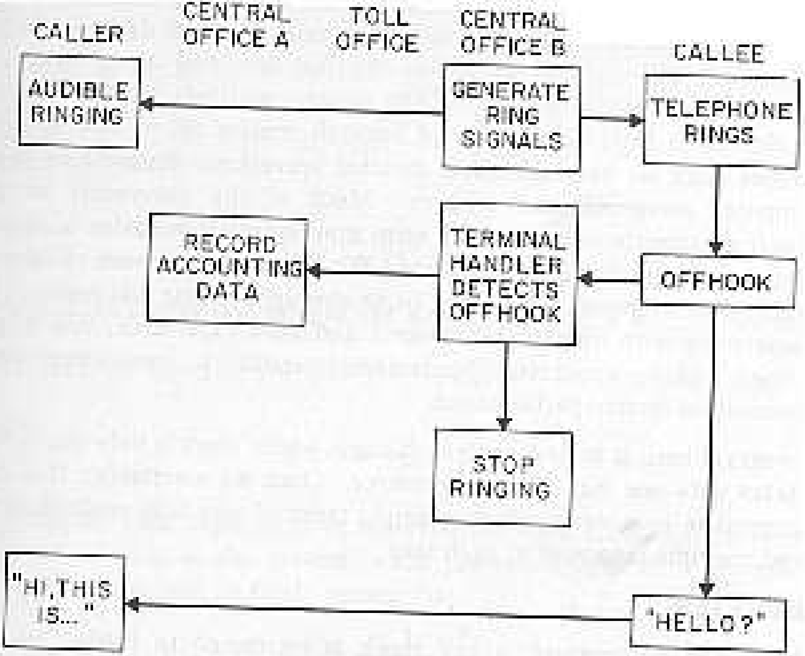

Figure 3.20. Final Steps in Call Set Up Processing