11

Computer System Reliability Modeling

11.1 INTRODUCTION

The scope of computer applications has steadily expanded since the early 1960s, encompassing numerous areas of critical importance. These applications include existing and proposed uses in real time control of communication and transportation systems, automated plant operations and space explorations. These systems generally demand high reliability since system failures can be very costly and hazardous. Very significant progress has been made in the field of computer reliability both in the simplex sense and in the use of redundancy to achieve this objective. This chapter reviews different basic approaches to computer reliability and classification of computer faults and describes modeling of permanent and transient faults.

The term reliability in the context of a computer or information processing system pertains to the correct execution of a program. The modeling and assurance of reliability in digital systems becomes somewhat different from many other systems because of the lack of ability of digital systems to tolerate temporary disturbances. A transient disturbance in many systems would degrade the system performance temporarily, after which the system is restored to normal operation. In digital systems, a transient disturbance could become fatal to a computation, if at a critical moment contents of a register are changed. This subtle difference between the analog and digital systems [2] has not generally been perceived by many reliability analysts.

11.2 CAUSES OF COMPUTER FAILURES [15-20]

For designing reliable computer-based systems, it is essential to identify causes of failures in computers. This section provides a list of the various sources of failure. It is to be noted that failures due to hardware components are only one of such causes. Therefore, reliability predictions based on these failures could be optimistic. A discussion of some important sources of errors is provided in this section. Readers interested in this topic will find references 15–20 of great value.

Memory and Processor Failures. Modern computers generally use a parity bit to detect failures in memory. Memory failures can be serious as they can cause the entire system to shut down as the operating system can not effectively deal with this problem. Power surges and failures can cause memory problems.

Processor errors are rare but generally catastrophic. Reference to index register *n* may, for example, be suddenly changed due to a dropped bit and this will undoubtedly cause the system to go berserk.

Peripheral Device Failures. Failure or degradation of peripheral device hardware can sometimes cause serious problems, although generally they do not result in system shut down.

Intermodule Communication Failures. It is generally accepted that communication failures do occur and will continue to occur. Various error-detecting and -correcting codes are employed but nevertheless some communication errors do finally result in outages of terminals and lines.

Human Errors. The two important sources of human errors are the operator errors and errors in the software. The matter of software reliability has been covered in Chapter 5. The operators occasionally cause a system crash by starting or shutting down system incorrectly or by incorrectly responding to a particular situation.

Environmental Failures. The environmental failures can result from electromagnetic interference due to inadequate shielding and by the failure of air conditioning equipment.

Power Failures. A strong power surge could seriously degrade the life expectancy of electronic components and cause many lingering problems. The computer systems are sensitive even to transient dips and surges and must be provided with proper protection.

The various possible sources of computer faults have been listed. When a computer failure does, however, occur it is not easy to classify the exact cause of failure. Many of the faults remain unexplained.

11.3 CLASSIFICATION OF FAULTS

Despite the numerous possible locations of faults described in the previous section, they basically originate either from permanent failures of hardware components, temporary malfunctions of these components, or external interference with the computer system operation. For the purposes of reliability modeling and evaluation, a useful way of classifying faults is on the basis of their duration. The faults may be classified as transient or permanent.

Permanent faults are often caused by the catastrophic failures of the components. The failure of the component in this case is irreversible and permanent and needs repair or replacement. These faults are characterized by long duration and have a failure rate depending on the environment. A component, for example, will have generally a different failure rate in power-on and power-off situations [12].

The transient faults, on the other hand, are caused by temporary malfunction of the components or by the external interference such as electrical noise, power dips, and glitches. These faults are of limited duration and although they require restoration do not involve repair or replacement. These faults are characterized by arrival modes and duration of transients [3].

11.4 BASIC APPROACHES TO COMPUTER RELIABILITY

Like other physical systems the reliability of computer systems may be enhanced either in a simplex manner or through the use of some form of redundancy. The simplex approach would involve use of high reliability components, conservative designs and adequate attention to fabrication and assembly. In this type of approach, which has also been called the fault intolerant approach, correct program execution implies fault-free hardware operation. There has been a tremendous improvement in the reliability of computer components, making simplex configurations more reliable. The improvement in component reliability, however, has been accompanied by a corresponding growth in the complexity of computer systems requiring an ever-increasing number of components. It has been recognized for a long time [13] that simplex configurations are not likely to provide the ultrareliable computer systems required for space explorations and real-time control of ground or airborne systems.

An alternative and complementary approach to reliable computer systems is that of fault-tolerance. Here the faults are expected to occur but disruptive effect of faults is avoided or minimized by providing some form of redundancy. The computer system can tolerate a predetermined number of faults and execute the programs correctly in their presence. The fault tolerance in computer systems is provided by protective redundancy, which may be implemented in three different ways.

1. Hardware redundancy, that is, additional hardware in terms of redundant logic and/or replication of entire computers.
2. Software redundancy, that is, additional programs for either masking faults or providing recovery.
3. Time redundancy in terms of repetition of machine operations.

Functionally redundancy may be either static or dynamic.

11.4.1 Static Redundancy

Fault Masking. The effect of faults may be masked by providing additional hardware such that the output of the module remains unaffected even though the fault has occurred in one of the redundant units. The effect of a faulty component is instantaneously and automatically masked by permanently connected and concurrently operating units [5, 7]. The reason for naming it static redundancy is that fault masking is autonomous and is without intervention through input-output terminals.

Majority voting redundancy which is the most prominent form of fault masking was proposed by John von Neuman [11] who developed and analyzed triplication of units with majority voting. This type of redundancy has been made economically feasible by integrated circuit technology. One of the interesting illustrations of this approach is SATURN V launch vehicle computer. The SATURN V computer employs a triple modular redundancy with voting elements in the central processor and duplication in the main memory [8].

Application of Coding Theory. Error detecting and correcting codes developed in connection with communication theory and special codes developed for high-speed encoding and decoding have been used for implementing fault tolerance in data transmission and storage functions. Reference 7 states that cost of implementing such schemes is less than 1.5 times the nonredundant configuration.

11.4.2 Dynamic Redundancy

In dynamic redundancy, the fault effect is allowed to appear at the terminals but means are provided for detection, diagnosis, and recovery. If human intervention is eliminated, dynamic redundancy results in computer system self-repair. The first operational computer with full self-repair is probably the JPL-STAR computer [1]. This type of redundancy has also been termed standby redundancy. Error correction is achieved by recomputation, possibly retracing the steps in the program to a roll back point. Reference 5 provides a qualitative and quantitative comparison of the static and dynamic redundancy techniques.

11.4.3 Hybrid Redundancy

In such a scheme, at any moment, three or more elements are connected to a majority element. When a module, however, fails its disagreement with its companions is detected and it is replaced by a spare unit.

11.5 MODELING PERMANENT FAULTS

Modeling of alternative configurations provides information for the judicious choice for a particular application. A number of redundant configurations have been proposed for computer architecture. Models for more commonly known configurations are discussed. The reliability measures depend on the nature of the application. For space-borne equipment, where the equipment is not accessible to repair, the probability of successful operation during the mission time is an appropriate measure. On the other hand, for applications like automated transit or plant control, a measure like mean up time, mean down time, and failure frequency also provide useful information. The failure rates and repair rates where applicable are assumed constant.

11.5.1 Simplex System

The reliability of a simplex system is well-known,

$$R = e^{-\lambda T} \tag{11.1}$$

where λ = failure rate of the system
T = mission time

Equation 11.1 gives the probability that the system will not have failed by mission time T. For repairable system

$$A = \frac{\mu}{\lambda + \mu} \tag{11.2}$$

$$\bar{A} = \frac{\lambda}{\lambda + \mu} \tag{11.3}$$

$$f_f = \frac{\lambda \mu}{\lambda + \mu} \tag{11.4}$$

where A = system availability, that is, long run average proportion of time in success state
$\bar{A}$ = unavailability, that is, long run proportion of time spent in failed state
f_f = frequency of failure, that is, average number of failures per unit time of system operation

11.5.2 Triple-Modular Redundant System

Triple modular redundancy (TMR) is probably the most tossed around term in computer redundancy reliability. The basic TMR system is shown in Figure 11.1, which consists of three identical units representing a single logical variable, the value of the variable being determined on the basis of majority voting. The function of the voter is illustrated in Table 11.1. If there is an independent fault in one of the units, it is evidently masked and output remains correct.

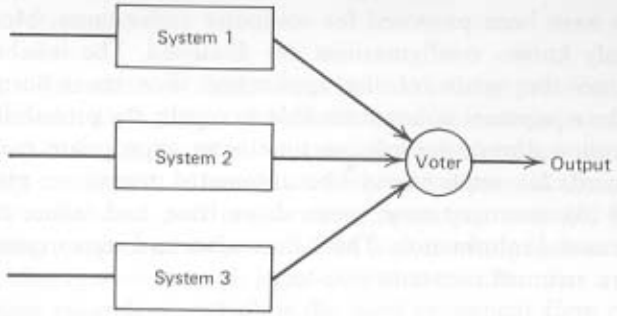

Figure 11.1 Triple-modular redundant system with voting.

The reliability of the TMR configuration is given by the following equation:

$$R_{\text{TMR}} = R^3 + 3(1-R)R^2$$
$$= R^2(3-2R) \qquad (11.5)$$

In deriving (11.5) the voter is assumed perfectly reliable and the effect of compensating failures is ignored. For a nonperfect voter, (11.5) becomes

$$R_{\text{TMR}} = R^2(3-2R)R_V \qquad (11.6)$$

where R = reliability of each unit
R_V = voter reliability

In reality some of the failures can be compensatory. For example if one of the units is stuck at 0 and the other is stuck at 1, then their votes cancel and if, the third unit is operating correctly, the output will be correct.

Table 11.1 *TMR Voting*

Unit Output			
System 1	System 2	System 3	Voter Output
0	0	0	0
0	0	1	0
0	1	0	0
1	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	1

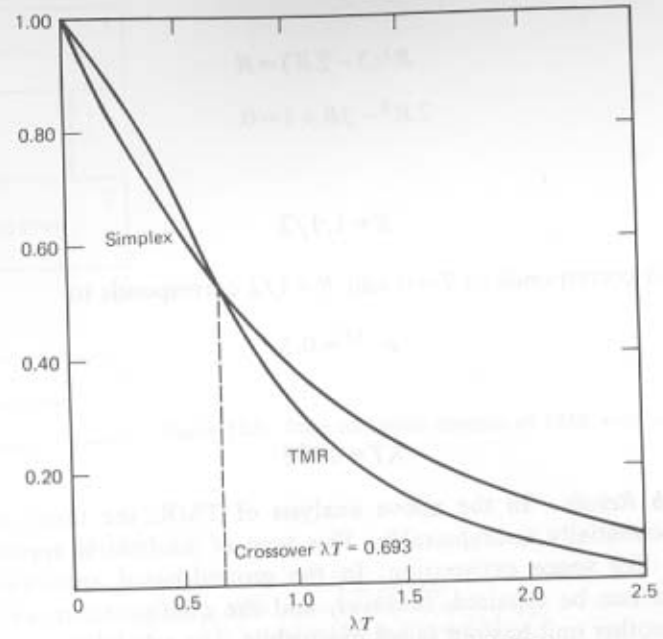

Figure 11.2 Reliability of TMR vs SIMPLEX.

Equations 11.5 and 11.6 therefore provide a pessimistic or conservative estimate. This topic is further discussed in reference 9. The behavior of TMR with respect to the simplex system is shown in Figure 11.2 over the normalized mission time. It is to be noted that whereas reliability of redundant configuration is better than the simplex system until the crossover point, the reliability of TMR is worse after this point. This type of relative behavior has led to the suggestion [4] that MTTF may be a misleading indicator of performance for mission-oriented systems. The MTTF is given by

$$\text{MTTF} = \int_0^\infty R\,dt \qquad (11.7)$$

That is, MTTF computation considers reliability over the interval $[0, \infty]$ which includes period after the crossover. For a mission-oriented system, the reliability function is of concern over the period $[0, T]$ which may be before the crossover. The crossover point can be determined by equating the reliability of simplex and TMR systems,

$$R_{\text{TMR}} = R$$

that is,

$$R^2(3-2R)=R$$

$$2R^2-3R+1=0$$

or

$$R=1, 1/2$$

Now $R=1$ corresponds to $T=0$ and $R=1/2$ corresponds to

$$e^{-\lambda T}=0.5$$

that is,

$$\lambda T=0.693$$

TMR with Repair. In the above analysis of TMR, the three units are assumed essentially nonrepairable. This type of analysis is applicable to situations like space exploration. In the ground-based applications the failed unit can be repaired, however, and the configuration restored to TMR if another unit has not failed meanwhile. The reliability equation for such an application can be derived using the state transition diagram shown in Figure 11.3, where λ and μ are the failure and repair rates of each unit. When a unit fails, it can be repaired and restored. The process of restoration (synchronization, etc.) is assumed to be accomplished with probability 1. State 3, where two units are failed, is made an absorbing state by not allowing repair from this state. Under this condition, the reliability of the TMR (TMRR) is

$$R_{\text{TMRR}}=P_1(T)+P_2(T) \tag{11.8}$$

where $P_i(t)$=probability of the system being in state i at time t.

The state differential equations for Fig. 11.3 are

$$\dot{P}_1(t)=\mu P_2(t)-3\lambda P_1(t) \tag{11.9}$$

$$\dot{P}_2(t)=3\lambda P_1(t)-(2\lambda+\mu)P_2(t) \tag{11.10}$$

$$\dot{P}_3(t)=2\lambda P_2(t) \tag{11.11}$$

Assuming $P_1(0)=1.0$ and taking Laplace transform of (11.9)–(11.11)

$$sP_1(s)-1=\mu P_2(s)-3\lambda P_1(s) \tag{11.12}$$

$$sP_2(s)=3\lambda P_1(s)-(2\lambda+\mu)P_2(s) \tag{11.13}$$

$$sP_3(s)=2\lambda P_2(s) \tag{11.14}$$

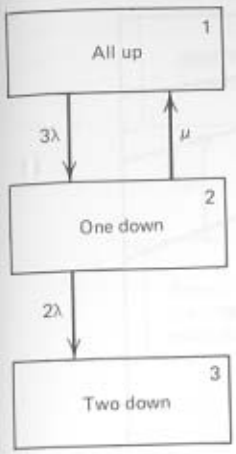

Figure 11.3 State transition diagram of TMR with repair.

From (11.12) and (11.13)

$$P_1(s)=\frac{s+2\lambda+\mu}{s^2+(5\lambda+\mu)s+6\lambda^2} \tag{11.15}$$

and

$$P_2(s)=\frac{3\lambda}{s^2+(5\lambda+\mu)s+6\lambda^2} \tag{11.16}$$

Taking inverse Laplace of (11.15) and (11.16)

$$P_1(t)=\frac{1}{r_1-r_2}\left[(2\lambda+\mu)(e^{r_1t}-e^{r_2t})+r_1e^{r_1t}-r_2e^{r_2t}\right] \tag{11.17}$$

and

$$P_2(t)=\frac{3\lambda}{r_1-r_2}\left[e^{r_1t}-e^{r_2t}\right] \tag{11.18}$$

where

$$r_1, r_2=\frac{-(5\lambda+\mu)\pm\sqrt{\lambda^2+\mu^2+10\lambda\mu}}{2}$$

Substituting into (11.8)

$$R_{\text{TMRR}}=\frac{1}{r_1-r_2}\left[(5\lambda+\mu)(e^{r_1T}-e^{r_2T})+r_1e^{r_1T}-r_2e^{r_2T}\right] \tag{11.19}$$

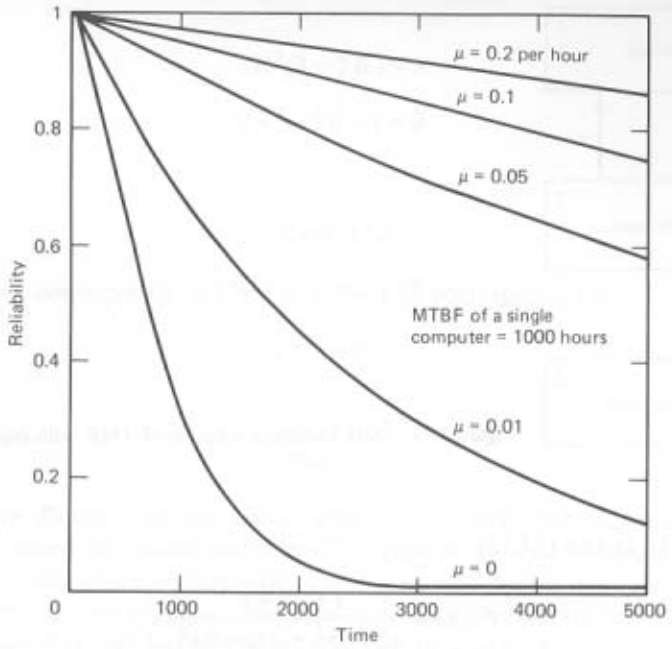

Figure 11.4 Effect of repair on TMR reliability.

Equation 11.19 gives the reliability of a TMR configuration with repair. Figure 11.4 shows the effect of repair rate parameter on the TMR reliability. If there is no repair, then substituting $\mu=0$ into (11.19)

$$\begin{aligned} R_{\text{TMRR}} &= 3e^{-2\lambda T} - 2e^{-3\lambda T} \\ &= R^2(3-2R) \\ &= R_{\text{TMR}} \end{aligned}$$

TMR With Repair and Common Mode Failures. Analysis of TMR up to this point is based on the independent failures of the units comprising the TMR. In practice, some faults like those due to external environment and software bugs can cause common mode failures. The state transition diagram for this situation is shown in Figure 11.5. Let

λ = failure rate of a single unit = $1/U_1$
U_1 = mean up time of a single computer
λ_c = rate for common mode failures
= $1/U_c$
U_c = mean operating time between consecutive common mode failures
μ = repair rate of a computer
= 1/(mean time to repair)

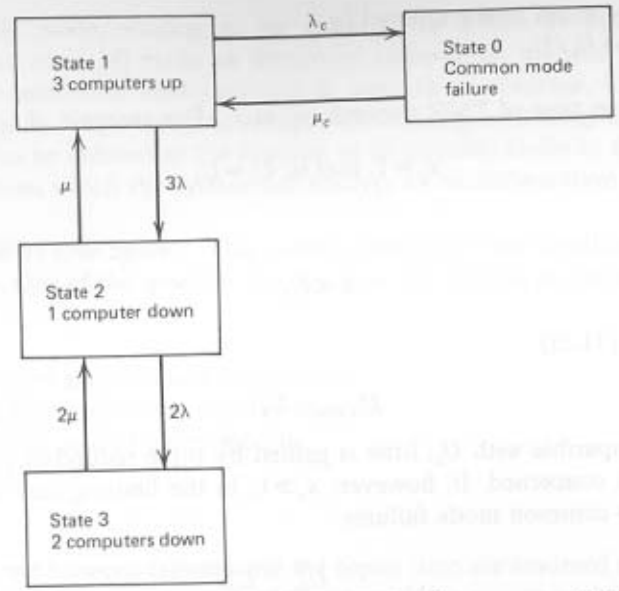

Figure 11.5 State transition diagram of TMR with common mode failure.

μ_c = repair rate for common mode failure
= 1/(mean time to repair common mode failure)

The frequency balance equations [14] for Figure 11.5 are given by (11.20)–(11.23),

$$(3\lambda+\lambda_c)P_1 = \mu P_2 + \mu_c P_0 \tag{11.20}$$

$$\lambda_c P_1 = \mu_c P_0 \tag{11.21}$$

$$(2\lambda+\mu)P_2 = 3\lambda P_1 + 2\mu P_3 \tag{11.22}$$

$$2\mu P_3 = 2\lambda P_2 \tag{11.23}$$

From (11.22) and (11.23)

$$P_1 = \frac{\mu}{3\lambda} P_2 \tag{11.24}$$

Mean up time of TMR is given by [14]

$$\begin{aligned} U_{\text{TMR}} &= \frac{P_1+P_2}{\lambda_c P_1 + 2\lambda P_2} \\ &= \left(\frac{\mu}{3\lambda}+1\right) \Big/ \left(\frac{\mu\lambda_c}{3\lambda}+2\lambda\right) \\ &= \frac{x_r U_1^2}{U_1+6x_r d} + \frac{3x_r d U_1}{U_1+6x_r d} \end{aligned} \tag{11.25}$$

where d = mean repair time $= 1/\mu$
$x_r = U_c/U_1$

The mean up time of TMR depends upon x_r. For example, if

$$x_r = 1, \text{ that is, } U_c = U_1$$

and

$$d \ll U_1$$

Then from (11.25)

$$U_{\text{TMR}} \simeq U_1$$

If U_c is comparible with U_1, little is gained by triple redundancy as far as reliability is concerned. If, however, $x_r \gg 1$; in the limiting case of $x_r \to \infty$ meaning no common mode failures

$$U_{\text{TMR}} = \frac{U_1^2}{6d} + \frac{U_1}{2} \tag{11.26}$$

It is worth noting that U_{TMR} is sensitive to the square of U_1. Therefore if U_1 is doubled, U_{TMR} is approximately quadrupled.

11.5.3 NMR Configuration

An NMR (N-tuple Modular Redundant) system consists of $N = 2n+1$ replicated units feeding a $(n+1)$-out-of N voter. For a majority among N units, at least $n+1$ units must survive for successful operation. Neglecting the effect of compensating failures, the reliability of an NMR configuration is given by (11.27).

$$R_{\text{NMR}} = \sum_{i=0}^{n} \binom{N}{i} (1-R)^i R^{N-i} \tag{11.27}$$

where $\binom{N}{i} = N!/(N-i)!i!$. If R_{NMR} is plotted as a function of normalized time λT, it is seen that as in the case of TMR, R_{NMR} is greater than simplex reliability before the crossover point and less than simplex after the crossover point. This effect is accentuated as N increases and in the limiting case of $N \to \infty$, the R_{NMR} is unity before crossover point and zero afterward.

11.5.4 Standby Redundancy

Several functionally identical units are employed in standby redundancy. Some of the units perform active function whereas the others are in a standby mode, waiting to be switched in if one of the units fails. Bouricious et al. [4] made an important observation that the reliability of stand-by redundant configurations is sensitive to coverage. Coverage can be defined as the probability of successful sensing, switching, and recovery. It can also be defined as the fraction of all possible faults in the computer system from which the system can recover by reconfiguration.

System With One Spare. This system consists of two identical computing modules. One of the modules is active and the other is in a stand-by mode. Let

λ_a = failure rate of the active module
λ_s = failure rate of the standby module
μ = repair rate of either module
c = coverage
$\bar{c} = 1 - c$

The time between failures and the repair time are assumed exponentially distributed. The state transition diagram of this system is shown in Figure 11.6 where A and S stand for active and stand-by, and G and F stand for good and failed. Both the modules are good in state 1. When the standby fails, the system enters state 3. When the active unit fails, there are two possible resulting states. If the failure of the active unit is sensed, the standby module is switched in and there is a successful recovery, the system enters state 2 where the standby now becomes active. In the case of unsuccessful recovery, both the units go down and there is system failure—state 4. The transition rate matrix for Figure 11.6 is

$$A \equiv \begin{bmatrix} -(\lambda_a+\lambda_s) & c\lambda_a & \lambda_s & \bar{c}\lambda_a \\ \mu & -(\mu+\lambda_a) & 0 & \lambda_a \\ \mu & 0 & -(\mu+\lambda_a) & \lambda_a \\ 0 & \mu & \mu & -2\mu \end{bmatrix}$$

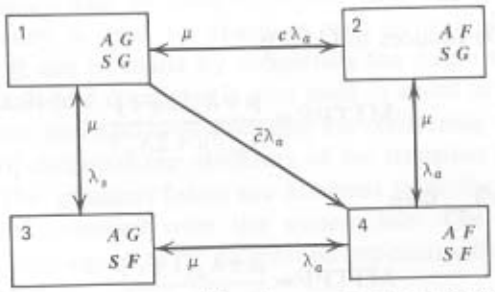

Figure 11.6 State transition diagram of a standby system.

Assuming the process starts in state 1, the mean time to first failure (MTTFF) can be obtained [14] from (11.28)

$$\text{MTTFF} = (1\ 0\ 0)[-Q]^{-1}U \tag{11.28}$$

where Q = is the matrix obtained from A by deleting the row and column corresponding to state 4 indicating system failure
U = is a unit column vector

Now

$$-Q = \begin{bmatrix} \lambda_a+\lambda_s & -c\lambda_a & -\lambda_s \\ -\mu & \mu+\lambda_a & 0 \\ -\mu & 0 & \mu+\lambda_a \end{bmatrix}$$

Denoting the element (ij) of the inverse of $-Q$ by m_{ij},

$$m_{11} = \frac{(\mu+\lambda_a)^2}{\Delta}$$

$$m_{12} = c\lambda_a \frac{(\mu+\lambda_a)}{\Delta}$$

$$m_{13} = \lambda_s \frac{(\mu+\lambda_a)}{\Delta}$$

where $\Delta = |-Q|$

$$= \lambda_a(\mu+\lambda_a)(\bar{c}\mu+\lambda_a+\lambda_s)$$

Using (11.28)

$$\text{MTTFF} = m_{11}+m_{12}+m_{13}$$

$$= \frac{\mu+\lambda_s+\lambda_a(1+c)}{\lambda_a(\bar{c}\mu+\lambda_a+\lambda_s)} \tag{11.29}$$

If $\lambda_a = \lambda_s$, (11.29) reduces to (11.30)

$$\text{MTTFF} = \frac{\mu+\lambda_a(2+c)}{\lambda_a(\bar{c}\mu+2\lambda_a)} \tag{11.30}$$

If, however, $\lambda_s \ll \lambda_a$, then

$$\text{MTTFF} = \frac{\mu+\lambda_a(1+c)}{\lambda_a(\bar{c}\mu+\lambda_a)} \tag{11.31}$$

Table 11.2 *MTTFF as a function of c*

$\lambda_a = 0.00025$/hour
$\mu = 0.25$/hour

	MTTFF (hours)	
c	For $\lambda_s = \lambda_a$	For $\lambda_s = 0$
1	2,006,000	4,008,000
0.99	334,330	364,360
0.98	182,360	190,853
0.95	77,150	78,584

As an example assume

$$\lambda_a = 0.00025/\text{hour}$$

and

$$\mu = 0.25/\text{hour}$$

The MTTFF for this system is shown as a function of c in Table 11.2. It can be seen that the MTTFF is very sensitive even to small variation in c. The analysis of this system with constant repair time is reported in reference 6

11.6 MODELING TRANSIENT FAULTS

A transient fault is of limited duration and disappears some time after its arrival [10]. During this brief period of its stay, it can, however, alter the contents of registers and interfere with the normal sequence of program execution. The transient fault, although of limited duration, can cause incorrect execution of the program and therefore result in computer system failure. Redundancy can be used to recover from the effect of transient faults. Redundancy is used to detect as well as provide recovery. The detection of fault can be made by comparing the outputs of two or more modules. The fault-free computer is also used to assist in restoring altered data and program and also synchronizing the recovering computer.

Reference [10] describes the modeling of the transient faults in a TMR configuration. The transient faults are assumed to arrive with an average rate λ_t, assumed constant over the system life. The duration of the transient faults is assumed to be distributed exponentially, given by

$$f_D = re^{-rt} \tag{11.32}$$

The recovery procedure consists of three steps:

1. *Detection*. Faults are detected by comparison, the time between comparisons being T_c. The time between the occurrence and detection of a fault is a random variable with an assumed probability density function $\delta \exp(-\delta t)$.
2. *Delay*. A certain time T_D is allowed between the detection of fault and initiation of recovery. This time is designed for the transient to die out.
3. *Recovery*. A certain time T_R is allowed to accomplish the recovery procedure.

If all the computers are fault-free, and now a transient occurs in one of the computers, then a transient recovery procedure will be initiated with the following possible outcomes:

1. Recovery is successful and therefore the computers are again fault-free.
2. The transient may be long enough to continue into the recovery period and be mistaken for a permanent fault. In this case this computer will be assumed faulted.
3. A previously fault-free computer may experience a fault during the recovery period resulting in TMR system failure.

The system is assumed to fail when a computer experiences a permanent or transient fault when another computer is having a permanent fault or is recovering from a transient. The probability of a system failing before time T is derived in reference 10 and is,

$$= a\left\{\frac{1-e^{-3bT}}{b} - \frac{3e^{-2cT}}{3b-2c}\left[1-e^{-(3b-2c)T}\right]\right\}$$

$$+ \frac{\bar{L}\lambda_t}{b}\left[\frac{2c+\delta(1-e^{-2cT_r})}{\delta+2c}\right](1-e^{-3bT}) \qquad (11.33)$$

where $a = \lambda + L\lambda_t$

$b = \lambda + (1-c_T)\lambda_t$

$c = \lambda + \lambda_t$

c_T = transient coverage
= Probability of recovery from a transient, given a transient occurs.

L = Probability of transient fault being interpreted as a permanent fault.

$$= 1 - \delta e^{-cT_r}\left[\frac{1}{\delta+c} - \frac{e^{-rT_D}}{c+r+\delta}\right]$$

$$\bar{L} = 1 - L$$

$$T_r = T_D + T_R$$

REFERENCES

1. Avizienis, A., G. C. Gilley, F. P. Mathur, D. A. Rennels, J. A. Rohr, and D. K. Rubin, "The STAR (Self-Testing-And-Repairing) Computer: An Investigation of the Theory and Practice of Fault-Tolerant Computer Design," *IEEE Trans. Comput.*, **C-20**, 1312–1321 (1971).
2. Avizienis, A., "Modeling and Evaluation of the Reliability of Computer Systems," *Third National Reliability Symposium*, Perros-Guirec, France Sept. 14–17, 1976.
3. Avizienis, A., "Fault-Tolerant Computing: Progress, Problems, AND Prospectus," Inf. Process '77, Proceedings of IFIP Congress, Toronto, Aug. 8–12, 1977. Published by North-Holland Publ. Co. (IFIP Congress series, vol. 7), Amsterdam and New York, NY, 1977, pp. 405–420.
4. Bouricius, W. G., W. C. Carter, and P. R. Schneider, "Reliability Modeling Techniques For Self-Repairing Computer Systems," *Proceedings of ACM National Conference*, ACM, New York, 1969.
5. Carter, W. C. and W. G. Bouricius, "A Survey of Fault Tolerant Computer Architecture and Its Evaluation," *Computer*, **4**, 9–16 (1971).
6. Chow, D. K., "Reliability of Some Redundant Systems with Repair," *IEEE Trans. Reliab.*, **R-22**, 223–228 (1973).
7. Goldberg, J., "A Survey of the Design and Analysis of Fault-Tolerant Computers," *Reliability and Fault Tree Analysis*, Theoretical and Applied Aspects of System Reliability and Safety Assessment, SIAM, Philadelphia, 1975, pp. 687–731.
8. Mathur, F. P. and A. Avizienis, "Reliability Analysis and Architecture of a Hybrid-Redundant Digital System: Generalized Triple Modular Redundancy With Self-Repair," 1970 Spring Joint Computer Conference, *AFIPS Conference Proceedings*, Vol. 36, Montvale, NJ, May 1970, pp. 375–387.
9. Mathur, F. P. and P. T. de Sousa, "Reliability Models of NMR Systems," *IEEE Trans. Reliability*, **R-24**, 108–113 (1975).
10. Merryman, P. M. and A. Avizienis, "Modeling Transient Faults in TMR Computer Systems," *Proceedings 1975 Annual Reliability and Maintainability Symposium*, IEEE, New York, 1975.
11. Von Neumann, J., "Probabilities Logics and the Synthesis of Reliable Organisms From Unreliable Components," In: *Automata Studies*, Princeton University Press, Princeton, NJ, 1956.
12. Nerber, P. O., "Power-Off Time Impact on Reliability Estimates," *IEEE Int. Conv. Rec.*, Part 10, March 1965, pp. 1–8.
13. Short, R. A., "The Attainment of Reliable Digital Systems Through the Use of Redundancy—A Survey," *Computer Group News*, **2**, 2–17 (March 1968).
14. Singh, C. and R. Billinton, *System Reliability Modelling and Evaluation*, Hutchinson, London, 1977.
15. Yourdon, E., "Reliability Measurements For Third Generation Computer Systems," *Proceedings 1972 Annual Reliability and Maintainability Symposium*, IEEE, New York, 1972.
16. Yourdon, E., "Reliability of Real-Time Systems—Part 1—Different Concepts of Reliability," *Modern Data*, **5**, 36–42 (Jan. 1972).

17. Yourdon, E., "Reliability of Real-Time Systems—Part 2—The Causes of System Failures," *Modern Data*, **5**, 50–56 (Feb. 1972).

18. Yourdon, E., "Reliability of Real-Time Systems—Part 3—The Causes of System Failures Continued," *Modern Data*, **5**, 36–40 (March 1972).

19. Yourdon, E., "Reliability of Real-Time Systems—Part 4—Examples of Real-Time System Failures," *Modern Data*, **5**, 52–57 (April 1972).

20. Yourdon, E., "Reliability of Real-Time Systems—Part 5—Approaches to Error Recovery," *Modern Data*, **5**, 38–52 (May 1972).