RELIABILITY MODELLING CONSIDERATIONS FOR A REAL-TIME CONTROL SYSTEM

Malcolm PR Hamer
Post Office Research Department,
Ipswich, Suffolk, UK

ABSTRACT

It may be shown that, in certain types of repairable processing system employing modular redundancy security techniques, the existence of a number of fault patterns, which do not theoretically represent a system failure but in practice cause the system to fail, can severely degrade the reliability of the system as predicted by simple theory. Hardware design errors and software design errors also result in a number of failures that are not accounted for by simple theory; these will also degrade the system reliability. This paper examines all these types of failure in the case of a real-time control system and attempts to show how an analysis of the overall system reliability might take account of such failures. A theorectical comparison of two different classes of secure system - the parallel-operation type system and the load-sharing type system - is made, from the point of view of system reliability. Also the merits of two types of load-sharing sytem - the multi-computer and multiprocessor - are discussed.

INTRODUCTION

The usual way of organising a repairable processor system secured by means of modular hardware redundancy is to have a number of modules (which may be complete processors or smaller units such as CPUs and store units) arranged so that a certain number of modules may fail and there still be enough hardware available for the system to function satisfactorily. A module is taken to be failed when it contains any fault, and it is then removed from the system automatically. The system may then be said to have reconfigured. When a faulty module is repaired it is restored to the system. A total system failure only takes place, in theory, when repair of one or more faulty modules is not completed before further failures leave the system with insufficient working modules for it to function. An analysis of the failure rate due to such failures yields values such as those given by the approximate formulae in Table 1. These are derived assuming constant failure rates for each module and independent repair of each failed module with a constant repair rate which is much greater than the module failure rate. (The mean time to repair each fault is taken as τ. This is reciprocal of the repair rate.)

TABLE 1

Type of System	Failure Rate	Symbols
Duplicated processor	$2\tau\lambda_p^2$	λ_p = failure rate of each processor
Triplicated processor with majority voting	$6\tau\lambda_p^2$	λ_p = failure rate of each processor
Mulitiple computer	$(n-e+1)\binom{n}{n-e+1}\lambda_p^{n-e+1}\tau^{n-e}$	λ_p = computer failure rate n = number provided e = number essential for service
Multiprocessor	$(c-a+1)\binom{c}{a-1}\lambda_c^{a-1}\tau^{c-a}$ $+(d-b+1)\binom{d}{b-1}\lambda_s^{b-1}\tau^{d-b}$	λ_c = CPU failure rate λ_s = store module failure rate a,b = essential CPUs, stores c,d = CPUs, stores provided

These values for the system failure rate may be called the intrinsic hardware failure rates. They determine the minimum failure rate that can be achieved with a particular system configuration. The actual failure rate achieved in practice will be greater than these values since:-

(a) Individual faults in modules and certain combinations of faults in one module or different modules may be such as to make the system unable to function, even though the number of working modules is, in theory, able to constitute a working system.

(b) Hardware design faults and software design faults (bugs), when encountered, may cause the system to enter a state from which it cannot recover.

(c) The system may be unable to cope with the effects of transient faults and may thus get into a state from which it cannot recover.

It can be shown that [1], the result of (a) can be to add a very large extra failure rate to the intrinsic hardware failure rate. Also, experiences with real-time systems to date have indicated that (b) and (c) may have a significant effect on the system reliability.

FAILURE MODE CLASSIFICATION

If an attempt is made to classify failures according to their causes, then a confused picture results. Each failure is the result, either directly or indirectly, of some incident, for example, a transient fault, the encounter of a design fault in the hardware or software, or a reconfiguration arising out of a hardware fault. Every such incident, when not associated with an intrinsic hardware failure, still has a certain probability of leading to a system failure. Such probabilities are both a function of the incident itself and of the processes that are going on at the time of the incident. If, however, one attempts to classify failures according to the eventual failed state, then a slightly clearer picture emerges.

In a real-time control system it is reasonable to assume that the security mechanisms include both error containment facilities, acting within certain modules or in association with certain processes, and more drastic recovery mechanisms. The most extreme recovery mechanism might be a complete restart with a re-load of the main program from a backing store. Such a mechanism would not be acceptable in a batch processing system, but in a control system a complete loss of data may not be so serious. For example, in a telephone switching control system important data produced by the system while it is operating, such as billing information, could be recorded on magnetic tape as it is generated. Data about telephone calls in progress at the time of a re-load could be built up again afterwards by interrogation of the switching unit. In such a system the possible failed states may be classified as follows:-

(1) Enough hardware modules have failed for the system to be unable to function satisfactorily (ie. an intrinsic hardware failure).

(2) A set of faults has occurred which, while not theoretically constituting a system failure, makes it impossible to re-load and restart or impossible to run for any length of time without another re-load.

(3) All copies of the program, including the backing store one, have been lost or corrupted.

(4) All copies of essential backing data have been lost or corrupted.

(5) The mechanisms for detecting that the system is in need or a re-load have failed to recognise that the system is not functioning properly.

An important point emerges if these failure modes are examined. Failures of type 2 are characterised by a static phenomenon: that is, a particular state of the system hardware which (given particular system software) is not a workable state. But failures of type 3, 4 and 5 are characterised by a dynamic phenomenon: namely that (given the system software and hardware, the probabilities of various kinds of transient fault, and the frequency of reconfigurations due to hard faults) there is a certain probability per unit time of system operation that one of these failures will occur. System failures may thus be placed in three categories:-

A. Intrinsic hardware failures (Failure mode 1).

B. Other static failures. (Failure mode 2).

C. Dynamic failures. (Failure modes 3, 4 & 5).

If it is assumed that these contribute independently to the total system failure rate, then three corresponding failure rate terms can be defined: the "A" term, Λ_A, the "B" term, Λ_B, and the "C" term, Λ_C. The total system failure rate may be derived by combining these terms, taking account of the probability distributions involved. For example, in the simplest case of all three terms being constant, the result is the arithmetic sum.

DETERMINATION OF THE B-TERM AND C-TERM

The determination of the value of the A-term of the failure rate, although not trivial in the case of more complex systems, is fairly straightforward. It does not call for information on how the system actually performs, but only on how it is intended to perform. To determine the B and C terms, however, calls for the solution of two sets of problems. For the B-term the problem is always static in nature: given a particular state of the system (ie. a particular set of faults), to find out whether this represents a type 2 failure. For the C-term the problem is always dynamic: given a particular state of the system, to determine the probability per unit time of a failed state of type 3, 4 or 5 occuring.

To investigate the B-term, let alone the C-term, in any detail would be a formidable task. For example, consider a system consisting of two CPUs and two store modules, operating in some unspecified way such that one store module and one CPU may fail and the system still function satisfactorily. Suppose that, counting every possible fault on every component and connection, there are 10^6 possible faults that can occur in a CPU and 10^5 in a store module. This means that there are:-

2.2×10^6 single faults

and

2.42×10^{12} pairs of faults

in the presence of which the system is meant to continue to function satisfactorily. A [illegible] of B-term failures thus amounts to setting up each of these states and starting the system off from a re-load, then checking that it will run for a reasonable length of time without another re-load being initiated automatically. Should the system be unable to re-load in the first place, or keep re-loading at short intervals so that it can do no useful work, then a B-term failure mode has been identified. By summing the probabilities of all states of this kind one should obtain the value of the B-term.

For example, if the mean time to repair all faults is τ and faults are assumed to be repaired independently and at a constant rate, if n B-term states are identified and each one characterised by x_j faults with constant rates of occurrence λ_{ij} ($i = 1, 2 \ldots x$), and if $\tau \gg \lambda_{ij}$ for all i and j, then the B-term failure rate is approximately given by:-

$$\Lambda_B \simeq \sum_{j=1}^{j=n} (x_j \tau^{x_j - 1}) \prod_{i=1}^{i=x_j} \lambda_{ij}$$

To investigate the C-term one would have to run the system for a long time, exposing it to every combination of input data and transient faults for every possible state investigated for the B-term, including the fault-free state. The probability of occurrence per unit time of failures of type 3, 4 and 5 would thus be determined and multiplied by the probability of the system state in which they occur, then all the results summed to give the C-term.

For example, using a similar notation and making the same assumptions as above, if there are n states identified in which C-term failures do occur, these states being characterised by x_j faults with rates of occurrence λ_{ij} (again assumed to be constant) and the probability of a C-term failure occurring in such a state is γ_j per unit time, then the C-term failure rate is approximately given by:-

$$\Lambda_C \simeq \sum_{j=1}^{j=n} \left(\frac{\gamma_j x_j \tau^{x_j}}{\tau \gamma_j + x_j} \right) \prod_{i=1}^{i=x_j} \lambda_{ij}$$

It is hardly necessary to point out that investigations of the actual form just described are completely out of the question because of the time they would take. Unfortunately there is no simple answer to the problem of practical measurement of the B-term and C-term, but some of the following techniques may prove fruitful in the future:-

(i) Simulation

Logic-level simulation of the system being investigated could make it possible to try out different fault patterns in a very short time in order to help identify type 2 failures.

(ii) Sampling Techniques

It could be that the number of type 2 failures in a particular system is sufficiently large for the simulation approach or another approach to generate meaningful results when operated on a sampling basis rather than a full test basis. This could cut down the time taken by the investigation considerably.

(iii) State Characterisation

It might be possible to characterise the detailed states that are investigated for the B-term analysis by means of a number of macroscopic parameters. Then, by determining from a sample of states the relationship between these macroscopic parameters and the probability of failures of type 3, 4, and 5 occuring, the value of the C-term might be predicted from mean values of the C-term as found by summing the parameters of all states weighted according to the probability of the state.

EFFECT OF SYSTEM DESIGN ON FAILURE RATE TERMS

There are many different possible forms of processing system that could be used in a real-time control application. However, they can be divided into two general categories: parallel-operation systems and load-sharing systems. The essential feature of the parallel-operation type of system is that extra hardware modules, provided for security, are used to replicate the functions performed by the other modules. This replication of functions may be either active (ie. micro-synchronised or "match-mode" working) or on a time-shared basis (ie. worker/standby operation). The reconfiguration of parallel-operation systems, when a module fails, may be either active (as in the case of a duplicated system where the output of the faulty module is inhibited) or passive (as in the case of a triplicated majority voting system where the faulty module if automatically out-voted.) The essential feature of the load-sharing type of system, on the other hand, is that replicated modules share the total workload and thus perform different functions to one another. The reconfiguration of such systems consists of removing the faulty module from service and re-distributing the workload between the remaining modules.

Within the general category of parallel - operation systems there are a number of discrete types of system, distinguished by:-

(a) The extent of the modularity. (eg. individual CPUs and store units may be replicated separately or the basic processor may be replicated as a whole.)

(b) The order of replication of modules. (eg. duplication, triplication.)

(c) The way in which reconfiguration takes place. (eg. removal of failed modules, changeover to standby, majority voting.)

However, within the general category of load-sharing systems there is a spectrum of systems, the position of a particular system in the spectrum being determined by the extent to which CPUs share storage facilities. At one extreme of the spectrum is the "pure" multiprocessor, with all CPUs and store modules sharing a common highway. At the other extreme is the multi-computer, with each CPU having its own private storage and communicating with other CPUs over special data paths (or even via the entity being controlled by the system).

In order to discuss the effects of the system design on the different failure rate terms it is convenient to first consider the general effects of the system being either parallel-operation or load-sharing and then go on to consider the reliability of specific types of parallel-operation system and load-sharing system.

COMPARISON OF PARALLEL-OPERATION AND LOAD-SHARING SYSTEMS

Because the way in which redundant hardware is used, on a modular basis, determines the intrinsic hardware failure rate, or A-term, one tends to ignore the effect of another aspect of the system design on the system reliability - the use of operational multiplicity. Operational multiplicity is a means by which serveral processing functions may be carried out at once and its use in a system has an important effect on the B-term and C-term failure rates. Before discussing this in detail, a few points in connection with the A-term failure rates are perhaps worth mentioning.

The A-term failure rates of some well-known types of system were given in Table 1. The first important point to be noted is that the reliability of parallel-operation systems has a fixed limit determined by the type of system and the failure rates of the individual modules. Without completely re-structuring the system (eg. changing from duplication to triplicated-majority-voting) or re-engineering its modules, it is impossible to change either the reliability of the system or its processing power by taking away or adding hardware modules. The economic implication of this lack of modular expandability is very serious where the workload rises during the system's lifetime, as is also the more basic point that one is providing 100% or 200% redundant hardware in order to achieve reliability while obtaining only the processing capacity of a single machine. The load-sharing type of system, however, offers plenty of scope for expansion of both capacity and reliability. In fact, the two can be balanced against one another, an increased capacity being offered at a reduced level of security or vice versa. There is no fixed limit to either capacity or reliability imposed by the system design. Also, in spite of the fact that this type of system is more sophisticated, the costs of this sophistication tend to be offset by a smaller proporation of redundant hardware being needed for security as compared with systems using complete replication. Typically, a load-sharing system may have 50% redundant hardware provided for security, as compared with 100% for a duplicated system.

Looking next at the B-term failure rate, it seems likely that this will tend to be higher in a load-sharing system than in a parallel operation system. This is because the complex nature of the design of a load-sharing system is more likely to result in unexpected failure modes involving combinations of faults. The C-term failure rate, on the other hand, whilst tending to be contributed to by the complexity of the load-sharing system, will be mainly determined by the role of operational multiplicity in the system. In a parallel-operation system there is, by definition, no operational multiplicity; each processor (or CPU) performs the same operation at the same time using the same data. It follows that the whole system will be involved in every error that occurs, no matter what the cause. In a load-sharing system, however, a number of processes will normally be going on at a time, so that any errors arising in a particular process may be contained within that process, provided that error containment mechanisms are effective and that corrupt results from the process are not passed on to other processes. It follows that in only a small proportion of cases will the opportunity for a C-term failure arise. It is therefore likely that the actual number of C-term failures will be considerably less in a load-sharing system than in a parallel-operation system, where there are many more opportunities for C-term failures to occur.

A further consideration is the extent to which the system software can be debugged. Well-debugged software will result in lower C-term failure rates, and to some extent also B-term, so the additional capacity for self-monitoring that a load-sharing system has by virtue of its operational multiplicity can lead to a faster reduction in the number of bugs. There will therefore be a progressive improvement in the performance of such a system with respect to a parallel-operation system.

COMPARISON OF DIFFERENT PARALLEL-OPERATION SYSTEMS

The above comments apply more or less equally to all parallel-operation systems. The limit on the system reliability determined by the particular design (ie. the intrinsic hardware reliability, as in Table 1) is worse for systems where the extent of the modularity tends towards the replicated processor type of arrangement. If the system can be broken down into many individually replicated modules, then the A-term can be considerably reduced. For example, if a system has two duplicated modules with the same failure rate, by duplicating modules of each type independently, as opposed to taking the two modules of each type as the unit to be duplicated, the failure rate A-term may be halved. The order of replication and method reconfiguration also have a significant effect; the higher order, the better the reliability in general, but the use of majority voting brings out the worst in a replicated arrangement. For example, a triplicated majority voting system has an A-term which is three times worse than that of the equivalent duplicated system, even though it contains 50% more hardware.

COMPARISON OF DIFFERENT LOAD-SHARING SYSTEMS

Since there is a complete spectrum of load-sharing systems, it is convenient to compare the two extremes of the spectrum: the "pure" multiprocessor and the "pure" multi-computer. This should provide some indication of the effects on the failure rate terms of moving from one end of the spectrum to the other. This is quite a useful comparison since, in the real-time control field, the multi-computer type of arrangement has often been put forward as a form of load-sharing that avoids some of the security risks inherent in having such a high degree of inter-module interaction as occurs in the multiprocessor. The multi-computer provides a very good error containment mechanism in that individual CPUs work almost in isolation from the rest of the system. It is argued that this will cause the system to have lower B-term and C-term failure rates and hence the overall system reliability for the multi-computer type of arrangement is likely to be much better than that of an equivalent multiprocessor. However, this argument avoids the question of which multi-computer system is being compared with which multiprocessor system. To compare the reliability of two different systems one must make some assumptions about the ways in which the two systems are to be considered a match for one another (eg. equivalent cost, equivalent capacity).

In order to make a reasonable comparison, a multi-computer and a multiprocessor of the same workload capacity probably provide the best starting point. How this affects the relative costs of the two systems is hard to say; the extra complexity of the multiprocessor design is to some extent offset by the larger number of hardware modules in the multi-computer. The important point to be considered is the number of store modules in the two systems. Since each CPU in the multi-computer must have its own copy of the main programs as well as its own working space, the total number of store modules in the multi-computer will be greater than in a multiprocessor with the same number of CPUs.

Assuming that two systems with the same capacity will have the same number of CPUs and considering just the A-term reliability of the main-frame system, it should be possible to draw up a comparitive table of A-term failure rates for different sizes of system if the failure rates of the CPUs and store modules are known. Table 2 shows the A-term reliabilities (expressed as MTBFs, since these are easier to appreciate at first sight) for a multiprocessor type system and a multi-computer type system. It should be remembered, as mentioned previously, that there is really a spectrum of systems ranging from the multiprocessor to the multi-computer; a comparison between the extremes is made here in order to obtain some information about which end of the spectrum might be more desirable. The module failure rates assumed to derive the MTBFs were 3×10^{-4} per hour for the CPUs in both systems and 9×10^{-4} per hour for the store modules in both systems and the mean time to repair any fault was taken to be 5 hours. The number of modules of each type are shown in the form (E + R), where E = number of modules essential to handle workload, and R = number of additional modules provided to achieve security. These values were adjusted, in the multiprocessor case, to achieve an arbitrary MTBF target of 500 years. The number of stores per CPU in the multi-computer were derived by assuming that about 60% of the storage is used for program and fixed data.

TABLE 2

MULTIPROCESSOR			MULTI-COMPUTER		
CPUs	Store Modules	System MTBF	CPUs	Stores per CPU	System MTBF
1+2	1+2	2012 yrs	1+2	1	880 yrs
2+2	2+2	503 yrs	2+2	2	41 yrs
3+2	3+3	4532 yrs	3+2	2	16 yrs
4+2	4+3	2195 yrs	4+2	3	3 yrs
5+2	5+3	1216 yrs	5+2	3	2 yrs
6+2	6+3	737 yrs	6+2	4	0.5 yrs

It is clear from Table 2 that the large number of store modules needed in the multi-computer makes the A-term reliability very much worse than that of the equivalent multiprocessor and this difference between the B-terms and C-terms for the two types of

system may well be swamped for the larger system and the claims made for the multi-computer may thus be invalidated.

CONCLUSIONS

In processing systems employing modular redundancy security techniques, the true system failure rate is higher than that predicted by a theoretical analysis of how the system is intended to behave. This is due to the occurrence, in practice, of a number of failures in addition to intrinsic hardware failures. These have a number of different and complex causes, but can be described in terms of the failure mode that results. It has been suggested that the failure modes of a real-time control system can be grouped into three categories and the contribution to the total system failure rate from these categories described by an A-term, a B-term and a C-term failure rate. The A-term is the intrinsic hardware failure rate; the B-term is the result of certain combinations of faults which are not meant to cause a system failure in theory but which, in practice, render the system unworkable; and the C-term is the result of other failures which do not follow directly from the state of the system but which have a certain probability per unit time of occurring when the system is in a given state. Some of the difficulties in determining the values of the B-term and C-term failure rates have been outlined.

The effects of different security techniques on the values of the A, B and C terms have been discussed, and the special properties of the load-sharing type of system outlined: the scope for varying the value of the A-term, the ability to achieve a low A-term with a smaller proportion of redundant hardware, and the role of operational multiplicity in determining the B and C terms. The existence of a spectrum of possible load-sharing systems has been mentioned, ranging from the "pure" multiprocessor to the "pure" multi-computer and it has been shown that a multiprocessor type system may have considerable advantages over the multi-computer system when the system is a large one.

ACKNOWLEDGEMENTS

Acknowledgement is made to the Director of the British Post Office for permission to publish this paper.

REFERENCES

1. Arnold, TF, "The Concept of Coverage and its Effect on the Reliability Model of a Repairable System", Digest of Papers of 1972 International Symposium on Fault-Tolerant Computing, pp 200-204.