Redundancy and Fault Tolerance in Elevator System Design

by Jon B. Halpern - Millar Elevator Industries, Inc.

Abstract

Redundancy has been applied in elevator system design particularly to mechanical safety systems. With the advent of the microprocessor technology issues of reliability, redundancy and fault tolerance are more often considered and evaluated in elevator control system design, group design and additional safety systems. A great deal has been learned in the discipline of redundancy and fault tolerance in the aircraft, spacecraft and related industries. It is important to understand these principles of fault tolerance when designing or evaluating the performance of a prospective elevator system. I will review some of the major principles and methods to obtain and assess the level of fault tolerance in a system and apply it to elevator systems.

Introduction

Safety critical real time systems became important when designers began to design guidance control and navigation systems for spacecraft. The first method used was fault avoidance, avoiding fault by designing high reliability into a system through reduced component failure rates through rigorous quality control methods. This proved to be very effective for the early spacecraft systems, however in the case when a single failure was exhibited, the total system failed. Additionally there was a tremendous cost penalty for designing high reliability through reduced component failure rates. The microprocessor has made it possible to trade off some of these costs by trading off fault avoidance for fault tolerance. [1]

Today in the elevator industry there is discussion and claims of fault tolerance particularly with relation to group control systems. Microprocessor Elevator Control Systems have been in commercial use since the mid 1970's. Microprocessor systems in use today are capable of various levels of fault detection, fault reporting and remote monitoring. Some systems even make claim to having redundant components to improve reliability with back up group supervisory systems or distributed supervisory systems where the supervisory system is distributed throughout many of the different elements of the system such as the individual elevator controllers. Individual elevator car processors incorporating failure detection reporting and recovery systems, have not replaced the many mechanical safety systems incorporated in elevators however, they have supplemented these safety systems so that the mechanical/electromechanical safety systems are a system of last resort.

Because redundant fault tolerant systems are becoming more common in the industry we must all become more familiar with the basics principles and methods of achieving and assessing high reliability and fault tolerance.

Designing and evaluating fault tolerant systems is a very difficult complex task that requires a very structured design environment, a very rigorous validation process, and is very expensive. I will review some of these concepts of redundancy and basic reliability measures as well as providing some simple examples to provide a basic understanding of the complexity of redundant systems. I will review the methods used to obtain true fault tolerance and high reliability in any system. The simple claim of having a redundant peace of hardware is clearly not an indication of the fault tolerance or reliability of any system.

History of Fault Tolerant Systems

In the 1970's highly reliable safety and mission critical applications were required for space and military applications. These systems used various methods to attain a high level of reliability that included dual and triplex systems to tolerate random hardware faults that are supposed to occur independently in like hardware. These methods proved effective for this class of faults, however redundancy greatly complicated the task of validation. You could easily develop redundant control systems that were more prone to failure than simplex systems due to the complexity and additional number of components required for the design.

In the last 10-15 years rigorous methods have been developed to design, implement and validate hardware systems that are fault tolerant to single independent hardware faults. This tolerance is considered a Byzantine Resilient fault tolerant system.

Methods that address Byzantine type faults unfortunately completely ignore a second class of failure modes, known as Common Mode Failures . Common Mode Failures are where the same failure occurs simultaneously in multiple copies of a redundant system, due to a single cause. Unlike independent hardware faults, the cause of common mode failures are so diverse that numerous desperate techniques are required to predict, avoid remove and tolerate them. Solutions have ranged from design diversity, redundant pieces of hardware or software that are designed by different teams, to formal methods, however there is no silver bullet to the solution.

Software systems that incorporate real time systems with boundaries and task deadlines further complicate the problem.

The use of digital computers for safety critical applications was pioneered by NASA on the Apollo Mission. The Saturn V launch vehicle was controlled by triply redundant IBM computers. The Navy's F8 fighter aircraft used a triply redundant computer as a means of Digital Fly By Wire controls. This system although reliable did not meet the requirements of Byzantine Resilience. The Space Shuttle used quad-redundant computer system with fully cross strapped connectivity between the computers. This system came very close to pure Byzantine Resilience. Additionally NASA added a fifth computer to watch the quad-redundant computers. The software in the fifth was developed by a completely different design team. [1]

Today there are many systems in use that provide all levels of fault tolerance and redundancy within. These examples are know quite commonly seen in to days commercial aircraft systems such as the Boeing 747,757,767,777 and the Airbus a-320. Particularly today with the common use of Category III landing systems, that automatically land commercial aircraft unassisted by a human pilot. Fault tolerant systems are becoming more familiar and widely used.

Reliability Measures

Reliability R(T) - The probability that a component or system will be running at time T.

Failure Rate - The rate at which components fail in Failures per Unit Time.

Hazard Rate H(T)- Instantaneous rate of failure.

MTBF - Mean Time Before Failure.

Basic Principals of Reliability

I will review how to calculate the reliability of several components in series where the failure of one of the series components causes the failure of the systems, then a parallel system where the failure of all the elements are required for the failure of the system.

Series Components:

$$R_S = R_1 * R_2 * R_3 * \ldots * R_N$$

Parallel:

$$R_P = 1 - [(1- R_1) * (1 - R_2) * (1 - R_3) * \ldots * (1 - R_N)]$$

Series- Parallel:

$$R_{SP} = [R_1 * R_2 * R_3 \ldots] * [1 - (1 - R_8) * (1 - R_9) * (1 - R_{10}) * \ldots]$$

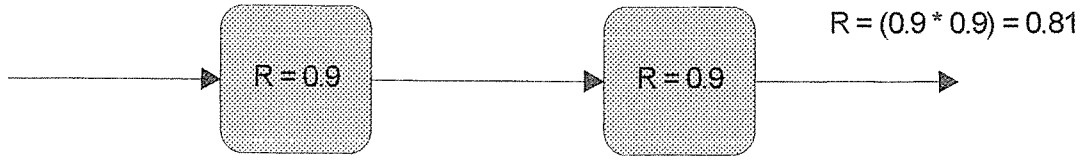
Complex Systems - Cut Set Method.

Very similar to the calculation of series parallel resistor circuits, the cut set method of calculating a cut set matrix and then computing a minimal cut set to determine the minimal number of components needed to fail for the system to fail. The series parallel system can be used to determine the reliability.

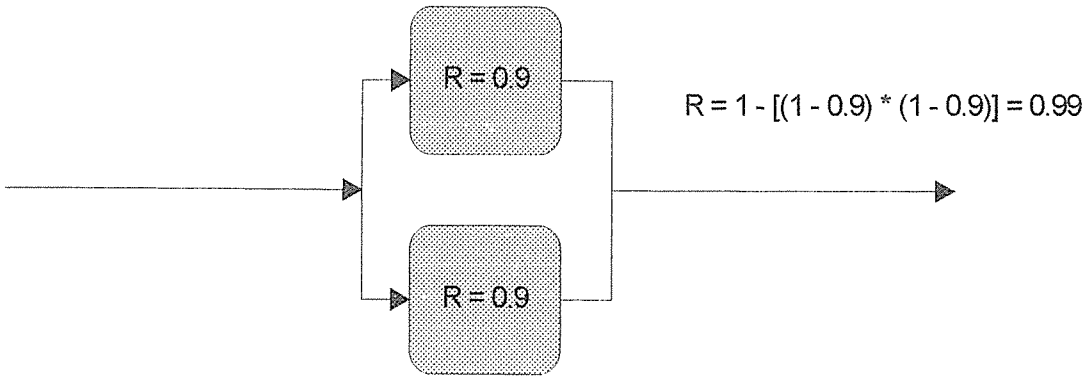
As a general rule parallel systems improve the reliability and series systems reduce the reliability. All complex systems are a combination of series and parallel systems with many modes of failure. The minimal cut set method finds the least reliable set of parallel series components that general drives the overall reliability of a system. As shown in figure 1, an example of a series, parallel and then a combination of series parallel are shown with there respective reliability's calculated. Please take special note that in the series parallel example the reliability is no better than any of the independent series elements and therefore the overall reliability of a system is no better than its weakest link.

Redundancy

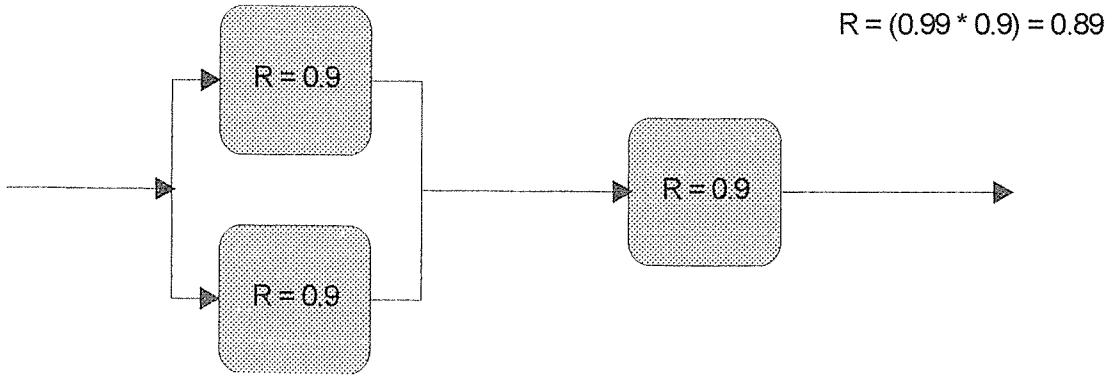
Redundancy alone does not guarantee fault tolerance. The only thing redundancy does guarantee is higher fault arrival rates compared to non-redundant systems of the same functionality. Redundancy management is the most important issue in determining a systems ultimate reliability and the performance penalties paid for redundancy. A fault tolerant computer can spend most of its computing power on the management of redundancy.



Series



Parallel



Series - Parallel

R = Reliability that block will be running at time T , R(t) = R

Figure 1.

Fault Containment Region (**FCR**) - A collection of components that operates correctly regardless of any arbitrary logical or electrical fault outside the region. [1] This usually requires total isolation from other FCR's.

The first step in addressing redundancy is to partition redundant elements into Fault Containment Regions with independent power and clocking sources. Additionally interfaces between FCR's must be electrically isolated and error containment to prevent erroneous data to propagate to other FCR's. This is usually realized through "voting planes" where errors are masked out. FCR's issue there output to the voting plane where a majority decision is calculated and applied to the output. In some instances the actual output may be the sum of outputs from FCR's and is dependent on the application.

Then finally the control actuators must be driven by multiple electrical or mechanical inputs so that the majority of the inputs can drive an actuator to its required position even if one or more of the inputs fail to one of its maximum values.

Byzantine Resilience

Byzantine Resilience - Is the resilience to arbitrary independent faults such as the failure of a single hardware component in a system. The resilient system will be completely operational when faced with this failure. In order to achieve Byzantine Resilience (**BR**), input congruency must be achieved in the presence if f arbitrary input faults. The following necessary conditions must be met to achieve **BR** and in affect fault tolerance to the f arbitrary fault.

1) The system must contain $3f + 1$ Fault Containment Regions **FCR's**.
2) Each **FCR** must be connected by $2f + 1$ disjoint paths for communication.
3) The inputs must be exchanged $f + 1$ times between **FCR's**.
4) **FCR's** must be synchronized.

Where f is the number arbitrary simultaneous faults that the system will be resilient to. Therefor a redundant system can reach exact consciences on a single arbitrary input fault if it uses four (4) independent fault containment regions connected by three (3) disjoint communication paths and exchanges the input information two (2) times between units. Quite complex.

Congruent Processes - Two or more redundant processes (computers) that are relying on the same inputs, operating in an identical manner are called congruent processes and follow a precise means for the detection of faults.

Fault Detection - When two or more congruent processes do not agree.

Fault Isolation - When a congruent process do not agree with the majority of congruent processes in a system.

Synchronization - Redundant channels or processes must be somehow synchronized so that inputs and outputs can be compared and faults can be detected.

Input Agreement - Input congruency occurs when each congruent process has a copy of a set of inputs and agree on their values. Input validity occurs when all channels have the correct value of the inputs and are not just in agreement.

Common Mode Failures and Fault Classification

Common Mode Failures are where the same failure occurs simultaneously in multiple copies of a redundant system due to a single cause. The source of common mode failures are diverse and difficult to prevent. Common Mode failures may be viewed according to their nature, we will consider only accidental faults as opposed to intentional faults:

1. Transient External Faults The result of temporary interference such as electro magnetic interference, lightning, heat.

2. Permanent External Faults - The result of heat, sand, dust, water, and physical damage.

3. Transient Internal Faults - The results of imperfections in specifications, design, software, which manifest themselves only part of the time.

4. Permanent Internal Faults - The same as transient internal faults but after time manifest themselves all of the time. [1]

Common Mode Fault Avoidance and Detection and Recovery

Techniques used to avoid common mode failures are complicated and involve may different methods all of which can reduce the likelihood of failure. These methods are not mutually exclusive and therefor must be considered in order to assure common mode fault avoidance.

Avoidance:
- Use of Mature and Formally Verified Components.
- Conformance to Standards
- Formal Methods - Mathematically based techniques for specifying and proving.
- Design Automation - Removing Human Error
- Simplifying Abstractions - Keep it simple
- Design Diversity - Confinement of design faults to a single Fault Containment Region.

Detection:
- Design Reviews
- Simulations
- Testing
- Watchdog Timers

Recovery:
- Hardware Exception Handlers
- Software Exception Handlers
- Memory Management Units
- Rollback or Roll forward Recovery and Re synchronization Methods

The use of these techniques again do not guarantee that common mode failures do not occur however they will certainly minimize the likelihood of such an occurrence.

Application to Elevator Group Supervisory Systems

Elevator group supervisory systems seem to be the first critical area by the industry and its customers when it comes to reliability. The past history of old relay control dispatch systems and the result of a single failure within this system making an entire bank of elevators non-functional is something that most building managers and owners may never forget. The simple transition of relay group supervisory systems to solid state microprocessors systems have improved the reliability of group supervisory systems by orders of magnitude, however a single failure can still result in a non-operational or severely degraded group of elevators.

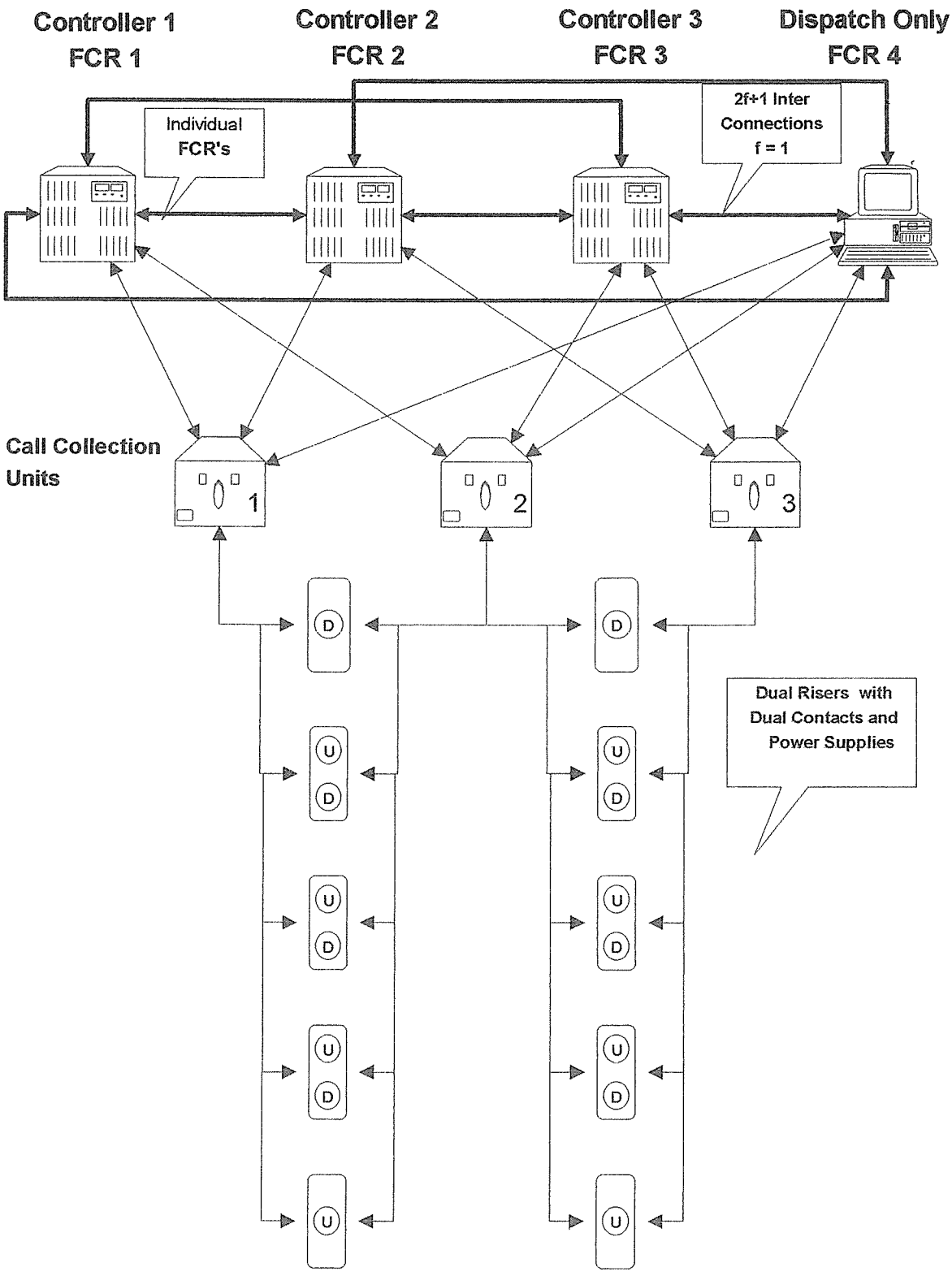
Due to commercial pressure we have seen systems that claim to be redundant or distributed with respect to the supervisory control. These systems do exist however it is questionable as to the level of fault tolerance these systems contain. Non-redundant well designed group supervisory systems could well be more reliable than complicated distributed or redundant systems as indicated in prior sections of this paper. Rarely are reliability figures published for elevator dispatch systems.

The complexity of the system as well as the installation environment play an important role in the true reliability, unlike an electronic component that is designed and manufactured in an environment that can be controlled. The field aspects of the elevator installation can clearly effect the overall reliability of a system. This not only includes the installation but also the environmental aspects of the building such as the temperature humidity, dust, dirt and electromagnetic compatibility.

Factors that clearly influence the effectiveness of a distributed elevator control system for group control are those components that are not redundant. First power supplies must be investigated thoroughly. For most machine rooms main AC power comes from a single source, this alone can be a source of failure or a source of fault introduction that violates fault containment regions. Other sources of common failure includes components such as data concentrators, hall call collection units, call push button risers, electro magnetic compatibility (electrical noise) and power supplies.

The introduction of a single fault into any one of these areas must be thoroughly investigated before the reliability of a system can be assessed. In figure 1, the example of a series parallel system, shows that the overall reliability of the system is based on the reliability of the individual parallel systems in series with each other. If any of the series component groups have a low reliability figure, the entire system will have a reliability measure that is at best equal to the reliability of this weakest group component or less.

Communication methods between redundant system components must also be assessed. In order to attain Byzantine resilience to a single independent fault there must be four independent fault containment regions, at least three disjoint paths of communication between fault containment regions and inputs of the system must be exchanged two times between



Example of a fault tolerant group supervisory system.

Figure 2.

regions for validation. The assessment of fault tolerance can be made in this case first by looking at the physical network between redundant components. If three disjoint paths of communications do not exist between fault containment regions then a fundamental necessary condition for Byzantine Resilience has been violated.

Common Mode Failure in redundant or distributed dispatch systems is a major area of concern. Dispatch algorithms alone are generally fairly complex and require usually a high level of computing power. Validation of non-redundant dispatch systems from a software perspective is a rather difficult task and the addition of redundancy further complicates the matter. To obtain design diversity a second dispatch algorithm would have to be designed by a second design team utilizing different hardware and software including different development tools. Common mode failures can be contained in the design tools used to develop a system such as a compiler or operating system or even a CPU of the same design.

In figure 2 a basic architecture of a Byzantine Resilient distributed group supervisory is shown. Redundant copies of the group algorithm must be contained in each FCR which in this case is three car controllers and a fourth independent dispatcher containing the same basic algorithm designed by a separate group of designers. This fourth FCR has completely different hardware and software as well as the operating system and compilers used to generate the operating code. Additionally, there must be complete electrical isolation of all FCR's from all inputs and outputs as well as other FCR's. It must follow all of the rules in the processing of inputs and output to achieve BR where the inputs are the collection of hall calls and the outputs are assignments of the respective hall calls to individual elevators. To achieve true fault tolerance the additional formal methods to eliminate common mode failures must be followed.

Application to Individual Elevators

The application of redundant elements to the design of individual car components are only necessary when designing safety systems and components. Clearly the reliability of most modern elevator car controllers clearly exceed the reliability of the electro-mechanical systems of an elevator, such as the doors and the shaft way contacts. It would be a waste of money and effort with very little gain to have redundant components in the most reliable system components. Money would be much better spent to improve the reliability of these electro-mechanical components before significantly improving the other elements. Remember that from a reliability standpoint a single door lock or similar component can take the system to failure, and the reliability level of a components such as these is much lower than the controller and therefor dictates the reliability of the system.

When it comes to safety critical circuits and components, this is an area that is worth redundancy from a prospective of , input validation, failure detection and accident avoidance and not necessarily continued reliability and operation.

Conclusions

The realization of true fault tolerant systems is a complex task. Prevention and detection of

both Byzantine and Common Mode faults requires complex designs, formal methods and rigorous testing. Due to the overall complexity of the complete elevator system and the general knowledge of the available systems in the market place I find it highly unlikely that a Byzantine Resilient fault tolerant elevator design exists today, even though there are many claims of redundancy and distributed control.

When assessing elevator control systems you must take into consideration all of the requirements of both Byzantine Resilience and Common Mode Fault Avoidance into the equation. Improperly designed fault tolerant systems can in the end be less reliable and more troublesome than well designed non-redundant simple systems.

Redundancy in elevator control systems will continue to be seen primarily in safety monitoring systems and possibly group supervisory systems. The reliability of an individual car controller is heavily dependent on the reliability of the many electro-mechanical components. Many of these components are required by law and the changing of the codes to allow more innovative methods is at times a difficult task.

Bibliography

[1] Jaynarayan H. and Harper R., "Architectural Principles for Safety-Critical Real-Time Applications", Proceedings of the IEEE, Vol. 82 No. 1, January 1994.