PAMUKKALE ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
PAMUKKALE UNIVERSITY ENGINEERING COLLEGE
MÜHENDİSLİK BİLİMLERİ DERGİSİ
JOURNAL OF ENGINEERING SCIENCES
YIL : 1998
CİLT : 4
SAYI : 1-2
SAYFA : 589-601

INVESTIGATION AND DESIGNING A FAULT-TOLERANT PROCEDURE FOR CONTROL COMPUTER SYSTEM

Rafig YAGUBOĞLU SAMEDOV, Ahmet ÇİFTÇİ
Pamukkale Üniversitesi, Mühendislik Fakültesi, Elektrik ve Elektronik Mühendisliği Bölümü, Denizli

ABSTRACT

Many computer systems have turned increasingly to control systems, requiring more sophisticated machinery over an ever-widening range. The reliability of the systems should be carefully considered in all its aspects. This paper analyses the structure of computer systems with redundancy and the types of faults which might appear in structure of these systems. This paper also describes the fault-tolerant procedures for computer systems with redundancy which counteract all forms of appearance of Nonbyzantine and Byzantine faults. The proposed mechanisms for execution of procedures support the fault-tolerance of redundant computer systems during degradation from N to 1. According to designed graphic model only part of procedures is executed depending on of absence, presence and sort of faults.

Key Words : Control computer system, Multi-computer system, Fault, Fault-tolerant procedure

BİLGİSAYAR KONTROL SİSTEMLERİ İÇİN ARIZA-KALDIRILABİLİR İŞLEMİN ARAŞTIRILMASI VE PROJELENMESİ

ÖZET

Son zamanlarda bilgisayar sistemlerinin geniş çapta kontrol alanlarında kullanılması çok mükemmel donanımı gerektirmektedir. Bu sistemlerin güvenilirliği tüm detaylarıyla dikkatli bir şekilde incelenmelidir. Bu makalede yedeklenmiş bilgisayar sistemlerinin yapısı ve bu yapıda ortaya çıkabilecek arızaların tipleri analiz edilmektedir. Aynı zamanda bu makalede yedeklenmiş bilgisayar sistemlerinde ortaya çıkabilecek arızaları etkisiz hale getirebilecek arıza-kaldırılabilir işlemler tanımlanmaktadır. Bu işlemleri yapmak için önerilmiş mekanizmalar yedeklenmiş bilgisayar sistemlerinin arıza-kaldırılabilirliğini N sayıdan 1'e kadar azalma seviyelerinde sağlamaktadır. Projelenmiş grafik modeline uygun olarak, arızaların yer alıp almamasına ve türüne göre işlemin ancak bir kısmı yapılmaktadır.

Anahtar Kelimeler : Bilgisayar kontrol sistemi, Çok bilgisayarlı sistem, Arıza, Arıza-kaldırılabilir işlem

1. INTRODUCTION

It is a well known fact that all the moving objects such as planes, satellites, space ships and other kinds of objects such as atomic power stations, continuous technological process etc. operate through using the control system. The basic function job of control system is executed by using the digital computer system. It is also known that if any fault occurs in a digital computer system then control system will be out of operation. In it turn will cause great losses in information, economy, ecology disasters (Avizienis, 1978).

Existing up-to-day technologies don't allow us to create absolutely reliable components for digital computer systems. Digital computer systems are designed and created by using existing components which don't provide the necessary reliability. Namely, the reliability of digital computer systems

for above mentioned control systems must not be less than P = 0.999 (the probability of faultless operation), (Hopkins et al., 1978). Such a high value for probability is too difficult (or maybe impossible) to reach by using the classical methods. For this purpose it is necessary to investigate and design the special methods for supporting the fault-tolerance of digital computer systems. In other words, it is necessary to develop the methods which allow to create the digital computer systems with required reliability by using the less reliable components. One solution to this problem is the creation of the fault-tolerant digital computer systems.

It is clear that the creation of the fault-tolerant digital computer systems is a very serious and complex problem. For this a lot of questions have to be solved. For example, investigation and designing of

1. The architecture,
2. The fault-tolerant procedure,
3. The real-time problems,
4. The hardware support of fault-tolerant procedureof the digital computer systems. In this paper only one of them was solved. Namely, a fault-tolerant procedure of digital computer systems is investigated and designed.

2. INVESTIGATION OF THE DIGITAL COMPUTER (MULTI-COMPUTER) SYSTEM

2. 1. The Analysis of the Structure of the Complex Control System

First of all define the place of the digital computer (multi-computer) system in the structure of the complex control system. Figure 1 shows the structure of the complex control system. It consists of the remote and automatic control sensors, systems and executive devices.

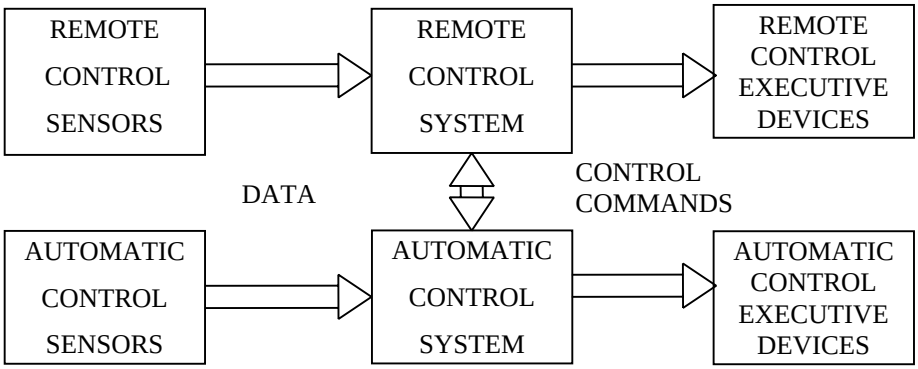


Figure 1. The structure of the complex control system

Remote and automatic control sensors are the data sources. They form the data about environment of the control objects and send them to the remote and automatic control system. These systems are the main elements of the complex control system. They receive and process the data, form and send the control commands to the remote and automatic control executive devices. These devices are the final control elements. They change the environment of the control objects by execution the control commands. Obviously the most important functions are executed by the remote and automatic control systems. Figure 2 shows the structure of the remote and automatic control systems.

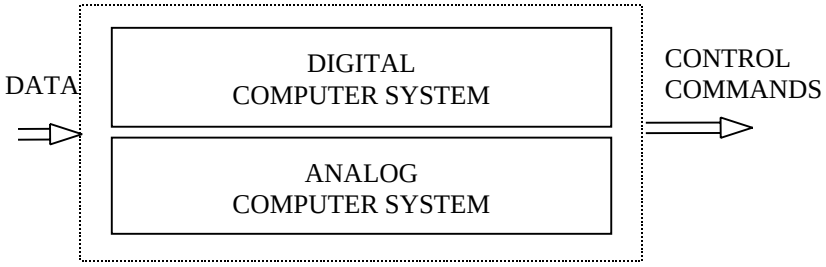


Figure 2. The structure of the remote and automatic control systems

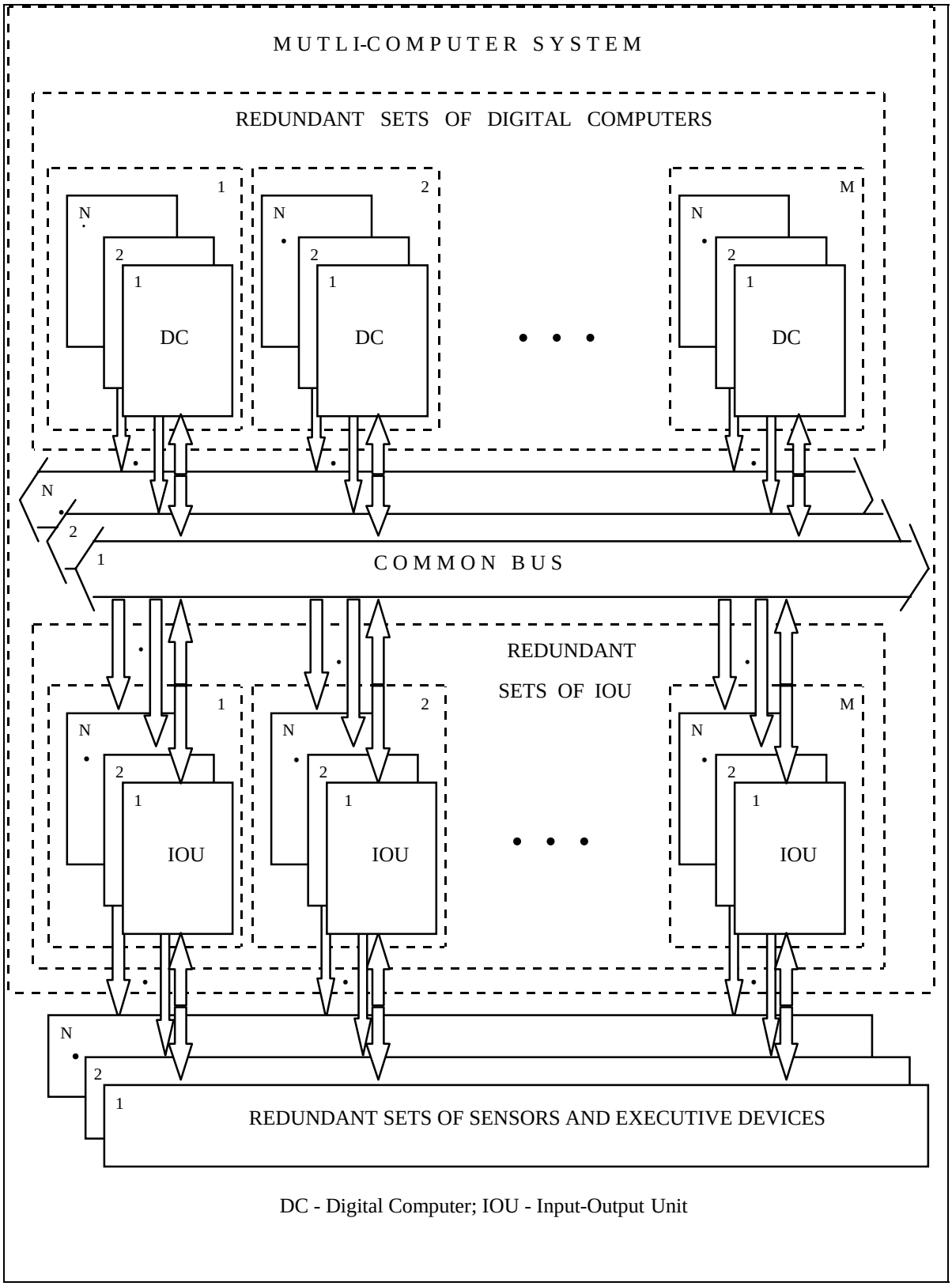


Figure 3. The structure of the digital computer system (Multi-computer system)

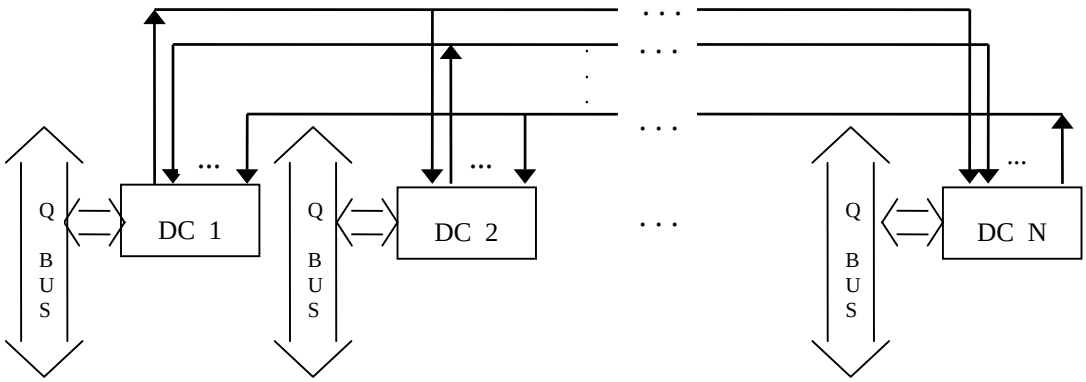
These systems have the same structure which consists of digital and analog computer systems. The basic function job of control system is executed by using the digital computer system. Figure 3 shows the structure of the digital computer system. This is the multi-computer system.

2. 2. The Analysis of the Structure of the Multi-Computer System

The structure of the multi-computer system consists of S sets of digital computers and input-output units. The interchange between all kinds of sets is executed by common bus. The environment of the multi-computer system is S sets of sensors and executive devices. At first the data enter from the sets of sensors through the sets of input units to the sets of the digital computers. Next the sets of digital computers process these data and form the control commands. At last the control commands enter from the sets of digital computer through the sets of output units to the sets of the executive devices. At the same time the multi-computer system execute S application tasks. It means that all sets of digital computers execute the different application tasks. In other words, each set of digital computers controls only one set of sensors and executive devices. All kinds of sets and buses are redundant. This is necessary to support the required reliability and fault-tolerance of the multi-computer system. The redundant elements (digital computers, input-output units, etc.) which belong to same set execute the same operations. The level of redundancy (the number of redundant elements in the sets) depends on the required probability of faultless operation for multi-computer system.

2. 3. The Analysis of the Structure of the Redundant Set of the Digital Computers

Figure 4 shows the structure of the redundant set of the digital computers.



To other sets of digital computers (DC) and input-output units (IOU)

Figure 4. The structure of the redundant set of the digital computers

All digital computers in the sets have the same architecture, namely, SISD (single instruction and single data) architecture (Flynn, 1972). The set is the group of the redundant digital computers which perform the same single instruction flow on the base of the same single data flow. The sets may be formed of double, triple, ..., N-multiple redundant digital computers. The level of the redundancy (L) is defined as the proportion of the general number of the redundant digital computers (N) to the minimum number of the necessary nonredundant digital computers (R) : L = N/R. R is determined by the volume of application task and the limitation of real time. In other words, R is the minimum number of nonredundant computers which can execute the certain volume of application task in real time. In this paper we assume that R = 1. During the computational process the faults appear in the set of the digital computers.

The fault-tolerant procedure localize and close the faulty computers. In the result N is decreased or the set of the digital computers is degraded. But application task is executed in full volume. The degradation level (D) is defined as the proportion of the number of the nonfaulty computers (E), where $E \leq N$, to the R : D = E/R. There are two groups of the degradation levels :

1. Normal degradation levels;
2. Critical degradation level.

In this paper we consider the sets of the digital computers which degrade from N to 1, where $N \geq 4$. Normal degradation level is when the set of digital computers executes the application task in full volume. Each set may have (N-1) normal degradation levels. Critical degradation level is when the appearance of another fault will be reason of the failure of the set of digital computers. Each set may have only one critical degradation level.

The computers in the set might operate in synchronous and asynchronous modes. The synchronous mode is when all of the computers in the same set start and finish the execution of the identical applications on the base of same data simultaneously. The asynchronous mode is when all of the computers in the same set execute the identical applications on the base of the different data. In addition they can start and finish execution of the identical applications in different time moments (the start and end of the execution for the identical applications in different computers may be slipped in the time).

The fault-tolerant procedures assume that every computer in a set of directly linked computers with redundancy executes the identical applications and the computation results are representable by a single value. The computation results of each job generated in all computers through communication of data between the computers form the initial data set (IDS) which is used by the every computer to check the state of the entire system. In the absence of faults, the values of all the IDS elements agree, i.e. they are either equal (for synchronous mode) or fall within the range of admissible deviations (for asynchronous mode). In the presence of a fault, the values of one or several elements differ from the other elements (disagreement appears). Fault-tolerance is a procedure by which all the normally operating computers in the set simultaneously and unambiguously identify the faulty computer and decide what to do with it.

2. 4. The Analysis of the Structure of the Operating System for the Multi-Computer System

Figure 5 shows the structure of the operating system that consists of the supervisor, gipervisor and kernel.

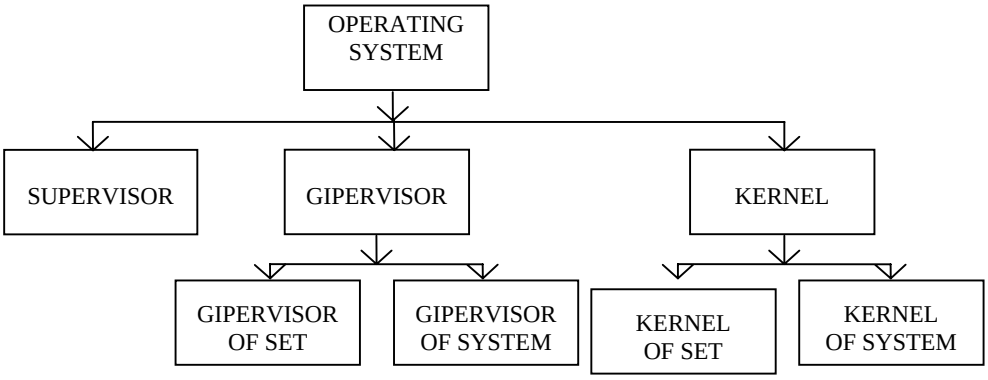


Figure 5. The structure of operating system for multi-computer system

The supervisor is a part of the operating system that controls the application tasks, the exchange between digital computers, IOU, sensors and executive devices. The supervisor also initiates the execution of the gipervisor. The gipervisor consists of the gipervisors for sets and system. First of them is the part of the operating system that realizes the fault-tolerant procedure of the sets of digital computers. Whereas the gipervisor of the system is also the part of the operating system that realizes the fault-tolerant procedure of the multi-computer system on the whole. The kernel consists of the kernels for sets and system. The kernel of the sets of digital computers is the part of the operating system that executes the functions of connections between the supervisor of digital computers and the gipervisor of the sets. The kernel of the sets also synchronizes the digital computers in the same sets. Whereas the kernel of the system is also the part of the operating system that executes the functions of connections between the supervisor of digital computers and the gipervisor of the system.

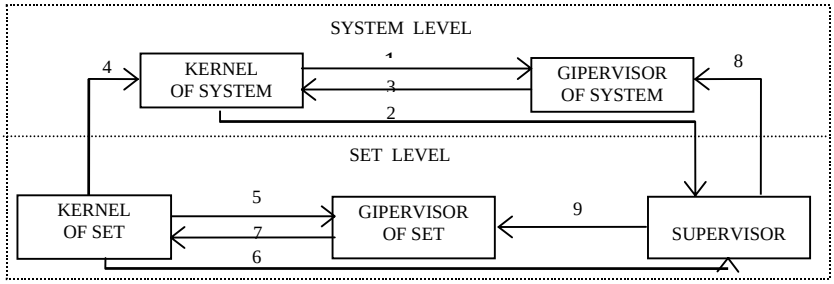


Figure 6. Functional connections between elements of operating system

Figure 6 shows the functional connections between elements of the operating system of the multi-computer system and Table 1 gives the meaning of these connections.

Table 1. The Meanings of Functional Connections Between Elements of Operating System

Designation	Content
1	Synchronization of interchange between sets. Initialization of gipervisor for system
2	System reconfiguration after failure of set of digital computers
3	Formulation of fault parameters for sets of digital computers and buses of interchange between sets
4	Initialization of fault of set
5	Synchronization of interchange between computers in same sets of digital computers
6	Request for realization of gipervisor for set of digital computers
7	Formulation of fault parameters for computers in set and buses of interchange between computers
8	Initialization and realization of gipervisor for system
9	Initialization and realization of gipervisor for set of digital computers

2. 5. The Analysis of the Structure of the Computational Process

The computational process executed in each computer consists of a number of operating cycles (Figure 7a). Each operating cycle consists of M logical segments (LS). In each LS one or certain number of application tasks are executed. After each LS the check point (CP) is realized. In each CP the fault-tolerant procedure is executed.

The computational process is periodically interrupted at CP by the execution of the fault-tolerant procedure. In each computer the computational process is formed by the execution LSs and CPs. As has been noted the computation results of execution LSs are representable by a single value. In CP all computers of the same set realize the interchange by the computation results which form IDS that consists of N elements (computation results). IDS is used by every computer for execution the fault-tolerant procedure.

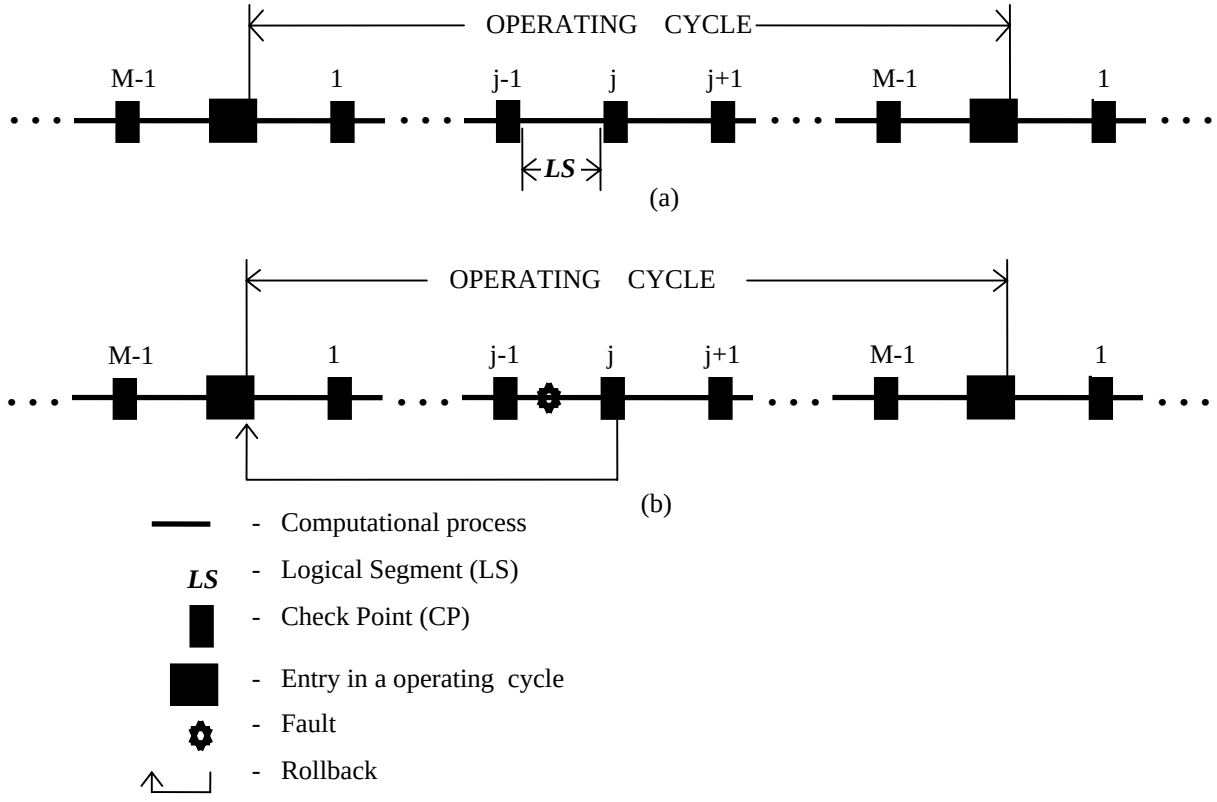


Figure 7. The structure of the computational process

3. DESIGNING OF THE FAULT-TOLERANT PROCEDURE

3. 1. The Analysis of the Faults

In computer systems with redundancy, we distinguish between Nonbyzantine and Byzantine faults according to their effect. Nonbyzantine faults cause the faulty computer to behave in a fixed manner relative to the normally operating computers. Byzantine faults cause transmission of random results from the faulty computer. As an illustration consider the set which consists of four computers (Figure 8).

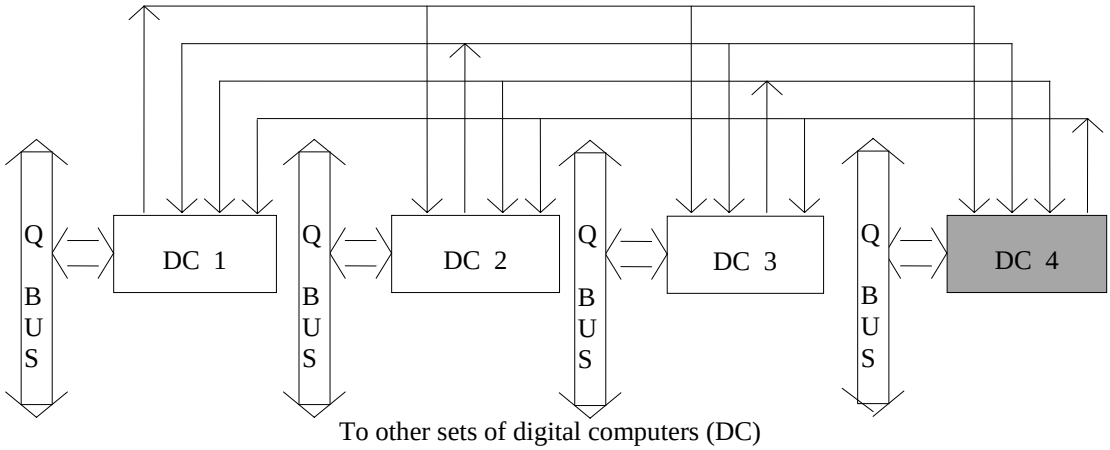


Figure 8. The structure of redundant set of digital computers

Suppose that fourth computer is faulty and the computation results of nonfaulty computers are “1”. If during the exchange by computation results fourth computer sends to all others the same result, namely, logical “0” it means that the fault is Nonbyzantine.

The Numbers of Computers	The IDS Formed in Suitable Computers
1	[1 1 1 0]
2	[1 1 1 0]
3	[1 1 1 0]
4	[X X X X]

As we see in all nonfaulty computers IDSs consist of same values. However if fourth computer sends to others the different computation results, namely, to first and third logical “0” and to second logical “1” it means that the fault is Byzantine .

The Numbers of Computers	The IDS Formed in Suitable Computers
1	[1 1 1 0]
2	[1 1 1 1]
3	[1 1 1 0]
4	[X X X X]

As we see in all nonfaulty computers IDSs consist of different values. The index of X (“0” or “1”) refers to values in faulty computers.

Each type (Nonbyzantine or Byzantine) of faults may occur either due to malfunction or due to failure. Malfunction is the instantaneous destruction of logical series for the tasks solving. For example, splashes in output voltage may result in the malfunction. Failure is the permanent destruction of logical series for the tasks solving. For example, short circuited computer components may result in the failure.

The type of faults depends on the modes of intercomputer communication. If the mode is broadcast only Nonbyzantine faults may appear in set of computers. On the other hand if the mode is time sharing both Nonbyzantine and Byzantine faults may appear in set of computers.

To sum up the appearance either Nonbyzantine or Byzantine faults depends on the structure of the set of computers. So the fault-tolerant procedure must be orientated on the concrete type of faults.

3. 2. The Analysis of the Types, Versions and Steps of the Fault-Tolerant Procedure

As has been noted two types of faults might appear in the structure of a multi-computer system. The approaches, consequently, the fault-tolerant procedures to counteract each type of faults are different.

There are three types of fault-tolerant procedures with different possibilities (Table 2).

Table 2. The Types, Versions, Steps and Mechanisms of Fault-Tolerant Procedures

The steps of the Fault-Tolerant Procedure	The types of the fault-tolerant procedure							
	1				2		3	
	The standard fault-tolerant procedure (it counteracts only all Nonbyzantine faults).				The mixed fault-tolerant procedure (it counteracts almost all Nonbyzantine faults and part of Byzantine faults).		The perfect fault-tolerant procedure (it counteracts all Nonbyzantine and Byzantine faults).	
	The versions of the fault-tolerant procedure.							
	A	B	C	D	E	F	G	H
	The number of computers in the set of digital computers							
	N = 2		N ≥ 3		N ≥ 4			
	The modes of interaction between computers in the set (Synchronous-S, Asynchronous-As)							
	S	As	S	As	S	As	S	As
	The mechanisms of the fault-tolerant procedure.							
1. Fault detection.	Each computer in the set detects the fault, independently of all other computers, by checking the values of all vector elements for equality (for synchronous mode) and falling within the range of admissible deviations (for asynchronous mode).				After execution a Byzantine Agreement Algorithm each computer in the set detects the fault, independently of all other computers, by checking the values of all vector elements for equality (for synchronous mode) and falling within the range of admissible deviations (for asynchronous mode).		After execution a Byzantine Agreement Algorithm each computer in the set detects the fault, independently of all other computers, by checking the values of all matrix elements for equality (for synchronous mode) and falling within the range of admissible deviations (for asynchronous mode).	
2. Fault localization (definition a number of faulty computer).	Each computer in the set identifies a number of faulty computer by the criterion of the disagreement between the constant and own computation result.	Each computer in the set identifies a number of faulty computer, independently of all other computers, by the diagnosis routine.	Each computer in the set identifies a number of faulty computer, independently of all other computers, by the criterion of the disagreement the values of all vector elements with the majority of values (for synchronous mode) and the range of admissible deviations (for asynchronous mode).				Each computer in the set identifies a number of faulty computer, independently of all other computers, by the form of appearance of all types of faults on the matrix.	
3. Definition of the sort of the fault types (malfunc- tion or failure).	Each computer in the set defines the sort of fault types, independently of all other computers, by counter of faults or by clock for measurement of fault appearance time.							
4. Recovery of a computational process after malfunction.	Each computer in the set recovers the computational process, independently of all other computers, by the rollback.		One of the nonfaulty computers recovers the computational process in the faulty computer by sending the majority of values for synchronous mode (the median of values for asynchronous mode) or by coping the memory (RAM) - back-up copy.					
5. Reconfiguration of a set of computers after failure.	Reconfiguration of the set of computers is executed by self-closing of the faulty computer.		Reconfiguration of the set of computers is executed by self-closing of the faulty computer or the nonfaulty computers close the faulty computer.					

1. The standard fault-tolerant procedure,
2. The mixed fault-tolerant procedure,
3. The perfect fault-tolerant procedure,

The standard fault-tolerant procedure (Avizienis, 1978) was designed to counteract only Nonbyzantine faults. To counteract of the Byzantine faults is a very difficult problem. For Byzantine faults, a Byzantine Agreement Algorithms were developed (Pease et al., 1980; Lamport et al., 1982; Choar and Coan, 1985) independently of the standard procedure. These Byzantine Agreement Algorithms or BAA, ensure that various normally operating computers receive the same data from the faulty computer in the presence of Byzantine fault. The use of these algorithms and their modifications in combination with the procedure of (Avizienis, 1978) leads to the mixed fault-tolerant procedure (Lala et al., 1986) that counteracts almost all Nonbyzantine faults (with the exception of faults that appear only in the second

round of the communication between computers) and part of Byzantine faults.
The perfect fault-tolerant procedure was designed on the base of the form of appearance of a fault (Samedov et al., 1992). By the form of appearance of a fault (FAF) we mean the character of the position where the disagreement is located in the IDS of all nonfaulty computers generated at the same CP. This procedure counteracts all Nonbyzantine and Byzantine faults.

The approaches used in (Avizienis, 1978; Lala et al., 1986) ensure that the fault-tolerant procedure is executed in its completeness at the same CP where the appearance of the fault is detected. Let N be the total number of computers in the set and k the number of faulty computers. The Nonbyzantine faults are localized when $N \geq 2k+1$ (Avizienis, 1978).

After the IDS is created, a single round of communication is performed and each nonfaulty computer generates a vector of N elements in which the locations of the disagreements match the indices of the faulty computers (Avizienis, 1978). All forms of appearance of Byzantine faults can be localized only when $N \geq 3k+1$ (Pease et al., 1980; Lamport et al., 1982; Choar and Coan, 1985). In this case the IDS has a much more complex structure, in particular multidimensional and sparse, so that position of the disagreements in various nonfaulty computers do not match and are no longer uniquely related with the indices of the faulty computers. The approaches used in (Samedov et al., 1992) accumulate information about the FAF which is sufficient for classifying it in one of three groups. A certain localization algorithm is used for each group. The types of the fault-tolerant procedure are divided into versions according to the value of N (the total number of computers in the set) and the modes of interaction between computers in the set (synchronous and asynchronous) (Table 2). The standard fault-tolerant procedure is divided into four versions A, B, C, D, the mixed fault-tolerant procedure is divided into two versions E, F and the perfect fault-tolerant procedure is also divided into two versions G, H. The versions of the fault-tolerant procedure are formed from the steps which execute certain function for counteracting the fault. Each version consists of five steps (Table 2):

1. Fault detection,
2. Fault localization (definition a number of faulty computer),
3. Definition of the sort of the fault types (malfunctions or failure),
4. Recovery of the computational process after malfunction,
5. Reconfiguration of a set of computers after failure.

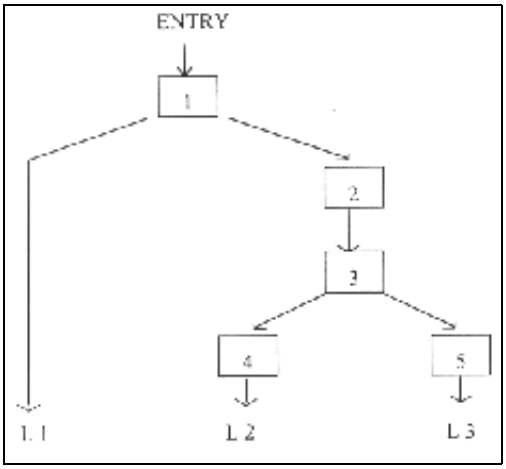


Figure 9. Graphic model of the fault-tolerant procedure

Figure 9 shows the graphic model of the fault-tolerant procedure. The numbers used in model are the numbers of steps for fault-tolerant procedure. There are three logical branches :

A hardware and software which realize any step of the fault-tolerant procedure are called mechanism. The mechanisms which realize different steps of the same versions are different. The mechanisms which realize the same step of the different versions may also be different (Table 2).

Thus the fault-tolerant procedure is executed by the gipervisor of the operating system in CP on the base of the IDS of computation results of the LSs. There are three types of the fault-tolerant procedure for the sets of computers with different possibilities. All versions of types of fault-tolerant procedures have five steps which are realized by suitable mechanisms. The fault-tolerant procedure is executed in each CP independently of absence, presence and sorts of faults. Depending on the absence, presence and sort of faults only one of logical branches is executed. It means that in each CP the different number of mechanisms is executed.

4. DESIGNING OF THE MECHANISMS FOR REALIZING THE STEPS OF THE FAULT-TOLERANT PROCEDURE

4.1. Fault Detection Mechanism

This mechanism is a very important part of the fault-tolerant procedures because it is executed in each CP independently of absence, presence, types and sorts of faults (Figure 9). Fault detection mechanism is executed on the base of IDS of computation results of LS formed in CP (Figure 7). Number of IDS elements is a number of computers in set. It means that each computer contains its own computation result and the computation results of other computers in the set.

We will now give the rules of formation of the IDS structure for all types of the fault-tolerant procedures of a single fault (k=1). A single fault is an FAF such that the probability of occurrence of another fault during the execution of the procedure is negligibly small.

4.1.1. Formation of the IDS Structure for the Standard Fault-Tolerant Procedure

Let n be the current index of the nonfaulty computer generating the IDS (n = 1, 2,...,N). The indices of the computers interacting with the computer n in each round of communication will be denoted by j (j = 1, 2,..., N, j ≠ n). The IDS generated in computer n after first round is represented by the following vector D_n :

$$D_n = \{ a_{11}^n, \ldots, a_{jj}^n, \ldots, a_{NN}^n \},$$

where the elements a_{jj}^n are the computation results of all computers j received by computer n.

4. 1. 2. Formation of the IDS Structure for the Mixed Fault-Tolerant Procedure

The BAA is executed. BAA reduce all possible FAFs at the same CP to the same form as for Nonbyzantine faults. For execution of BAA it is necessary to have $N \geq 3k + 1$ computers in the set and to perform m = (k+1) round of communication (Pease et al., 1980; Lamport et al., 1982; Choar and Coan, 1985). As has been defined k = 1, consequently, m = 2.

After first round of communication the IDS generated in computers n is represented by the vector D_n . The computers whose computation results are transmitted in the first round to computer j and then in the second round from computer j to computer n will be indexed i (I = 1, 2,..., N, I ≠ j, I ≠ n). After second round the IDS generated in computer n is represented by the following matrix A_n:

$$A_n = \begin{bmatrix} a_{11}^n & a_{12}^n & \bullet & \bullet & \bullet & a_{1N}^n \\ a_{21}^n & a_{22}^n & \bullet & \bullet & \bullet & a_{2N}^n \\ \bullet & \bullet & \bullet & a_{nn}^n & \bullet & \bullet \\ a_{N1}^n & a_{N2}^n & \bullet & \bullet & \bullet & a_{NN}^n \end{bmatrix}$$

- The elements a_{jj}^n on the main diagonal are the computation results of all computers j received by computer n in the first round,
- The n-th row and n-th column contain a single element a_{nn}^n which corresponds to the computation result of computer n,
- The i-th row contains the computation results of the same computer i received from all computers j,
- The off-diagonal elements in the j-th column are the computation results of all computers i received from the same computer j in the second round.

The matrix elements generated in nonfaulty computers n and j are related by the following equalities:

$a_{ij}^n = a_{ii}^j$ (i, j, n = 1,2,...N, j ≠ n, i ≠ n, i ≠ j),

$a_{jj}^n = a_{jj}^j$.

FAFs are analyzed using data sets of three kinds:

$D_n = \{a_{ii}^n\}$, where i = 1, 2,..., N (the main diagonal of A_n),

$R_n(i) = \{a_{ij}^n\}$, where i = const, j = 1, 2,..., N, j ≠ n (i-th row of A_n),

$C_n\{j\} = \{a_{ij}^n\}$, where j=const, i = 1, 2,..., N, I ≠ n (j-th column of A_n).

At the end of BAA transformation in j-th column $C_n(j)$ of A_n the majority of values is chosen by majority voting:

$a_j^n = maj\{C_n(j)\}$, j = 1, 2,..., N, j ≠ n.

Element a_{nn}^n is chosen as majority of values in the n-th column, i.e.

$a_n^n = a_{nn}^n$.

After execution of BAA in the n-th computer the following IDS is formed:

$B_n = \{a_1^n, \ldots, a_j^n, \ldots, a_N^n\}$, j=1,2,...,N.

Under BAA transformation, the value of each B_n element is generated by majority voting, i.e., it is equal to the value of the majority of agreements in the C_n (j). As a result, the majority voting procedure hides the FAFs for which the number of disagreements is less than one half of the total number of elements in a group, i.e., these FAFs may build up and ultimately lead to system failure.

4.1. 3. Formation of the IDS Structure for the Perfect Fault-Tolerant Procedure

For formation of the IDS structure, it is also necessary to execute the BAA. But in this case the majority voting procedure is not executed, i.e., the B_n is not formed. The perfect fault-tolerant procedure is realized on the base of $R_n(i)$ and $C_n(j)$ of A_n.

Thus the fault-tolerant procedure will be executed on the base of following IDS structures:

- Vector D_n - for standard procedure,
- Vector B_n - for mixed procedure,
- Matrix A_n - for perfect procedure.

Let's consider the fault detection rules by using these IDS structures. The fault detection mechanism on the base of vectors is executed only on the computer level, but on the base of matrix - on the computer and set levels.

4. 1. 4. Fault Detection Rules on the Base of Vectors

1. On the base of D_n or B_n n-th (n = 1, 2,..., N) computer choose the majority of values by majority voting for synchronous mode and the median of values according to median algorithms for asynchronous mode.
2. n-th (n = 1, 2,..., N) computer detects the fault by checking for equality all D_n or B_n values with majority of values for synchronous mode and for falling within the range of admissible deviations of disagreement of all D_n or B_n values with median of values for asynchronous mode.

So in the absence of faults, the values of all the D_n or B_n (n = 1,2,...,N) agree, i.e., they are either equal or fall within the range of admissible deviations. In the presence of fault, the values of the one or several elements differ from the other elements (disagreement appears).

4. 1. 5. Fault Detection Rules on the Base of Matrix

1. The operations which are executed on the computer level.

Each C_n (j) (n = 1, 2,..., N, $n \neq j$) of A_n is considered as vector. Then according to fault detection rules on the base of vectors (see above) the fault is detected. If there is no fault then the cod of absence, in contrary case, the cod of presence of fault is formed.

2. The operations which are executed on the set level.

The BAA is executed for formed cods. As a result, all computers will have the same vector of formed cods.

If all values of vector are same there is no fault in the set of computers, in contrary case, there is fault in the set.

Thus the fault detection is realized by following mechanisms (Table 2);

1. IDS has vector structure: by checking the values of all IDS elements for equality (for versions A, C, E),
2. IDS has vector structure: by checking the values of all IDS elements for falling within the range of admissible deviations (for version B, D, F),
3. IDS has matrix structure: by checking the values of all matrix elements for equality (for version G),
4. IDS has matrix structure: by checking the values of all matrix elements for falling within the range of admissible deviations (for version H).

Finally let's analyze the volume of executed operations for fault-detection mechanism. For versions A, B, C, D, the operations for fault detection are only executed. For versions E, F the operations for BAA and fault detection are executed. For versions G, H, the operations for two BAA and fault detection are executed. For versions A, B, C, D only one, for versions E, F two and for versions G, H four rounds of communication are executed. So according to the volume of executed operations, all versions of fault tolerance procedure may be arranged by increase: A, B, C, D, E, F, G, H.

4. 2. Fault Localization Mechanism

Fault localization mechanism is executed in presence of faults and enter to L2 and L3 of graphic model of the fault-tolerant procedure. The aim of this mechanism is to find the faulty computer in the set. We will now consider the mechanism which realize this aim. All mechanisms are executed on the

base of suitable IDS formed by fault detection mechanism.

For version A, the rollback mechanism is used. In this case, we made certain assumption: the data and program for execution each LS and its computation result as a constant are saved in each computer. According to this mechanism, in presence of fault in doubly set of computers in j-th LS the rollback is realized to beginning of operating cycle (Figure 7b). Then the first LS is executed repeatedly. After that the repetition computation result and the constant are compared. If they are same then suitable computer is nonfaulty, in contrary case, the computer is faulty.

For version B, the faulty computer is localized by diagnosis routine.

For versions C, D, E, F, the fault localization mechanism is realized by definition the accordance between the number of computers and disagreements in IDS.

For version G and H, the fault localization mechanism is based on the regular features that characterize the appearance of faults (Samedov et al., 1992). The used method generates a set of admissible localization results for every FAF in every nonfaulty computer (this set may be nonunique in some computers) and the ambiguity in the identification of the faulty computer is then resolved by transforming the IDS and the diverging localization results at several CP. The localization procedure implementing this method is multistage for different FAFs. Despite the arbitrary character of the data received from the faulty computer in each round of communication, the disagreements have a regular location pattern in the IDS generated by the nonfaulty computers, which makes it possible to develop appropriate localization rules.

Thus the fault localization is realized by following mechanisms (Table 2);

1. By the criterion of disagreement between the constant and own computation result (for version A),
2. By the diagnosis routine (for version B),
3. By the criterion of disagreement the values of all IDS elements with the majority of values (for version C, E),
4. By the criterion of disagreement the values of all IDS elements with the range of admissible deviations (for version D, E),
5. By the form of appearance of all types of faults (for version G, H).

For execution the fault localization mechanism in version A only one and in version G, H some rounds of communication are realized. For other versions B, C, D, E, F the rounds of communication are not realized. Versions A, B, G, H require the most processing time for execution of rollback (A), diagnosis routine (B) and some rounds of communication and the large volume of operation (G, H).

4. 3. The Mechanism for Definition of the Sort of the Fault Types

As has been noted a fault may occur either due to malfunction or due to failure. If a reason of fault is a malfunction it will disappear by next CP. However if a reason of fault is a failure it will remain by the following CPs. So a mechanism for definition of the sort of the fault types gives an answer to the question which reason is a fault? Malfunction or failure?

The sort of fault types is defined by using one of two mechanisms: by counter of faults or by the clock for measurement of fault appearance time. According to the first mechanism, the number f is recorded in counter (f = 1, 2,...,F). If during f consistent CPs the fault is disappeared it means that the reason of fault is malfunction, in contrary case, the reason is failure. According to the second mechanism, the time period t is recorded in clock. If during t the fault is disappeared it means that the reason of fault is malfunction, in contrary case, the reason is failure.

Thus the definition of the sort (malfunction or failure) of the fault types (Nonbyzantine and Byzantine) is realized by one of following mechanisms for all versions of fault-tolerant procedure (Table 2);

1. By the counter of faults,
2. By the clock for measurement of fault appearance time.

For execution of this mechanism, the rounds of communication are not realized and it requires a small value of operations.

4. 4. The Mechanism for Recovery of a Computational Process After Malfunction

The main aim of this mechanism is the recovery of an application and system data in the faulty computer. For N = 2, this aim is achieved by execution the LS, in which the fault was appeared, repeatedly. For this purpose the rollback mechanism is realized. For $N \geq 3$, the recovery of computational

process is executed on the "word" or "massive" levels. For this purpose, one of the nonfaulty computers recovers the computational process in faulty computer by sending the majority of values for synchronous mode (the median of values for asynchronous mode) or by coping the memory (RAM) - back-up copy.
Thus there are three mechanisms for recovery of a computational process (Table 2);

1. Rollback - by the repetition of faulty LS (for versions A, B),
2. On the "word" level - by sending the majority or median of values (for versions C, D, E, F, G, H),
3. On the "massive" level - by coping the RAM (back-up copy), (for versions C, D, E, F, G, H).

The most processing time is required for the rollback mechanism. On the "word" level mechanism is required the least processing time. For execution of these mechanisms the round of communication is performed between one of nonfaulty computers and faulty computer.

4. 5. The Mechanism of Reconfiguration of a Set of Computers After Failure

After failure the faulty computer must be turned off or isolated. For this purpose two mechanisms are used;

1. Self-closing of the faulty computer,
2. Closing of the faulty computer by nonfaulty computers.

First of them is used for $N \geq 2$, second - for $N \geq 3$. After this mechanisms the computational process is executed by less number of computers. It means that a set of computers is degraded.

Thus there are two mechanisms for reconfiguration of a set of computers;
1. By self-closing of faulty computer (for all versions),
2. Closing of faulty computer by nonfaulty computers (for versions C, D, E, F, G, H).

These mechanisms are realized by execution a small volume of operations.

5. CONCLUSİONS

This paper has given a description of the fault-tolerant procedures of a multi-computer system, which counteract all types of faults during degradation from N to 3 and only Nonbyzantine faults during degradation from 3 to 1.

The types of faults (Nonbyzantine and Byzantine)

and the sorts of all types of faults (malfunction and failure) which might appear in the structure of multi-computer system have been analyzed. The types of the fault-tolerant procedures (standard, mixed and perfect) which counteract different types and sorts of faults have been designed.
The graphic model, the steps and mechanisms of the fault-tolerant procedure have been described. It has been proved that depending on an absence, presence and sort of faults only part of steps of procedure was executed in each check point. This fact minimizes the hardware and software instrumentation impact, also processing time. Especially this is very important for highly responsive hard real-time control systems.

6. REFERENCES

Avizienis, A. 1978. Fault Tolerance: a Property that Ensures Constant Availability of Digital System, Proc. IEEE, Vol. 66, No.10, pp. 5-25.

Choar, B. and Coan, B. 1985. A Simple and Efficient Randomized Byzantine Agreement Algorithm, IEEE Soft. Eng., Vol. 11, No.6, pp. 531-539.

Flynn, M. F. 1972. Some Computer Organizations and Their Effectiveness, IEEETC, pp. 948-960.

Hopkins, A. L., Smith, T. B., Lala, J. H. 1978. FTMP- a Highly Reliable Fault-Tolerant Multiprocessor for Aircraft, Proc. IEEE, Vol. 66, No.10, pp.142-165.

Lala, J. H., Alger, L. S., Ganthie, R. J. and Dzwonczyk, M. J. 1986. A Fault Tolerant Processor to Meet Rigorous Failure Requirements, Proc. 7th Dig. Avionics System Conf., pp. 555-562.

Lamport, L., Shostak, R. and Pease, M. 1982. The Byzantine Generals Problem, J. ACM Trans. Program Lan. and System, Vol. 4, No.3, pp.382-401.

Pease, M., Shostak, R. and Lamport, L. 1980. Reaching Agreement in the Presence of fault, J. ACM, Vol.27, No.2, pp. 228-237.

Samedov, Y. R., Mamedli, E. M. and Sobolev, N. A. 1992. A Method for Localization of Byzantine and Non-Byzantine Faults, J. Automation and Remote Control, Vol. 53, (5), pp. 734-744.