·体系结构与软件技术· 文章编号：1000-3428(2019)07-0041-05 文献标志码：A 中图分类号：TP393

Survivability Mechanisms of Complex Computer Systems Based on Common Information Space

DODONOV Oleksandr[1], JIANG Bo[2], DODONOV Vadym[3]
(1. Institute for Information Recording of National Academy of Sciences of Ukraine, Kyiv 03113, Ukraine;
2. The 32nd Research Institute of China Electronics Technology Group Corporation, Shanghai 201808, China;
3. Institute for System Research and Information Technologies of the Ukrainian Academy of Sciences, Kyiv 03179, Ukraine)

【Abstract】The survivability of computer systems should be guaranteed in order to improve its operation efficiency, especially for the efficiency of its critical functions. This paper proposes a decentralized mechanism based on Software-Defined Architecture (SDA). The concepts of critical functions and critical states are defined, and then, the critical functional parameters of the target system are collected and analyzed. Experiments based on the analysis results are performed for reconfiguring the implementations of the whole system. A formal model is presented for analyzing and improving the survivability of the system, and the problem investigated in this paper is reduced to an optimization problem for increasing the system survival time.

【Key words】survivability; evaluation; common information space; Software-Defined Architecture (SDA); complex computer systems
Reference Format: DODONOV Oleksandr, JIANG Bo, DODONOV Vadym. Survivability mechanisms of complex computer systems based on common information space[J]. Computer Engineering, 2019, 45(7): 41-45.
DOI: 10.19678/j.issn.1000-3428.0053440

0 Introduction

In general, a system may be impaired by external and internal factors. The external factors, which usually derive from the interactions and mutual influences between a system and its operating environment, may affect the system and lead to system degradation at the structural and/or functional level. To solve the problem, a system protection circuit can be established for mitigating (though not eliminating) the external risks[1-2]. In terms of adverse internal factors, one feasible approach is to adjust the system properties that have direct effects on systematically critical functions. The approach requires pre-planning at the initial design stage, as the system structure has to encompass the possibilities of reconfiguration, reorganization, reconstruction and changes in component functions.

Survivability is the ability of a system to perform its basic functions under adverse external and internal conditions at a certain level, which is enough to sustain the operation of the system at least. Survivability can be described as consisting of 3 parts: structural vitality, functional vitality and informational vitality[2-4]. The 3 parts of survivability are interrelated, and the implementation of them may ensure the functionality of a system. As the system structure should allow possible changes in component functions, the survivability of a system has to be lumped with that of its organizational management components. From this point of view, it is necessary to establish a specific organizational management subsystem, with a focus on its interaction features with the operating environment[5-6].

In order to ensure the survivability of a given system, this paper proposes a mechanism for building up an Organizational Management Subsystem (OMS). The OMS is responsible for the management of the system functionality and the analysis of the parameters of the external environment. In response to detected adverse external factors, the OMS performs flexible reconfiguration, reorganization and reconstruction of a system, as well as redistribution of basic functions of an individual component. In terms of the internal system environment, the OMS provides continuous monitoring of the critical parameters of the system, and the analysis of the behavior with reference to the increasingly changed parameters. If an adverse internal factor is detected, the OMS can respond readily to

Biography: DODONOV Oleksandr (1941—), male, professor, academician, doctor, main research direction is the technology of survivability; JIANG Bo, professor; DODONOV Vadym, professor, academician, doctor.
Receipt Date: 2018-12-19 **Revised Date**: 2019-03-27 **E-mail**: jiangbo@ecict.com

establish new objectives. Based on the previous monitoring and analysis, the OMS can estimate the state of the system to prevent its transition to a potentially dangerous state. It is advisable to recognize potentially dangerous states at the initial development stage, which can enable a prompt and cost-effective response to parameter changes.

1 Design of OMS

1.1 Architecture of OMS

OMS is responsible for managing the basic or control functions of its host system. It supports the structural and/or functional redistribution of system functions among its components. Thus, for computer systems consisting of several processors or controllers, it is possible for them to perform both basic and control functions depending on which program is currently loaded into the RAM. One of the options for implementing the OMS is a Software-Defined Architecture(SDA), which is shown in Fig. 1.

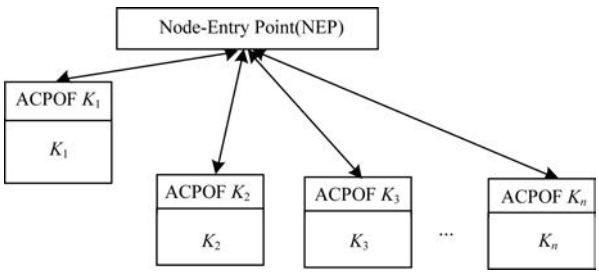

Fig. 1 The interaction scheme of ACPOF component with a NEP

During the running process of a given system, a task from an external or internal queue is arranged to a free or released component K_i ($i=1,2,\cdots,n$), the parameters of which will allow processing the task at a given time. In the future, this component will act as the entry point of the task and take over the management of its processing, including scheduling, selecting system resources, collecting intermediate results and obtaining the final result. To implement efficient processing of a given task, the Node-Entry Point (NEP) must collect information about the system components, with a focus on the components where the task or its parts will be sent for execution.

In general, system components can transmit information about their current state in the synchronous mode, such as sending messages to the NEP periodically, or in the asynchronous mode upon request from the NEP. In order to reduce the analysis cost of the components, it is advisable to transfer the information to a special element——the Agent for Collecting Parameters for the Operation of the Facility (ACPOF). In fact, each component of the system maintains a local ACPOF, which supports the collection and analysis of NEP and provides NEP with information of that component. The scheme of ACPOF interacting with NEP is shown in Fig. 1.

ACPOF collects certain parameters of the components and transfers them to the NEP. The specific list of parameters is determined based on user requirements. Parameter transfer is realized either synchronously at regular intervals or asynchronously at the request of the NEP. It is advisable to write information about the component to its local allocated memory at certain intervals, and to set up control points of the system for continuous analysis of its function.

1.2 Critical states and functions

During the running process of OMS, it is necessary to recognize functions that are critical to maintain the survivability of the OMS, and to recognize states in which a system may be equipped with a certain probability of suspension. Each node of the system maintains a database that contains "templates" for critical functions and potentially dangerous states. This database is distributed among components, and only some of the templates about dangerous states are stored on corresponding components, while others are periodically exchanged between the components. Upon detecting the fact of an attempt to implement potentially critical functions of the system, the triggered "template" will be copied to the bases of all components.

Normally, the objectives of a given system are not fixed. As a consequence, for certain reasons, if the system is in a critical state or its function may lead to a transition to a potentially critical state with a high probability, the objectives should be changed to get out of the critical state. In this way, all the available resources will be reoriented to the solution of this task, and the initial objectives of the system are in the so-called "frozen" state. After recovering from the critical state, the system will restore its original objectives.

2 Formalization of the survivability mechanism

The establishment of a functional model for analyzing the survivability of a given system requires the following concepts. The functionality level of a component is defined as the ability to support basic functions including critical functions, while the criticality of operations is defined in the light of their impact on the overall function of the component, primarily in the light of its efficiency.

Suppose that a system consists of interrelated and interdependent components, each of which operates independently with one another. Given the criticality of the operations, the functionality level Lf_i of the *i-th* component can be calculated as follows:

$$Lf_i(t) = Lf_0 \cdot \left[\frac{Q_{\lim}}{Q_i(t)+1}\right] \cdot Cr(t) \tag{1}$$

Where Lf_0 is the initial functionality, $Cr(t)$ is the criticality of the functions performed by the *i-th* component over a time interval $(0, t)$, $Q_i(t)$ is the number of failures caused by the *i-th* component over a time interval $(0, t)$, and $Q_{\lim}$ is the critical number of failures based on survivability over a time interval $(0, t)$.

To formalize the functionality level of a given component over an interval $(0, t)$, the highest possible functionality level $Lf_{\max}$ of that component can be defined as a fixed initial functionality level as follows:

$$Lf_{\max} = Lf_0 \cdot Q_{\lim} \cdot Cr_{\max} \tag{2}$$

where $Cr_{\max}$ is the maximum possible value of the criticality of the functions performed by the component over a time interval $(0, t)$.

Further more, the normalized functionality level $Lf_{\text{norm}}(t)$ can be carried out as follows:

$$Lf_{\text{norm}}(t) = \frac{Lf_i(t)}{Lf_{\max}} \tag{3}$$

Fig. 2 shows plenty of the normalized $Lf_{\text{norm}}(t)$ during the running process of a system under the varying functionality level performed by the *i-th* component over a time interval $(0, t)$. It is apparent that the functionality level of the components increases and decreases regularly because of 2 following factors:

1) Number of failures caused by the *i-th* component over the interval (in the 1st and 3rd sections, the number of failures is lower than those of the critical level, while in the 2nd section is higher).

2) Dynamics of the criticality of functions performed by the *i-th* component.

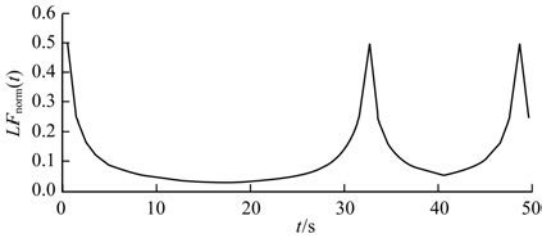

Fig. 2 The dependence of the normalized functionality level $Lf_{\text{norm}}(t)$

As a result, the survivability level $V(t)$ of the whole system (the system is deemed as a set of interrelated components) can be calculated as follows:

$$V(t) = \prod_{i=1}^{m} Lf_i(t) \tag{4}$$

In the meantime, the survival time T of the system, taking into account the survivability level, is defined as follows:

$$T = \int_0^1 V(t)\,\mathrm{d}t \tag{5}$$

In the discretized form, the survival time T of the system, taking into account the level of survivability, is presented as in Reference [7-8] as follows:

$$T = \sum_{i=1}^{N} V(t_i)\,\Delta t \tag{6}$$

$$L_{\min} \leqslant V(t_i) \leqslant L_{\max}$$

where $L_{\min}$ and $L_{\max}$ are the minimum and maximum possible value of functionality level over an interval $(0, t)$ respectively.

Therefore, the question of providing the maximum possible working time T can formally be converted as an optimization problem [9-11].

To describe the survival time T in a formal way, $v_1(t)$, $v_2(t)$ and $v_3(t)$ are defined to denote the survivability indicators while $c_1(t)$ and $c_2(t)$ the control variables, leading to the $f_1(t, v_1(t), v_2(t), v_3(t), c_1(t), c_2(t))$ form of survivability function at time t, which can be estimated as follows:

$$T = \int_0^1 (f_1(t, v_1(t), v_2(t), v_3(t), c_1(t), c_2(t)))\,\mathrm{d}t \tag{7}$$

Formally, the optimization problem of increasing

the survival time can be described as follows:

$$\frac{dv_1}{dt} = f_1(t, v_1(t), v_2(t), v_3(t), c_1(t), c_2(t)) \quad (8)$$

With T to be maximized, the time of survival can be described as follows:

$$T = \int_0^1 (f_0(t, v_1(t), v_2(t), v_3(t), c_1(t), c_2(t))) \, dt \quad (9)$$

Where $v_1(t)$ is the probability of system failure, $v_1(t, n) = P\{f = 0, A_n\}$, A_n is the n-time occurrence of adverse effects, $v_2(t)$ is system survival rate, $v_2(t, n) = P\{f = 1, A_n\}$, $v_3(t) = Q_{\lim} - 1$ is system survivability margin, $c_1(t)$ is the number of affected system components, $c_2(t)$ is the number of adverse effects on the system.

The solution of this problem will contribute to extend the survivability of a complex system.

3 Implementation of OMS

Suppose that there is a computer system consisting of complex heterogeneous hardware and software components, each of the components responsible for implementing the software located in common or local memory, as shown in Fig. 3[6,12-13]. During the running process of that system, each component with the right to access the common memory will periodically submit the current state of its local program to the common memory and set up a rollback point. If some failures occur in one of the components, the control component or OMS will migrate the program being executed and its corresponding data in common memory to a free component, while the faulty one is temporarily put into the sleep mode.

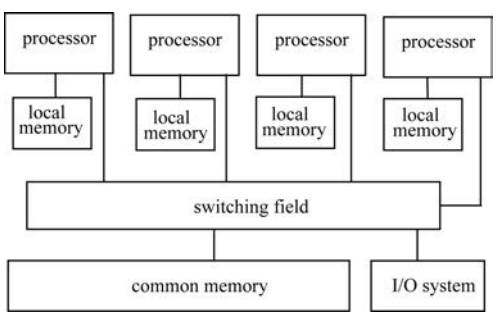

Fig. 3 The general scheme of a computer system with common memory

In fact, the choice of a free component is carried out either by the principle of random sampling, or in reference to the functional parameters, such as performance, the speed of communication channels, reliability, security and etc. Additionally, the control assignment is performed in a decentralized model. Once a component is assigned to a task, it will be declared as the initiator of this task, and act as a control component. Meanwhile, a general supervisor also exists in the system to coordinate the actions of control components.

An alternative way to ensure the survivability of a computer system is the interaction scheme of processor modules without shared memory[14-15], as shown in Fig. 4. In this case, the initiator and control component of the task will utilize the dispatching mechanisms to perform initial resources distribution. All resources (processors) maintain the history data about its operation, as well as data exchange with other resources. Moreover, the history data involves fixing the addresses which data was sent to or received from, and backing up a copy of the sent data and program code for processing.

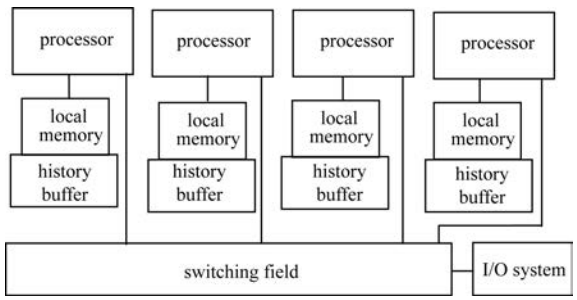

Fig. 4 The general scheme of a computer system with local history buffer

As a matter of fact, background history data varies dynamically. Once the processing of the next part of some data is finished and the correct response is received, the previous data will be reset. For example, if the absence of a response/result continues for a certain span of time, the control component will confirm the address of the faulty processor, restore the history of the system and return the system to the rollback point where all the processors were correct. Then, the control component will send a copy of the

data and the program code that was not finished by the faulty processor to another (healthy) one. After a correct response from the available processor is received, the prehistory data will be reset, leaving the current history of the system only the data that is being processed and the program information about which the response/reaction of the processor has not been received.

4 Conclusions

A mechanism for enhancing the survivability of complex systems based on the architecture with distributed control and history buffer is proposed in this paper. Furthermore, in this work, not only the formalization of the survivability mechanism on the basis of the functional model is presented, but also the question of increasing the system survival time has been reduced to the solution of the optimization problem.

However, it should be noted that this paradigm is confined to complex computer systems. Hopefully, in the future, we will further explore the applications of the survivability mechanisms over complex information systems. And as with formal techniques, the question of increasing the system survival time has been simplified to the solution of the optimization problem. Thus, we are greatly motivated to develop a more efficient model to formalize the critical functions and states.

References

[1] DODONOV A, LANDE D. The survivability of information systems[M]. Kyiv, Ukraine: Naukova Dumka, 2011.

[2] KLYACHKO L, OSTRETSOV G. The projecting of highly reliable automatic ship motion control systems [M]. Moscow, Russia: Nauka PhysMathLit Publ., 2010.

[3] KORYTCHENKO T. Application of methods of survivability increase for security ensuring in distributed telecommunication systems[J]. Information and Control Systems on the Railway Transport, 2015, 2: 1-8.

[4] DODONOV A G, FLYTMAN D V. Corporate information systems ensuring vitality mathematical machines and systems[M]. Kyiv, Ukraine: [s. n.], 2005.

[5] DODONOV A G, GORBACHYK O S, KUZNYETSOVA M G. Survivability of informational-analytical systems: conseptual basis, models for analysis and evaluation [M]. Kyiv, Ukraine: [s. n.], 2007.

[6] MUKHIN V. The rating mechanism for the trusted relationship establishment for the security of the distributed computer systems[EB/OL]. [2018-12-01]. http://www. woiz. polsl. pl/znwoiz/z67/Mukhin% 20-% 20art. pdf.

[7] SHNITMAN V, KUZNETSOV S. Secrets of computer kitchen or PC chefs: problem solving [M]. Moscow, Russia: BHV Publishing GmbH, 2003.

[8] CHERKESOV G, NEDOSEKYN A. The survivability of complex objects estimation with high precision multiply impacts[EB/OL]. [2018-12-01]. http://www. ifel. ru/surv/Res_3. pdf.

[9] GRISHCHENKO I. The method of increasing the survivability of the information and communication network [J]. Refrigeration Technic and Technology, 2013, 146(3): 66-70.

[10] ORLOV A. Decision-making theory study guide [M]. Moscow, Russia: [s. n.], 2004.

[11] MUKHIN V, VOLOKYTA A N. Integrated safety mechanisms based on security risks minimization for the distributed computer systems[J]. International Journal of Computer Network and Information Security, 2013, 5(2): 1-10.

[12] STALLINGS W. Operating systems-internals and design principles[M]. Englewood, USA: Prentice Hall, 2011.

[13] KHOROSHEVSKY V G. Architecture of the number of systems[M]. Moscow, Russia: [s. n.], 2008.

[14] GARSVA E. Computer system survivability modelling by using stochastic activity network [C]//Proceedings of the 25th International Conference on Computer Safety, Reliability, and Security. New York, USA: ACM Press, 2006: 71-84.

[15] SANDERS W H. Probabilistic validation of computer system survivability [C]//Proceedings of Latin-American Conference on Dependable Computing. Berlin, Germany: Springer, 2005: 1-5.

编辑 樊丽娜