UDK 681.516

BUILDING FAULT-TOLERANT DECENTRALIZED SYSTEMS

E.K. Mailybaev *– PhD doctoral student, Kazakh university ways and communications, Almaty, Republic of Kazakhstan, 050063, e-mail: ersind@mail.ru*
U.U. Umbetov*-Dr.Sci.Tech., professor, Vice-Rector for Academic Affairs Kh.A. Yasavi International Kazakh-Turkish University, Turkestan, Republic of Kazakhstan, e-mail: uumbetov@mail.ru*
Zh.I. Batyrkanov*, professor, Kyrgyz State Technical University after I. Razzakov, Bishkek, Republic of Kyrgyzstan*
Kossyakov Igor Olegovich*, PhD student Kazakh University Ways of Communications, Republic of Kazakhstan, Almaty, microdistrict Zhetysu-1, 32A, e-mail: heimmdal@mail.ru*
A.B. Shynykulova *- PhD doctoral student, Kazakh university ways and communications, Almaty, Republic of Kazakhstan, 050063, sh.anel14@mail.ru*

Annotation. The safety of technological processes, communication systems, onboard systems and other control objects is largely determined by the reliability, fault tolerance and survivability of the control computing systems, especially in the event of emergency situations, accidents, sabotage that impede and sometimes exclude repairs. Technical failures, as well as the unreliable functioning of information and telecommunication systems in the field of information security are noted as one of the main threats to information security. The development of methods and means of adaptation at the same time to the flow of failures and requests in control systems, computing nodes of which are implemented on the basis of industrial computers and controllers, equipped with a set of functional modules, which determine the multifunctionality of the nodes, help in addressing fault tolerance. Fault tolerance property of a technical system to maintain its operability after the failure of one or more composite components. Fault tolerance is determined by the number of any consecutive single component failures, after which the health of the system as a whole is maintained. The advantages of fault-tolerant decentralized systems are given. The basic principles of building systems with fault-tolerant characteristics are listed. Various technologies for system stability are considered.

Keywords:fault tolerance, automation, decentralization, processes, shutdowns, monitoring.

ПОСТРОЕНИЕ ОТКАЗОУСТОЙЧИВЫХ ДЕЦЕНТРАЛИЗОВАННЫХ СИСТЕМ

Е.К. Майлыбаев *– докторант PhD, Казахский университет путей сообщения, г.Алматы, Республика Казахстан, e-mail: ersind@mail.ru.*
У.У. Умбетов*, д.т.н., профессор, Проректор по учебной работе Международного казахско-турецкого университета имени Х.А.Ясави, Туркестан, Республика Казахстан, uumbetov@mail.ru*
Ж.И. Батырканов*, профессор, Кыргызский Государственный Технический Университет им. И. Раззакова, Бишкек, Республика Кыргызстан*
Косяков Игорь Олегович*, докторант PhD Казахского Университета Путей Сообщения, Республика Казахстан, г. Алматы, микрорайон Жетысу-1, 32А, e-mail: heimmdal@mail.ru*
А.Б. Шыныкулова *- докторант PhD, Казахский университет путей сообщения, г.Алматы, Республика Казахстан, e-mail: sh.anel14@mail.ru.*

Аннотация. Безопасность технологических процессов, систем связи, бортовых систем и иных объектов управления во многом определяется надежностью, отказоустойчивостью и живучестью управляющих вычислительных систем, особенно при возникновении нештатных ситуаций, аварии, диверсии затрудняющих, а порой исключающих ремонтные работы. Отказы технических средств, а также ненадежное функционирование информационных и телекоммуникационных систем в сфере информационной безопасности отмечены как одни из основных угроз информационной безопасности. Разработка методов и средств адаптации одновременно к потокам отказов и запросов в управляющих системах, вычислительные узлы которых реализуются на основе промышленных компьютеров и контроллеров, укомплектованных набором функциональных модулей, обусловливающих многофункциональностью узлов помогают в решении вопросов отказоустойчивости. Отказоустойчивость свойство технической системы сохранять свою работоспособность после отказа одного или нескольких составных компонентов. Отказоустойчивость определяется количеством любых последовательных единичных отказов компонентов, после которого сохраняется работоспособность системы в целом. Приведены преимущества отказоустойчивых децентрализованных систем. Перечислены основные принципы построения систем с отказоустойчивыми характеристиками. Рассмотрены различные технологии для устойчивости системы.

Ключевые слова: отказоустойчивость, автоматизация, децентрализация, процессы, остановы, мониторинг.

Introduction

Fail-safe decentralized systems- are one of the promising areas for the development of optimal control of equipment shutdowns of technological processes. Fail-safe decentralized are able to continuously operate in the event of failure of individual nodes and communication channels without the need for remedial repair due to the presence in their architecture of special hardware and algorithmic tools that automatically detect failures, isolate them and replace them with backup resources ensuring restoration of the logical integrity of the communication environment. Due to these properties, fail-safe decentralized can be successfully used as a basis for creating critical, important facilities of a hazardous production and process management system. Failures of the modules and bonds of the fail-safe decentralized systems lead to the appearance of heterogeneityin its physical structure.

As a result, the number of possible data transfer routes decreases, while their average length increases, which complicates the routing of data between healthy modules after system reconfiguration. As a result, the average time for the exchange of information flow increases and the actual system performance decreases [1].

The base of the article description was the scientific and production work of scientists in the field of the theory of the fault-tolerant organization of decentralized systems. In particular, the following works were studied: S.E. Baranov, V.V. Voevodin, Y.Y. Gromov, E.A. Kalyaev, E.E. Levin, V.G. Khoroshevsky, I.V. Zotov, Borisenko Y.V etc

THE MAIN PART

There are two fundamentally different approaches that can be independently combined when building a fail-safe decentralized systems. Figure 1 shows an example of a multi-level decentralized control system. The first approach is implemented using disaster recovery, when a complete copy of the entire system can be restored in another data center. This method is relevant in almost any situation, however, it can have a very long period of inactivity. The second way: it is the software implementation of fault tolerance for each of the components and their interaction. Below we consider various technologies for system stability. [2,3].

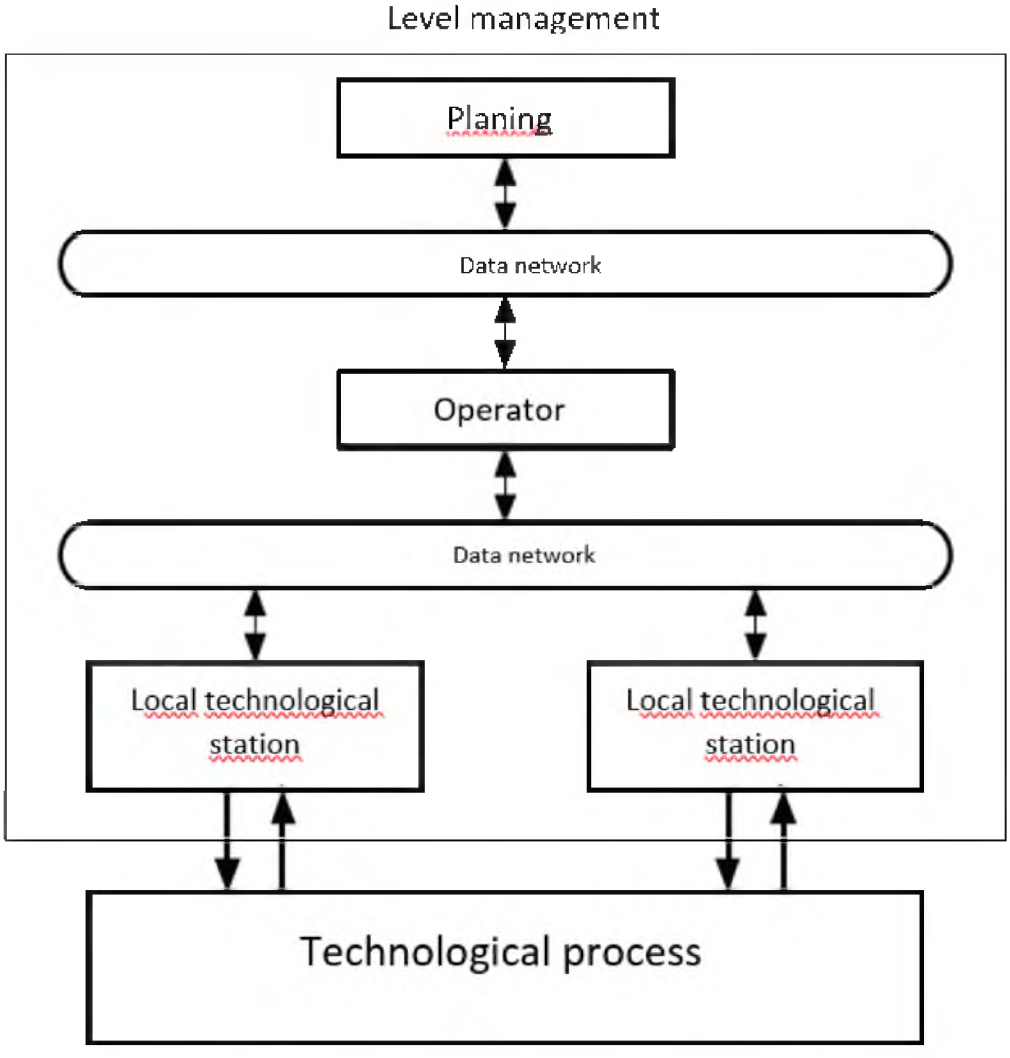

Figure 1- is an example of a multi-level decentralized control system.

Low level fault tolerance services. These systems should consists of more or less independent of each other subsystems, and each of them must be fault tolerant. Single point of failure. Architecture should be avoided in which the entire system collapses when one of the components is stopped. This can be achieved either by using the principle of redundancy, or by making the components as independent as possible so that if one of the components fails, only part of the functionality stops working, and the rest of the system continues to work. This solution is not suitable for the main

functionality of the system, but in case of problems of secondary elements of the system that provide auxiliary functions, the specified shutdown can completelydisable the system. Redundancy system with the presence of an excess amount of necessary components. Redundancy can be seen on the example of a mirrored RAID array as well as two network adapters connected to two different switches also represent a redundancy model. And when one of these redundant components is stopped, all other components should continue to work.

With this design approach, two strategies can be distinguished: active-active and active-passive.

In an active-active strategy, you can work simultaneously with two identical components at the same time. For example, in a system where an operator simultaneously receives data from a workshop using two different components from two different places. If one of these components fails, the operator will not notice that there was a problem in the system, which is an undoubted advantage of this approach. As minuses it is possible to allocate doubling the amount of traffic and time for its processing, as well as additional server infrastructure, which is constantly in operation and consumes resources. The active-passive strategy is only one constantly working component, in the case of which it stops, the second component automatically turns on, which restores the state and takes over all the work. But with this strategy, problems may arise with the bandwidth of the component. It is also important to understand the complexity of implementing an active-passive strategy compared to an active-active strategy, since you need a reliable way to verify that the active component is functioning and you need to be able to restore the state at the time of the shutdown or constantly synchronize it.

Also, this solution will always have some delay in the operation when the component is stopped, while the passive component will consume a lot less resources and you can get a fault-tolerant solution on a weaker hardware resource.

Load balancing is used when building heavily loaded systems.

However, for the fail-safe decentralized systems this principle also applies. Load balancing evenly distributes the entire load among identical components.

Unlike the above-described active-active strategy, here only one component performs each task. For example, in the case of web servers, making session replication is difficult without load balancing. In this solution, it is very important to have at least N+1 redundancy; if peak loads require N components operating at full capacity, then N+1 such components should be present in the system, otherwise, when one of the elements stops, then all other loads increase and the whole system will crash.

Defensive coding. To achieve maximum resiliency, you need to pay attention to resiliency during the design and programming stage of the system. The code of the processing program must be able to work with continuous cycles, mechanisms to protect the processor from thermal damage when the system overheats, access points, null division.

When an error occurs during processing, the system should not enter into an infinite loop, which constantly tries to perform this operation, especially if the message comes from outside, and there is no guarantee that it is correct. Sending an error message to the monitoring system and proceeding to the processing of the next task will provide an opportunity to interrupt such a cycle. When you receive a message from outside, you need to consider the situation with the number of messages that exceed the processor`s processing ability. The system from time to time must maintain its state, so that in case of big problems there will always be an opportunity to roll back into the last consistent state.

To avoid problematic situations with division by null, the program code should be created with the rule to never pass a null between components in the system.

Monitoring. Increase the resiliency of the system as possible through monitoring. Very often, having learned about upcoming problems in advance, you can take certain actions to avoid their occurrence. For example, seeing that the free disk space is running out, you can start the process of clearing old event logs. For monitoring there are many ready-made solutions, both paid and with open-source license, such as Triton, Nagious. Standard monitoring functions are monitoring disk,

processors and traffic. There are also various plugins that allow you to monitor log files and send a message to the monitoring system when errors occur. [4].

Another type of monitoring is the health monitor, when the application sends special heartbeat periodic signals generated by hardware or software to indicate normal operation or to synchronize other parts, and if there was no return from the application for a certain period of time, a message pops up in the tracking system about malfunction.

Findings

It is necessary to think through all the phases of system resiliency as early as possible, at the design stage, adhering to bring the system to the possibility of their components to resist unplanned failures or violations, and also to recover in a certain temporary period in the event of such an event. It is very important to investigate every problem that has occurred in the system, find the true cause of the errors and make an analysis of the event.

LIST OF SOURCES:

1. Borisenko Yu.V. Method, algorithms and hardware for online program relocation in fault-tolerant multicomputer systems: Abstract dis. can.tech.science. Kursk, 2014.

2. Chernyshev N.N. Distributed automatic control system for a hydrogen sulphide gas combustion unit. -Lugansk.: Праці луганського відділення Міжнародної Академії інформатизації №1(23), 2011. 14с

3. Kruchinin S.V. Types of decentralized networks and the option of building a decentralized network of a full protocol stack. -Volgograd.: Proceedings of the Volgograd state technical university №11(190), 2016. 161с

4. Javaspecialist. Building a fault-tolerant system (fault tolerant) [electronic resource] https://habr.com/ru/post/118496/