Section 1. FT (Security of computer systems and networks. Fault-tolerant distributed computing).

Oleksandr Honcharenko, Artem Volokyta, Heorhii Loutskii.

METHOD OF FAULT-TOLERANT DISTRIBUTED SYSTEMS' REALIZATION BY EXCESS DE BRUJIN TOPOLOGY

The article discusses variants of fault-tolerant system realization by usage of excess de Brujin topology in the switch network. Methods of counteracting failures are considered both among computing elements and inside a switched system.

Key words: fault tolerance, excess de Brujin, distributed system

Fig.: 17. Tabl.: 2. Bibl.: 8.

Urgency of the research. In modern world the distributed computing is an important branch of the development of computer technology. Improving of their fault-tolerance is a one of most important tasks. One of perspective methods of its solution is a method of fault-tolerant topologies synthesis. But the use of fault-tolerant topologies is complicated by the fact that the topology proposed in the framework of graph theory and the current possibilities for its implementation differ significantly. This leads to the fact that the fault tolerance of the system may be less than the fault tolerance of the topology per se. Therefore, it is important not only to review and synthesize topologies per se, but also to consider the possibilities of using their properties to design computing systems and networks. Thus, the topic of this article is urgent.

Target setting. Quite often, topology offers good properties that cannot be used into practice. Conversely, reviewing only the topology of the system without considering its possible implementation does not allow to take full advantage of all possibilities for ensuring fault tolerance. Therefore, this article proposes to consider an excess de Brujin topology (called as quasi-quantum) using a complicated model that includes a switch network and computational nodes as separate entities, and consider how a high fault-tolerance can be ensured for such a system.

Actual scientific researches and issues analysis. Well known at the moment are switched and hybrid networks. There are also many methods of fault tolerance ensuring. Quasi-quantum topology, its capabilities for simple routing and fault tolerance compared to its closest counterparts have already been explored. Previous publications have proposed clustering of nodes with the same numbers for this topology.

Uninvestigated parts of general matters defining. So far, quasi-quantum topology has been considered only in terms of graph theory as a point-to-point network. An overview of the capabilities of switched network has not been completed.

Not taking into account the fact that most modern processors have autonomous controllers to access the communication environment. It is proposed to use quasi-quantum routing to bypassing failures. However, the possibility of actual implementation of this idea is not considered given the above features.

The research objective. The purpose of the article is to consider and analyze options for implementing a multicomputer system with quasi-quantum topology and a network of switches as a communication medium.

The statement of basic materials. Multicomputer and cluster systems use communication network (CN) with a certain topology to communicate between processors. Working with a communication network isn't simple. There are a lot of additional routing-related actions are required to send the message. Therefore, it is quite common to use additional equipment to work with the communication network: controllers, switches, routers, etc. This approach allows you to route messages in parallel with calculations. This improves system performance. Therefore, this approach is quite common. Therefore, when considering systems, it is advisable not only to consider abstract nodes and topologies, but also consider the presence of additional equipment.

In Fig. 1 shows a model of a multicomputer system with a linear topology as an example of model.

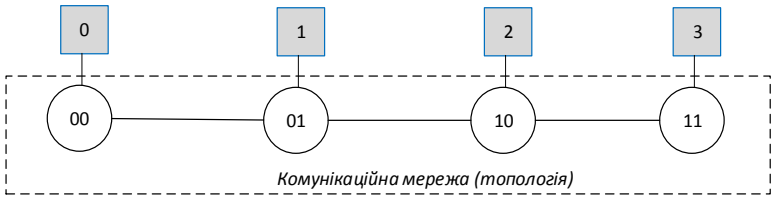

Fig. 1. Multicomputer system with linear topology in the proposed model

Use of the proposed model for quasi-quantum topology. Quasi-quantum topology is based on the De Brujin graph using code transformations - left and right shifts - to form a network. However, unlike the De Brujin graph, the quasi-quantum topology uses excess code 0/1 / -1 to encode node numbers. It leads to redundancy. A topology may contain multiple nodes with different code, but the same number. This, in turn, can be used for fault-tolerant routing. Also, this topology has good topological characteristics: constant degree 6 and diameter growing as $\log_3 N$, where N is the number of nodes. n Fig. 2 shows a normal representation of this topology according to graph theory.

However, as stated above, this presentation does not quite match the current technical capabilities and therefore the forecasts made on such a model may be inaccurate. Therefore, the following model is proposed: the system contains computing nodes (processors) and switches. Computing nodes have decimal numbers (visible

from the outside), switches - a unique code in the excess numeric system. Each computing node is connected to a switch whose code represents the node number. Switches are connected by the “excess de Brujin” (quasi-quantum) topology. Fig. 3 illustrates this model of a computing system. Gray squares indicate computing nodes, white circles are switches.

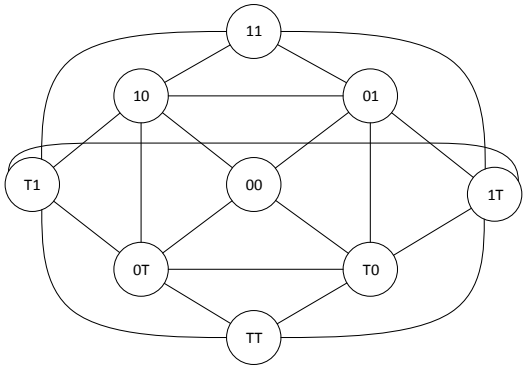

Fig. 2. Quasi-quantum topology, rank 2

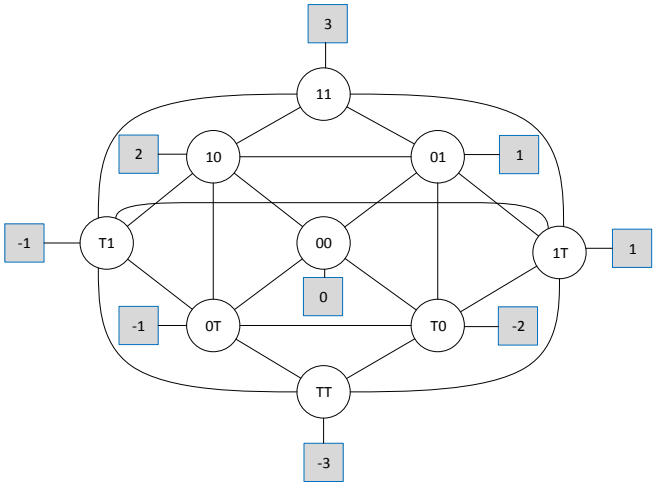

Fig. 3. Model of a system with a quasi-quantum topology of rank 2.

As can be seen from Fig. 3, for rank topology 2 there are 2 nodes with non-unique number: 1 and -1. Because the numbers of the compute nodes are visible from the outside, such representation is undesirable and may lead to conflict.

Virtualization of conflicting computing nodes. Because ID conflict is undesirable for external representation, the following solution is proposed: make conflicting processors virtual. Each virtual processor (virtual node, VN) has a unique number, but several physical processors are hidden within it. The system "knows" which processor is involved for what task and therefore can perform the messages transfer correctly, however, only virtual processors are visible to the user and to the routing. Fig. 4 shows a model of the system with virtual nodes 1 and -1.

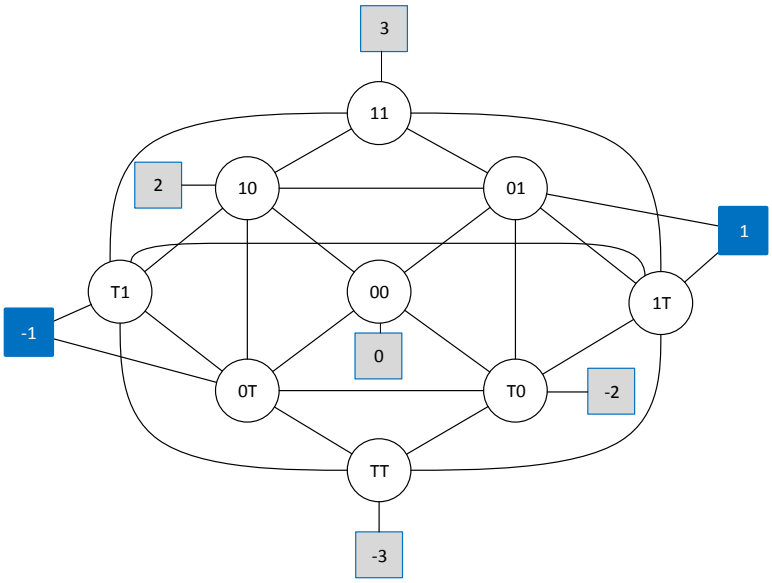

Fig. 4. Model of the system with virtual redundant nodes

There are also several principles for virtual node systems that aim to provide high fault-tolerance:

1. A virtual processor may contain multiple physical processors

2. Any message assigned to a virtual node can be delivered to any component of it from any connected router (principle of universality).

3. In the event of failure of any component of the virtual node, its role can assume any other component of this VN (the principle of replacement)

4. Any message passing through a router connected to a virtual node can be transmitted to any other router connected to this node ("Quantum transition" or "quantum tunnel")

5. Additional processors can be added inside the virtual node, which can be used for backup or to increase system performance (internal scaling).

Options for implementing the described principles. Although node virtualization avoids some problems, it does create new ones. First of all, it is necessary to deal with the structure of virtual nodes. Several implementations are offered.

Simple implementation involves direct links between physical processors and switches. This structure is the simplest and cheapest, but has a lot of disadvantages. Fig. 5 shows a simple implementation. Blue dashed line indicates additional connections between computing nodes and switches. These links allow you to connect a physical node to an alternate switch as needed. The red dotted line indicates additional direct connections between nodes inside the virtual node. Such connections allow data to be transmitted directly between nodes with the same number without having to access the communication network. This allows you to speed up transmission inside the virtual node, unload your communications network and

increase fault-tolerance. The black dotted line outlines virtual processors. The IDs of physical processors inside the virtual ones are indicated in parentheses.

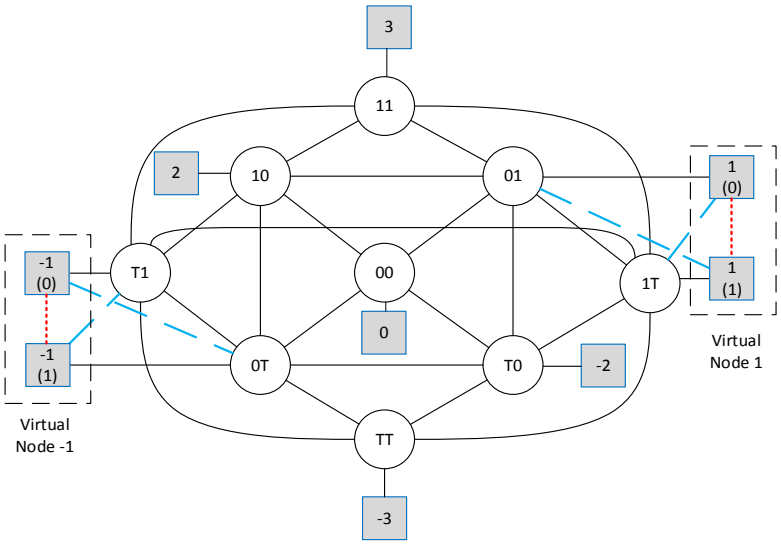

Fig. 5. Simple implementation of virtual nodes

It is worth considering how this model enables the implementation of such principles as the quantum tunnel principle and the principle of internal scaling. In Fig. 6 shows an example of transmitting a message from node -3 to node 2. Due to the large number of link failures, the switches TT and 0T were isolated. This causes the topology to lose connectivity. However, since the code of node 0T represents the number -1, and this number has several representations in the excess code, a "quantum tunnel" can be used to bypass the fault. In a simple implementation, one of the processors from the virtual node -1 must be used for this purpose. The green arrow indicates the path to which the message is transmitted: from the source (processor -3) to the corresponding switch, from TT to OT, then through the "quantum tunnel" through processor -1 (0) to switch T1, whose code also represents the number -1, from there to 10, and from it to the associated destination processor 2.

In Fig. 7 shows, how internal scaling can be performed. Additional physical processors are highlighted in green and have a lighter fill. The links between processors and switches are highlighted in blue. The virtual links of the virtual node is red.

After analyzing the two examples above, you can immediately identify key disadvantages of a simple model. This is a rapid increase in degree of switches on internal scaling, the using of processors to perform the quantum transition. In addition, with the increase in the number of components complicates the transmission of messages inside the virtual node. Either you need to implement a fully connected system or use processors for routing, which eliminates all the benefits of such a decision.

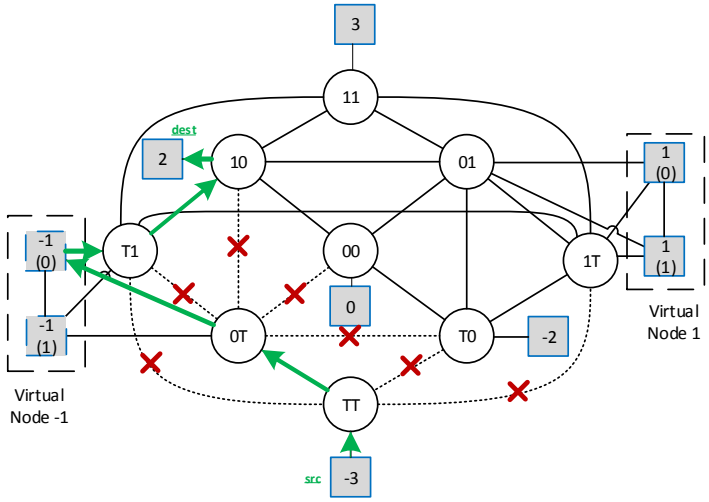

Fig. 6. The principle of "quantum tunnel" working in simple implementation

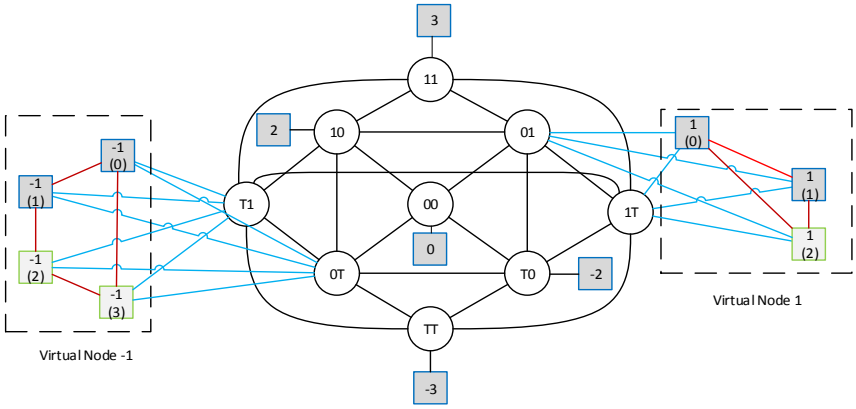

Fig. 7. Internal scaling in a simple model.

Another implementation offers the opposite approach. The message is transmitted through an intermediate access device to the communication network, - Virtual Node Switch (VNS). VNS does not perform routing by itself, only transmits to or receives messages from the network. As result, its fault probability is less. In Fig. 8 illustrates this model.

This implementation solves a number of problems. VNS is one for the entire virtual node, regardless of the number of hidden processors. As result, the degree of switches on the network is not growing. In addition, there is no need to use processors for "quantum tunnel". It can be performed by VNS. But such a system has a very significant disadvantage. VNS is a single point of failure. VNS failure causes a complete node failure. This is not acceptable in terms of fault tolerance. However, there is a solution: additional VNS. Using one or more backup VNS eliminates single point of failure issues while providing the same advantages. In Fig. 9 shows an advanced switched implementation. It also shows internal scaling. The markings are

the same, except that the internal links are replaced by the IN (Inner Network) pseudo-switch for simplify of presentation.

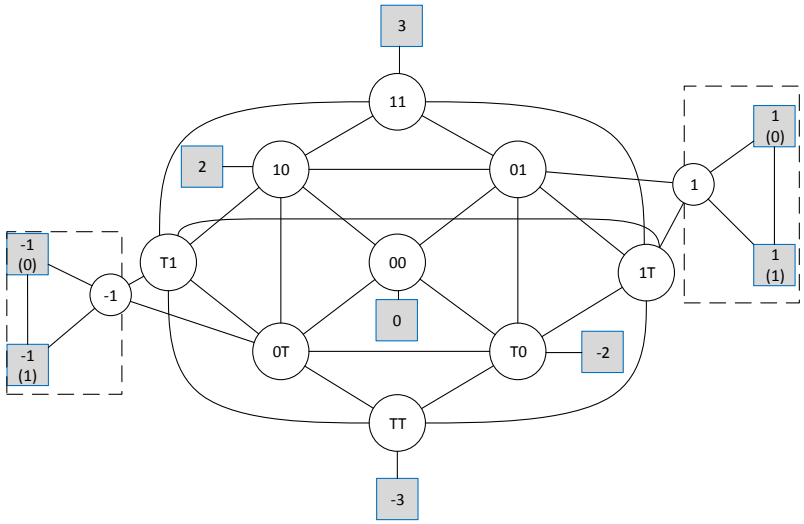

Fig. 8. Implementation using VNS (switched implementation). Basic version

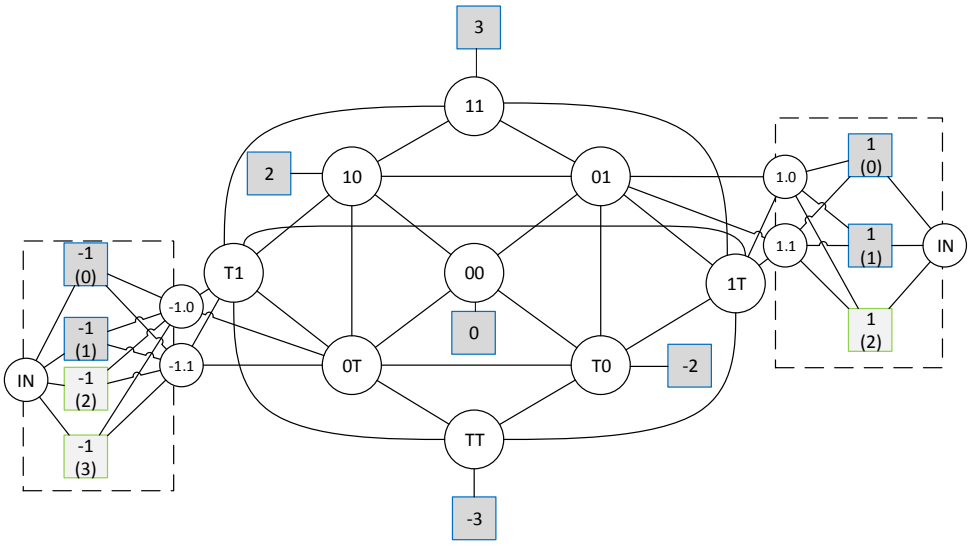

Fig. 9. Realization with VNS. Advanced.

The third implementation combines the two previous ones. This implementation proposes the following connection scheme: each processor is directly connected to one and only one switch in the topology directly. At the same time, the node also has a VNS (1 or more) that provides access to any switch connected to this virtual node. Such an implementation is a trade-off: simpler than switching and at the same time avoids the disadvantages of simple implementation. In Fig. 10 presents a hybrid implementation.

After analyzing the proposed models, you can compare them with each other. For this purpose, it is suggested to evaluate the influence of a particular virtual node

architecture on various parameters. Let the topology have a virtual node with N "hidden" nodes. This node is connected to the K topology switches. At the node is M VNS (for simple architecture M = 0

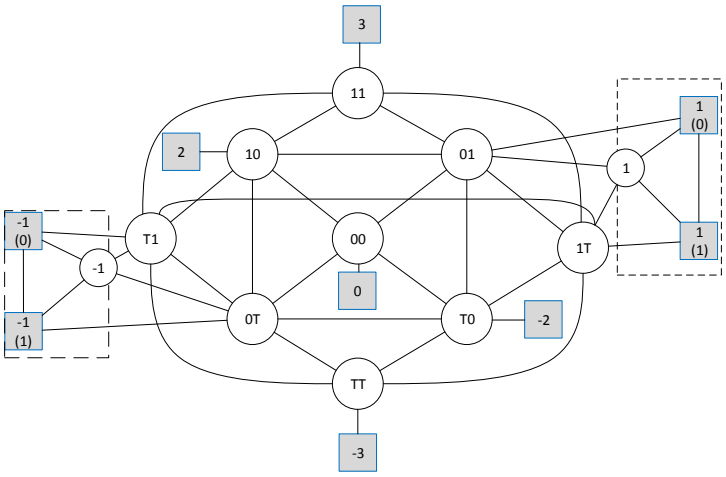

Fig. 10. Hybrid implementation.

Table 1

Implementations comparison

#	Parameter	Realization of virtual node			
		Simple	**Switched, base**	**Switched, adv.**	**Hybrid**
1	Count of immediate links between procs. and switches (L)	NK	0	0	N
2	Minimal VNS count (M_{min})	0	1	2	1
3	Count of links between procs. and VNSs	0	N	NM	NM
4	Count of links between VNS and topological switches	0	K	KM	KM
5	Maximal degree of one VNS	0	N+K	N+K	N+K
6	Degree of connected topological switch grows as…	N	1	M	M + 1
7	Routing is independent from processing.	No	Yes	Yes	Yes
8	Minimal delay of access to CN (delay of link =x, delay of VNS = y, x<y)	x	y	y	x
9	Minimal cost (cost of link=x, cost of VNS=y, M=M_{min}, x<y).	NKx	(N+K)x+y	2(N+K)x+2y	(2N+K)x+y
10	Fault tolerance of VN-CN connection	NK	1	MK	MK+N

As can be seen, the simple implementation has the highest fault-tolerance (since N> = M). But it leads to the highest growth in the degree of topology switches. Switching implementation in the base version has the least fault tolerance (since it has

a single failure point). On average, the most balanced is the hybrid implementation.

Processor-level fault-tolerance. Since there are 2 types of nodes (processors and switches) in the model described, it is worth considering the fault tolerance separately for each type.

As discussed above, the mechanism of virtual computing node (virtual processor, virtual node, VN) allows to solve the problem of fault tolerance for those processors whose numbers have several code representations. It was postulated that when one processor of VN fails, another processor should take over its role and thus restore the virtual node's work This was called the replacement principle (replacement mechanism). It is suggested to consider this mechanism in more detail. The essence of the replacement mechanism is this: let there be a failure. One of the processors does not work. The task of the system is to do the following:

1. Its role and computational task may be assumed by another processor

2. At the user and routing levels, the structure of the system remained unchanged.

Allowed performance loss due to failure, but no communication network or the user does not have to "see" the processor's fault. In Fig. 11 shows an example of how such a replacement is performed (in simple implementation). Node -1 contains several processors. In regular mode processor -1(0) works with switch T1, processor -1(1) – with switch 0T. Processor -1 (0) fails. But, due to the fact that node -1 has 2 processors, processor -1 (1) can take its role. In decrease performance of virtual node -1 (since it will no longer be possible to use -1 (0) and -1 (1) processors in parallel), but nothing will change for the routing. Messages received on 0T and T1 routers will be delivered. For the user, similarly, the model remains the same: processor -1 works.

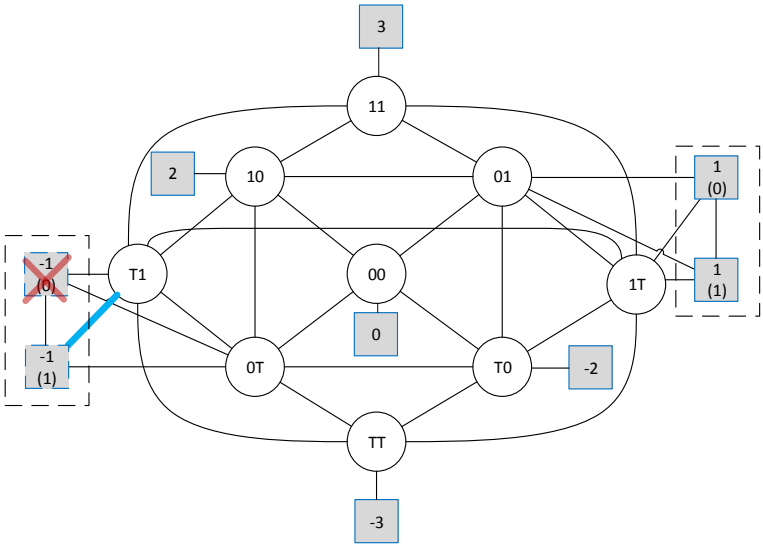

Fig. 11. Example of replacement in virtual node -1

However, there is a disadvantage: it seems that this replacement only works for node 1 and -1. If, for example, node-3 fails, it will change the model of the user. It is

suggested to make all nodes virtual to solve the problem. This will allow the implementation of the replacement principle even for those nodes that do not have multiple code representations. In Fig. 11 shows a new model of the system as seen by the user.

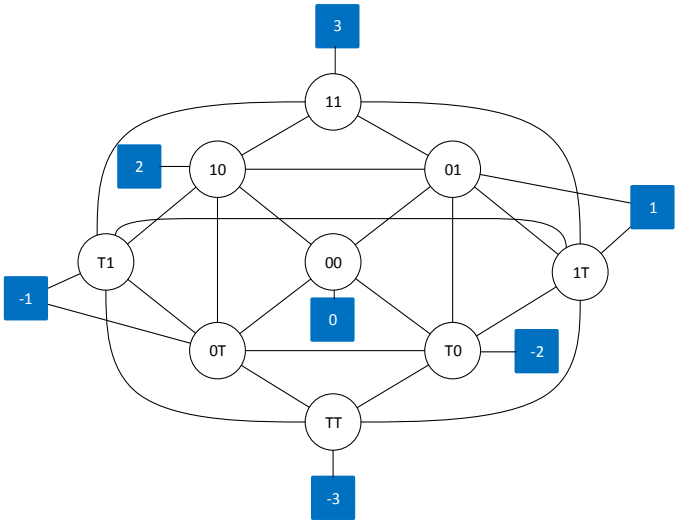

Fig. 11. User model with full virtualization

However, the problem is: where to get the processors for replacement? Of course, you can add additional processors to the virtual nodes. But this greatly increases the cost of the system. Therefore, a different approach is needed. A shared processor mechanism is proposed as such an approach. The basic idea is: let the same processor belong to multiple virtual nodes. Then, when all the processors of one virtual node fail, the other node will be able to "give away" one processor in order to restore the structure of the system. If the virtual node has only one working processor, it can go into "shared" mode. In this mode, physically the same processor is used for several different tasks and is perceived by the system as several different processors of lower performance. This will reduce the performance of both virtual processors, but in the critical case will restore the system model and take time to fix the failure. In Fig. 12 shows an example of replacement by this mechanism. In example processor -1 (1) uses as shared by VN's -1 and -3. While once «own» processor of VN -3 fails, node -1 «give away» processor -1 (1), thus restores the structure of the system.

Network-level fault tolerance. The fault-tolerant routing and redundancy provided by the topology allow us to bypass some failures with the help of "quantum tunnels". However, there is a problem: there are not many "quantum tunnels" in the topology.

To resolve this issue is proposed to create additional links. This allows you to bypass the fault when routing does not allow it the usual way. Of course, bypassing using "artificial quantum tunnel" is not as simple as bypass using fault-tolerant

routing. However, it will gain time to fix the problem in a critical situation. The following links are suggested. For all topology nodes having only one code representation, the duplicate link is executed by inversion (for example, 2 and -2). Node 0 communicates with nodes 11..1 and TT..T. The topology is symmetrical with respect to 0, so such a linkage will allow for even distribution of additional links. With respect to node 0, it and node 11 and TT are especially important when using tree-based routing. Therefore, it makes sense to connect them directly to go from one root to another in the event of a failure.

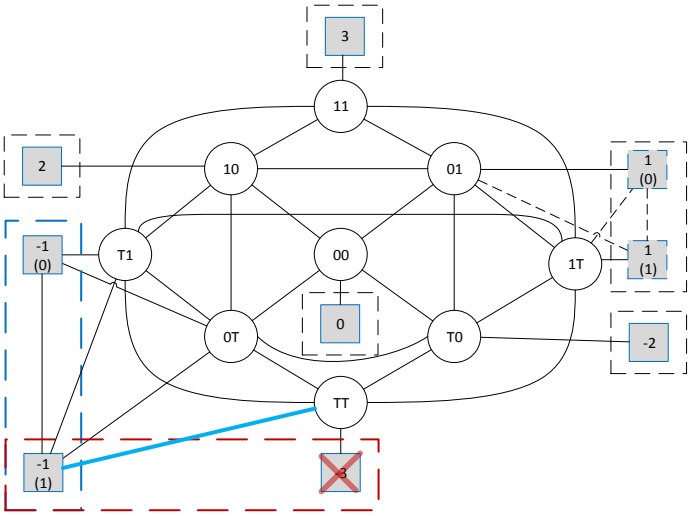

Fig. 12. An example of a shared processor mechanism

There are several possible implementations of such connections. Figure 13 shows the implementation of connections between VNs and topology switches.

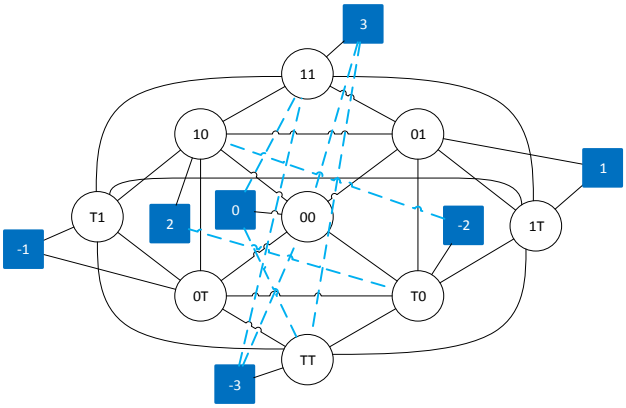

Fig. 13. Additional links created between VN and switches

This implementation is similar to conventional "quantum tunnels" in topology, allowing them to be used in the same mode. But there is a disanvantage: additional connections increase the degree of topology switches.

Therefore, another option is offered: direct virtual processor connections. Fig. 14 shows a model of this.

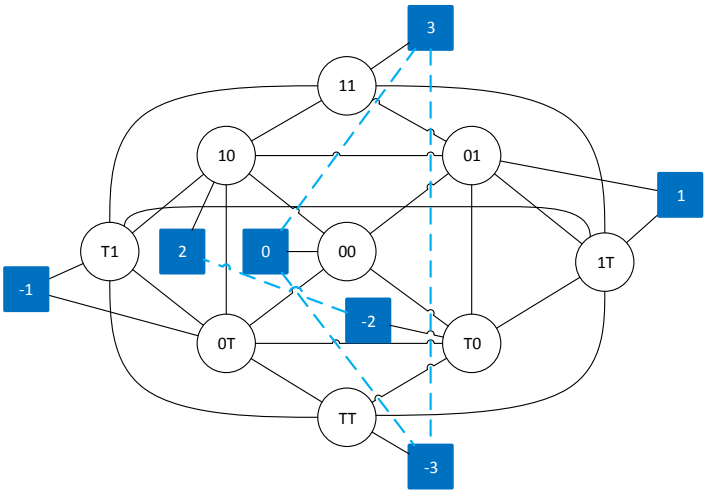

Fig. 14. Additional links created between VNs

Example of system. An example is how suggestions made allow the system to work with many failures. A hybrid implementation of virtual nodes is chosen as an example. It is suggested to use the fact that nodes -1 and 1 have 2 physical processors per node and use them to duplicate processors 3, 2, -2 and -3. It is also worth duplicating processor 0 with processors 3 and -3 (it is possible to use already existing duplicate links). Additional links are suggested to be made between virtual nodes. Fig. 15 shows a model of the described system.

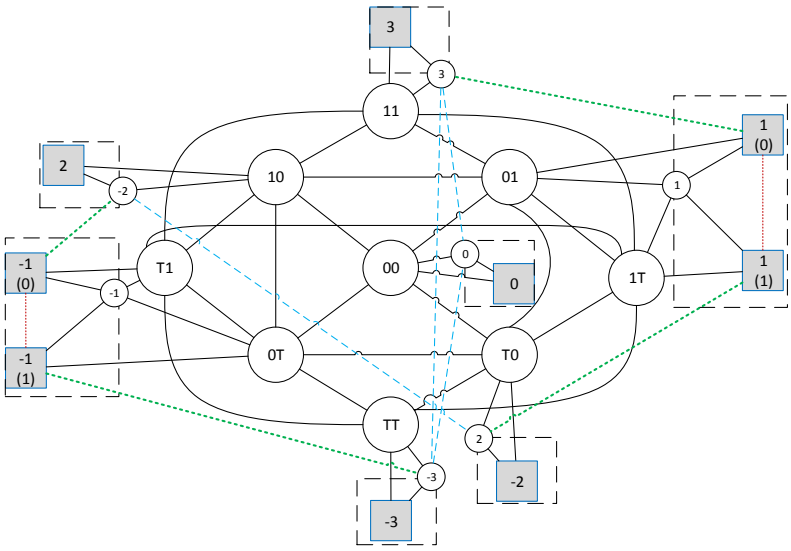

Fig. 15. Example of a system

Fig. 16 shows the failures made to the system as an example.

The next failures are suggested for example:

- Processors: 3, 2, -1(1), -2
- Switches: 10, 00, TT, T1

Switch of virtual node (VNS) № 1.

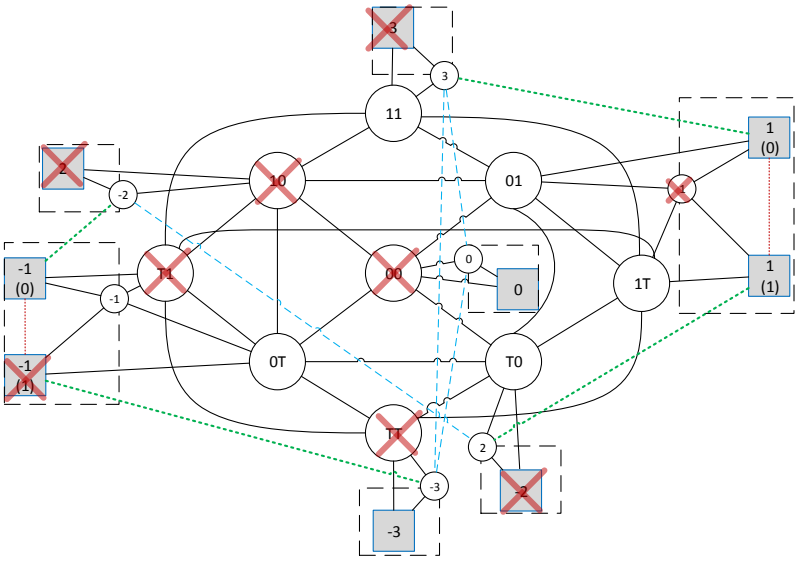

Fig. 16. Faults in system

This can be solved in the following way:

1. Processor 3 is reserved by processor 1(0). VNS 3 works, so 1(0) will be «given away» to node 3 and communicates with switch 11 through VNS 3.

2. Processor 2 is reserved by processor -1 (0). But processor -1 (1) fails, as result, -1 (0) is only working processor of node -1. Therefore the «shared usage» mechanism will be used: processor -1 (0) goes to multitasking mode and takes both roles (roles of processors 2 and -1(1)).

3. Processor -2 is reserved by processor 1(1). Because the processor 1(0) has already been used to replace the node 3, there are 2 variants: 1(0) and 1(1) goes to shared usage mode (providing performance P for node 1, and performance 0.5 * P for nodes 3 and 2). Or, only one of the processors will be shared and the other is fully "gives away". Given the failure of VNS 1, the first solution is optimal.

4. Switch 10 fails and node 2 is unreachable. But switch T0 works. Usage of «artificial quantum tunnel» allows restore access to node even in this critical case.

5. Switches 00 and TT fails, so their roles taken by switch 11.

6. Switch T1 fails, but switch 0T works. Those switches have same number and both connected to node -1. So, messages to node -1 may be delivered through switch 0T.

Fig. 17 depicts what described system restore looks like.

Analysis of proposed solution. Now proposals are shown and the fault-tolerance-associated possibilities are highlighted on example of system. As shown in the model, the combination of fault-tolerant topology and additional fault-tolerance ensuring methods allows to keep the system operational even in the presence of many failures. The proposed solution allows you to do this even without the use of additional

nodes. But it reduces system performance. Let each physical processor have speed P and the system is homogeneous. Table 2 shows, how reduces computing system elements' performance due to failures (see. Fig. 17).

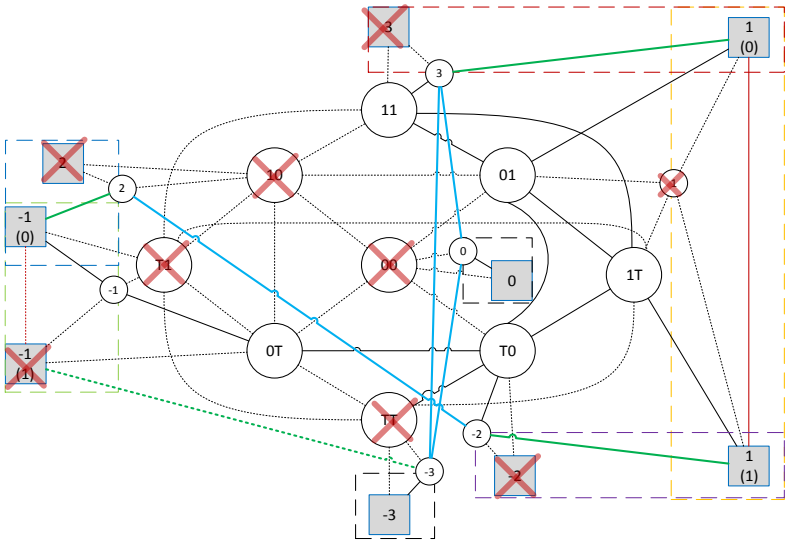

Fig. 17. Restored system

Table 2

Performance of each virtual node in cases with failures and without failures

Virtual node	No failures	With failures in example
-3	P	P
-2	P	P/2
-1	2P	P/2
0	P	P
1	2P	P
2	P	P/2
3	P	P/2
System's performance	9P	5P

Thus, the described solution allows you to redistribute the performance of the system to restore its structure. This allows for a high level of fault tolerance.

However, there are a number of requirements and disadvantages. First, the shared processor mechanism requires appropriate software support. The system should distribute tasks, providing multitasking. The same support is required for the routing algorithm. The algorithm must change the destination node if it fails. Another disadvantage is that the system has a fairly high degree and diameter. The third disadvantage is the difficulty of using such a solution for large systems. The proposed solution is universal for any system with a quasi-quantum topology. However, increasing the number of nodes will increase the complexity of fault-tolerance ensuring tools. The last drawback is the irregularity of additional backup connections.

The links shown are rigidly tied to the example and may vary. This prevents the system from analytically determining which node is reserving which node.

Conclusions. The article considers the method of implementing a fault-tolerant system with a switched communication system based on a fault-tolerant topology in switch network. The basic principles that can be implemented by the system to ensure fault tolerance are put forward. Variants of realization of these principles are described. Measures to additional increase the fault tolerance of the system are considered. An example of how these measures allow the system to work in case of failure is shown.

The advantages of the proposed solutions are high fault tolerance. Using a combination of hardware structure and the necessary software, the described solution allows you to perform the redistribution of computing resources, thereby hiding the failure.

Disadvantages include unsatisfactory topological characteristics, unresolved scaling issues, and irregularity of additional redundancy links.

Future publications may focus on regularization of backup links, ways to minimize the degree and diameter, and review the system when scaling.

References

1. Goncharenko Olexandr, Pavlo Rehida, Artem Volokyta, Heorhii Loutskii, and Vu Duc Thinh: Routing Method Based on the Excess Code for Fault Tolerant Clusters with InfiniBand. Advances in Intelligent Systems and Computing, vol. 938, pp. 335-345. Springer, Heidelberg (2019)

2. Washington N., Perros H.: Performance Analysis of Traffic-Groomed Optical Networks Employing Alternate Routing Techniques. Lecture Notes in Computer Science, vol. 4516, pp. 1048-1059. Springer, Berlin, Heidelberg (2007).

3. Hu, Z., Mukhin, V., Kornaga, Y., Volokyta, A., & Herasymenko, O. The scheduler for distributed computer systems based on the network centric approach to resources control. In: Proceedings of the 2017 IEEE 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2017, pp. 518-523(2017).

4. Emanouilidis, E and Bell, R. Latin squares and their inverses. Math. Gaz., vol 88(511), pp. 127–128 (2004)

5. Richard J.Cole, Bruce M.Maggs, Ramesh K.Sitaraman, On the Benefit of Supporting Virtual Channels in Wormhole Routers, Journal of Computer and System Sciences, vol. 62(1), pp 152-177 (2001)

6. Ian M. Wanless. Cycle Switches in Latin Squares, Graphs and Combinatorics, vol. 20(4), pp 545-570 (2004).

7. H. Loutskii, A. Volokyta, P. Rehida, O. Honcharenko, B. Ivanishchev and A. Kaplunov, "Increasing the fault tolerance of distributed systems for the Hyper de Bruijn topology with excess code," 2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT), Kyiv, Ukraine, 2019, pp. 1-6, doi: 10.1109/ATIT49449.2019.9030487.

AUTHORS

Oleksandr Honcharenko – student, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute” (Solomenskiy district, ave. Pobedy, 37, 03056, Kyiv, Ukraine).

E-mail: alexandr.ik97@ukr.net

Artem Volokyta (supervisor) – associate professor, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

E-mail: artem.volokita@kpi.ua

Heorhii Loutskii (supervisor) – professor, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.