*I. J. Computer Network and Information Security*, **2022, 6, 23-36**
Published Online on December 8, 2022 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2022.06.03

# Extended DragonDeBrujin Topology Synthesis Method

**Artem Volokyta***
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"/ Department of computer engineering, Kyiv, 03056, Ukraine
E-mail: artem.volokita@kpi.ua
ORCID iD: https://orcid.org/0000-0001-9069-5544
*Corresponding Author

**Heorhii Loutskii**
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"/ Department of computer engineering, Department of information systems and technologies, Kyiv, 03056, Ukraine
E-mail: georgijluckij80@gmail.com
ORCID iD: https://orcid.org/0000-0002-3155-8301

**Pavlo Rehida**
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"/ Department of computer engineering, Department of information systems and technologies, Kyiv, 03056, Ukraine
E-mail: pavel.regida@gmail.com
ORCID iD: https://orcid.org/0000-0002-6591-7069

**Artem Kaplunov**
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"/ Department of computer engineering, Department of information systems and technologies, Kyiv, 03056, Ukraine
E-mail: art.kaplunov@gmail.com
ORCID iD: https://orcid.org/0000-0002-2408-1015

**Bohdan Ivanishchev**
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"/ Department of computer engineering, Department of information systems and technologies, Kyiv, 03056, Ukraine
E-mail: callidus.iv@gmail.com
ORCID iD: https://orcid.org/0000-0003-4726-0689

**Oleksandr Honcharenko**
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"/ Department of computer engineering, Department of information systems and technologies, Kyiv, 03056, Ukraine
E-mail: alexandr.ik97@ukr.net
ORCID iD: https://orcid.org/0000-0002-9086-6988

**Dmytro Korenko**
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"/ Department of computer engineering, Department of information systems and technologies, Kyiv, 03056, Ukraine
E-mail: korenko.dima98@gmail.com
ORCID iD: https://orcid.org/0000-0003-0463-189X

Received: 28 April 2022; Revised: 17 July 2022; Accepted: 27 October 2022; Published: 08 December 2022

**Abstract:** Scaling high performance computer systems needs increasing the fault tolerance at the design stage of a topology. There are several approaches of designing simple fast routing with fault tolerance. One of effective approach is to ensure fault tolerance at the topology level. This article discusses two methods for optimizing topologies synthesized using Dragonfly and Excess De Brujin. Methods of topology saturation are discussing, which allow to increase the dimension of the system without deterioration of topological characteristics due to the optimization of the

 

synthesis method. Three scaling constraint methods are also proposed to reduce the topology dimension to the desired performance.

**Index Terms:** Dragonfly, De Bruijn, Multilevel Topological Organization.

## 1. Introduction and Related Work

The topology is the main factor, that determines parameters of a distributed system, such as speed of message transmission, complexity of routing, fault tolerance and the price of the system. The issue of optimal topology finding has no simple solution due to the mutual exclusivity of the requirements. High fault tolerance, acceptable topological characteristics, simple and efficient routing, good scalability but at the same time a wide range of system size choices are needed. As the result, issues of topological synthesis are relevant [1-6].

There are several issues, that are important for topology developing. First of all, it is fault tolerance ensuring. Modern high-performance systems are a large (up to millions of nodes and billions of cores), so, the failure rate is very high. Another important issue is the routing. Ease of routing directly affects the effectiveness of message transmission. Commonly used methods, such as tabular routing, are optimal for systems with a dynamic topology, but they are not a good choice for the static-structured system.

The key objective of the research is to propose methods for optimizing topologies synthesized using Dragonfly and Excess De Brujin. Consider methods of saturation of the topology, which allow to increase the size of the system without deterioration of topological characteristics, and scaling constraint methods to reduce the topology size to the desired values.

The key methodologies, used on this research, are a graphs theory and practice of fault tolerant routing. The structure of paper is presented. Introduction highlights actuality of this research, the objective and methodology are shown. In 2nd section the synthesis of multilevel topology based on Excess de Bruijn clusters is described. In 3rd section considered the DragonDeBruijn system. Chapter 4 discusses the scaling restrictions. Chapter 5 considers the issue of expanding (saturating) the topology. In 6rd section considered the obtained results. The 7th section it is conclusion. The main features of these are highlighted.

## 2. Review of Previous Studies

### *2.1. Excess de Bruijn as the Basis of Multilevel Structure*

This topology is an extension of the usual de Bruijn topology [7-10] using a redundant binary representation for nodes encoding [11-16]. The link system forms by operations of code shift, as the result, the degree of this graph is constant and independent from its rank. The other advantage of shift-based synthesis is the simplicity of routing. Furthermore, the issues of fault tolerance for this topology were discovered, the number of approaches were proposed. Fig 1 shows a topology of rank 2 as the example.

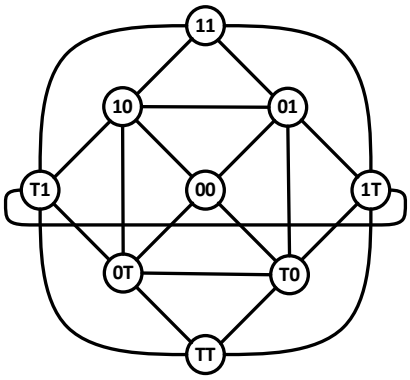


Fig.1. Excess de Bruijn rank 2

The main restriction of each topology is the characteristics, such a degree and diameter (that can be assessed complexly as SD characteristic). Among two topologies with a similar number of nodes, the best one is the one with the minimum SD characteristic. Among two topologies with a different number of nodes, the optimality of the SD characteristic depends on how much the number of nodes in them differs. This makes it relevant to estimate not only the pure SD but also the derivative characteristic of the ratio of SD to the number of nodes (N) in the topology. The second important consequence of this is related to the synthesis. In order to ensure the optimal SD of topology compared to other topological organizations with similar N, it is necessary to maximize the growth of N with scaling while

minimizing the growth in degree and diameter. This problem can be easily solved using Dragonfly-like interconnection approach [2, 16].

*2.2. Synthesis of Dragonfly-like Multilevel Structure*

The main idea of synthesis is next: in other next step of scaling topology must include self in previous rank as cluster. It allows to maximize the N growth rate and removes needs to lower-stage structure rebuilding. The simplest topology with 2 elements is proposed as example to show basic synthesis principles. Consider this topological structure as a basic cluster - a system of rank 1. It is shown in fig. 2, left part. Supposing, that nodes in this topology are encoded in usual binary system as 0 and 1. Respectively, the system of rank 2 must contents 4 nodes, encoded as 00, 01, 10 and 11. So, if the lower part of each code represents the in-cluster address, higher part will show the cluster address. Denote these addresses as a (cluster) and b (in-cluster). So, full address may be denoted in next view:

$$A = a.b \tag{1}$$

Supposing, that every node has one and only one external neighbor, that can be founded only by using code transformation on each current node. So, the rule is next:

$$a_{dest} = b, b_{dest} = a \tag{2}$$

And, the final destination address may be presented in next view:

$$A_{dest} = a_{dest} b_{dest} = b.a \tag{3}$$

Using these synthesis rules with respect to the presented addresses, one can obtain the rank-2 system shown in fig.2.

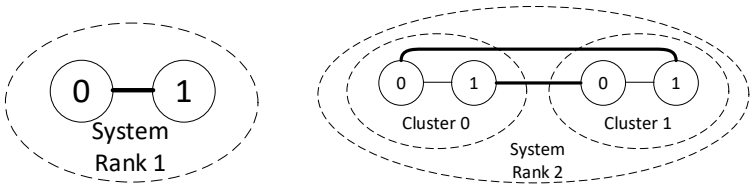


Fig.2. Exampled cluster (left) and system of rank 2 (right)

Continuing the synthesis, we should consider a system of rank 2 as a cluster. From the above synthesis rules, it can be seen that parts a and b are assumed to be equal in length and interchangeable. Therefore, a rank 3 topology must include a rank 2 system as a cluster and, as a result, allocate 4 bits for node addresses (2 for in-cluster, 2 for cluster address). Using the rules, presented in formulas 1-3, the connection system may be built. The result is shown in fig. 3.

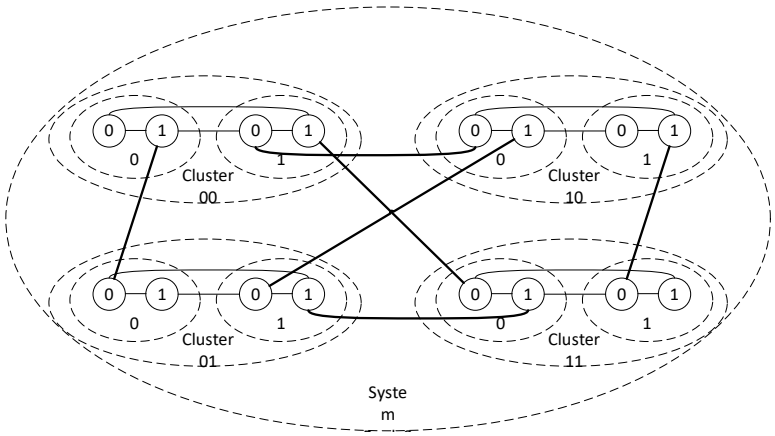


Fig.3. System for example, rank 3

Regarding encoding, the synthesis is as follows: the internal address of the node becomes its full address from the previous step, and the external address is the number of the cluster, which includes this node. This recursive principle of synthesis may be simply used for routing.

### *2.3. DragonDeBruijn System*

Using the same principle for excess de Brujin cluster, the DragonDeBrujin topology been obtained [16]. This topology of rank 2 is shown in fig. 4.

One of the interesting aspects of the DragonDeBruijn is excess clusters, that duplicates others due to the same number, but different representation in code. In this case, such clusters are pairs 01-1T and 0T-T1. The approaches of using excess nodes for improving fault tolerance was considered and these are same for the clusters. Actually, if even all excess cluster fails, it can be replaced by other and continue its computations.

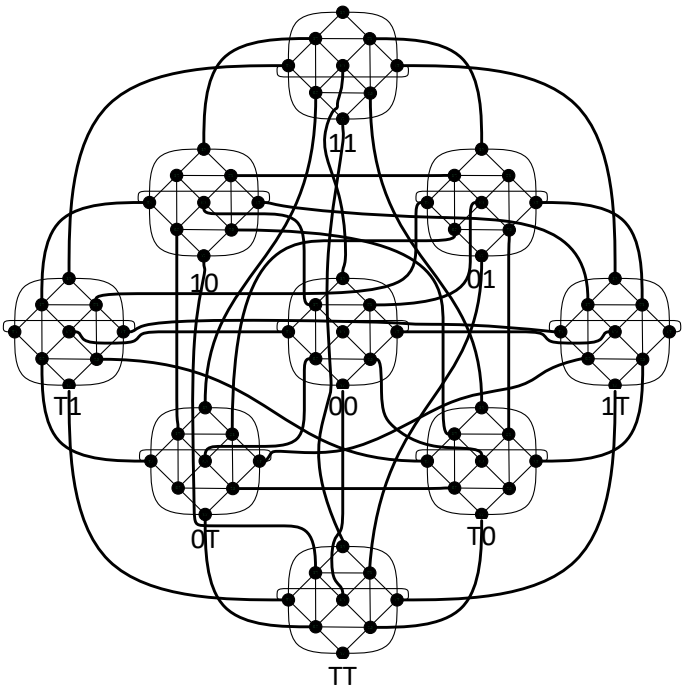


Fig.4. DragonDeBruijn system, rank 2 [16]

Moreover, it makes possible to use the quasi-quantum routing in addition to usual algorithm, that allows to bypass failures by sending data through excess nodes [15]. In the case of this topology, this allows you to maintain connectivity even in a situation where all inter-cluster links are in a failure state [16].

### *2.4. Disagvantages of Basic DragonDeBrujin Topology*

The key disadvantage of the proposed topology is its too fast scaling ($N = 9^{2^R}$). This makes it difficult to expand the system, and also makes it impossible to choose from the available ranks of the topology that would meet the needs for computing resources.

Another problem is that the topology does not make full use of the scaling resource. At each scaling step, there is always one node in each cluster that has no external connection. This makes the degree of topology irregular and does not allow to fully achieve the goals set for this system (linear growth of the degree and moderate SD with an explosive increase in dimension).

## 3. Proposed Methods

### *3.1. Topological Truncation (Restriction of Scaling)*

As the number of nodes in this type of topology grows explosively, there is a need to limit growth. Create a "cap" that has a lower growth rate. The simplest method to do this is to perform a Cartesian product on another topology, which will be external to this one. In fig. 5 shows an example of the product of a simple topology of rank 2 on a line of 2 processors.

On the one hand, this method allows you to get any number of nodes, and the correct choice of the plug makes it possible to reduce the diameter. On the other hand, this method of restriction destroys the entire routing structure.

The topology parameters will be described by the following formulas (lowercase parameters indicate the parameters related to the stub):

$$N_{R+1,P} = n_P N_R = n_p N_1^{2^R} \tag{4}$$

$$S_{R+1,P} = S_R + s_P = S_1 + R + s_P \tag{5}$$

$$D_{R+1,P} = D_R + d_P = 2^R (D_1 + 1) - 1 + d_P \tag{6}$$

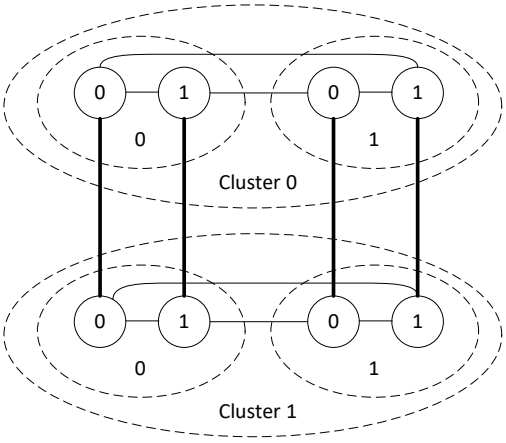


Fig.5. Cartesian product Dragonfly-like rank 2 on a line of 2 processors

An alternative way is to reduce the external number. Its essence is as follows. It is known that at each stage of scaling the vertex code consists of 2 parts: internal number and external. Both have the same bit rate. To find the outer neighbor of the vertex, these numbers are rearranged. The essence of the method is to reduce the bit size of the external number so as to obtain a topology of the desired order. When changing places, only the junior ranks of the internal number take part in the substitution, the senior ones are ignored. This allows you to keep the logic of the synthesis unchanged and ensure the operation of the rule "1 neighbor from the outside".

As for routing, it's a bit more complicated. Since it is now impossible to perform the transformation $ab \rightarrow ac$ due to different bits a, c and b, d, the algorithm needs to be clarified.

1. Let $A_{src} = axb, A_{dest} = cyd$. In this case, xb and yd are internal numbers, a and c are external numbers.
2. Perform transfer intra-cluster.

$$axb \rightarrow axc$$

3. Transfer inter-cluster.

$$axc \rightarrow cxa$$

4. Perform transfer intra-cluster.

$$cxa \rightarrow cyd$$

Similarly, all previously proposed improvements to the algorithm remain valid, only the internal parts of the code x and y are added.

In fig. 6 shows an example of a reduced system (for example, Dragonfly-like).

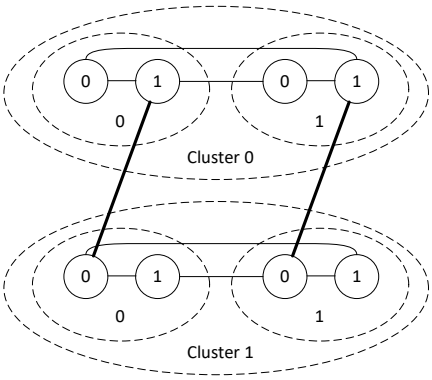


Fig.6. Reduced Dragonfly-like system of rank 3

With a clear example, you can do the same for the DragonDeBruijn topology and get connections for a reduced order system of the 2nd order. If a normal system of rank 2 has 81 nodes and is described by 4 redundant digits, then the reduced one consists of 27 nodes and contains 3 digits. Table 1 shows the links for such a system.

Table 1. Different measurement measuring mixed state estimation results

| Node's number | | Linked to | | Node's number | | Linked to | | Node's number | | Linked to | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Outer part | Inner part | Outer part | Inner part | Outer part | Inner part | Outer part | Inner part | Outer part | Inner part | Outer part | Inner part |
| 0 | 00 | 0 | 00 | 1 | 00 | 0 | 01 | T | 00 | 0 | 0T |
| 0 | 01 | 1 | 00 | 1 | 01 | 1 | 01 | T | 01 | 1 | 0T |
| 0 | 0T | T | 00 | 1 | 0T | T | 01 | T | 0T | T | 0T |
| 0 | 10 | 0 | 10 | 1 | 10 | 0 | 11 | T | 10 | 0 | 1T |
| 0 | 11 | 1 | 10 | 1 | 11 | 1 | 11 | T | 11 | 1 | 1T |
| 0 | 1T | T | 10 | 1 | 1T | T | 11 | T | 1T | T | 1T |
| 0 | T0 | 0 | T0 | 1 | T0 | 0 | T1 | T | T0 | 0 | TT |
| 0 | T1 | 1 | T0 | 1 | T1 | 1 | T1 | T | T1 | 1 | TT |
| 0 | TT | T | T0 | 1 | TT | T | T1 | T | TT | T | TT |

As you can see, all the nodes in which the junior and senior digits coincide have no neighbors. All others have exactly 1 neighbor, all of which are strictly external, and the frequency of inter-cluster connections is uniform.

The example system has only 1 reduction method (from 4 to 3 digits). Of course, this will not be so easy for higher order systems. For a rank 2 system, there are at least 3 reduction methods (up to 3, 2, or 1 external discharge, respectively). For a system of rank 3, there are 7 ways (in which 7 to 1 external digits are stored, respectively). And so on. Reduction is strongly associated with a number system that encodes the nodes of the base cluster. Thus, this method allows reducing the growth of the topology from a two-level power dependence to BP, where B is the number of digits in the number system that encodes the vertex numbers, P is the number of digits of the external number. Knowing this, you can derive a formula for the number of nodes.

$$N_{R+1,P} = B^P N_R = B^P N_1^{2^{R-1}} \tag{7}$$

It is easy to get a formula for a degree. Since the rule "1 neighbor from the outside" is maintained, the degree of reduced topology will be the same as in a conventional topology of the same rank.

$$S_{R+1,P} = S_{R+1} = S_1 + R \tag{8}$$

As for the diameter, it is much more difficult to derive a formula here. Obviously, the diameter of the reduced topology cannot exceed the diameter of the non-reduced topology, but at the same time it cannot be smaller than the diameter of the cluster. However, it can be argued that a reduction of the topology by B times leads to a corresponding increase in the density of inter-cluster connections. Thus reducing intra-cluster routing: if previously there was only 1 path to the destination cluster, now they are B, and therefore, the step of inter-cluster transition can be performed much earlier than through DR steps. However, it is unfortunately not possible to accurately estimate this reduction.

$$D_{R+1,P} < D_{R+1} \tag{9}$$

The third method that reduces the number of nodes of the topology is its rarefaction. The idea is to remove some of the clusters along with all the connections that belong to them. An example of a sparse Dragonfly-like system of rank 3 is shown in fig. 7

The advantage is that the increase in the number of nodes will now be described by the following linear formula.

$$N_{R+1,P} = P N_R = P N_1^{2^{R-1}} \tag{10}$$

The degree and diameter will be equivalent to the parameters of the undiluted topology of the corresponding rank.

$$S_{R+1,P} = S_{R+1} = S_1 + R \tag{11}$$

$$D_{R+1,P} = D_{R+1} = 2^R (D_1 + 1) - 1 \tag{12}$$

A key drawback is the need to remember which clusters are "forbidden" when looking for a workaround. With normal routing, such a problem should not arise: it is obvious that for any pair of numbers $A_{src} = ab$, $A_{dest} = cd$, it is impossible when the component a or c refers to the "forbidden". As for components b and d, they are important only for intra-cluster routing.

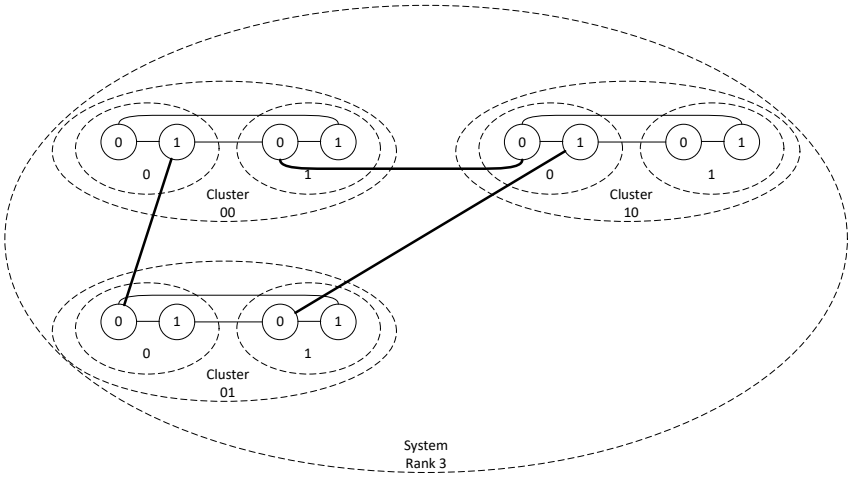


Fig.7. Sparse (without cluster 11) Dragonfly-like rank 3

### *3.2. Expansion (Saturation) of the Topology*

Along with the reduction, the topology extension method should also be considered. The proposed synthesis system provides more or less uniform connectivity between parts of the system, but assumes that at each iteration in each cluster there is exactly one vertex that has no external connection. This is the vertex A = aa, ie, one in which the internal and external numbers coincide. The presence of an unconnected vertex dysregulates the topology and leads to the failure to use all the opportunities to increase productivity. Since, regardless of the scaling step, such nodes in the topology will be exactly NR-1 (equal to the number of clusters and the number of nodes in each cluster), this makes it possible to add another 1 cluster to the topology. This will bring the topology to the saturation point, when all nodes on each iteration will be used for expansion. This addition will not change the topological characteristics, will not affect the routing and will increase the fault tolerance of the system. In fig. 8 on the example of the Dragonfly-like system shows how this can be done.

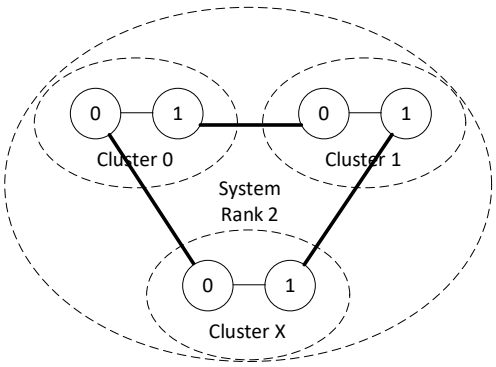


Fig.8. Additional cluster in Dragonfly-like (Saturated Dragonfly-like), rank 2

Because the nodes are encoded in binary code, there is a problem: there is no unique notation for the new cluster. Therefore, it is denoted as X. When applying the method only to the upper level of the system, there are no problems: an additional cluster can be used purely as a source of nodes to replace failures, as well as an additional bypass. However, scaling such a topology causes this "broken" encoding to propagate, and there is a need to fix it in some way. In fig. 9 shows the same topology scaled to rank 3.

As can be seen, the number X introduced at the previous stage has spread, "creating" clusters X0 and X1. Interestingly, clusters 0X and 1X did not appear, that is, a pure binary system is still used for the lower digit of the number. But a Y cluster appeared. It is logical to assume that the same thing will happen in the next iteration, and will lead to a bunch of "strange" numbers, such as ZYX0, 0000Y00, 000010X1, Z0110, and so on. As you can see, this will lead to a catastrophe at the level of number coding: if previously it was described by a clear number of bits, now its bit has become variable.

Multilevel coding can be used to solve this problem. Yes, now each level has its own numbering, and the full node number is described by several numbers. In fig. 10 shows the same system in multilevel coding.

Multilevel coding can be used to solve this problem. Yes, now each level has its own numbering, and the full node number is described by several numbers. In fig. 10 shows the same system in multilevel coding.

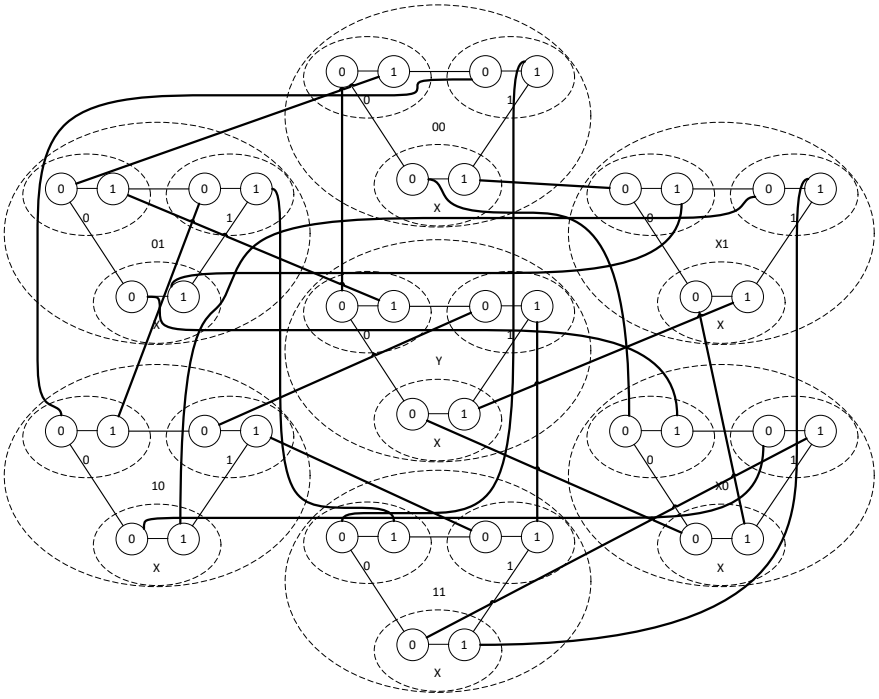

Fig.9. Saturated Dragonfly-like, rank 3

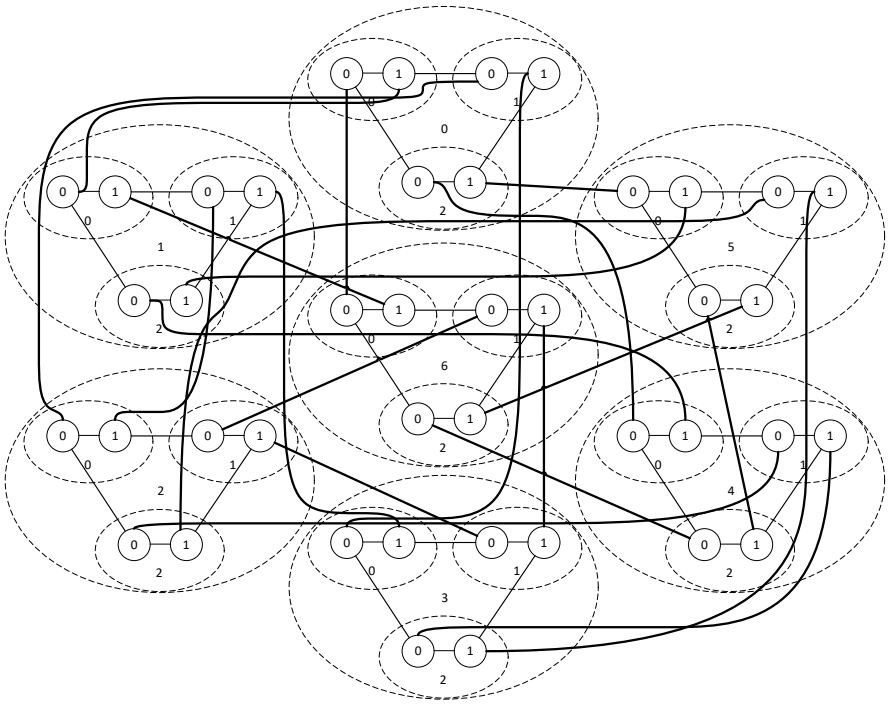

Fig.10. Saturated Dragonfly-like rank 3, multilevel encoding

Of course, the disadvantage of this solution is the change in routing. Now to find an external neighbor, you must perform the following steps.

$$A = ab = a.b_{R-1}.b_{R-2}\dots b_1.b_0 \tag{13}$$

$$a_{linked} = value(b) = \sum_{i=0}^{R-1} value(b_i) = \\ = \sum_{i=0}^{R-1} b_i \prod_{j=0}^{i} basis(j) = \sum_{i=0}^{R-1} b_i \prod_{j=0}^{i} N_j \tag{14}$$

Where $N_j$ is the number of clusters on the next iteration of scaling. It should be noted that although such a "code" cannot be fully considered an integral code, each b will give its unique meaning a. Accordingly, to obtain the value of b, it is necessary to perform the inverse operation, and convert the value of a into "irregular code".

$$b_{linked} = expxpand(a) \tag{15}$$

This operation can be performed iteratively, through sequential taking of the module and integer division on the basis. That is, in the same way as it is performed when converting between number systems, with the difference that in this case the "number system" is not regular.

An even more difficult issue is the application of the topology saturation method to the DragonDeBruijn network, as it involves the use of redundant code. This means that there is no way to guarantee a unique match between nodes when building external links. The search for such a method, as well as the study of the properties of irregular code and methods of generating redundant irregular codes is a possible direction for further research.

As for the characteristics of the saturated topology, they completely coincide with the usual topology of the same type, only the growth rate of the number of nodes differs. For a saturated topology, it is described by the following recurrent formula

$$N_{R+1} = N_R^2 + N_R = N_R(N_R + 1) \tag{16}$$

## 4. Results

Table 2 presents the results for both the topologies already considered and their saturated nodes. In addition to the classical topological characteristics, 2 additional ones were also used for the analysis: the ratio of SD to N (denoted as J) and the ratio of the increase of SD to the increase of N with rank (denoted as M).

Also, these results may be represented as a chart, shown in the Fig. 11. In this chart the relation between count of nodes and SD are presented. The given data show that the use of the proposed method makes it possible to significantly optimize the use of the external system of connections and, due to their regularization, to achieve an improvement in the ratio of the number of nodes to the SD characteristic.

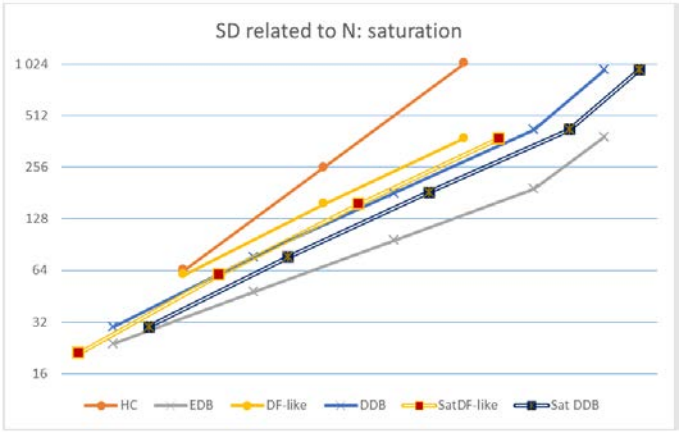


Fig.11. Dependency SD(N) for considered topologies before and after saturation

Another solution is scaling restrictions. While saturation eliminates gaps on the link system, constraints, on the other hand, create gaps to reduce the number of processors required to build a system and thus adapt it to the developer's financial capabilities and users' computing needs.

Table 3 shows the characteristics of the topologies selected for the desired number of processors.

The assumption of results may be graphically presented in a next chart. Fig. 12 shows dependencies between SD-characteristic and count of nodes (N) for 8 topologies considered above.

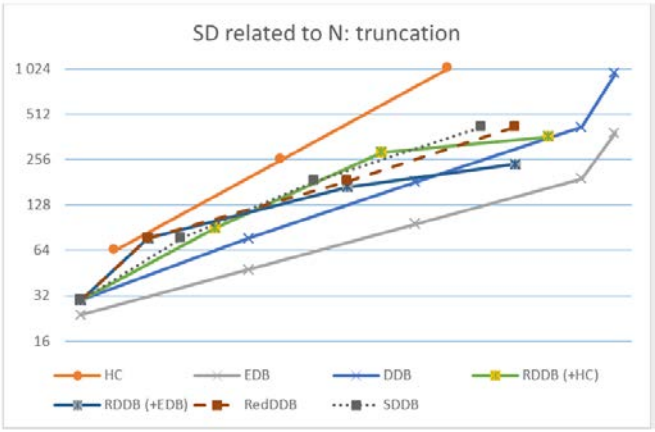


Fig.12. Dependency SD(N) for considered topologies before and after different truncation methods use.

Table 2. Topological characteristics

| Rank | | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Topology | Param. | | | | | | |
| Hypercube (1) | N | 2 | 4 | 8 | 16 | 32 | 64 |
| | S | 1 | 2 | 3 | 4 | 5 | 6 |
| | D | 1 | 2 | 3 | 4 | 5 | 6 |
| | SD | 1 | 4 | 9 | 16 | 25 | 36 |
| | J | 0.5 | 1 | 1.125 | 1 | 0.78 | 0.56 |
| | M | - | 1.5 | 1.25 | 0.875 | 0.56 | 0.34 |
| Excess De Bruijn (2) | N | 3 | 9 | 27 | 81 | 243 | 729 |
| | S | 2 | 5 | 6 | 6 | 6 | 6 |
| | D | 1 | 2 | 3 | 4 | 5 | 6 |
| | SD | 2 | 10 | 18 | 24 | 30 | 36 |
| | J | 0.66 | 1.11 | 0.66 | 0.29 | 0.12 | 0.04 |
| | M | - | 1.33 | 0.44 | 0.11 | 0.03 | 0.01 |
| Dragonfly-like (3) | N | 2 | 4 | 16 | 256 | 65536 | 429496729 |
| | S | 1 | 2 | 3 | 4 | 5 | 6 |
| | D | 1 | 3 | 7 | 15 | 31 | 63 |
| | SD | 1 | 6 | 21 | 60 | 155 | 378 |
| | J | 0.5 | 1.5 | 1.3125 | 0.23 | 0.002 | 8.801E-08 |
| | M | - | 2.5 | 1.25 | 0.1625 | 0.0014 | 5.19E-08 |
| Dragon DeBruijn (4) | N | 9 | 81 | 6561 | 43046721 | 1.85E+15 | 3.43E+30 |
| | S | 5 | 6 | 7 | 8 | 9 | 10 |
| | D | 2 | 5 | 11 | 23 | 47 | 95 |
| | SD | 10 | 30 | 77 | 184 | 423 | 950 |
| | J | 1.11 | 0.37 | 0.011 | 4.27E-06 | 2.28E-13 | 2.76E-28 |
| | M | - | 0.27 | 0.007 | 2.48E-06 | 1.28E-13 | 1.53E-28 |
| Saturated Dragonfly-like (5) | N | 2 | 6 | 42 | 1806 | 3263442 | 1.07E+13 |
| | S | 1 | 2 | 3 | 4 | 5 | 6 |
| | D | 1 | 3 | 7 | 15 | 31 | 63 |
| | SD | 1 | 6 | 21 | 60 | 155 | 378 |
| | J | 0.5 | 1 | 0.5 | 0.033223 | 4.75E-05 | 3.55E-11 |
| | M | - | 1.25 | 0.4167 | 0.022108 | 2.91E-05 | 2.09E-11 |
| Saturated Dragon DeBruijn (6) | N | 9 | 90 | 8190 | 67084290 | 4.5E+15 | 2.03E+31 |
| | S | 5 | 6 | 7 | 8 | 9 | 10 |
| | D | 2 | 5 | 11 | 23 | 47 | 95 |
| | SD | 10 | 30 | 77 | 184 | 423 | 950 |
| | J | 1.1111 | 0.3333 | 0.0094 | 2.74E-06 | 9.4E-14 | 4.69E-29 |
| | M | - | 0.2469 | 0.0058 | 1.6E-06 | 5.31E-14 | 2.60E-29 |

Table 3. Topological characteristics for pre-determined count of processors

| Desired N | | 100 | 1 000 | 1 000 000 | 1 000 000 000 |
|---|---|---|---|---|---|
| Topology | Param. | | | | |
| Hypercube (HC) | $N_{Real}$ | 128 | 1 024 | 1 048 576 | 1 073 741 824 |
| | S | 7 | 10 | 20 | 30 |
| | D | 7 | 10 | 20 | 30 |
| | SD | 49 | 100 | 400 | 900 |
| | J | 0.38 | 0.098 | 3.8147E-4 | 8.3819E-07 |
| Excess De Bruijn (EDB) | $N_{Real}$ | 81 | 729 | 1 594 323 | 1 162 261 467 |
| | S | 6 | 6 | 6 | 6 |
| | D | 4 | 6 | 13 | 19 |
| | SD | 24 | 36 | 78 | 114 |
| | J | 0.29 | 0.049 | 4.89E-05 | 9.81E-08 |
| Dragon DeBruijn (DDB) | $N_{Real}$ | 81 | 6561 | 43 046 721 | 1.85302E+15 |
| | S | 6 | 7 | 8 | 9 |
| | D | 5 | 11 | 23 | 47 |
| | SD | 30 | 77 | 184 | 423 |
| | J | 0.37 | 0.011 | 4.27E-06 | 2.28E-13 |
| Restricted DDB (by Cartesian prod. with hypercube) | $N_{Real}$ | 81 | 1296 (81x16) | 1 679 616 (6561x256) | 1 377 495 072 (43046721x32) |
| | S | 6 | 10 | 15 | 13 |
| | D | 5 | 9 | 19 | 28 |
| | SD | 30 | 90 | 285 | 364 |
| | J | 0.37 | 0.069 | 1,697E-4 | 2.6425E-7 |
| Restricted DDB (by Cartesian prod. with EDB) | $N_{Real}$ | 81 | 729 | 1 594 323 | 1 162 261 467 |
| | S | 6 | 11 | 12 | 12 |
| | D | 5 | 7 | 14 | 20 |
| | SD | 30 | 77 | 168 | 240 |
| | J | 0.37 | 0.1056 | 1.0537E-4 | 2,0659E-7 |
| Reduced DDB (RedDDB) | $N_{Real}$ | 81 | 729 | 1 594 323 | 1 162 261 467 |
| | S | 6 | 7 | 8 | 9 |
| | D | 5 | 11 | 23 | 47 |
| | SD | 30 | 77 | 184 | 423 |
| | J | 0.37 | 0.1056 | 1,154E-4 | 3.639E-7 |
| Sparsed DDB (SDDB) | $N_{Real}$ | 81 | 1053 (81x13) | 1 003 833 (6561x153) | 1 033 121 304 (43046721x24) |
| | S | 6 | 7 | 8 | 9 |
| | D | 5 | 11 | 23 | 47 |
| | SD | 30 | 77 | 184 | 423 |
| | J | 0.37 | 0.073 | 1.83E-04 | 4.094E-07 |

## 5. Discussions

The main advantage of proposed saturation method is possibility to keep degree and diameter while adding some additional nodes. This is possible due to the fact that the basic formulas for the synthesis of Dragonfly-like topologies (and, as a result, DragonDeBrujin too) contain shortcomings that create irregularities in the topology and reduce the density of nodes. Thus, topology regularization, i.e., elimination of irregularities, allows you to use previously unused opportunities to linking and, as a result, apply this to connect a certain number of new nodes. However, the problem is a significant complication of synthesis formulas, which negatively affects routing.

Regarding truncation methods, the presented data show that, despite all the proposed approaches, the best solution in terms of SD characteristics is EDB. However, the key feature of dragonfly-like structures is the ability to modify them using the proposed methods, which makes it possible to achieve the maximum approximation to the desired characteristics and size of the system. As can be seen, sparsing is optimal for size selection, but final characteristics are not satisfactory. Too high a diameter, which is usually compensated by a large number of nodes, in this case nullifies all the gains from the low degree.

Reducing approach allows reduce the size of the system according to the properties of the encoding (binary or

RBR), which limits it in the context of approaching the desired performance. On the other hand, this approach, with some addition, could partly solve the problem of a large diameter at the cost of increasing the connectivity of clusters.

Anyway, with a large system size, both of these solutions offer better SD performance than the classic hypercube, although significantly inferior to the excess De Brujin. The variant of the restriction, based on the Cartesian product with the excess De Brujin, showed the best results.

In addition to limiting scaling, the proposed methods also address a number of other issues. For example, if parts of the system fail, this system can be considered sparse, and therefore, to replace "bypass routing" with the necessary routing. Regarding the reduced system, the division of topology into equal parts is one of the important tasks in graph theory, and the availability of standard methods of working with such separation (including appropriate routing methods) allows to use it in planning, balancing traffic and fault tolerance.

## 6. Conclusions

This paper proposes two methods for optimizing topologies synthesized using Dragonfly and Excess De Brujin. Methods of topology saturation are considered, which allow to increase the dimension of the system without deterioration of topological characteristics due to the optimization of the synthesis method. Three scaling constraint methods are also proposed to reduce the topology dimension to the desired performance.

It is proved that the use of the proposed method of saturation to the DDB topology allows (while maintaining the topological characteristics) to achieve an expansion of the topology from 10% (at the first scaling step) to 591% (at the fifth step). Similarly, the proposed truncation methods make it possible to minimize the discrepancy between the actual and desired number of nodes to a difference of 20-40%, while without their use, these values can differ by orders of magnitude. The main disadvantage of the proposed solutions is their rigid attachment to the properties of the DDB topology, which complicates their use in other cases.

## References


[1] Alverson, Robert, Duncan Roweth, and Larry Kaplan. "The gemini system interconnect." In 2010 18th IEEE Symposium on High Performance Interconnects, pp. 83-87. IEEE, 2010.

[2] Kim, John, Wiliam J. Dally, Steve Scott, and Dennis Abts. "Technology-driven, highly-scalable dragonfly topology." In 2008 International Symposium on Computer Architecture, pp. 77-88. IEEE, 2008.

[3] Ajima, Yuichiro, Shinji Sumimoto, and Toshiyuki Shimizu. "Tofu: A 6D mesh/torus interconnect for exascale computers." Computer 11 (2009): 36-40.

[4] Guan K. C., Chan V. W. S. Cost-efficient fiber connection topology design for metropolitan area WDM networks //IEEE/OSA Journal of Optical Communications and Networking. – 2009. – Т. 1. – №. 1. – С. 158-175.

[5] Yasir Arfat, Fathy Elbouraey Eassa,"A Survey on Fault Tolerant Multi Agent System", International Journal of Information Technology and Computer Science, Vol.8, No.9, pp.39-48, 2016.

[6] A. A. Atallah, G. B. Hamad and O. A. Mohamed, "Fault-Resilient Topology Planning and Traffic Configuration for IEEE 802.1Qbv TSN Networks," 2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS), 2018, pp. 151-156, doi: 10.1109/IOLTS.2018.8474201.

[7] Ganesan, Elango, and Dhiraj K. Pradhan. "The hyper-debruijn networks: Scalable versatile architecture." IEEE Transactions on parallel and distributed systems 4.9 (1993): 962-978.

[8] Dürr, F. (2016). A Flat and Scalable Data Center Network Topology Based on De Bruijn Graphs. arXiv preprint arXiv:1610.03245.

[9] Kamal, Md Sarwar, et al. "De-Bruijn graph with MapReduce framework towards metagenomic data classification." International Journal of Information Technology 9.1 (2017): 59-75.

[10] Peng, G., Ji, P. & Zhao, F. A novel codon-based de Bruijn graph algorithm for gene construction from unassembled transcriptomes. Genome Biol 17, 232 (2016). https://doi.org/10.1186/s13059-016-1094-x

[11] Olexandr G., Rehida P., Volokyta A., Loutskii H., Thinh V.D. (2020) Routing Method Based on the Excess Code for Fault Tolerant Clusters with InfiniBand. Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing, vol 938. Springer, Cham

[12] Oleksandr Honcharenko, Artem Volokyta, Heorhii Loutskii. Fault-tolerant topologies synthesis based on excess code usign the latin square. The International Conference on Security, Fault Tolerance, Intelligence ICSFTI2019, Ukraine, Kyiv, 14-15 May, 2019, pp. 72-81 (2019).

[13] Loutskii, H., Volokyta, A., Rehida, P.,Goncharenko, O. (2019). Using excess code to design fault-tolerant topologies. Technical sciences and technologies, 1 (15), 134–144.; DOI - https://dx.doi.org/DOI: 10.25140/2411-5363-2019-1(15)-134-144.

[14] Loutskii H., Volokyta A., Rehida P., Honcharenko O., Thinh V.D. (2021) Method for Synthesis Scalable Fault-Tolerant Multi-level Topological Organizations Based on Excess Code. In: Hu Z., Petoukhov S., Dychka I., He M. (eds) Advances in Computer Science for Engineering and Education III. ICCSEEA 2020. Advances in Intelligent Systems and Computing, vol 1247. Springer, Cham. https://doi.org/10.1007/978-3-030-55506-1_32

[15] 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.

[16] Loutskii, H., Volokyta, A., Rehida, P., Kaplunov, A., Ivanishchev, B., Honcharenko, O., Korenko, D. (2021, January). Topology Synthesis Method Based on Excess De Bruijn and Dragonfly. In International Conference on Computer Science,

Engineering and Education Applications (pp. 315-325). Springer, Cham. https://doi.org/10.1007/978-3-030-80472-5_27

## Authors' Profiles

**Associate professor Artem Volokyta**, Department of Computer Engineering, National Technical University of Ukraine, "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine.
(ORCID ID https://orcid.org/0000-0001-9069-5544)

Major interests: High-performance computer systems and networks: theory, methods and means of hardware and software implementation; design of fault-tolerant distributed computing systems; network topological organization.

**Professor Heorhii Loutskii**, Department of Computer Engineering, National Technical University of Ukraine, "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine
(ORCID ID https://orcid.org/0000-0002-3155-8301)

Major interests: High-performance computer systems and networks: theory, methods and means of hardware and software implementation; design of fault-tolerant distributed computing systems; network topological organization.

**Assistant Pavlo Rehida**, Department of Computer Engineering, National Technical University of Ukraine, "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine
(ORCID ID https://orcid.org/0000-0002-6591-7069)

Major interests: High-performance computer systems and networks: theory, methods and means of hardware and software implementation; design of fault-tolerant distributed computing systems; network topological organization.

**Assistant Artem Kaplunov**, Department of Computer Engineering, National Technical University of Ukraine, "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine
(ORCID ID https://orcid.org/0000-0002-2408-1015)

Major interests: High-performance computer systems and networks: theory, methods and means of hardware and software implementation; design of fault-tolerant distributed computing systems; network topological organization.

**Assistant Bohdan Ivanishchev**, Department of Computer Engineering, National Technical University of Ukraine, "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine
(ORCID ID https://orcid.org/0000-0003-4726-0689)

Major interests: High-performance computer systems and networks: theory, methods and means of hardware and software implementation; design of fault-tolerant distributed computing systems; network topological organization.

**PHD Student Oleksandr Honcharenko**, Department of Computer Engineering, National Technical University of Ukraine, "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine
(ORCID ID https://orcid.org/0000-0002-9086-6988)

Major interests: High-performance computer systems and networks: theory, methods and means of hardware and software implementation; design of fault-tolerant distributed computing systems; network topological organization.

**PHD Student Dmytro Korenko**, Department of Computer Engineering, National Technical University of Ukraine, “Igor Sikorsky Kyiv Polytechnic Institute”, Ukraine
(ORCID ID https://orcid.org/0000-0003-0463-189X)

Major interests: High-performance computer systems and networks: theory, methods and means of hardware and software implementation; design of fault-tolerant distributed computing systems; network topological organization.

**How to cite this paper:** Artem Volokyta, Heorhii Loutskii, Pavlo Rehida, Artem Kaplunov, Bohdan Ivanishchev, Oleksandr Honcharenko, Dmytro Korenko, "Extended DragonDeBrujin Topology Synthesis Method", International Journal of Computer Network and Information Security(IJCNIS), Vol.14, No.6, pp.23-36, 2022. DOI:10.5815/ijcnis.2022.06.03