Network optimization

TOMAS FENCL, JAN BILEK
Department of Control Engineering, Faculty of Electrical Engineering
Czech Technical University in Prague
K335 ; Technická 2; 166 27 PRAHA 6
CZECH REPUBLIC

Abstract: - This document deals with a design of a physical and a logical topology of communication networks that are applied in the control engineering. The design of the physical topology works with aspects of demands for redundant links between nodes. Thanks to knowledge of the physical topology, we can design the logical topology according to permitted delays of delivered communication frames. The whole procedure of a network design proceeds like an iterative task on the basis of an evolution algorithm.

Key-Words: - Network optimization, Genetic algorithm, Physical topology, Logical topology, Resilient network, Redundant links, Graph algorithm.

1 Introduction
A design of the physical and the logical topology is more and more important nowadays, because every part of techniques needs some kind of a communication. Therefore, we have to design it very carefully with an aspect of the most important attribute of communication networks; a reliability of the communication. The design of the communication networks consists of the design of a physical topology and the design of a logical topology. The first of them says how we have to connect devices in the network via "wire" and the second says how data are sent in the network with the known physical topology. Our whole algorithm puts together the design of the both topologies and gains the reliable communication network with the lowest possible acquisition costs.

2 Problem Formulation
We can design the physical topology with different objects. The first object is the lowest possible costs of the whole network. The solution of this issue results in an application of a minimum spanning tree algorithm, because if the length of a network is the shortest, acquisition costs will be the lowest, as well. Another object is a reliability of the network. The often used solution is a design of the ring topology. This solution ensures if one communication link is interrupted so it will be possible to communicate in the network but if the network is interrupted at two different places, parts of the communication network will be disconnected. We introduce the solution that allows the design of the physical topology with the lowest acquisition costs with preservation of a possibility of a usage of more than only one redundant communication link.

3 Problem Solution
The input parameters for the design of the physical topology are always almost the same, although it is possible to use different way how design it. Therefore we describe the input parameters. However, we describe the input parameters for the design of the physical topology and parameters of the applied genetic algorithm in the following paragraphs.

The inputs into the first part of our algorithm are matrixes of the acquisition costs and the desired redundant links and a heap of inappropriate solutions. The matrix (the matrix of the acquisition costs) describes a price of the communication links between each pair of the nodes. If it is not possible to place the communication link between the nodes, a corresponding element in the matrix will be equal to infinity. At each position of the matrix of the desired redundant communication links is a number of the desired redundant communication links. If the redundant communication link is not needed, the corresponding position in the matrix will be zero. Inappropriate solutions are stored in the heap of the inappropriate solutions, which usage will be described later. Output from this part of the algorithm is an incidence matrix. The incidence matrix is also used for a representation of the network in a genetic algorithm that allows a design of the desired physical topology.

The inputs into the first part of our algorithm are matrixes of the acquisition costs and the desired redundant links and a heap of inappropriate solutions. The matrix (the matrix of the acquisition costs) describes a price of the communication links between each pair of the nodes. If it is not possible to place the communication link between the nodes, a corresponding

element in the matrix will be equal to infinity. At each position of the matrix of the desired redundant communication links is a number of the desired redundant communication links. If the redundant communication link is not needed, the corresponding position in the matrix will be zero. Inappropriate solutions are stored in the heap of the inappropriate solutions, which usage will be described later. Output from this part of the algorithm is an incidence matrix. The incidence matrix is also used for a representation of the network in a genetic algorithm that allows a design of the desired physical topology.

The incidence matrix is empty at start of the algorithm and it is initialized randomly. The representation does not describe the whole incidence matrix but only the upper triangular part. This part is big enough for us because the incidence matrix is symmetrical and we have to describe only the upper triangular part without diagonal elements. A creation of a chromosome that describes the physical topology is shown in the Fig. 1.

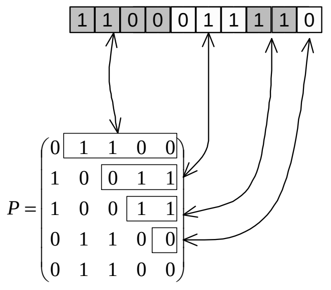


Fig 1. – Creation of chromosome

Then length of the chromosome is given by the equation (1).

$$l = \frac{N(N-1)}{2} \tag{1}$$

We can find space of possible solutions from the equation (1). It is given by:

$$S = 2^l \tag{2}$$

If we want to recognize in this huge space (2) the best solution, we have to evaluate every chromosome. We use for this purpose a value of the physical topology as a fitness function. We gain the price of the designed topology as a sum of all elements from the matrix Cp.

$$C_P = CP \tag{3}$$

The acquisition costs are not only one point of view. Other demands are the number of redundant communication links and network interconnections. Firstly, we check the interconnection of the network we use breadth-first search algorithm for this purpose. The incidence matrix is the input parameter for this algorithm. We have to control the number of redundant communication links as well. We apply for the verification the incidence matrix and the breadth-first search algorithm. The results of verification have an influence on evolution of chromosomes thanks to a penalty function. The final price of the physical topology is given by.

$$c = \sum_{i,j=1}^{N} c_{P,i,j} + Pen \tag{4}$$

Where N is a number of nodes and penalty is:

$$Pen = 1 + N^2 c_{MAX} \tag{5}$$

where c_{MAX} is a maximal element of the matrix of acquisition costs. An equation (5) ensures that chromosome, which is penalized, has bigger price than allowable chromosomes. The chromosomes are penalized if they do not correspond to the number of redundant communication links or if they are the same as the one of the former unsuitable solutions. We apply a genetic algorithm with one bit mutation and a tournament selection for the design of the physical topology. The speed of a population improvement is used like a stop rules.

3.1 Parameters of network

The design of the physical topology belongs to the NP hard problem as well. Therefore, every heuristics, which decreases the number of possible solutions, is very useful and one of them is applied in our algorithm. We will explain it at the following example.

We can see the physical topology in the Fig. 2.

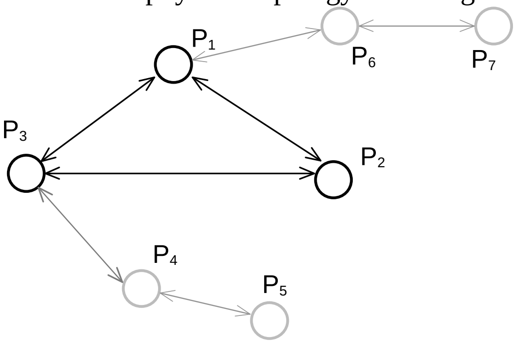


Fig. 2. – Physical topology

The physical topology in the figure 2 meets with demands for a redundant communication link between nodes P_1, P_2 and P_1, P_3. It is one of inputs into our algorithm the others are D - matrix of permitted delays of delivered frames and F - matrix of data-flows among nodes. These matrixes are symetrical positive matrixes and determine traffic and its attribute in the network exactly. Unfortunately, we need to compare data-flows and permitted time delays later and it is not possible to do it directly in those matrixes therefore we have to transform one of them into the another or vice versa.

We know that the data-flows have a poisson distribution and also service times have poisson distribution therefore we can apply a poisson distribution for a description of the network behaviour. An average time delay for the next sent bit for the poisson distribution is given by [10]:

$$D_p = \frac{n}{C_p - L_p} \qquad (6)$$

Where n is number of the communication links, C_p is data-link capacity and L_p is an amount of the data-flow. Naturally, the condition $C_p > L_p$ must be valid. The equation (6) is applied for a transfer from value of delay into value of maximal data-flow.

$$L_p = C_p - \frac{1}{D_p} \qquad (7)$$

We transform the matrix of permitted delay D into the matrix of maximal incoming data-flows $F^\bullet$. Then, we can transform the matrix $F^\bullet$ into a vector of maximal incoming data-flow into the node very easy.

$$\vec{f} = \begin{pmatrix} f_1^\bullet & \cdots & f_n^\bullet \end{pmatrix} \qquad (8)$$

where

$$f_i^\bullet = \min\left(F_{x,i}^\bullet\right) \text{ for } \forall i, x \in \langle 1, N \rangle \qquad (9)$$

3.2 Design of logical topology

It is possible to divide the physical topology in a few parts as we can see in the Fig. 2. First of them are the branches of the network. The logical topology in these branches is strictly given by the physical topology and it is not possible to change it without a modification of the physical topology. In this part are nodes P_4, P_5 and P_6, P_7.

All redundant communication links are in the second part (P_1, P_2, P_3). Nodes P_1, P_3 are also the first nodes of branches.

We have influence only on the second part with redundant communication links but firstly we have to confirm whether it is possible to delivered frames in branches up to the permitted delay. We can detect all branches very easily because the last node has the degree one. We find the whole branch by the following procedure:

1. Find the last node (node with the degree one).
2. Find all nodes that are connected to the last node.
3. If you have found out node with degree equal or bigger than three, it is the first node of the branch.

This procedure is repeating till finding of all branches. Then we have to verify the time delay of data-frames in branches.

1. Sum all input data-flows of the last node
2. If a result is smaller than permitted data-flow, continue with following steps. If a result is bigger, go back to the design of the physical topology and put a current physical topology in a heap of wrong physical topologies.

After successful check of last node, continue in verification of data-flows in the rest of branch.

1. Sum all input data-flows to the node (they correspond to column in the data-flows matrix); add result to all data-flows that go via this node.
2. If result is smaller than permitted data-flow, continue with the following nodes. If result is bigger, go back to the design of the physical topology and put current physical topology in the heap of wrong physical topologies.

This procedure decreases number of possible logical topology and therefore increases speed of finding of the logical topology because we focus only on part of the network, in that we are able to change delays of data-flows by changes of the logical topology. The new matrix of data-flows will be smaller but big enough for logical topology description. Also, the incidence matrix will be smaller; its dimension is the same as the dimension of the data-flows matrix. At the incidence matrix will be only lines and columns that correspond to nodes with redundant communication links.

Original data-flows for our network looks like this:

$$F = \begin{pmatrix} 0 & f_{1,2} & f_{1,3} & f_{1,4} & 0 & 0 & f_{1,7} \\ f_{2,1} & 0 & f_{2,3} & 0 & f_{2,5} & f_{2,6} & 0 \\ f_{3,1} & f_{3,2} & 0 & f_{3,4} & 0 & 0 & 0 \\ f_{4,1} & 0 & 0 & 0 & f_{4,5} & 0 & f_{4,7} \\ 0 & f_{5,2} & 0 & f_{5,4} & 0 & f_{5,6} & 0 \\ 0 & f_{6,2} & 0 & 0 & f_{6,5} & 0 & f_{6,7} \\ f_{7,1} & 0 & 0 & f_{7,4} & 0 & f_{7,6} & 0 \end{pmatrix} \qquad (10)$$

Matrix (10) is transformed into the new matrix of data-flows:

$$F_M = \begin{pmatrix} 0 & f_{M,1,2} & f_{M,1,3} \\ f_{M,2,1} & 0 & f_{M,2,3} \\ f_{M,3,1} & f_{M,3,2} & 0 \end{pmatrix} \qquad (11)$$

where elements of matrix of F_M are “symbolic” they comprise the following data-flows. It is the “initial” logical topology that could be changed by our algorithm if it is necessary.

$$\begin{aligned} f_{M,1,2} &= f_{1,2} + f_{6,2} \\ f_{M,1,3} &= f_{1,3} + f_{6,5} + f_{7,4} \\ f_{M,3,1} &= f_{3,1} + f_{4,1} + f_{4,7} + f_{5,6} \\ f_{M,3,2} &= f_{3,2} + f_{5,2} \end{aligned} \qquad (12)$$

We want to gain balanced load of the network we use for this purpose genetic algorithm. We expect smaller capacity of the communication links than a real communication capacity because of the redundant communication links. If one of the redundant communication links is destroyed, then the load of communication links is sent via other links. If the other communication links are fully utilised, data will be delivered too late therefore we can count only with smaller capacity that is given by:

$$k = 1 - \frac{1}{N} \qquad (13)$$

Where N is a number of the redundant communication links (main link is included) then the new capacity is given by:

$$C_{P,N} = k\,C_P \qquad (14)$$

3.2.1 Logical topology representation

The chromosomes for the logical topology description code every possible path for every data-flow. The chromosomes could be too long, but thanks to decrease of the space of possible solutions they are noticeable shorter than chromosomes that describe the whole logical topology. The Length of chromosome is given by:

$$l = nm \qquad (15)$$

Where n is number of the data-flows and m is the number of communication links in the part of the network with the redundant communication links. In our case, n=13 and m=3 and therefore the length of chromosome is l=39.

3.3 Genetic operators

We have to modify genetic operators for our chromosome configuration. The crossover function can be modified very easy; we know that chromosome is comprised by groups of bits that represent the logical path between two nods. Therefore, our one-point crossover function cross chromosome exactly at the borders of those groups. If chromosomes are crossed somewhere else, the resultant logical path could be impermissible. We can see the crossover function in the Fig. 3.

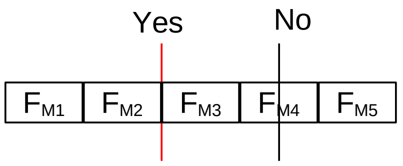


Fig. 3 -Crossover

We have to use our own mutation operator for our description of the logical topology. If we use common one bit mutation for our problem, the results gained by this operator will be completely wrong because they will represent an impermissible logical topology.

We show application of mutation at the following example for the data-flow $f_{1,2}$. Firstly, we have to create a list of every possible logical path for every data-flow. The Possible paths are represented by only two binary chains 001 and 110. The LSB (the Least Significant Bit) corresponds to the path between nodes P_1 and P_2. The MSB (the Most Significant Bit) corresponds to the path between nodes P_3 and P_1. The middle bit corresponds to the path between P_2 and P_3. If bit chain 001 is mutated, its value will be transformed into chain 110 and vice versa. We have to expect that there are more than two possible binary chains. In this case, the result of mutation is chosen randomly from these possible paths. The lists of the possible paths are also applied during the chromosomes initialization. Chromosomes are initialized by randomly chosen logical path for every data-flow.

3.3.1 Fitness function

Important part of genetic algorithm is a fitness function and we have to define it for our problems. We can calculate the delay in the network very easy because each chromosome exactly defines the whole logical topology for the redundant links and the rest of the logical topology (the logical topology in branches) is known from the past. We can use the average delay or the sum of the delays in every link as the fitness function. If average fitness of chromosomes decreases, then quality of the logical topology will increase. If the average fitness of chromosomes increases, then quality of the logical topology will decrease.

It is possible that average quality of chromosomes is satisfactory but delay in one path is bigger than permitted delay and therefore the whole logical topology is wrong. We should recognize some level of inappropriate designs and therefore we add a penalization to the fitness function. We need different penalization for each rank of inappropriate chromosomes according to the number of delays that are bigger than permitted delays.

$$Penalty = k \max(D) \qquad (16)$$

$$Fit = \sum d_i + Penalty \qquad (17)$$

Equation (17) is our fitness function; d_I is delay for i^{th} data-flow. Equation (16) describes the penalty function; k is the number of delays that are bigger than the permitted delay.

3.3.2 Selection and end condition

We use a tournament selection because it is efficient and not complicated. We know that the best chromosome has the smallest fitness (the smallest delay).

Our end condition is speed of improvement in last ten iteration of genetic algorithm. If improvement in last ten iteration is smaller than 0,5%, algorithm will be stopped.

3.4 Space of possible solution

It is not possible to say size of space without knowledge of the physical topology. We know only lower constraint of this space. We can expect that the lower constraint corresponds to demand for the ring topology (one redundant communication link) then space of possible solutions is given by:

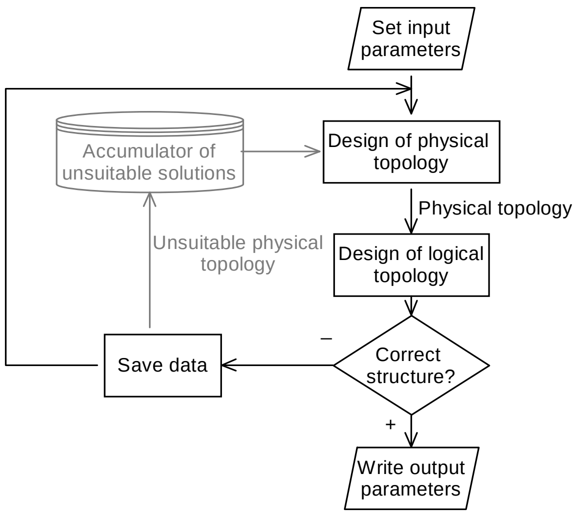
$$S_R = 2^m \qquad (18)$$

where m is the number of data-flow. Then we can expect that the size of the space possible solutions is given by

$$S \geq S_R \qquad (19)$$

3.5 Whole algorithm

As I wrote before the whole algorithm is an iterative task on basis of genetics algorithm. We can see structure of the whole algorithm in the figure 4.



Set input parameters
Accumulator of unsuitable solutions
Design of physical topology
Physical topology
Unsuitable physical topology
Design of logical topology
–
Save data
Correct structure?
+
Write output parameters

Figure. 4 – Whole algorithm

The first step of algorithm is settings of the input parameters; that means matrixes of data-flows, acquisition costs, permitted delays and so on. Then procedure continues with the design of the physical topology. Result of this algorithm part is the physical topology with small acquisition costs and desired redundant communication links. This physical topology is used as an input parameter for the design of the logical topology. The logical topology is verified if it meets with our requirements for attributes of the final network. If network corresponds to our demands, results are written to output and our task is finished. If not, the physical topology is stored to the accumulator of the unsuitable solutions and the next iteration of algorithm follows. It is necessary to set the number of iteration of the whole algorithm because it is possible that our demands are unreal and algorithm could iterate to the infinity.

4 Acknowledgment

This paper has been supported by Grant Agency of the Czech Republic, project number 102/05/0467 “Architectures of Embedded System Networks”

5 Conclusion

We have presented the solution of the topology design problem on the basis of genetic algorithm. This technique allows finding the optimal physical topology that includes the desired number of the redundant communication links. The physical topology is not achieved only by the optimization of acquisition costs but also by the verification of times in which data-frames are delivered.

This verification is ensured by the second step of our algorithm, in which we can gain the logical topology like the end product of the verification of the time of data-frames delivery.

Thanks to that we can say whether the resultant topology corresponds to our requirements for the number of redundant communication links and the average time delay of delivered data-frames.

Our future work at this topic is connected to the performance verification and the verification of the resultant topology. The performance of our algorithm will be compared with another method (branch and bound method). The resultant topology will be verified at a model of ProfiNet that will be implemented in Opnet Modeler.

Thanks to that, we will be able to say whether our expectations will be fulfilled.

References:

[1] J. Demel. *Grafy a jejich aplikace.* Praha: Academia, 2002. ISBN 80-200-0990-6

[2] J. Dinger, H. Hartenstein. On the challenge of assessing overlay topology adaptation mechanisms. Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing, 2005.

[3] A. Dutta, S. Mitra. Integrating Heuristic Knowledge and Optimization Models for Communication Network Design. IEEE Transactions of knowladge and data engineering, vol. 5, 1993, pp. 999-1017

[4] R. Elbaum, M. Sidi. Topological design of local area network using genetic algorithms. Proceedings of the Fourteenth Annual Joint Conference of the IEEE Computer and Communication Societies, 1995.

[5] J. Han, G. Robert Malan, F. Jahanian. Fault-Tolerant Virtual Private Networks within An Autonomous System. Proceedings of the 21 st IEEE Symposium on Reliable Distributed Systeme, 2002.

[6] C. Chekuri, A. Gupta, A. Kumar, J. Naor, D. Raz, „Building Edge-Failure Resilient Networks”, Algorithmica, vol. 43, 2005, pp. 17-41.

[7] King-Tim Ko, Kit-Sang Tang, Cheung-Yau Chan,

Kim-Fung Man, Sam Kong. Using Genetic Algorithms to Design Mesh Network. Computer, Vol. 30 , No 8, 1997, pp: 56 – 61.

[8] G. Kumar, N. Narang, C.P. Ravikumar. Efficient Algorithms for Delay-bounded Minimum Cost Path Problem in Communication Network. Proceedings of the Fifth International Conference on High Performance Computing. ISBN:0-8186-9194-8, 1998

[9] L. Layuan , L. Chunlin. QoS Multicast Routing in Networks with Uncertain Parameter. Proceedings of the International Parallel and Distributed Processing Symposium. 2003

[10] Z. Liu, A. Jaekel, S. Bandyopadhyay. A genetic algorithm for optimization of logical topologies in optical network. Proceedings of the International Parallel and Distributed Processing Symposium, 2002.

[11] V. Mařík, O. Štěpánková, J. Lažanský a kolektiv. *Umělá inteligence 3*.
Praha: Academia, 2001. ISBN 80-200-0472-6

[12] V. Mařík, O. Štěpánková, Jiří Lažanský a kolektiv. *Umělá inteligence 4*. Praha: Academia, 2003. ISBN 80-200-1044-0

[13] M.A. Sridhar , C. S. Raghavendra. Fault-Tolerant Networks Based on the de Bruijn Graph. IEEE TRANSACTIONS ON COMPUTERS, VOL. 40, NO. 10, 1991

[14] E. Szlachcic. Fault tolerant topological design for computer network. Proceedings of the International Conference on Dependability of Computer Systemes, 2006

[15] D. R. Thompson, G. L. Bilbro. Comparison of Two Swap Heuristics with a Genetic Algorithm for the Design of an ATM Network. Proceedings of the International Conference on Computer Communications and Networks, 1998

[16] Marek Obitko Genetic algorithms [online] http://cs.felk.cvut.cz/~xobitko/ga/ [cite 2007-04-04]