Using Simulated Annealing to Improve Reliability of Grid Computing Systems

Ehsan Gholami
Islamic Azad University Shoushtar Branch, Shoushtar, Iran
eh.gholami@gmail.com

Amir Masoud Rahmani
Islamic Azad University Science and Research Branch, Tehran, Iran
rahmani@srbiau.ac.ir

Reza Farshidi
Islamic Azad University Dezful Branch, Dezful, Iran
farshidi@iaud.ac.ir

Abstract—**Grid computing system has been suggested as an effective technology in distributed resource coupling for applications that require large-scale resource sharing, where processors and communication have significantly influence on grid computing system reliability. A grid computing system contains a series of machines, resources, programs and communication network. Each machine includes a series of resources and programs. In designing a grid computing system, a topology is specified by distributing the resources and programs in grid computing system machines and defining communication network between the machines, which have significant effect on grid computing system reliability. Finding such a topology that guarantees the maximum reliability is difficult and considered as an NP-Hard problem. In this work, a novel method is proposed for designing and improving the grid computing system reliability based on Simulated Annealing. The proposed method has this ability to obtain a new topology with more reliability in comparison to the initial given grid computing system.**

Keywords-topology; reliability; Grid computing system; simulated annealing.

I. INTRODUCTION

In the recent years, Grid Computing Systems (GCS) have been created as a way to increase the capacity of computing resources like processing and memory resources by integrating them and use computing resources, which are distributed geographically [1]. In GCS, the integration of shared computing resources, e.g., computer, software, data, and peripheral equipments are achieved through a communication network [1][2].

GCS can be built in various sizes, ranging from just a few machines in a department to groups of machines organized as a hierarchy spanning the world. Machines that participate in the GCS may include systems from multiple departments with the same organization. Such a grid is also referred as an intragrid [1]. In this paper, we focus on reliability of this type of grid and propose a new algorithm to design and improve the reliability of intragrid that guarantees the maximum reliability.

A GCS contains a series of machines, resources, programs and a communication networks. Each machine in GCSs can be a computer, data storage device or other devices. Every machine includes a series of programs and resources such as software, data, etc. These programs can start their execution from this machine. Machines are connected via the communication network. In a GCS, each program completes its own execution by connecting to some grid system machines via communication network and exchanging the information with their resources.

The reliability of a program on a GCS is the probability that one program may exchange information with other machines successfully, and completes its own execution. Likewise, the reliability of a grid computing system is the probability that all programs on a grid computing system are executed successfully [2][3]. This reliability can be calculated by determining the reliability of each component using Monte Carlo [14] or any other method such as Grid system reliability [2] and Markov model [3]. Therefore, the way that programs access to required resources in remote machines and also the circumstances of distributing resources and programs between machines have significant effect on GCS reliability. To define the GCS topology, communication network between machines and distributing resources and programs in GCS machines must be defined in a way that guarantees maximum reliability against failures in GCS components [2][14]. Many different topologies can be defined for a GCS with specific number machines, programs and resources. Finding the best GCS topology from the viewpoint of reliability between these topologies is an NP-Hard problem. Since there is not any standard algorithm, which guarantees an optimized reliability of GCS, in this paper, Simulated Annealing as a meta-heuristic method is presented for designing GCS topology and improving reliability of GCS. Furthermore, other methods such as Genetic algorithms do not offer any guarantee of global convergence to an optimal point to solve such problems [6]. But the proposed method has better results for designing GCS topology in shorter run time.

The rest of this paper is organized as below. Section II describes the simulated annealing. Section III defines the designing of GCS topology. Sections IV, V and VI develop the algorithm for improving reliability

 ISBN: 978-1-61208-101-4

of GCS. In Section VII, experimental results are illustrated. Section VIII concludes the paper.

II. SIMULATED ANNEALING

Simulated Annealing (SA) is one of the probabilistic algorithms presented by Kirkpatrick in 1987 to solve optimizing problems with large search span of solutions [10]. Basically, SA is a random-search technique, which exploits an analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process) and the search for a minimum in a more general system; it forms the basis of an optimization technique for combinatorial problems [16]. Search span of a solution for a problem is called *S* and each solution in this search span is called *s,* where $s \in S$. The SA algorithm begins from an initial solution s_0 as current solution and gradually obtains the optimized solution by passing from a solution to another solution in search span and choosing a neighbor solution of current solution. The solution such as s_1 from search span is chosen as neighbor solution of the current solution. Current solution is replaced with neighbor solution or remains unchanged by a probability. This process iterates until finding an optimized solution or reaching to maximum number of iteration in SA.

In this algorithm, a better solution for the problem has a better cost. Accepting the neighbor solution as current solution in next iteration is based on a strategy. According to this strategy if the cost of neighbor solution is better than current solution it will be accepted as the current solution in next algorithm iteration and if the cost of neighbor solution is worse, it will be accepted by a probability, which will be mentioned later. Therefore, by this manner SA can leave local optimized solution. In SA, temperature parameter *T* is defined to obtain probability of acceptance of neighbor solution. When the algorithm starts, initial value of parameter *T* is defined such that most neighbor solutions in algorithm iterations are accepted as alternative for current solution. Parameter *T* is decreased gradually by iterations of SA algorithm such that it reaches to zero before the number of iterations of algorithm reaches to last (maximum) iteration. Therefore, by increasing the number of iterations, parameter *T* is decreased and the probability of accepting neighbor solution is decreased. So, in last iteration of algorithm, only the neighbor solution which has better cost than current solution is replaced as current solution in next algorithm iteration. In this algorithm, decreasing the parameter *T* in each iteration, obtaining neighbor solution, determining initial temperature T_0 and maximum number of iterations of algorithm *(MAX)* have significant effect on obtaining best or optimized solution. SA pseudo-code is shown in Fig. 1.

III. PROBLEM DEFINITION

In this paper, GCS topology is designed in two separate steps: first step is distributing the GCS resources and programs between GCS machines. The second step is defining a communication network for these machines. To design such a GCS, one of the problems is the definition of its topology to guarantee a maximum reliability for the GCS. In this work, we tackle the following version of this problem: given an user-defined network topology, find a more reliable alternative GCS in real system. Therefore, reaching to a GCS topology with high reliability is possible by changing the results of each step or both of them and combining them to obtain a topology with a more reliability. Searching for such a topology is an NP-hard problem, because there are many different topologies during implementation of first and second steps. Solving these problems with conventional algorithms is so difficult and time consuming. In this work, an intra GCS is assumed as a graph *G(V, E)* with *n* nodes $V=\{v_1,v_2, \ldots ,v_n\}$ and *m* edges $E=\{e_1,e_2, \ldots ,e_m\}$, where GCS machines are its nodes and communication network links are its edges.

```
t = T0
Initial solution = S0
Best solution = S0
OF0 = Objective fumction(S0)
OF_Best solution = OF0
for i = 1 to Max do
 { S1 = Generate neighbor(S0)
   OF1 = Objective function(S1)
   If OF1 ≥ OF0 then
     { S0 = S1
       OF0 = OF1
       If OF1 ≥ OF_Best solution then
          { Best solution = S1
            OF_Best solution = OF1 }
     }
   else { Accept = e^(-((OF0 - OF1)/t))
          r = Random(0,1)
          if r > Accept then
          { S0 = S1
            OF0 = OF1 }
   }
  t = decrease(t)
 } // Endfor
Return (Best solution)
```

Figure 1. SA Algorithm pseudo-code

Reliability of each machine depends on number of its resources and programs and reliability of each link depends on programs and machines that transfer their data through this link [2][14]. Also, in this work reliability of GCS is calculated based on Monte Carlo simulation according to the reliability of machines and links [14].

According to SA, algorithm must begin from an initial solution. In this work, the SA algorithm begins from initial topology as initial solution and improves the reliability of GCS, too. The initial topology can be defined by user or randomly. Implementing SA algorithm to find a GCS topology and improving its reliability include some parts, which are considerable for obtaining best solution. These parts are mentioned in next sections.

IV. NEIGHBORHOOD STRUCTURE

In designing GCS, resources and programs in GCS machines are distributed by the designer before implementing and creating GCS, such that the designer denotes how the software, data storage devices and other devices are distributed in GCS machines and also denotes machines, from which programs start their execution. Thus, the proposed method can be used in designing real intragrid system to achieve at more reliability. Obtaining a neighbor topology from current topology of GCS has significant effect on faster and better convergence of SA to global optimum topology from the viewpoint of GCS reliability. In this work, two methods are proposed to obtain a neighbor topology from current topology in GCS:

A. Obtain Neighbor Topology by Exchanging the Resources and Programs of Machines

In this method, at first, two machines are selected randomly from current topology of GCS. The neighbor topology can be obtained by changing all resources and programs or part of them in these two selected machines of current topology. Note that the resources and programs of other machines remain unchangeable. For example if *i, j* machines are selected in current topology the neighbor topology is obtained by exchanging resources and programs of these two machines. It has been shown in Fig. 2. The reason of obtaining neighbor in such manner is that in GCS, distributing resources and programs has direct effect on machine reliability and amount of data exchanging between machines, which consequently have affects on reliability of machines and GCS [2][3][11]. Therefore, exchanging the resources and programs might decrease the workload and amount of data exchanging, which consequently increases reliability of GCS.

Node j	
Program	P_3, P_4
Resource	R_1, R_2, R_5
Neighbors	$N1, N2, N5$

Node i	
Program	P_1, P_2
Resource	R_1, R_3, R_4
Neighbors	$N3, N4$

(a)

Node j	
Program	P_1, P_2
Resource	R_1, R_3, R_4
Neighbors	$N1, N2, N5$

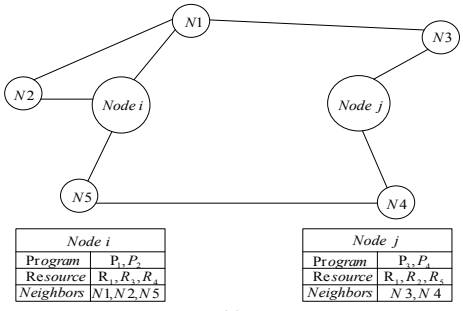
Node i	
Program	P_3, P_4
Resource	R_1, R_2, R_5
Neighbors	$N3, N4$

(b)

Figure 2. Neighbor topology: (a) Machines specifications in current topology, (b) Machines specification in neighbor topology.

B. Obtain Neighbor Topology by Exchanging Links

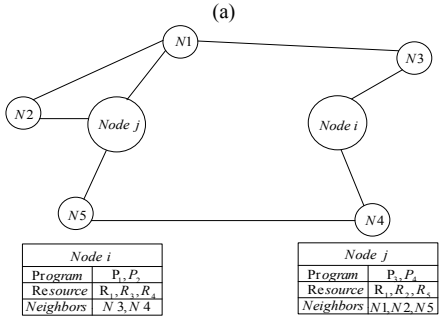
In this method, similarly to the previous method, two machines are selected from current topology of GCS. Neighbor topology is a topology in which resources and programs of machines remain immutable and all or some of connected links of two selected machines are exchanged. Also, it is controlled to be just one direct link between two machines and each machine has not any link to itself (loop). For example if *i, j* machines are selected for exchanging the data, the resulted neighbor topology is as shown in Fig. 3. The reason of obtaining neighbor topology in this manner is that GCS programs access to required resources through links in communication network [3][12]. Therefore, it is possible to decrease the amount of data exchanging through links that consequently can increase the reliability of the links and GCS.



Node i	
Program	P_1, P_2
Resource	R_1, R_3, R_4
Neighbors	$N1, N2, N5$

Node j	
Program	P_3, P_4
Resource	R_1, R_2, R_5
Neighbors	$N3, N4$

(a)



Node i	
Program	P_1, P_2
Resource	R_1, R_3, R_4
Neighbors	$N3, N4$

Node j	
Program	P_3, P_4
Resource	R_1, R_2, R_5
Neighbors	$N1, N2, N5$

(b)

Figure 3. Neighbor topology: (a) Communication network in current topology, (b) communication network in neighbor topology.

In spite of original SA algorithms, which obtain a neighborhood structure as a fix structure [8][9][14], a different strategy is used in proposed method. In the proposed SA algorithm, in each step the number of programs and resources or links that must be exchanged are variable to obtain neighbor topology. It means that in the first step all resources and programs or links between two selected machines are exchanged and by approaching to the maximum number of SA iterations, in both A and B methods the number of exchanges in neighborhood structure is decreased such that in last iteration of algorithm the number of exchanges from current topology to neighbor topology is restricted to just one exchange. Therefore, with this strategy the speed of convergence to the global optimum topology is increased. In the proposed algorithm the percentage of exchanges in k-th iteration is calculated in (1).

$$(Number\ of\ changes)\% = \frac{T_k}{T_0} \tag{1}$$

where T_0, T_k are initial temperature and temperature of k-th iteration respectively. According to this equation the percentage of exchanges is decreased as k is increased.

V. ACCEPTANCE RULE OF NEIGHBOR TOPOLOGY

After obtaining a neighbor topology its reliability is calculated. If reliability of neighbor topology is better than current topology, it will be selected as the new current topology. But if this reliability is worse than reliability of current topology, it will be selected as new current topology according to a probability. This probability is calculated in (2).

$$P(Accept) = \begin{cases} 1 & if \quad \Delta \le 0 \\ e^{-\Delta / T_k} & if \quad \Delta > 0 \end{cases} \tag{2}$$

where Δ is equal to the difference between reliability of current and neighbor topology and T_k is stand for temperature parameter in k-th iteration. Since the temperature parameter T_k in SA algorithm is decreased as iteration increases, so according to (2), decreasing T_k results in the reduction of acceptance probability.

Therefore, in this rule when the reliability of neighbor topology is worse than the current topology a number is selected between zero and one randomly that is named r. If r is more than acceptance probability, the neighbor topology is selected as new current topology.

VI. SCHEDULING RULE TO DECREASE THE TEMPERATURE PARAMETER AND INITIALING THE PARAMETER

In SA algorithm, scheduling the reduction of temperature parameter during algorithm iteration has significant effect on the way the algorithm operates. Scheduling rule for decreasing the temperature parameter T must be in a manner that reaches to zero before the last iteration of SA algorithm. In this paper, the applied method to decrease the temperature is based on geometrical progression, which is defined as (3).

$$T_{k+1} = \alpha T_k \tag{3}$$

where α=095. Reduction of temperature parameter T as (3) is used in SA algorithm usually. In (3), α accepts different values. In our work, the different values for α are selected experimentally and shows this reality that $\alpha = 0.95$ results in better solution. In fact, α determines the speed of reduction for temperature parameter T.

```
t = T0
Initial topology = S0
Best topogy = S0
GR0 = reliability (S0)
GRBest topology = GR0
for i = 1 to Max do
{   NC = t / T0
    S1 = Generate neighbor (S0, NC)
    GR1 = reliabilty (S1)
    If GR1 ≥ GR0 then
    {   S0 = S1
        GR0 = GR1
        If GR1 ≥ GRBest topology then
        { Best topology = S1
          GRBest topology = GR1 }
    }
else { Accept = e^-((GR0 - GR1)/t)
       r = Random (0,1)
       if r > Accept then
       { S0 = S1
         GR0 = GR1 }
    }
    t = 0.95 * t
} //for
Return (Best topology)
```

Figure 4. SA pseudo-code for improve GCS reliability

TABLE I. Program Specifications

Program	Necessary Resources(Exchanged information (KB))
P1	R3(100),R4(300),R6(200),R7(400)
P2	R1(250),R2(200),R5(300)
P3	R2(100),R3(100),R4(200),R5(300),R7(300)
P4	R1(200),R2(100)

The parameters, iteration number (*max*) and initial temperature (T_0) are initialized by trial and error method. The value of *max* must be chosen in such manner that finds the global optimized topology with high probability before reaching to maximum number of iterations. Also, selection of parameter T_0 is in such a way that reaches to zero before reaching to the maximum number of iterations. Proposed algorithm for improving reliability of GCS has been shown as pseudo-code in Fig. 4 according to above explanations.

VII. EXPERIMENTAL RESULTS

In this section, the proposed algorithm is run to design a sample GCS with below characteristics and improve its reliability. To simulate the algorithm GridSim software is used. In this simulation initial GCS consisted of six machines, seven types of resources and four different programs. The failure rate and data transfer rate of each link of communication network are assumed to be 0.003 failures per second and 150 kbps respectively. The specifications of programs, which include the types of required resources and amount of data that must be exchanged to them, are assumed as shown in Table I. The corresponding graph of the initial GCS has been shown in Fig. 5. The specifications of machines in initial GCS that contain resources, programs and failure rate have been shown in Table II. The proposed algorithm was run with 0.5 and 100 for T_0 and *max* respectively. The GCS topology resulted from executing the proposed algorithm by using method B to obtain neighbor topology is shown in Fig. 6. (in this method specification of machines are immutable). Table III illustrates the specifications of obtained GCS topology using method A to obtain neighbor topology (in this method communication network is immutable).

TABLE II. Machines Specification of Initial GCS

Machine	Resource	Programs	Failure rate (Failure / sec)
M1	R3,R4	P1,P2	0.001
M2	R1,R2,R5	P1	0.002
M3	R2,R7	-	0.003
M4	R2,R4,R7	P4	0.004
M5	R3,R4	P2,P3	0.005
M6	R1,R2,R5	P3,P4	0.006

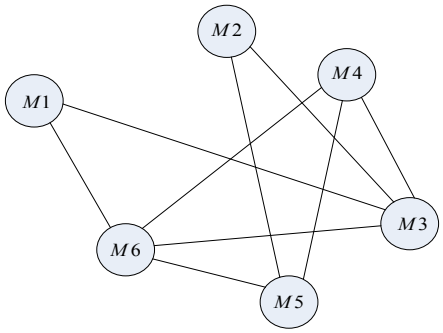


Figure 5. Initial GCS Communication Network

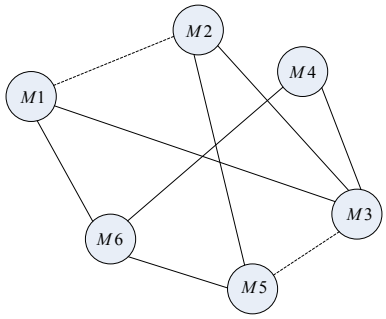


Figure 6. Resulted GCS topology by neighborhood structure method B

According to Fig. 6, in resulted GCS topology which uses method B to obtain neighbor topology, the communication links that are shown with dotted lines have been changed rather than original communication network in Fig. 5. According to Table III, in resulted GCS topology, which applies method A to obtain neighbor topology, the resources or programs that are shown with gray color have been changed rather than original specification of machines in Table II.

In this paper, the method proposed in [14] was used to estimate the reliability of GCS. Therefore, the reliability of initial GCS topology with specification of machines in Table II and communication network in Fig. 5 is 0.91324. The reliability of resulted GCS topology from executing proposed algorithm and methods A and B to obtain neighbor topology is 0.94347 and 0.96512. According to results, the reliability of obtained GCS by using proposed algorithm and neighborhood structures A or B is improved rather than initial GCS.

TABLE III. Machines Specification of Resulted GCS Topology by Neighborhood Structure Method A

Machine	Resource	Programs	Failure rate (Failure / sec)
M1	R3,R5	P1,P2	0.001
M2	R2,R5,R7	P1	0.002
M3	R2,R4	-	0.003
M4	R2,R5,R7	P3	0.004
M5	R2,R4	P2,P4	0.005
M6	R1,R2	P3,P4	0.006

 ISBN: 978-1-61208-101-4

TABLE IV. Results of SA by Different Max Iterations

Max-Iteration	1	2	3	4	5	Best	Average
1000	0.94061	0.93832	0.93154	0.94928	0.94867	0.94928	0.94166
10000	0.94402	0.96653	0.95436	0.97624	0.95931	0.97624	0.96009

It is clear from the obtained results that the proposed algorithm improves GCS reliability by using each one of neighborhood structure.

Since choosing initial values of parameters *max* iterations and T_0 is important in proposed algorithm; so to better evaluation of proposed algorithm, it was run with two different values of *max* iterations for five times. The results of reliability of obtained GCS topology are shown in Table IV. According to the results of Table IV, in each execution of proposed algorithm the resulted reliability of obtained GCS topology has been improved. Therefore, the percentage of reliability improvement for max=1000 is 2 to 4 percent and 3.5 to 7 percent for max= 10000, which shows this reality that changing the initial parameters can change the percentage of reliability improvement. Also it is clear from the obtained results that the proposed algorithm is convergent to a GCS topology with better reliability but improved percentage is different.

VIII. CONCLUSION AND FUTURE WORK

Since designing and obtaining the reliability of a GCS is an NP-Hard problem, so in this work SA has been applied to obtain a GCS topology and improve its reliability. The obtained results of running the proposed algorithm show that SA has this capability to be applied for designing a GCS topology and improving its reliability. It should be noted that, proposed algorithm can be improved efficiently by exploiting other techniques for reduction of parameter T or percentage of exchanges in obtained neighbor structure. This algorithm also can be used to obtain a GCS topology, which considers other factors like decreasing the congestion, rapid performing of programs, etc. Using combination of two neighbor structures together in proposed algorithm may improve proposed algorithm to represent better performance. This algorithm is used as a tool for initial designing of GCS to improve its reliability. In future work, we will focus on applying this method for real GCS in which the topology has been designed before. So, in order to improve the reliability and performance of GCS, the proposed method is used wherein resources are changeable such as databases or software.

REFERENCES

[1] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," International Journal of High Performance Computing Applications, vol. 15, pp. 200-222, 2001.

[2] Y. S. Dai, M. Xie, and K. L. Poh, "Reliability analysis of grid computing systems," Proc. IEEE Pacific Rim Int'l Symp. Dependable Computing (PRDC'02), pp. 97–104, 2002.

[3] Y. S. Dai, M. Xie, and K. L. Poh, "Markov renewal models for correlated software failures of multiple types," IEEE Trans. Rel., vol. 54, no. 1, pp. 100–106, 2005.

[4] M. Daoud and Q. H. Mahmoud, "Monte Carlo simulation-based algorithms for estimating the reliability of mobile agent-based systems," Journal of Network and Computer Applications, pp. 19–31, 2008.

[5] G. S. Fishman, "A comparison of four Monte Carlo method for estimating the probability of set conectedness," IEEE Transactions on Reliabilty, vol. R-35, pp. 145-155, 1986.

[6] F. Altiparkmak, B. Dengiz, and A. E. Smith, "Reliability optimization of computer communication networks using genetic algorithms," IEEE international conference on systems. San Diego, California, USA: Man and Cybernetics, pp. 4676–81, Oct. 1998.

[7] C. C. Hsieh and Y. C. Hsieh, "Reliability and cost optimization in distributed computing systems," Computer Operation Research, vol. 30, pp. 1103–19, 2003.

[8] S. Anily and A. Federgrune, "Simulated annealing methods with general acceptance probabilities," Journal of Applied Probability, vol. 24, pp. 657-667, 1987.

[9] D. Connoly, "General purpose simulated annealing," European Journal of Operational Research, vol. 43, pp. 495-505, 1992.

[10] S. Kirkpatric, C. D. Gellat Jr, and M.P. Vecchi, " Optimization by simulated annealing," Science, vol. 220, pp. 671-681, 1987.

[11] A. Kumar, S. Rai, and D. P. Agarwal, "On computer communication network reliability under program execution constraints," IEEE Journal of Selected Area in Communication, vol. 6, pp. 1393-1400, Oct. 1988.

[12] D. J. Chen and T. H. Huang, "Reliability analysis of distributed systems based on a fast reliability algorithm," IEEE Transaction on Parallel and Distributed System, vol. 3, pp. 139-154, March 1992.

[13] B. Shirazi, M. Wang, and G. Pathak, "Analysis and Evaluation of Heuristic Methods for Static Task Scheduling," Journal of Parallel and Distributed Computing, vol. 10, pp. 222-232, 1990.

[14] E. Gholami, A. M. Rahmani, and A. Habibizad Navin, "Using Monte Carlo Simulation in Grid Computing Systems for Reliability Estimation," The Eight International Conference on Networks, Cancun, Mexico, pp. 380-384, March 2009.

[15] L. Ingber, "Simulated annealing: practice versus theory," Mathl. Comput. Modelling, vol. 18, no. 11, pp. 29-57, 1993.

[16] H. R Anderson and J. P. McGeehan, "Optimizing microcell base station locations using simulated annealing techniques," in 44th IEEE Vehicular Technology Conference, pp. 858–862, IEEE, 1994.

[17] S. Z. Selim and K. Alsultan, "A Simulated Annealing Algorithm for the Clustering Problem," Pattern Recognition, vol. 24, no. 10, pp. 1003-1008, 1991.

[18] M. Gerla and L. Kleinrock, "On the topological design of Distributed Computer Networks," IEEE Transactions on Communications, vol. 25, no. 1, pp. 55-67, 1977.

 ISBN: 978-1-61208-101-4