Selecting the Best Spanning Tree in Metro Ethernet Networks using Genetic Algorithm

Farhad Faghani[†] **and Ghasem Mirjalily**[††],
faghani_farhad@yahoo.com mirjalily@yazduni.ac.ir
[†] Instructor, Electrical Engeering Department, Islamic Azad University (Najafabad Branch), Isfahan, Iran .
[††] Assistant Professor, Electrical and Computer Engineering Department, Yazd University, Yazd, Iran.

Summary

IEEE Spanning Tree Protocol (STP) is a layer-2 protocol which ensures a loop free topology in Metro Ethernet networks. It is based on Minimum Spanning Tree solution that involves determining the links which can join all the nodes of a network together such that the sum of the costs of the chosen links is minimized. In STP, all customers need to use the same spanning tree and there isn't any traffic engineering mechanism for load balancing. This results in uneven load distribution and bottlenecks, especially close to the root. A solution for this problem is using the multi-criteria Minimum Spanning Tree by considering criterions such as load balance over links and switches. In our previous work, the algorithm was based on computation of the total cost for each possible spanning tree and then selection the best one with minimum total cost. This algorithm is very time consuming, especially when our Metro Ethernet network is large. In this paper, we propose a new approach using Genetic Algorithm. It reduces the computational complexity by selecting the best spanning tree in a stochastic manner.

Key words:
Genetic Algorithm, Minimum Spanning Tree, Metro Ethernet Network, load balance.

1. Introduction

Ethernet technology has so far been widely accepted in enterprise deployments and it is said that today 90% of data traffic is Ethernet encapsulated. The significant advancement of Ethernet technology is pushing Ethernet from the local area network environment to metropolitan and wide area network environments [1,2]. One of the most important protocols of Metro Ethernet Networks is Spanning Tree Protocol (STP). It is a link management protocol that provides path redundancy while preventing undesirable loops in the network. For an Ethernet network to function properly, only one active path can exist between two nodes. Multiple active paths between nodes cause loops in the network. If a loop exists in the network topology, it confuses the forwarding algorithm and allows duplicate frames to be forwarded.

To provide path redundancy, STP defines a tree that spans all switches in an extended network. STP forces certain redundant data paths into a standby (blocked) state. If one network segment in the STP becomes unreachable, or if STP costs change, the spanning tree reconfigures the spanning tree topology and reestablishes the link by activating the standby path.

Current Metro Ethernet Protocols relies on IEEE STP or IEEE Rapid Spanning Tree Protocol (RSTP). In both STP and RSTP, all customers need to use the same spanning tree and there isn't any traffic engineering mechanism for load balancing. This results in uneven load distribution and bottlenecks, especially close to the root, and this leads to inefficient utilization of expensive fiber links in metropolitan area networks.

Traffic engineering of Ethernet using spanning tree is a widely researched topic because of performance issues. Several enhancements have been proposed in the literature in order to solve this problem by mitigating congestion near the root. Some previous related works are reviewed in[3].

In our previous work [3], we introduced a graph theoretic approach named Best Spanning Tree (BST) algorithm for ranking all of the possible spanning trees in a Metro Ethernet network and for finding the best tree. It was designed to be backward-compatible, as it relies on the IEEE STP protocol.

Although, BST algorithm can find the best answer for small networks, but its complexity is too large in great networks. To solve its huge complexity, in this paper we proposed a new approach using Genetic Algorithm.

This paper is organized as follows: Section II gives some definitions and notations. The proposed algorithm is explained in Section III. In Section 4, some simulation

Manuscript received June 5, 2008
Manuscript revised June 20, 2008

results are presented and some conclusions are drawn in the section 5.

2. Some Definitions and Notations

In this paper, we want to select the best spanning tree for a given Metro Ethernet network not only based on shortest path criterion but also based on load balancing on links and switches. So, we define three major criterions: load balancing over links, load balancing on switches and shortest path selection. We can define three coefficients *α, β,* and *γ* corresponding to above criterions respectively. This allows us to weight the importance of each criterion based on our goal.

We model a metro Ethernet network by a directed graph *G(V;E)*, where *V* is a set of *N* nodes (vertices) representing switches and *E* is a set of M Links(edges) connecting nodes. Assume $b_k(k=1,2,\cdots,M)$ denotes the bandwidth of k^{th} link in Mbps, $C_i(i=1,2,\cdots,N)$ denotes the switching capacity of i^{th} node (switch) in Gbps and $d_{ij}(i,j=1,2,\cdots,N \text{ and } i\neq j)$ denote the traffic demands, that is, the average rate of customer traffic between nodes *i* and *j* in Mbps. Furthermore, α, β, γ $(0\leq\alpha, \beta, \gamma \leq 1, \ \alpha+\beta+\gamma=1)$ are user defined parameters indicate the importance of the criterions: Link Load Balance(LLB), Switch Load Balance(SLB), and Shortest Path Selection(SPS). For each tree, after distributing traffic demands, we can define l_k $(k=1,2,\cdots,M)$ as the traffic load on the k^{th} link in Mbps and s_i $(i=1,2,\cdots,N)$ as the traffic load cross i^{th} switch in Gbps. In this work we assume symmetric full duplex links and demands.

The parameters used in our GA approach to define evaluation function are defined below. These parameters are related to defined criterions (LLB, SLB and SPS).

For LLB criterion, we define the variance of normalized link loads (σ_l^2) as:

$$\sigma_l^2 = \frac{1}{N-1}\sum_{k=1}^{N-1}\left(\frac{l_k}{b_k} - \bar{l}\right)^2 \qquad (1)$$

Where:

$$\bar{l} = \frac{1}{N-1}\sum_{k=1}^{N-1}\frac{l_k}{b_k} \qquad (2)$$

is the average of normalized link loads. For each spanning tree σ_l^2 is a useful parameter that indicates the degree of switch load balancing. In LLB criterion, the goal is to find a spanning tree with minimum σ_l^2.

For SLB criterion, we define the variance of normalized switch loads (σ_s^2) as:

$$\sigma_s^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{s_i}{c_i} - \bar{s}\right)^2 \qquad (3)$$

Where:

$$\bar{s} = \frac{1}{N}\sum_{i=1}^{N}\frac{s_i}{c_i} \qquad (4)$$

is the average of normalized switch loads. For each spanning tree, σ_s^2 is a useful parameter that indicates the degree of switch load balancing. In SLB criterion the goal is to find a spanning tree with minimum σ_s^2.

In SPS criterion, the goal is to find a spanning tree with maximum bandwidth and minimum hop count paths. This is equal to select a tree with minimum *L* defined as:

$$L = \frac{\sum_{k=1}^{N-1} l_k}{\sum_{k=1}^{N-1} b_k} \qquad (5)$$

In large networks, it is difficult to find the best spanning tree by evaluating all possible spanning trees. Therefore, in next section we propose a new Genetic Algorithm approach to solve this problem.

3. Genetic Algorithm Approach

Genetic Algorithm (GA) is a search technique used to find the approximate solution. Moreover, it improves all potential solutions step by step through biological evolutionary processes like crossover, mutation, etc. Because the process in the GA approach is not wholly operated randomly but includes both directed and stochastic search embedded with a survival of the fittest mechanism, it is possible to enforce the search to reach the optimal solution.

In [4], a genetic algorithm is proposed for degree-constrained Minimum Spanning Tree problem. There, a novel model that transfers degree-constrained Minimum Spanning Tree problem into a preference two objective Minimum Spanning Tree problem is presented.

In our paper, we use this main idea to develop a new genetic algorithm for selecting the best spanning tree in Metro Ethernet networks based on load balance criterion.

Before running a GA, it is necessary to define:

- The individual on which it operates (encoding),
- The operators it uses,
- Some parameters such as the population size, etc.,
- An objective function.

3.1 Chromosome Representation

For GA approach, it is important to figure out the adequate chromosome representation of problem. One of the classical theorems in graphical enumeration is Cayleys's theorem [5]. It states that there are $N^{(N-2)}$ distinct labeled trees for a complete graph with *N* vertices. Prüfer provided a constructive proof of Cayley's theorem establishing a one-to-one correspondence between such trees and the set of all permutation of *N-2* digits. This means we can describe our tree with *N-2* uniquely digits for *N* vertices. We name it Prüfer Number. You can see a simple tree and its Prüfer number in figure (1). We use Prüfer number as a chromosome in GA approach. The Prüfer number encoding procedure is as below:

Step 1) Let *i* be the smallest leaf node and node *j* be incident to node *i*. Set *j* be the first digit in the encoding. The encoding is built by appending digits to the right.

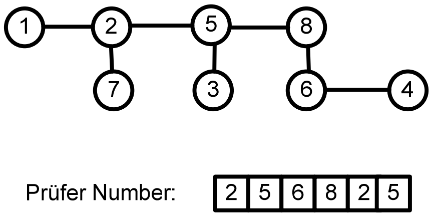
Step 2) Remove node *i* and the edge from *i* to *j*.

Step 3) Repeat above operation until only one edge is left. In a Prüfer number encoding, a tree is encoding as a Prüfer vector *P* and a set of its eligible nodes $\bar{P}$ (the set of all node not included in *P*).

The decoding procedure is as below:

Step 1) Let node *i* be the smallest eligible node of $\bar{P}$ and node *j* be the leftmost element of P. If $i \neq j$, add the edge (*i,j*) into the tree T. If *i* is no longer eligible, then remove node *i* from $\bar{P}$. Delete *j* from *P*. If *j* does not occur anywhere in the remaining part of *P*, then put it into $\bar{P}$. Repeat the process until *P* is empty.

Step 2) For the remaining last two nodes *u* and *v* of $\bar{P}$, add the edge (*u,v*) into the tree T.



Prüfer Number: 2 5 6 8 2 5

Fig.1: A tree and corresponding Prüfer Number.

Genotypes (chromosome values) are uniquely mapped on to the decision variables (phenotypic) domain.

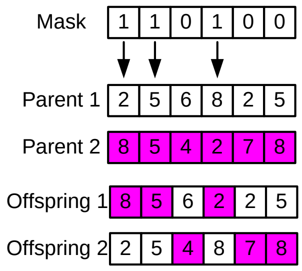
3.2 Crossover and Mutation

The genetic algorithm uses the individuals in the current generation to create the children that make up the next generation. Besides elite offspring, which correspond to the individuals in the current generation with the best fitness values, the algorithm creates:

- Crossover offspring by selecting vector entries, or genes, from a pair of individuals in the current generation and combines them to form a child.
- Mutation offspring by applying random changes to a single individual in the current generation to create a child.

Crossover and Mutation are two crucial factors in the biological evolutionary process. There are several types of crossover operator such as: single point, two point, uniform, and etc. In this paper, we use uniform crossover operator. Uniform crossover at first generates a uniformly random crossover mask and then exchanges relative genes between parents according to the mask. Figure (2) illustrates the crossover operation.

Mutation operator forces random changes in various chromosomes. It guarantees that we will not be placed in local minimum. There are several types of mutation operation, such as: uniform, Gaussian, and etc. In this paper, we use uniform mutation operator. Figure (3) illustrates the mutation operation.



Mask 1 1 0 1 0 0

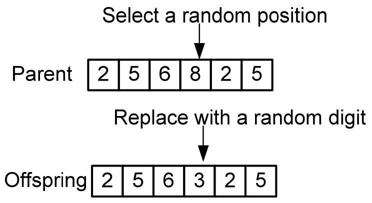
Parent 1 2 5 6 8 2 5

Parent 2 8 5 4 2 7 8

Offspring 1 8 5 6 2 2 5

Offspring 2 2 5 4 8 7 8

Fig. 2: Illustration of crossover operation



Select a random position

Parent 2 5 6 8 2 5

Replace with a random digit

Offspring 2 5 6 3 2 5

Fig. 3: Illustration of mutation operation

3.3 Evaluation and Selection Operator

In GA approach, we need to evaluate our population by an evaluation function. We name it *f_obj(X),* in which X is the Chromosome (In our research is Prüfer number). The main idea is to minimize the *f_obj(X)*. It is clear that for the maximization, we need to minimize the *g_obj(X)=-f_obj(X)*. In GA approach, the evaluation function is used to select the best chromosomes from the population.

There are several ways to select the selection operator, such as: Stochastic Uniform, Uniform, Roulette, and etc. In this paper, we use Roulette wheel. Each slice in Roulette wheel is proportional to its fitness value. Figure(4) illustrates an example of Roulette wheel.

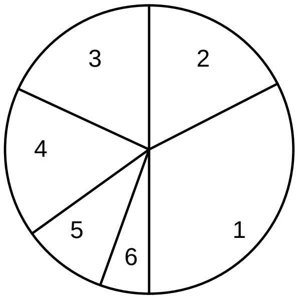


Fig. 4: An example of Roulette Wheel.

To define the evaluation function, we first define the normalized costs as:

$$\widetilde{C_l} = \frac{\sigma_l^2}{Max(\sigma_l^2)} \qquad (6)$$

$$\widetilde{C_s} = \frac{\sigma_s^2}{Max(\sigma_s^2)} \qquad (7)$$

$$\widetilde{C_L} = \frac{L}{Max(L)} \qquad (8)$$

The total cost that must be minimized is:

$$total\ cost = \left(\alpha \times \widetilde{C_l}\right) + \left(\beta \times \widetilde{C_s}\right) + (\gamma \times \widetilde{C_L}) \qquad (9)$$

We use the total cost as our evaluation function. So, the GA approach, will select the spanning tree with the minimum total cost as the best.

The overall GA procedure is outlined as follows:

Step 1) (Initialization) Choose the population size N, proper crossover probability P_c and mutation probability P_m, respectively. Generate initial population *P(0)*. Let the generation number *t=0*;

Step 2) (Crossover) Choose the parents from *P(t)* with probability P_c for crossover. If the number of parents chosen is odd, then randomly choose additional one from *P(t)*. Afterwards, randomly match every two parents as a pair and use the proposed crossover operator to each pair to generate two offspring. All offspring constitute a set denoted by *S*.

Step 3) (Mutation) Selection the parents for mutation from set *S* with probability P_m. For each chosen parent, the proposed mutation operator is applied to it to generate a new offspring. These new offspring constitute a set denoted by S_1.

Step 4) (Selection) Select the best *N* individuals among the set $P(t) \cup S_1$ as the next generation population *P(t+1)*, let *t=t+1*;

Step 5) (Termination) If termination condition hold, then stop, and keep the best solution obtained as the approximate global optimal solution of the problem; otherwise, go to step 2. We used 500 generation as our termination condition.

For GA approach, we need chromosome and evaluation function. We define the chromosome with three genes as defined in Figure (5). The chromosome length depends on the number of nodes we will use in our network.

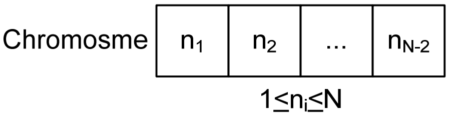


Fig. 5: Chromosome with three genes

After finding the best spanning tree, we can force STP to select it by assigning proper values to link costs and switch IDs.

4. Simulation Results

In this section, we show the computational complexity efficiency of proposed GA approach compared with our previous work [3]. We simulated our algorithm with MATLAB R2007b [6].

Figure (6) shows a simple network graph with 5 switches and 8 links.

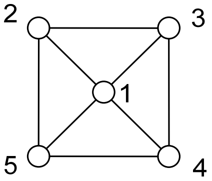


Figure 6. A simple network graph

Assume the same links with bandwidth equal to 1Gbps and the same switches with switching capacity equal to 4Gbps. There is 50 Mbps traffic demands from node 5 to node 2 and 100 Mbps from node 5 to all other nodes.

Now, consider the link load balancing (LLB) criterion $(\alpha = 1, \beta = 0, \gamma = 0)$. First, we examine the default spanning tree created by standard STP showed in figure (7). For simulation of standard STP, we used OPNET simulator tool [7].

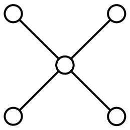


Figure7. Spanning tree created by standard STP

The best spanning tree selected by our GA approach based on LLB is shown in figure (8):

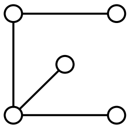


Figure 8. Spanning tree created by GA approach

Now, we can see the effectiveness of our approach in terms of link load balancing. As mentioned before, variance of the normalized link loads (σ_l^2) is a good metric for comparison. This metric is 0.216 for STP and 0.114 for our approach.

In our previous work [3], we compute the total cost for each spanning tree and then we select the best one with minimum total cost. This algorithm is very time consuming, especially when our network is large. In this paper, we proposed a new approach using Genetic Algorithm. It reduces the computational complexity by selecting the best spanning tree in a stochastic manner.

Prüfer number can be encoded and decoded in times that are *O(NlogN)*[4]. This part of calculation is common between our previous algorithm and our new GA approach. In our previous work, there was $N^{(N-2)}$ spanning trees that must be evaluated, and for each spanning tree, we must calculate its costs that is *O(N)*. Therefore, for each spanning tree, we have a computational complexity of $O(N^2 logN)$ so, the computational complexity order of our previous method is :

$$AN^{N-2}N^2 log\, N = AN^N \log N \qquad (10)$$

Where A is a constant.

In Table (1), you can see the complexity values of our previous algorithm calculated according to formula (10) for different values of nodes *(N).*

It is difficult to determine the computational complexity of GA approach, but we can estimate it by heuristic methods. One of the proposed heuristic methods in literature is evaluating the complexity by $O(x^K)$ [8], where x is complexity order of coding and decoding in GA approach and K is an integer number. As mentioned before, the coding and decoding complexity is *O(NlogN)*. To estimate the complexity in our GA approach, we simulated four different cases and the real complexity is measured for each case. Results can be seen in Table (2). By considering these results and above mentioned notes, the complexity of our GA approach can be described by:

$$GA\ order = B(N\log(N))^K \qquad (11)$$

where B is a constant coefficient. By replacing different values of K, *(K=2,3,4)* in (11) and comparing it with values indicated in Table (2), we found that K=3 is the best estimation. Therefore, the computational complexity of our GA approach can be estimated as:

$$GA\ order = B(N\log(N))^3 \qquad (12)$$

The comparison between measured (real) complexity values and estimated values is indicated in Table (3). GA is a stochastic process, so the measured complexity values may vary in different simulation tries for the same network. Therefore, although the estimated values are not exactly the same as measured ones, they are good estimations for the complexity order of our GA algorithm.

Figures (9) and (10) show the complexity of our previous approach and GA approach, respectively. As you can see, the complexity of our previous approach is very high for medium and large networks. In such cases, GA approach is helpful.

Table 1: Complexity of our previous approach

	Traditional Algorithm Complexity
Nodes=5	1.0059e+003
Nodes=10	2.3026e+009
Nodes=20	1.5706e+025
Nodes=50	6.9491e+083

Table 2: Measured Complexity of GA approach

	GA Complexity
Nodes=5	330
Nodes=10	43.878e+004
Nodes=20	0.899e+005
Nodes=50	23.6001e+006

Table 3: Comparison between measured complexity values and estimated values for GA approach

	GA Complexity (Real)	GA Complexity (Estimated)
Nodes=5	330	521.1139
Nodes=10	43.878e+004	1.2208e+004
Nodes=20	0.899e+005	2.1508e+005
Nodes=50	23.6001e+006	7.4837e+006

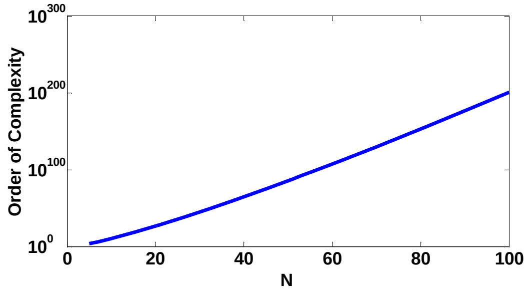


Fig. 9: Complexity Our previous algorithm

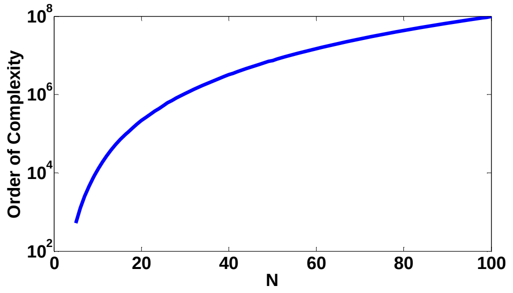


Fig. 10: Estimated Complexity of GA algorithm

5. Conclusion

In this paper, we introduced a new Genetic Algorithm approach for finding the best spanning tree in a Metro Ethernet network. We select the best tree not only based on shortest path selection but also based on load balancing on links and switches. We defined three coefficients corresponding to above criterions respectively. This allows us to weight the importance of each criterion based on our goal. After finding the best spanning tree, we can force STP to select it by assigning proper values to link costs and switch IDs.

In our previous algorithm, we search in all possible spanning trees whereas in this paper we showed that GA approach can reduce the computational complexity.

References

[1] M. Huynh, and P. Mohapatra, "Metropolitan Ethernet Network: A move from LAN to MAN," *Computer Networks 17, Elsevier,* pp. 4867-4894, 2007.

[2] P.M.V. Nair, "Quality of service in Metro Ethernet," *Ph.D. thesis*, Southern Methodist University, 2006.

[3] G. Mirjalily, H. Karimi, F. Adibnia and S. Rajai, "A new approach to select best spanning tree in Metro Ethernet Networks", *accepted in 8th IEEE CIT conf.*, Sydney, Australia, July 2008.

[4] L. Hanr, and Y. Wang, "A novel Genetic Algorithm for Degree-Constrained Minimum Spanning Tree Problem," *IJCSNS, vol. 6, no. 7A*, pp. 50-57, 2006.

[5] A. Cayley , "A theorem on trees," *Quarterly Journal of mathematics 23*, pp.376-378, 1889.

[6] A. Chipperfield, P. Fleming, H. Pohlheim, and C. Fonseca, "Genetic Algorithm TOOLBOX for use MATLAB," *MATLAB user's guide*, V. 1.2, 2007.

[7] OPNET Technologies, www.opnet.com

[8] J. A. Foster, "Genetic Algorithm hardness and Approximation Complexity," *Technical report No. LAL 9504*, University of Idaho, 1995.

Farhad Faghani was born in Iran in 1980. He received the B.S. degree in Electrical Engineering (Electronic) from Isfahan University of Technology (IUT) in 2004 and M.S. degree in Electrical Engineering (Communication Systems) from Yazd University in 2007 with honors. Now, he is Ph.D. student in Electrical Engineering of Yazd University and Instructor of Islamic Azad University (Najaf-Abad Branch). He has published three books in Electrical Engineering filed for undergraduate students. He is working on Data Communication Networks, specially on IP QoS, Traffic Engineering and Metro Ethernet Networks.

Mirjalily was born in Iran in 1969. He received the B.Eng. degree from Mashhad University, Iran, in 1991 and the MS. degree from K. N. Toosi University of Technology, Iran, in 1994, both in electrical engineering and with honors. In 2000 he completed the Ph.D. program at Tarbiat Modarres University, Iran. He has been a visiting researcher at the Communications Research Laboratory, McMaster University, Canada in 1998.

Since Fall 2001, he has been with Yazd University, Iran as an assistant professor. His current research interests include IP QoS, Traffic Engineering and Metro Ethernet Networks.

Dr. Mirjalily is a member of IAEE, IAENG and IEEE. He has published over 30 papers in national and international journals and conference proceedings. He has a grant from Iranian Telecommunication Research Center (ITRC) to design Metro Ethernet Network for Yazd state.