Journal of the Korean Institute
of Industrial Engineers
Vol.22, No.2, June, 1996

Balancing Loads on SONET Rings without Demand Splitting

Chae Y. Lee* and Seon G. Chang*

Abstract

The Self Healing Ring (SHR) is one of the most intriguing schemes which provide survivability for telecommunication networks. To design a cost effective SONET ring it is necessary to consider load balancing problems by which the link capacity is determined.

The load balancing problem in SONET ring when demand splitting is not allowed is considered in this paper. An efficient algorithm is presented which provides the best solution starting from various initial solutions. The initial solution is obtained by routing all demands such that no demands pass through an arc in the ring. The proposed algorithm iteratively improves the initial solution by examining each demand and selecting the maximum load arc in its path. The demand whose maximum arc load is biggest is selected to be routed in opposite direction. Computational results show that the proposed algorithm is excellent both in the solution quality and in the computational time requirement. The average error bound of the algorithm is 0.11% of the optimum and compared to dual-ascent approach which has good computational results than other heuristics.

1. Introduction

The networks employing fiber-optic technology use automatic diverse protection routing and dual homing to protect networks from fiber cable cuts and major hub failures. Such networks can evolve to Self Healing Rings (SHR) if they are proved to be economical. A SHR is a ring network that provides redundant bandwidth in which disrupted services can be automatically restored following network failures. The technology that facilitates this arrangement is based on the Synchronous Optical Network (SONET) standard for opti-

* Dept. of Industrial Management, KAIST, 373-1 Kusung-Dong, Taejon, Korea

cal transmission. Among many challenging problems in network planning that SONET gives rise to, the immediate one is that of a cost effective, survivable network design using SONET ring components. Since the cost of a SONET ring is dependent on its capacity, it is an important issue to find a routing for the demands such that the total capacity required to accommodate the traffic is as small as possible [3].

SONET ring is classified into two types: unidirectional ring and bi-directional ring. In unidirectional ring, all demands are routed in the same direction. However, in bi-directional ring a demand between two nodes can be assigned to any of the two direction. In the USHR (unidirectional SHR) two fibers are used. Working traffic is carried around a fiber in one direction only. The other fiber is used for the protection when the network component fails [5]. The BSHR (bi-directional SHR) uses two or four fibers depending on the spare capacity arrangement. In a four-fiber BSHR traffics are served in both directions. Each direction requires two fibers: one for working and the other for protection. In a two-fiber BSHR working and protection channels use the same fiber, in which half of the bandwidth is reserved for protection [5]. Therefore load balancing problem arises in the BSHR.

Shyur *et al.* [1] studied load balancing problem in which demand splitting is allowed by integer. Good computational results are obtained compared to the Demand Loading Balancing Algorithm by Liese [4]. However, no mathematical formulation of the problem and the convergence of the algorithm to the optimal solution are appearing in the study either by Shyur *et al.* [1] or others. Lee and Chang [2] formulated the problem and suggested a bounded approximation to the optimal solution. They provided an efficient algorithm INDES [2] that always satisfies either the minimum capacity or the capacity which is at most one unit higher than the minimum. Myung *et al.* [6] also studied load balancing problem in which demand splitting is allowed. They formulated it by linear programming and presented an efficient exact solution procedure which is far superior to any LP solution method.

Cosares and Saniee [3] proved NP-completeness of the load balancing problem when demand splitting is not allowed. They proposed a formulation by integer programming and a heuristic solution based on dual-ascent approach. The result points out that the dual-ascent scheme is an excellent approach to solving the load balancing problem particularly for instances with high demand volume. For the problem without demand splitting Myung *et al.* [6] developed a 2-approximation method. They have shown that the algorithm produced a solution whose objective value is within twice the optimal value of its linear programming relaxation. The computational results of the algorithm are not excellent but the bound of the solution is tight.

In this paper the load balancing problem when demand splitting is not allowed is considered. An efficient algorithm is developed which presents the best solution starting from several initial solutions. At each iteration, the algorithm selects a demand to route opposite direction and reduces the maximum arc load. The performance of the proposed algorithm is examined and compared to the dual-ascent approach [3].

2. Notation and An Example

We consider a ring with n nodes. Each node is numbered in ascending order in clockwise direction. We represent the arc $(i, i+1)$ as a_i and arc $(n, 1)$ as a_n. Each demand from node s to t, where $s < t$, is assumed to be symmetric such that the same amount of demand exists from node t to s. To solve the load balancing problem with m different demands, we present the following basic notations.

- $\mathbf{d}$: the demand vector $(d_1, d_2, \cdots, d_m)$, where d_j denotes the amount of demand j
- $\mathbf{x}$: a solution vector $(x_1, x_2, \cdots, x_m)$, where $x_j = 1$ means that demand j is routed in clockwise direction and $x_j = 0$ means demand j is routed counter-clockwise
- $z(\mathbf{x})$: the maximum arc load when demands are routed following the solution $\mathbf{x}$

The objective is to obtain the routing x_j of each demand j such that the maximum arc load is minimized. Consider a five-node example as shown in Figure 1 which is used to explain dual-ascent approach by Cosares and Saniee [3]. Figure 2 shows demand patterns in the SONET ring. The solution obtained by dual-ascent approach is $\mathbf{x} = (1, 1, 1, 0, 1, 1)$. From this solution the arc with the maximum load is a_1 and the maximum load is 18. Table 1 shows the solution of the example. In the table each demand has two columns: the first column represents amount of the demand routed clockwise and the second one represents routed counter-clockwise. That is, number on each cell represents load on the arc caused by the demand routed clockwise or counter-clockwise. The last column represents total load

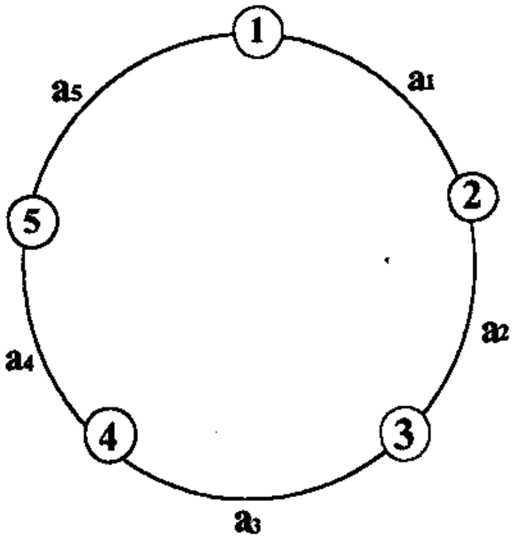

Figure 1. An Example of the SONET Ring

demand j	(s, t)	d_j
1	(1, 2)	2
2	(1, 4)	5
3	(2, 3)	9
4	(2, 4)	11
5	(3, 4)	4
6	(3, 5)	3

Figure 2. Demand Patterns in the Ring

Table 1. Solution by Dual-ascent Approach [3]

arc	demand 1		demand 2		demand 3		demand 4		demand 5		demand 6		load
a_1	2	0	5	0	0	0	0	11	0	0	0	0	18
a_2	0	0	5	0	9	0	0	0	0	0	0	0	14
a_3	0	0	5	0	0	0	0	0	4	0	3	0	12
a_4	0	0	0	0	0	0	0	11	0	0	3	0	14
a_5	0	0	0	0	0	0	0	11	0	0	0	0	11

on each arc.

3. Getting an Initial Solution and the Algorithm DENS

It is well known that the load balancing problem without the demand splitting is in the class of NP-complete. Thus, to find the best near-optimal solution the algorithm developed in the following section compares n different solutions each of which is obtained by its own initial solution. The initial solution is obtained by routing all demands such that no demands pass through a_i. For instance, suppose that the initial demands are routed such that no demands pass through a_5 in the example of Section 2. Then the initial solution becomes $\mathbf{x} = (1, 1, 1, 1, 1, 1)$. In the solution all demands are routed in the clockwise direction. The initial solution obtained by the above scheme has the effect that all demands are routed clockwise direction in the ring which is rotated such that the node $i+1$ is at the top of it.

The initial solution can be considered as an example of Weight-based routing [3]. It is shown that the objective value of the solution from Weight-based routing is at most twice of the optimal solution. Therefore the objective value of the initial solution is at most twice of the optimal solution.

Now, the objective of the load balancing problem is to minimize the maximum load in a ring. Once the initial solution is given, some demands are routed in opposite direction to improve the solution. To determine the demands to be routed in opposite direction the following theorem is applied.

Let $\mathbf{x}_{\text{ini}}$ and $\mathbf{x}_{\text{opt}}$ denote the initial solution and the optimal solution, respectively. Let D_i denote set of demands which pass through a_i in the initial solution. Also let D_R denote set of demands which is routed in opposite direction in the optimal solution.

Theorem 1. $\sum_{j\in D_i} d_j - z(x_{opt}) \le \sum_{j\in(D_R\cap D_i)} d_j - \sum_{j\in(D_R\cap D_i^c)} d_j$ for all i

Proof. In the optimal solution the load on a_i is given by $\sum_{j\in D_i} d_j - \sum_{j\in(D_R\cap D_i)} d_j + \sum_{j\in(D_R\cap D_i^c)} d_j$.

The first term denotes the sum of amount of demands that are passed through a_i in $\mathbf{x}_{\rm ini}$. The second term represents the sum of demands which are routed in opposite to the initial direction in $\mathbf{x}_{\rm opt}$ among demands in the first term. Also the third term denotes the sum of demands which are routed in opposite to the initial direction in $\mathbf{x}_{\rm opt}$ among demands that are not included in the first term.

Since $z(\mathbf{x}_{\rm opt})$ is greater than or equal to the load on a_i for all i, the following inequality holds:

$$\sum_{j\in D_i} d_j - \sum_{j\in(D_R\cap D_i)} d_j + \sum_{j\in(D_R\cap D_i^c)} d_j \le z(x_{opt})$$
for all i

Therefore,

$$\sum_{j\in D_i} d_j - z(x_{opt}) \le \sum_{j\in(D_R\cap D_i)} d_j - \sum_{j\in(D_R\cap D_i^c)} d_j$$
for all i. ■

Theorem 1 states that the bigger the load on a_i is in the initial routing $\mathbf{x}_{\rm ini}$, the more restricted portion of demand are allowed to pass through the arc in $\mathbf{x}_{\rm opt}$. Note that after a demand is routed in opposite direction some arcs are loaded by the demand and others are not. However all arcs must satisfy Theorem 1 in the optimal solution. Therefore, demands which pass through arcs with relatively big load must be selected to be routed in opposite direction.

After generating an initial solution the proposed algorithm DENS examines each demand and selects the maximum load arc contained in the path. The demand whose maximum arc load is biggest is selected to be routed in opposite direction. In case of a tie the algorithm breaks it by replacing the maximum load by the second to the maximum, the third to the maximum and so forth. The demand selected is routed in opposite direction if the maximum load is decreased by the routing. Otherwise, the above procedure is repeated for the other demands. When there is no demand to be routed the algorithm terminates. This process is repeated for every initial solution generated.

Algorithm DENS

Repeat the following process from $i = 1$ to n and choose the best solution.

STEP 1. Route all demands in one of the two directions such that no demands pass through a_i.

STEP 2. Let S = {1, 2,⋯, m} be the candidate set of demands.

STEP 3. For each demand in the set S, determine the maximum load among arcs contained in the path. Select the demand j^* which has the biggest maximum load. The ties are broken by replacing the maximum load by the second to the maximum, third to the maximum, and so forth.

STEP 4. (a) Route the demand selected in STEP 3 in opposite direction, if it decreases the maximum load of the ring. Go to step 2.

(b) Otherwise, let S = S − {j^*}. If the set S is empty, stop. Otherwise Go to STEP 3.

The operation of selecting the demand to be routed requires O(mn) time. Since the algorithm decreases the objective value iteratively, selecting demands is repeated at most D time, where D denotes the sum of the amount of all demands. Therefore each procedure requires O(Dmn) time and the time bound of the algorithm is O(Dmn^2).

The algorithm is illustrated with the example of Section 2. The initial solution generated is obtained by routing all demands in the path such that no demands pass through a_2.

i) Iteration 1

Initial solution is given as follows:

$\mathbf{x}$ = (1, 0, 0, 0, 1, 1)

Table 2. The Initial Solution

arc	demand 1		demand 2		demand 3		demand 4		demand 5		demand 6		load
a_1	2	0	0	0	0	9	0	11	0	0	0	0	22
a_2	0	0	0	0	0	0	0	0	0	0	0	0	0
a_3	0	0	0	0	0	9	0	0	4	0	3	0	16
a_4	0	0	0	5	0	9	0	11	0	0	3	0	28
a_5	0	0	0	5	0	9	0	11	0	0	0	0	25

a_4 is the arc with the maximum load and the objective is 28. Set S = {1, 2, 3, 4, 5, 6}. Among them, demands 2, 3, 4 and 6 pass through a_4. They equally have the biggest maximum arc load. The second to the maximum load on arcs in the path of demands 2, 3 and 4 is 25 while that of demand 6 is 16. The third to the maximum load on arcs in the path of demands 3 and 4 is 22. Note that demand 2 has no other arcs in its path. Finally demand 3 is selected since it has the load on a_3. Note that the maximum load is decreased by routing demand 3 in opposite direction. Thus, demand 3 is routed in clockwise direction, and the new solution becomes $\mathbf{x}$ = (1, 0, 1, 0, 1, 1).

ii) Iteration 2

After iteration 1, the following solution is obtained :

$\mathbf{x}$ = (1, 0, 1, 0, 1, 1)

Table 3. Solution After Iteration 1

arc	demand 1		demand 2		demand 3		demand 4		demand 5		demand 6		load
a_1	2	0	0	0	0	0	0	11	0	0	0	0	13
a_2	0	0	0	0	9	0	0	0	0	0	0	0	9
a_3	0	0	0	0	0	0	0	0	4	0	3	0	7
a_4	0	0	0	5	0	0	0	11	0	0	3	0	19
a_5	0	0	0	5	0	0	0	11	0	0	0	0	16

The maximum load is reduced to 19 units. S = {1, 2, 3, 4, 5, 6}. Still a_4 is the arc with the maximum load. Now demands 2, 4 and 6 have the biggest maximum load. By comparing the arcs in their paths, demand 4 is selected. However, the objective can never be improved by routing demand 4 in clockwise direction. Therefore, S becomes {1, 2, 3, 5, 6}. Thus demand 2 is selected to be routed at this iteration.

iii) iteration 3

$\mathbf{x} = (1, 1, 1, 0, 1, 1)$

Table 4. Solution After Iteration 2

arc	demand 1		demand 2		demand 3		demand 4		demand 5		demand 6		load
a_1	2	0	5	0	0	0	0	11	0	0	0	0	18
a_2	0	0	5	0	9	0	0	0	0	0	0	0	14
a_3	0	0	5	0	0	0	0	0	4	0	3	0	12
a_4	0	0	0	0	0	0	0	11	0	0	3	0	14
a_5	0	0	0	0	0	0	0	11	0	0	0	0	11

The maximum load is reduced to 18 units. Now a_1 is the arc with the maximum load. Demands 1, 2 and 5 are considered. Since demand 1 is the only demand which reduces the maximum load by routing opposite direction, it is selected and routed counter-clockwise direction.

iv) iteration 4

$\mathbf{x} = (0, 1, 1, 0, 1, 1)$

Table 5. Solution After Iteration 3

arc	demand 1		demand 2		demand 3		demand 4		demand 5		demand 6		load
a_1	0	0	5	0	0	0	0	11	0	0	0	0	16
a_2	0	2	5	0	9	0	0	0	0	0	0	0	16
a_3	0	2	5	0	0	0	0	0	4	0	3	0	14
a_4	0	2	0	0	0	0	0	11	0	0	3	0	16
a_5	0	2	0	0	0	0	0	11	0	0	0	0	13

The maximum load is reduced to 16 units. Now a_1, a_2 and a_4 are arcs with the maximum load. Since the maximum load cannot be reduced by routing of any demand, the algorithm terminates. Note that the solution obtained by the dual ascent approach is 18.

4. Computational Results

The performance of the proposed algorithm DENS is examined and compared with the dual-ascent approach [3]. Experiments are performed with six different ring sizes and three demand volumes. The demand volumes are classified such that the probability that a demand exists between any pair of nodes is 25%, 50% and 100%. Thus 18 cases are examined each with five problems. In each problem demands are randomly generated between 5 and 100 units.

Each cell in the Table 6, 7 and 8 represents the average of the five problems. The optimal solution is obtained by using the CPLEX optimizer code. Each number in parentheses denotes the gap from the optimum computed by (objective value - optimum value)*100/optimum value. All experiments are performed at 586 IBM PC.

Of the 90 cases examined in this paper, DENS has better results than the dual-ascent approach in 63 cases and the same results in 24 cases. The dual-ascent scheme has better results in three cases in which the differences are only one unit. In the average, algorithm DENS attained 0.11% of the optimum while

Table 6. Solutions for the Demands with Probability of 25%

Number of nodes in the ring	Optimum value	Algorithm DENS (%)	CPU seconds	Dual Ascent (%)	CPU seconds
5	75.8	75.8 (0.00)	0.01	75.8 (0.00)	0.65
10	244.2	244.8 (0.25)	0.01	248.6 (1.80)	1.1
15	560.6	562.8 (0.39)	0.09	574.2 (2.43)	1.92
20	937.2	937.8 (0.06)	0.36	974.4 (3.97)	3.56
25	1456*	1458.2 (0.15)	1.14	1515.2 (4.07)	6.35
30	2084*	2085.6 (0.08)	2.65	2098.8 (0.71)	9.83

* represents lower bound when the optimal solution cannot be obtained.

Table 7. Solutions for the Demands with Probability of 50%

Number of nodes in the ring	Optimum value	Algorithm DENS (%)	CPU seconds	Dual Ascent (%)	CPU seconds
5	139.6	139.6 (0.00)	0.01	139.6 (0.00)	0.9
10	380.6	382.6 (0.53)	0.03	393.2 (3.31)	1.13
15	945	945.6 (0.06)	0.16	946 (0.11)	2.13
20	1643.2	1645.4 (0.13)	0.73	1663.2 (1.22)	3.55
25	2379.6*	2380.4 (0.03)	2.31	2426.2 (1.96)	6.13
30	3506.8*	3509 (0.06)	6.23	3532.2 (0.72)	9.83

* represents lower bound when the optimal solution cannot be obtained.

dual-ascent scheme did 1.35% of the optimum. The dual-ascent approach has better results as the demand volume increases. However the proposed algorithm DENS shows excellent results without regard to the demand volume.

5. Conclusion

An algorithm DENS is proposed to solve the load balancing problem on SONET rings.

Table 8. Solutions for the Demands with Probability of 100%

Number of nodes in the ring	Optimum value	Algorithm DENS (%)	CPU seconds	Dual Ascent (%)	CPU seconds
5	209.2	209.2 (0.00)	0.01	212.8 (1.72)	0.84
10	730.2	730.4 (0.03)	0.03	734.8 (0.63)	1.26
15	1545.2	1547 (0.12)	0.34	1555.8 (0.69)	1.84
20	2881.4	2881.8 (0.19)	1.63	2886.8 (0.19)	3.66
25	4291.4*	4294.8 (0.08)	5.51	4313 (0.50)	6.09
30	6145*	6145 (0.00)	15.2	6156.6 (0.19)	9.99

* represents lower bound when the optimal solution cannot be obtained.

The initial solution is obtained by routing all demands such that no demands pass through an arc in the ring. It is shown that the objective of the initial solution is at most twice of the optimal solution. The algorithm DENS iteratively improves the initial solution. The algorithm examines each demand and selects the maximum load arc contained in the path. The demand whose maximum arc load is biggest is selected to be routed in opposite direction. The performance of the proposed algorithm is compared with the dual-ascent approach. The algorithm DENS demonstrated better computational results than the dual ascent scheme. The average error bound of the solution obtained by the DENS is 0.11% of the optimum in the average.

References

[1] C. C. Shyur, Y. M. Wu and C. H. Chen, A Capacity Comparison for SONET Self-

Healing Ring Networks. *IEEE GLOBECOM*. 3, 1574-1578 (1993).

[2] Chae Y. Lee and Seon G. Chang, Balancing Loads on SONET Rings with Integer Demand Splitting. Working Paper.

[3] S. Cosares and I. Saniee, An Optimization Problem Related to Balancing Loads on SONET Rings. *Telecommunication Systems*. 3, 165-182 (1994).

[4] Stephen T. Liese, Surviving Today's Competitive Climate. *Telephony*, 113-122 (1992).

[5] Tsong-Ho Wu, *Fiber Network Service* Survivability. Artech House, London (1992).

[6] Young-soo Myung, Hu-gon Kim and Dong-wan Tcha, Optimal Load Balancing on SONET Bidirectional Rings. Working Paper.

96년 3월 최초 접수, 96년 5월 최종 수정