Efficient Load Balancing Algorithms for a Resilient Packet Ring

Kwang Soo Cho, Un Gi Joo, Heyung Sub Lee, Bong Tae Kim, and Won Don Lee

ABSTRACT—The resilient packet ring (RPR) is a data optimized ring network, where one of the key issues is on load balancing for competing streams of elastic traffic. This paper suggests three efficient traffic loading algorithms on the RPR. For the algorithms, we evaluate their efficiency via analysis or simulation.

Keywords—Resilient packet ring (RPR), loading problem, counter-rotating ring, routing, load balancing.

I. Introduction

This paper concerns load balancing problems on a resilient packet ring (RPR), where the RPR is offered by IEEE 802.17 [1]. The RPR is well suited for a metropolitan area network with two counter-rotating rings where multiple stations share the bandwidth. The ring loading (load balancing) problems can be classified into two kinds: ones with and ones without demand splitting. Split loading allows the splitting of a demand into two portions to be carried out in two directions, while an unsplit loading is one in which each demand must be entirely carried out in either the clockwise or counter-clockwise direction. Since the cost of the ring increases with its capacity, it is desirable to route the demands so as to minimize this maximum load. The min-max problem is stated as follows: given a set of nodes on a ring network and a set of demands between pairs of nodes, how to allocate bandwidths to each traffic demand so that the maximum of the loads on the links in the network is as small as possible [2], [3].

Manuscript received June 03, 2004; revised Dec. 16, 2004.

Kwang Soo Cho (phone: +82 42 860 6191, email: choks@etri.re.kr), Heyung Sub Lee (email: leehs@etri.re.kr), and Bong Tae Kim (email: bkim@etri.re.kr) are with Broadband Convergence Network Research Division, ETRI, Daejeon, Korea.

Un Gi Joo (email: ugjoo@sunmoon.ac.kr) is with the Department of Knowledge and Industrial Engineering, Sunmoon University, Chungnam, Korea.

Won Don Lee (email: wlee@cnu.ac.kr) is with the Department of Computer Science, Chungnam National University, Daejeon, Korea.

For research on the unsplit min-max problem, Cosares and Saniee [4] and Dell'Amico et al. [5] studied the problems on bi-directional synchronous optical network (SONET) rings. For the split loading problem, Myung et al. [6] and Wan and Yang [7] considered the min-max loading problem on the SONET rings. Most research on the loading problems has been on the two-fiber bi-directional SONET ring except for Wan and Yang [7]. They noticed that each link on the two-fiber bi-directional SONET ring requires a capacity of at least the sum of both directional working traffic due to the protection requirement. Wan and Yang [7] considered the min-max loading problem on an unidirectional SONET ring with two working counter rotated fibers.

This paper develops efficient loading algorithms for the split and unsplit loading problems on the RPR.

II. Problem Description

Consider an RPR network with n nodes sequentially numbered from 1 to n in the clockwise direction, where two counter-rotating data links exist between two consecutive nodes. Suppose that there are K types of demands d_k on the ring bandwidth, $k=1,2,\ldots, K$. For each demand d_k, let O_k and D_k denote the originating and terminating nodes, respectively. The demand traffic d_k of the RPR is composed of mainly internet traffic and measured usually as bits per second.

Let x_k be a variable denoting the fraction of the total demand between O_k and D_k routed in the clockwise direction. Thus, $x_k=1$ and $x_k=0$ indicate that all the demand d_k are routed in a clockwise and counter-clockwise direction, respectively.

Let L_k^+ denote the set of links contained in the clockwise path from O_k to D_k for a demand type k as depicted in Fig. 1. Similarly, let $L_k^- = L - L_k^+$ denote the set of links contained in

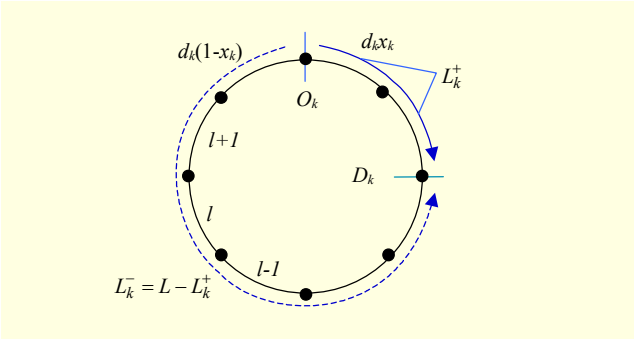

Fig. 1. Definition of L_k^+ and L_k^-.

the counter-clockwise path from O_k to D_k for a demand type k, where L denotes the link set of the given ring. For each link $l \in L$, let $K_l^+ = \{k | l \in L_k^+\}$ and $K_l^- = \{k | l \in L_k^-\}$ denote the demand index set of the origin-destination pairs whose clockwise and counter-clockwise paths contain the link l, respectively.

Our problem is to find the optimal loading of the RPR either with or without demand splitting. For the problem, let z represent the required ring capacity associated with a loading. Additionally, let's denote the link loads as R_l^+ and R_l^- for the clockwise and counter-clockwise directional link l, respectively. In other words, $R_l^+ = \sum_{k \in K_l^+} d_k x_k$ and $R_l^- = \sum_{k \in K_l^-} d_k (1 - x_k)$. The splitting problem can then be expressed as $\min_{x_k} \{z | z \geq R_l^+, z \geq R_l^-$, and $0 \leq x_k \leq 1$, for all l and $k\}$, termed P1 in this paper. The objective of P1 is to find $\{x_k\}$ which minimizes the ring load z. The constraints of $z \geq R_l^+$ and $z \geq R_l^-$ for all l denote that the ring load z is the maximum of the clockwise and counter-clockwise link load, respectively. The third constraint $0 \leq x_k \leq 1$ for all k implies that the splitting is allowed for each demand d_k. If the decision variable x_k is restricted to $x_k = 0$ or 1, then the problem becomes an unsplit loading one, termed P2 in this paper.

The researches of [4], [5] and [6] are on the problems of $\min_{x_k} \{z | z \geq R_l^+ + R_l^-\}$, and there is no previous research on P1 and P2 except Wan and Yang [7], where Wan and Yang discussed P1 without the development of any solution algorithm. We noticed that the loading problem is only a subproblem within the comprehensive ring planning and the problem has to be solved multiple times in practical applications. Therefore, the efficiency of a loading algorithm has a big effect on the overall performance of the planning procedure. This paper develops an efficient optimal solution algorithm for P1 in section III and suggests two heuristic algorithms for P2 in section IV.

III. Algorithm for Split Loading Problem

There exist two types of the ring loading algorithm; one is a construction type and the other is an improvement type. This section develops the improvement type algorithm for problem (P1) with the initial solution of $x_k = 1$ for all k.

The following property is derived by Wan and Yang [7].

Property 1. The ring load z is a convex function with respect to $\sum_{k=1}^{K} d_k x_k$.

Even though Property 1 characterizes the convexity of the ring load z, finding the optimal loading is difficult since there are many alternatives $\{x_k\}$ with the same value of $\sum_{k=1}^{K} d_k x_k$. Therefore, we further characterize the optimal routing by explicitly considering each demand type. For the characterization, let's define additional notations x_k^i and δ_i such that $\delta_i = \max_{1 \leq l \leq n} \{\sum_{k \in K_l^+} d_k x_k^i\} - \max_{1 \leq l \leq n} \{\sum_{k \in K_l^-} d_k (1 - x_k^i)\} = \max_{1 \leq l \leq n} \{R_l^+\} - \max_{1 \leq l \leq n} \{R_l^-\}$, where x_k^i denotes the variable x_k at the i-th iteration. Let $x_j^{i+1} = x_j^i - \Delta_i / d_j$ for a demand type j and $x_k^{i+1} = x_k^i$ for all k, $k \neq j$, where $\Delta_i = \min\{\delta_i / 2, d_j\}$. Then, δ_i has the following property, where its proof is given in the Appendix.

Property 2. The values of $\{\delta_i\}$ have a relationship of $\delta_i \geq \delta_{i+1}$ for each i, i=1,2,…, K-1.

According to Property 2, if we find out a solution with $\delta_i = 0$, then the current solution is an optimal solution. Based upon Property 2, we develop an optimal solution algorithm, denoted as the min-max algorithm.

Min-Max Algorithm

Step 0. Sequence all the K demands as follows. First, for the demands with $O_k > D_k$ if $O_{k_1} > O_{k_2}$, then $k_1 < k_2$; if $O_{k_1} = O_{k_2}$ and $D_{k_1} < D_{k_2}$, then $k_1 < k_2$. Then, for the demands with $O_k < D_k$, if $O_{k_1} < O_{k_2}$, then $k_1 < k_2$. Finally, if $O_{k_1} = O_{k_2}$ and $D_{k_1} > D_{k_2}$, then $k_1 < k_2$. If sequencing is finished, set i as 1.

Step 1. Let $x_k^i = 1$ for all k, k=1,2,…, K, where x_k^i denotes all the demand d_k routed in a clockwise direction at the i-th iteration. Calculate each initial clockwise link load as $R_l^+ = \sum_{k \in K_l^+} d_k x_k^i$ and let $R_l^- = 0$ for all links l.

Step 2. Compute $\delta_i = \max_{1 \leq l \leq n} \{R_l^+\} - \max_{1 \leq l \leq n} \{R_l^-\}$. If $\delta_i > 0$, go to step 3. Otherwise, stop.

Step 3. (1) For a link l_1 such that $\max_{1 \leq l \leq n} \{R_l^+\} = R_{l_1}^+$, find a demand type j with maximum clockwise load on the link l_1, i.e., $j = \arg \max_{k \in K_{l_1}^+} \{d_k x_k^i\}$.

(2) Calculate Δ_i for the demand j as $\Delta_i = \min\{\delta_i / 2, d_j\}$.

(3) Let $x_j^{i+1} = x_j^i - \Delta_i / d_j$ and $x_k^{i+1} = x_k^i$ for all k, k=1,2,…, K, k $\neq j$.

(4) For each link l, let $R_l^+ = R_l^+ - \Delta_i$ if $l \in L_j^+$, and let $R_l^- = R_l^- + \Delta_i$, otherwise. Go to step 2 with i = i+1 while $i \le K$.

Step 3 of the min-max algorithm improves the current solution $\{x_k^i\}$ by rerouting the amount of Δ_i traffic demand of demand type j, where the rerouting of demand j makes that either the resulting solution $\{x_k^{i+1}\}$ satisfies $\max_{1\le l\le n}\{R_l^+\} = \max_{1\le l\le n}\{R_l^-\}$ or all the demand d_j is routed in the counter-clockwise direction. Step 3 is activated only when $\delta_i > 0$ for demand type j, $j \in K_{l_1}^+$; therefore, $\delta_i > \delta_{i+1}$, and the algorithm terminates with $\delta_i = 0$. The rerouting step requires $O(n)$ for each demand type k and there are K demand types; thus, the overall computational time complexity of the min-max algorithm is $O(nK)$.

IV. Algorithms for Unsplit Loading Problem

The unsplit loading problem (P2) is NP-complete; therefore a heuristic algorithm is required. Wan and Yang [7] suggested heuristic algorithms for P2 on a unidirectional SONET ring and evaluated their worst case performance. This section suggests two heuristic algorithms of the construction type for RPR networks and evaluates their average performance.

Heuristic Algorithm H1

Step 0. Find an optimal solution $\{x_k^*\}$ by using the min-max algorithm.

Step 1. If all x_k^*'s are integers, stop. Otherwise, let $S = \{x_k^* \mid 0 < x_k^* < 1\}$, $R_l^+ = \sum_{k\in K_l^+} d_k x_k^*$, $R_l^- = \sum_{k\in K_l^-} d_k(1-x_k^*)$, and go to step 2.

Step 2. For each k, $k \in S$, set the value of x_k^* as zero or one as follows.

(1) Calculate $\lambda_1 = \max\{\max_{l\in L_k^+}\{R_l^+\} - d_k x_k^*, \max_{l\in L_k^-}\{R_l^-\} + d_k x_k^*\}$ and $\lambda_2 = \max\{\max_{l\in L_k^+}\{R_l^+\} + d_k(1-x_k^*), \max_{l\in L_k^-}\{R_l^-\} - d_k(1-x_k^*)\}$.

(2) If $\lambda_1 < \lambda_2$, then set $R_l^+ = R_l^+ - d_k x_k^*$ for $l \in L_k^+$, $R_l^- = R_l^- + d_k x_k^*$ for $l \in L_k^-$, and $x_k^* = 0$. Otherwise, set $R_l^+ = R_l^+ + d_k(1-x_k^*)$ for $l \in L_k^+$, $R_l^- = R_l^- - d_k(1-x_k^*)$ for $l \in L_k^-$, and $x_k^* = 1$.

The heuristic algorithm H1 uses the min-max algorithm of problem [P1] at Step 0. Step 2 reroutes the demand type k in set S to one of the two directions which would result in the smaller increase of ring load, where λ_1 and λ_2 denote the amount of increasing ring load resulting from entirely rerouting the demand k in the counter-clockwise and clockwise directions, respectively.

We can also consider the shortest-distance routing as another heuristic algorithm, denoted as H2.

Heuristic Algorithm H2

For each k, if $2(D_k - O_k) > n$ when $O_k < D_k$, or if $2(D_k - O_k) > -n$ when $O_k > D_k$, set $x_k = 0$. Otherwise, set $x_k = 1$.

For H1 and H2 algorithms, we compare their average performance as shown in Table 1. The first column in Table 1 represents the number of nodes n in the ring and total number of demand types K. For each instance (n, K), we randomly generate ten problems with demands between 5 and 100 for randomly selected pairs of originating and terminating nodes, and find out the resulted average ring load (Load) and calculate the average computation time (Time) of H1 and H2 using Visual C/C++ code on a Pentium IV PC (1.0 GHz, Windows XP). For an evaluation of the solution quality, we calculate the relative deviations as

$$\frac{(\text{load of H1 or H2}) - (\text{load of min-max algorithm})}{(\text{load of min-max algorithm})} \times 100,$$

since the solution of P1 is a lower bound of the optimal solution

Table 1. Mean value of solutions and CPU times.

Problem Size (n, K)	H1 Algorithm		H2 Algorithm	
	Load (Deviation)	Time (s)	Load (Deviation)	Time (s)
(5, 6)	117.4 (12.72)	0.000076	164.6 (58.04)	0.000006
(5, 8)	143.7 (9.44)	0.000091	253.7 (93.22)	0.000007
(5, 10)	160.3 (2.89)	0.000109	311.6 (100)	0.000007
(10, 12)	224.4 (11.20)	0.000203	307.2 (52.23)	0.000013
(10, 23)	368.4 (1.68)	0.000499	724.6 (100)	0.000017
(10, 45)	679.2 (2.86)	0.001782	1320.6 (100)	0.000031
(15, 25)	419.9 (3.11)	0.000923	775.6 (90.45)	0.000029
(15, 50)	744.1 (4.68)	0.003085	1387.1 (95.15)	0.000043
(15, 105)	1519.0 (2.82)	0.011498	2954.8 (100)	0.000095
(20, 40)	578.6 (5.09)	0.002274	1085.9 (97.24)	0.000045
(20, 95)	1336.9 (0.92)	0.012608	2627.3 (98.32)	0.000102
(20, 190)	2650.1 (2.10)	0.039352	5191.0 (100)	0.000217
(25, 60)	901.5 (4.21)	0.005879	4230.7 (100)	0.000083
(25, 150)	2099.1 (2.64)	0.033224	4090.2 (100)	0.000187
(25, 300)	4213.3 (1.71)	0.105620	8284.8 (100)	0.000372
(30, 90)	1277.5 (3.55)	0.016318	2434.0 (97.29)	0.000136
(30, 200)	2739.6 (1.64)	0.061990	5390.7 (100)	0.000288
(30, 435)	5992.7 (1.45)	0.291000	11814.6 (100)	0.000676

of P2. We can observe that the deviation of H1 does not increase as (n, K) increases and the mean deviation of H1 is 4.15%. However, H2 has a mean deviation of 93.44% and its deviation increases as (n, K) increases. Even if the CPU times of both algorithms increase as (n, K) increases, the computation time is not large. Therefore, we can conclude that H1 and the min-max algorithm can be used for good load balancing even when the ring has a large value of (n, K).

V. Conclusion

This paper considers two min-max loading problems either with or without load splitting on an RPR. For the load splitting problem, we characterize its optimal solution to develop an efficient algorithm, the min-max algorithm. For the unsplit loading problem, we suggested two heuristic algorithms based upon the min-max algorithm and the short-way routing concept and showed their efficiency by using various numerical examples. We expect that our algorithms will be used for efficient ring loading on an RPR or a unidirectional dual ring network to improve the ring utilization. However, the development of improvement type algorithms for the effective loading of problem [P2] remains as a further study.

References

[1] RPR Alliance, A Summary and Overview of the IEEE 802.17 Resilient Packet Ring Standard, June 24, 2004.

[2] L. Massoulie and J. Roberts, "Bandwidth Sharing : Objectives and Algorithms," *INFOCOM'99,* 1999, pp.1395-1403.

[3] H-S. Lee, L-G. Joo, H-H. Lee, and W-W. Kim, "Optimal Time Slot Assignment Algorithm for Combined Unicast and Multicast Packets," *ETRI J.,* vol. 24, no. 2, 2002, pp. 172-175.

[4] S. Cosares and I. Saniee, "An Optimization Problem Related to Balancing Loads on SONET Rings," *Telecomm. Sys.,* vol.3, 1994, pp.165-181.

[5] M. Dell'Amico, M. Labbe, and F. Maffioli, "Exact Solution of the SONET Ring Loading Problem," *Oper. Res. Lett.*, vol.25, 1999, pp.119-129.

[6] Y.-S. Myung, H.-G. Kim, and D.-W. Tcha, "Optimal Load Balancing on SONET Bidirectional Rings," *Oper. Res.,* vol.45, 1997, pp.148-152.

[7] P.-J. Wan and Y. Yang, "Load-Balanced Routing in Counter Rotated SONET Rings," *Networks,* vol.35, 2000, pp.279-286.

Appendix : Proof of Property 2

Since $\max_{1\le l\le n}\{R_l^+\} = \max\{\max_{l\in L_j^+}\{R_l^+\},\ \max_{l\in L_j^-}\{R_l^+\}\}$

and

$\underset{1\le l\le n}{Max}\{R_l^-\} = \max\{\max_{l\in L_j^+}\{R_l^-\},\ \max_{l\in L_j^-}\{R_l^-\}\},$

$\delta_i = \max\{\max_{l\in L_j^+}\{R_l^+\},\ \max_{l\in L_j^-}\{R_l^+\}\} - \max\{\max_{l\in L_j^+}\{R_l^-\},\ \max_{l\in L_j^-}\{R_l^-\}\}$

and

$\delta_{i+1} = \max\{\max_{l\in L_j^+}\{R_l^+\} - \Delta_i,\ \max_{l\in L_j^-}\{R_l^+\}\}$

$\quad - \max\{\max_{l\in L_j^+}\{R_l^-\},\ \max_{l\in L_j^-}\{R_l^-\} + \Delta_i\},$

where

$R_l^+ = \sum_{k\in K_l^+} d_k x_k^i,\ R_l^- = \sum_{k\in K_l^-} d_k (1-x_k^i)$ and $\Delta_i = \min\{\delta_i/2, d_j\}.$

There are four case situations such that

Case 1: $\max_{l\in L_j^-}\{R_l^+\} \ge \max_{l\in L_j^+}\{R_l^+\} - \Delta_i$ and $\max_{l\in L_j^+}\{R_l^-\} \ge \max_{l\in L_j^-}\{R_l^-\} + \Delta_i$,

Case 2: $\max_{l\in L_j^-}\{R_l^+\} \ge \max_{l\in L_j^+}\{R_l^+\} - \Delta_i$ and $\max_{l\in L_j^+}\{R_l^-\} < \max_{l\in L_j^-}\{R_l^-\} + \Delta_i$,

Case 3: $\max_{l\in L_j^-}\{R_l^+\} < \max_{l\in L_j^+}\{R_l^+\} - \Delta_i$ and $\max_{l\in L_j^+}\{R_l^-\} \ge \max_{l\in L_j^-}\{R_l^-\} + \Delta_i$,

Case 4: $\max_{l\in L_j^-}\{R_l^+\} < \max_{l\in L_j^+}\{R_l^+\} - \Delta_i$ and $\max_{l\in L_j^+}\{R_l^-\} < \max_{l\in L_j^-}\{R_l^-\} + \Delta_i$.

For case 1, δ_{i+1} becomes $\max_{l\in L_j^-}\{R_l^+\} - \max_{l\in L_j^+}\{R_l^-\}$ and we can easily notice that $\delta_i \ge \delta_{i+1}$ since $\max_{l\in L_j^-}\{R_l^+\} \le \max\{\max_{l\in L_j^+}\{R_l^+\}, \max_{l\in L_j^-}\{R_l^+\}\}$ and $\max_{l\in L_j^+}\{R_l^-\} = \max\{\max_{l\in L_j^+}\{R_l^-\}, \max_{l\in L_j^-}\{R_l^-\}\}$ by the supposition of the case.

The relationship $\delta_i \ge \delta_{i+1}$ can be proven for all the other cases similarly.