SBIBD를 이용한 분산시스템의 부하 균형 알고리즘

김성열*

Synchronous Distributed Load Balancing Algorithm Employing SBIBD

Seong-yeol Kim*

요 약

분산시스템에서 비집중화된 방법으로 부하균형을 유지하기 위해서는 네트워크상의 각 노드는 다른 노드들의 부하상태정보를 가져야만 한다. 네트워크상에 v 개의 노드가 존재할 때 모든 노드간에 부하정보를 교환하기 위해서는 $O(v^2)$의 트래픽 오버헤드가 필요하게 된다.

이 논문에서는 분산된 노드간에 동기적으로 동작하는 부하균형 알고리즘을 제시한다. 이를 위해 먼저 SBIBD(Symmetric Balanced Incomplete Block Design)에 근거하여 $(v, k+1, 1)$구조에 의해 $v=k^2+k+1$ 개의 노드간의 통신을 위해 $2k$ 정규그래프 구조를 갖는 네트워크 토폴로지를 구성하였다. 이 망에서 동작하도록 고안된 부하균형 알고리즘은 $O(v\sqrt{v})$의 메시지 오버헤드를 가지면서 각각의 노드가 v개의 모든 노드에 대한 부하상태정보를 가지도록 한다. 또한 이 알고리즘은 모든 링크가 부하상태정보 전송을 위해 $\sqrt{v}$의 동일한 트래픽을 갖도록 설계되었다.

ABSTRACT

In order to maintain load balancing in distributed systems in a decentralized manner, every node should obtain workload information from all the nodes on the network. It requires $O(v^2)$ traffic overheads, where v is the number of nodes. This paper presents a new synchronous dynamic distributed load balancing algorithm for a $(v, k+1, 1)$-configured network topology, which is a kind of $2k$ regular graph, based on symmetric balanced incomplete block design, where v equals k^2+k+1. Our algorithm needs only $O(v\sqrt{v})$ message overheads and each node receives workload information from all the nodes without redundancy. And load balancing in this algorithm is maintained so that every link has same amount of traffic by $\sqrt{v}$ for transferring workload information.

키워드

SBIBD, Block Design, 분산시스템, 부하균형, 네트워크 토폴로지

Ⅰ. 서 론

부하분산 기법은 부하상태에 대한 정보의 유지 및 전파, 프로세스이동시점의 결정, 이동시킬 프로세스 선택, 프로세스가 이동될 컴퓨터의 결정, 전송된 프로세스의 상태복구 등을 고려해야한다. 이때 컴퓨터의 부하상태를 측정하거나 부하상태정보를 전파하기 위한 방식은 최소의 오버헤드를 가져야한다. 컴퓨터의 작업량을 나타내는 지수는 CPU의 대기열에 의해 표현되는 것이 가장 효과적이라고 알려져 있다[1]. 부하균형 알고리즘은

* 울산과학대학 컴퓨터정보학부
접수일자 : 2004. 2. 12

정적(static)이거나 동적(dynamic)일 수 있다. 분산시스템이 v개의 컴퓨터로 구성되어 있을 때 정적인 시스템에서는 i번째 작업을 $i \pmod v$번째 컴퓨터에 할당한다. 동적인 시스템에서는 각 시스템의 부하상태 정보를 이용하여 부하가 적은 컴퓨터에 작업을 할당한다. 또한 작업량 분산 방식은 특정 시스템에서 모든 노드에 대한 부하정보를 가지고 있으면서 불균형된 노드 간에 부하를 분산하도록 중재하는 집중형과 각각의 노드가 다른 노드에 대한 부하정보를 가지고 부하분산을 실행하는 분산형으로 구분될 수 있다. 분산시스템에서 부하 분산을 위해서는 분산형의 동적인 알고리즘을 사용하는 것이 바람직하다[2][3]. 부하 분산을 위해 각각의 노드에게 주어지는 부하상태정보는 최신의 정보이어야 할 필요가 있다. 유효시간이 지나버린 정보는 각각의 노드에서 시스템의 전체에 대한 상황을 올바로 파악할 수 없게 하고 더불어 올바른 부하분산을 수행할 수 없게 한다. 이렇게 최신의 정보를 유지하기 위해서는 주기적으로 갱신되는 정보전송을 위한 통신비용이 심각한 문제가 될 수 있다.

[4][5][6][7][8] 등의 연구에서는 전체 노드 간에 각 노드들의 부하상태 정보를 공유하기 위한 과다한 통신비용을 줄이기 위해 인접한 노드 사이에만 부하상태정보를 교환하고 국부적으로 부하균등화 수행을 반복함으로써 전체적으로 균등부하에 수렴하게 됨을 주장하고 있다, 그러나 만일 적은통신비용으로 네트워크상의 모든 노드의 부하상태를 알 수 있다면 전체적으로 부하균등화를 이루는 시간은 상당히 단축될 수 있다.

[9][10][11][12] 등의 연구에서는 CWA(Cube Walking Algorithm)을 이용하여 하이퍼큐브 상에서 모든 노드에 대한 부하상태정보에 근거하여 신속한 부하균등화를 수행할 수 있음을 보여주었다. 그러나 이 방법은 전체노드의 부하상태정보를 얻기까지 v^2의 통신비용을 요구하고 또한 정보전송거리는 $O(\log_2 v)$가 된다. 통신횟수를 줄이기 위해 각 노드가 얻은 정보를 플러딩(flooding)하는 방법을 사용하는 경우에도 $v(\log_2 v)^2$의 통신비용이 요구되며 여기에는 전송정보의 중복이 존재하게 된다.

다른 형태의 연구로는 부하분산을 위해 SBN (Symmetric Broadcast Networks)에 근거하여 토폴로지에 독립적인 방법으로 각 노드간의 통신패턴을 형성하는 방법이 연구된 바 있으나 마찬가지로 부하상태정보 수집을 위해서는 v^2의 통신비용을 요구하고 있으며 정보전송거리는 $O(\log_2 v)$가 된다[13][14].

그리고 [15]에서 전체노드들의 적은 통신량으로 부하상태를 전송하기 위해 projective geometry를 이용한 방법이 제안된바 있지만 $v = k^2+k+1$ 개의 노드를 갖는 멀티컴퓨터시스템의 전송을 위한 projective plan의 구체적인 생성법이 없이 이론적인 수준에 머물렀다.

이 논문에서는 k가 소수인 경우의 SBIBD의 생성법을 고안하고 이를 이용하여 $v=k^2+k+1$ 개의 노드와 $v\times k$ 개의 링크를 가지며 각각의 노드는 균등하게 $2k$ 개의 인접노드를 갖는 정규그래프 형태의 네트워크 토폴로지를 설계하였다.

이와 같은 네트워크 상에서 각각의 노드는 k개의 인접노드로 k개의 노드에 대한 부하상태정보를 전송한다. 즉, 각각의 노드는 k 개의 노드로부터 정보를 수신하는데 각각 하나의 노드로부터 k 개의 노드에 대한 정보를 수신한다. 이렇게 수신된 모든 정보는 정보 발생지로부터 2단계 이내에서 수신 완료된 것이다. 또한 이렇게 수신된 정보에는 중복이 존재하지 않는다. 따라서 각각의 노드는 매 주기마다 k^2개의 노드에 대한 상태정보를 수신하게 된다. 또한 시간주기 T_{2t}와 T_{2t+1}에 수신하는 정보의 차(difference)가 k가 됨으로써 각 노드는 네트워크상의 모든 노드에 대한 부하상태정보를 얻을 수 있게 된다. 또한 이 알고리즘은 부하상태정보 전송을 위해 각각의 링크가 균등한 트래픽 오버헤드를 갖도록 설계되었다.

II. SBIBD(Symmetric Balanced Incomplete Block Design)

SBIBD는 다음과 같이 정의될 수 있다.

$V=\{v_0, v_1, \ldots, v_{v-1}\}$를 v개의 원소를 갖는 집합이라고 하자. 그리고 B를 다음과 같이 k 개의 원소를 갖는 부분집합들로 구성된 집합족 이라고 하자.

$$B=\{B_0, B_1, \ldots, B_{b-1}\},\ B_i=\{b_0, b_1, \ldots, b_k\}$$

유한결합구조 $\sigma=\{V, B\}$에 대하여 σ가 다음조건을 만족할 때 이는 BIBD(Balanced Incomplete Block Design)이다. 그리고 이를 (b, v, r, k, λ)-configuration이라 한다 [16][17].

(1) B는 블럭이라 불리는 b개의 부분집합으로 구성되며 각각의 부분집합은 V 에 속하는 k 개의 원소를 갖는다.

(2) V의 원소 각각은 r 개의 블록에만 존재한다

(3) V의 원소중 임의의 두 원소 쌍(pair)은 B에서 λ 번 나타난다.

(4) $k<v$

임의의 (b, v, r, k, λ)-configuration에 대하여 만일 다음 두 가지 조건을 만족하면 이는 SBIBD(symmetric balanced incomplete block design)이고 (v, k, λ)-configuration 이라 한다.

(1) $k=r$

(2) $b=v$

BIBD를 구성하기 위한 파라미터 b, v, r, k, λ 간에는 특별한 관계가 존재한다.

예를 들면 (b, v, r, k, λ)-configuration에서는 $bk=vr$ 이 성립하고 $r(k-1)=\lambda(v-1)$이 성립한다. 그리고 (v, k, λ)-configuration의 경우 모든 두 블록 사이에는 λ 개의 교집합이 존재한다. 그렇지만 임의의 (b, v, r, k, λ)-configuration 또는 (v, k, λ)-configuration이 존재하기 위한 조건 및 생성하기 위한 조건에 대해서는 알려진 바가 없다.

III. $(v, k+1, 1)$-configuration 생성과 네트워크 토폴로지 설계

이 장에서는 k가 소수인 경우에 $(v, k+1, 1)$-configuration의 조건을 만족하는 유한결합구조 $\sigma=\{V, B\}$를 생성하는 알고리즘을 제시한다.

이렇게 생성된 σ는 분산시스템 구성을 위한 네트워크 토폴로지를 설계하는데 사용된다. 이 논문의 3장과 4장에서 사용된 표기법과 의미는 다음과 같다.

V : v 개의 원소를 갖는 집합

B : 다음과 같이 b 개의 부분집합으로 이루어진 집합족

$\sigma[i]$: 유한결합구조 $\sigma=\{V, B\}$의 블럭 B_i

$\sigma[i, j]$: $\sigma[i]$의 원소 b_j

3.1 알고리즘 1: $(v, k+1, 1)$-configuration 생성 알고리즘

(1) 임의의 소수 k를 선택한 후 $v=k^2+k+1=k(k+1)+1$를 구한다.

(2) 유한결합구조 X, Y, Z를 다음과 같이 정의한다.

$Z=\{V, B\}$, $B=\{B_0, B_1, \ldots, B_{v-1}\}$, $|B_i|=k$

$Y=\{V, D\}$, $D=\{D_0, D_1, \ldots, C_{k^2-1}\}$, $|D_i|=k$

$X=\{V, C\}$, $C=\{C_0, C_1, \ldots, C_k\}$, $|C_i|=k$

(3) 유한결합구조 X와 Y를 다음과 같이 결정한다.

(a) $X[i, j]=\begin{pmatrix} 0 & \text{if} \quad j=0 \\ t \quad t=i\times k+j & \text{if} \quad j\geq 1 \end{pmatrix}$

(b) $Y[i, j]$

$$=\begin{pmatrix} X[0, t], \quad t=\lfloor i/k \rfloor+1 & \text{if } j=0 \\ X[j, t], \quad t=(i+(j-1)\times\lfloor i/k \rfloor) \bmod k+1 & \text{if } j\geq 1 \end{pmatrix}$$

(4) X와 Y로부터 다음과 같이 Z를 결정짓는다.

$Z[i] \leftarrow X[i]$

$Z[i+k+1] \leftarrow Y[i]$

예를 들면 알고리즘 1에 의해 구해진 $(13, 4, 1)$

-configuration은 표 1.과 같다.

표 1. (13,4,1)-configuration
Table. 1 (13,4,1)-configuration

V= { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 }		
X[0]=	Z[0]=	{0, 1, 2, 3 }
X[1]=	Z[1]=	{0, 4, 5, 6 }
X[2]=	Z[2]=	{0, 7, 8, 9 }
X[3]=	Z[3]=	{0, 10, 11, 12 }
Y[0]=	Z[4]=	{1, 4, 7, 10 }
Y[1]=	Z[5]=	{1, 5, 8, 11 }
Y[2]=	Z[6]=	{1, 6, 9, 12 }
Y[3]=	Z[7]=	{2, 4, 8, 12 }
Y[4]=	Z[8]=	{2, 5, 9, 10 }
Y[5]=	Z[9]=	{2, 6, 7, 11 }
Y[6]=	Z[10]=	{3, 4, 9, 11 }
Y[7]=	Z[11]=	{3, 5, 7, 12 }
Y[8]=	Z[12]=	{3, 6, 8, 10 }

이제 이 알고리즘이 $(v, k+1, 1)$ -configuration의 조건을 만족함을 증명하기로 한다.

정의 1. 유한결합구조 Y에서 , 다음과 같이 섹션 S_i를 정의한다. 여기에서 S_i는 k개의 블럭으로 이루어진 집합족이다.

$$\lfloor j/k \rfloor = i \Rightarrow Y[j] \in S_i$$

보조정리 1. Y의 두 원소 $Y[i1, j1]$, $Y[i2, j2]$에 대하여 다음이 성립한다.

$$j1 \neq j2 \Rightarrow Y[i1, j1] \neq Y[i2, j2]$$

증명 : 알고리즘 1-3-(a)에서, $j \geq 1$이면 $X[i,j] = i \times k + j$이 되는데 이는 모든 원소가 서로 다름을 의미한다. 알고리즘 1-3-(b)에서 보는바와 같이 $Y[i,j] = X[j,t]$ 이므로 Y를 구성하는 블럭의 j번째 원소들은 X의 j번째 행으로부터 왔음을 알 수 있다. 그러므로 $j1 \neq j2$이면 $Y[i1, j1]$와 $Y[i2, j2]$는 같을 수 없다.

보조정리 2. 동일한 섹션에 존재하는 k개의 블럭에 대하여 각 섹션의 첫 번째 원소는 모두 동일하고 첫 번째 원소를 뺀 나머지 k^2개의 원소들은 $V - X[0]$과 같다.

증명 : $Y[i,0] = X[0, \lfloor i/k \rfloor + 1]$ 이므로 , 어떤 섹션 s의 k개 블럭의 첫번째 원소는 $X[0, s+1]$을 값으로 갖는다. 그리고 알고리즘 1-3-(b)에 의하면 $Y[i,j] = X[j,t]$,

$t = (i + (j-1) \lfloor i/k \rfloor) \bmod k + 1$인데, 여기에서 k는 소수이므로 t는 1부터 k까지의 값을 가지며 j 또한 1부터 k까지의 값을 가진다. 그러므로 첫번째 원소를 뺀 k^2개의 원소는 $X[1,1]$부터 $X[k,k]$까지의 값을 갖게 되고 이는 $V - X[0]$와 같다.

보조정리 3. 유한결합구조 Y에서 $j \geq 1$일 때 a와 b사이에 다음과 같은 관계가 성립하는 경우에만 $Y[a,j] = Y[b,j]$이다.

$$b = ((a - c(j-1)) \bmod k + k(\lfloor a/k \rfloor + c)) \bmod k^2$$

증명: 알고리즘 1-3-(b)로부터

$b = ((a - c(j-1)) \bmod k + k(\lfloor a/k \rfloor + c)) \bmod k^2$

인 경우에 $Y[a,j] = Y[b,j] = X[j,t]$이 성립함을 증명할 수 있다.

$$\begin{aligned} t &= (b + (j-1) \times \lfloor b/k \rfloor) \bmod k + 1 \\ &= (((a - c(j-1)) \bmod k + k(\lfloor a/k \rfloor + c)) \\ &\quad + (j-1) \lfloor ((a - c(j-1)) \bmod k \\ &\quad + k(\lfloor a/k \rfloor + c))/k \rfloor) \bmod k + 1 \\ &= (((a - c(j-1)) + (j-1) \times (\lfloor a/k \rfloor + c) \bmod k + 1 \\ &= (a + (j-1) \lfloor a/k \rfloor) \bmod k + 1 \end{aligned}$$

위의 전개식으로부터 $Y[a,j]$가 취하는 값 $X[j,t]$와 $Y[b,j]$가 취하는 값 $X[j,t]$가 동일하다는 것을 알 수 있다. 그리고 보조정리 1에서 보인바와 같이 $j1 \neq j2 \Rightarrow Y[i1, j1] \neq Y[i2, j2]$이므로 정의된 조건 이외에 $Y[a,j]$와 $Y[b,j]$가 동일한 값을 갖는 경우는 존재하지 않는다.

여기에서 $Y[a,j]$가 섹션 S_s에 있으면 $Y[b,j]$는 섹션 $S_{(s+c) \bmod k}$에 있다. 그리고 $c \equiv 0 \bmod k$ 인 경우는 $a = b$의 경우이다.

보조정리 4.. V의 모든 원소는 Z에 $k+1$번 나타난다.

증명: 알고리즘 1-3-(a)에 의하면 $X[i,0]=0$이고 $X[i,j]=t,\ t\neq 0,\ j\geq 1$이다. 따라서 원소 0은 X에 $k+1$번 나타난다. 그리고 알고리즘 1-3-(b)에 의하면 $Y[i,j]=X[j,t]$인데 여기에서 t는 0이 될 수 없으므로 Y에는 원소 0이 나타나지 않는다. 그러므로 원소 0은 Z에 $k+1$번 나타난다.

알고리즘 1-3-(a)에 의하면 $v-1$개의 다른 원소 $V-\{0\}$는 X에서 단 한번 나타남을 알 수 있다. 그리고 이 원소들 $V-\{0\}$은 Y에서 k번 나타난다. 왜냐하면, 보조정리 2에 보인바와 같이 $X[0,j],\ 1\leq j\leq k$는 섹션 S_{j-1}에 k번 나타나고 각각의 섹션에는 $V-\{0\}$이 한번씩 나타나기 때문이다.

보조정리 5. V의 두 원소로 이루어진 모든 쌍은 Z에 단 한번 나타난다.

증명: 알고리즘 1-3-(a)에 의하면 X에서 원소 0은 모든 다른 원소들과 1번씩 쌍이 된다. 그리고 보조정리 2에 의해서 $X[0,j],\ 1\leq j\leq k$가 섹션 $j-1$에서 $V-X[0]$과 한번씩 쌍이 됨을 알 수 있다. 그리고 $X[0]$의 원소들끼리는 $X[0]$에서 서로 쌍이 되었다. 또한 X에서 쌍을 만들었던 원소쌍은 Y에서 쌍을 이룰 수 없다. Y는 X와 행열을 교차하여 만들어졌기 때문이다. 이제 Y에 존재하는 $Y[a,j1]$과 $Y[a,j2]$가 다른 블럭에서 쌍이 될 수 없음을 보임으로써 이 정리를 증명할 수 있다. $Y[a,j1],\ Y[a,j2]$의 동일한 쌍이 존재한다고 가정하고 이 쌍이 존재하는 블럭을 구해보기로 하자. 보조정리 3에 의해 $Y[a,j1]$과 같은 값을 갖는 블럭은

$$b1=((a-c(j1-1)) \bmod k + k(\lfloor a/k \rfloor + c)) \bmod k^2$$

이 된다.

그리고 $Y[a,j2]$와 동일한 값을 갖는 블럭은

$$b2=((a-c(j2-1)) \bmod k + k(\lfloor a/k \rfloor + c)) \bmod k^2$$

이 된다.

이 식에서 $j1\neq j2$인 경우에는 $b1=b2$가 될 수 없으므로 $Y[a,j1],\ Y[a,j2]$의 쌍은 더 이상 존재하지 않는다. 그러므로 모든 쌍이 1번씩 나타남을 알 수 있다.

정리 1. 알고리즘 1에 의해 만들어진 Z는 $(v,k+1,1)$-configuration의 조건을 만족한다.

증명: 보조정리 4와 5에 의해 Z는 SBIBD가 되기 위한 조건을 만족하고 있다.

3.2 알고리즘 2 : 분산시스템의 네트워크 토폴로지 설계

분산시스템의 부하분산을 위해 최소의 링크비용과 트래픽 오버헤드를 갖는 네트워크 토폴로지 설계를 위해 $(v,k+1,1)$-configuration을 도입하기로 한다. 어떤 유한결합구조 $Z=\{V,B\}$가 $(v,k+1,1)$-configuration 이고 G를 Z에 대한 이진결합행렬이라 하자. 이때 행렬 G는 그래프 $G=\{V,E\}$의 인접행렬로 변환될 수 있다. 이러한 접근법에 의해 $(v,k+1,1)$-configuration 으로부터 네트워크 토폴로지를 얻을 수 있다.

(1) 알고리즘1에 의해 $(v,k+1,1)$-configuration을 만족하는 유한결합구조 $Z=\{V,B\}$을 생성한다.

(2) Z로 부터 각각의 블럭 i가 원소 i를 포함하도록 $L=\{V,B\}$를 생성한다.

```
L[0] ⇐ Z[0]
for ( i=1 ; i<v ; i=i+1 ) {
      if ( i≤k ) {j ← (i×k+1); t ← Z[j,i];}
      else if ( i% k=1 ) t ← Z[i,0] ;
      else { j ← ⌈i/k⌉ −1 ;  t ← Z[i,j] ;}
       L[t] ← Z[i] }
```

(3) L로부터 그래프 G의 인접행렬 $A(G)$를 생성한다. 여기에서 G는 v개의 프로세서들로 이루어진 네트워크 토폴로지가 된다.

$$a(i,j)=\begin{cases}1 & \text{if } i\neq j \text{ and if } i\in L[j] \text{ or } j\in L[i]\\ 0 & \text{if } i=j \text{ or if } i\notin L[j] \text{ and } j\notin L[i]\end{cases}$$

정리 2. 알고리즘 2에 의해 얻어진 네트워크 G는 v개의 노드와 $v\times k$개의 링크를 가지며 각 노드의 차수가 $2k$인 정규그래프이다.

증명: G는 $(v,k+1,1)$-configuration으로부터 생성되었으므로 노드의 수는 v가 된다. 알고리즘 2-(2)으로부터 L을 구성하는 각각의 블록 $L[i]$는 $k+1$개의 원소 중 한 원소는 i임을 알 수 있다. 알고리즘 2-(3)에 의해 각 노드의 링크가 결정되는데, 식에서 보는바와 같이 노드 i는 $L[i]$ 중 자신을 제외한 k개의 원소와 연결되어 있고 동시에 블럭 $L[j]$ 가 노드 i를 포함하는 경우 노드 j와 연결되어 있다. 그러므로 SBIBD의 정의에 의하여 노드 i는 $2k$개의 링크를 갖는다. 따라서 네트워크의 링크의 수는 $(2k\times v)/2 = vk$임을 알 수 있다.

Ⅳ. $(2k)$-정규그래프상의 부하 분산을 위한 부하상태정보 전파 알고리즘

(1) 네트워크상의 각각의 노드 n은 다음과 같이 집합 S_n과 R_n을 정의한다. 여기에서 S_n은 시간 T_{2t}에 노드 n으로부터 상태정보를 수신하는 n에 인접한 k개의 노드이다. 그리고 R_n은 T_{2t}에 노드 n에 정보를 제공하는 n에 인접한 또다른 k개의 노드집합이다. 시간 T_{2t+1}에 정보전송의 방향을 거꾸로 한다.

$S_n=\{v \mid v\in L[n] \text{ and } n\neq v\}$

$R_n=\{v \mid n\in L[v] \text{ and } n\neq v\}$

(2) 네트워크상의 각각의 노드 n은 다음과 같은 집합족 SF_n과 RF_n을 정의한다.

$SF_n=\{SF_n(i) \mid i\in S_n,\ SF_n(i)=\{\{n\}\cup S_n-\{i\}\}$

$RF_n=\{RF_n(i) \mid i\in R_n,\ RF_n(i)=\{\{n\}\cup R_n-\{i\}\}$

(3) 각 노드 n은 부하상태정보를 갱신하고 노드집합 S_n의 각 원소 i에게 노드집합 $SF_n(i)$ 의 상태정보를 전송한다.

(4) 각 노드 n은 부하상태정보를 갱신하고 노드집합 R_n의 각 원소 i에게 노드집합 $RF_n(i)$ 의 상태정보를 전송한다.

(5) (3)을 반복한다.

정리 3. 알고리즘 3에 의해 각각의 노드는 매 전송 주기마다 k^2개의 노드에 대한 부하상태정보를 얻는다.

증명: 노드 n에 대하여, 노드 n으로부터 T_{2t}에 정보를 수신하는 k개 노드들의 집합을 $S_n=\{s_0,s_1,\ldots,s_{k-1}\}$라 하자. 그러면 $R_n=\{r_0,r_1,\ldots,r_{k-1}\}$는 T_{2t}에 노드 n에 정보를 제공하는 노드집합이 된다. 왜냐하면 r_i는 n을 원소로 갖는 $L[i]$로부터 생성되었기 때문이다. 그러므로 시간 T_{2t}에 r_i는 k개의 노드집합 $SF_{r_i}(n)$ 에 대한 부하상태정보를 노드 n에 전송한다.

그런데 k개의 노드 R_n으로부터 전송되는 정보에는 중복이 존재하지 않는다. 왜냐하면 SBIBD에는 λ개의 공통요소가 존재하는데 $SF_{r_i}(n)$에 존재하는 공통의 요소는 바로 n이기 때문이다. 따라서 노드 n은 시간 T_{2t}에 k^2개의 노드에 대한 부하상태정보를 수신한다. 또한 T_{2t+1}에도 마찬가지로 동작한다.

예를 들면 노드 1을 중심으로 T_{2t}에 전송되는 정보의 흐름은 그림 1.과 같다.

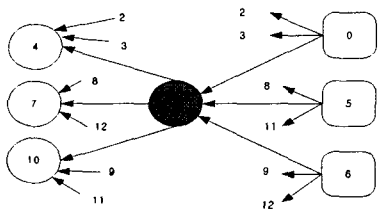

그림 4. 시간 T_{2t}에서 노드 1의 정보 송수신
Fig. 1 Communication of node 1 at T_{2t}

시간 T_{2t+1}에는 정보전송의 방향이 거꾸로 된다. 즉, 각 노드는 $2k$개의 링크를 가지고 각 시점에서 k개의 링크를 통해 부하상태정보를 수신하고 나머지 k개의 링크를 통해 정보를 송신한다.

이때 노드 1의 부하상태정보 수신 상태는 표 2.와 같다.

표 2.. 노드 1의 부하상태정보 수신상태
Table. 2 Received state of node 1

T_{2t}	T_{2t+1}
from 0 => 0, 2, 3	from 4 => 4, 2, 3
from 5 => 5, 8, 11	from 7 => 7, 8, 12
from 6 => 6, 9, 12	from 10 => 10, 9, 11

정리 4. 각 노드에서 수신한 부하상태정보의 도달거리는 2 이내이다.

증명: 노드 n이 노드 r_i로부터 T_{2t}에 수신한 k개의 부하상태정보는 T_{2t-1}에 생성된 r_i에 대한 상태정보와 T_{2t-2}에 생성된 $SF_{r_i}(n)-\{r_i\}$에 대한 상태정보로 구성되어있다. 그러므로 정보도달 거리는 2 이내임을 알 수 있다.

예를 들면 표 2.의 노드 1이 수신한 정보의 도달거리는 그림 2.와 같다.

정리 5. 각각의 노드는 알고리즘 3에 의해 네트워크상의 모든 노드에 대한 상태정보를 수신한다.

증명: 정리 3에 의하면 노드 n은 T_{2t}와 T_{2t+1}에 k개의 노드에 대한 부하상태정보를 수신하지 못한다. 여기에서 우리는 T_{2t}에 수신하지 못한 k개의 정보를 T_{2t+1}에 수신함을 증명함으로써 각각의 노드가 모든 노드에 대한 상태정보를 수신함을 증명하려한다.

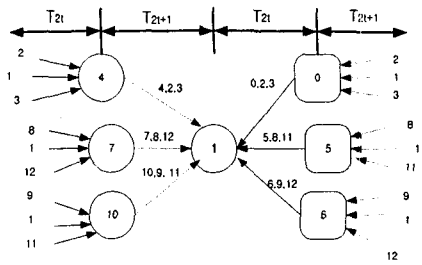

그림 5. 노드 1의 수신상태와 수신된 정보의 도달거리
Fig. 2 Received state and distance of node 1

$S_n=\{s_0,s_1,\ldots,s_k\}$는 $L[n]=\{n,s_0,s_1,\ldots,s_k\}$으로부터 유도된 것이다.

그리고 $R_i=\{r_0,r_1,\ldots,r_k\}$에 의해 다음과 같은 블럭이 존재했음을 알 수 있다.

$$L[r_0]=\{n,r_0,\ldots\}$$
$$L[r_1]=\{n,r_1,\ldots\}$$
$$\vdots$$
$$L[r_k]=\{n,r_k,\ldots\}$$

여기에서 이미 $L[n]$에 n과 s_i의 쌍이 존재하므로 상이 $L[i_j]$에는 s_i가 나올 수 없다. 물론 반대의 경우도 성립한다. 따라서 T_{2t}에 수신하지 못한 k개의 정보를 T_{2t+1}에 수신하게 됨으로써 각 노드는 모든 노드에 대한 부하상태정보를 얻게 됨을 알 수 있다.

V. 결 론

분산시스템에서 비집중화된 방법으로 부하 균형을 유지하기 위해 각각의 노드가 네트워크상의 다른 모든 노드에게 주기적으로 자신의 부하상태정보를 전송하는 데에 필요한 통신비용이 심각한 문제가 될 수 있다. 이 논문에서는 SBIBD를 이용하여 $(2k)$-정규그래프 형태를 갖는 네트워크 토폴로지를 설계하고 이 네트워크 상에서 각각의 노드가 다른 모든 노드의 부하상태정보를 얻기 위해 $O(v\sqrt{(v)})$의 통신량을 갖고 전송시간을 상수 2를 필요로 하는 알고리즘을 제시하였다. 이 알고리즘

은 또한 부하상태정보를 전송을 위해 각각의 노드와 링크가 균등한 트래픽 오버헤드를 갖도록 설계되었다. 또한 이 알고리즘은 최소의 링크비용과 메시지 오버헤드를 보장하는데, 이는 모든 링크가 균등한 트래픽을 가지며 동시에 전송된 정보에 중복이 없다는 사실로부터 증명될 수 있다.

참고 문헌

[1] R.Knuz, The Influence of Different Workload Description on a Heuristic Load Balancing Scheme, IEEE Trans. on Software Eng., Vol. 17, No. 7, pp.725-730, July 1991.

[2] N. G. Shivaratri and P. Krueger, "Load distributing for locally distributed systems", IEEE Comput. pp. 33-44, Dec. 1992.

[3] M. Willebeek-Lemair and A. P. Reeves, "Strategies for dynamic load-balancing on highly parallel computers" , IEEE Trans. Parallel Distrib. Systems, Vol. 4, No. 9, pp. 979-993, 1993.

[4] John A. Stankovic, "Stability and distributed scheduling algorithm", IEEE trans. on Software Engineering, Vol. 11, No. 10, pp.1141-1152, 1985.

[5] Frank C.H. Lin, Robert M. Keller, "the gradient model load balancing method", IEEE trans. on Software Engineering, Vol. 13, No. 11, pp. 32-38, 1987.

[6] Tomas L. Casavant, Jon G.Kuhl, "Effects of response and stability on scheduling in distributed computing systems", IEEE trans. on Software Engineering, Vol. 14, No. 11, pp. 1578-1588, 1988.

[7] S.H.Hosseini, B.Litow, M.Malkawi, "Analysis of a graph coloring based distributed load balancin algorithmg", Journal of Parallel and Distributed Computing, Vol. 10, No. 2, pp. 160-166, 1990.

[8] C.Hui, S.Chanson "Hydrodynamic Load Balancing", IEEE trans. on Parallel and Distributed System, vol.10, No.11, 1999.

[9] M. Y. Wu and W. Shu, A load-balancing algorithm for n-cubes, in "Proc. 1996 International Conference on Parallel Processing," IEEE Computer Society, pp. 148-155 , 1996,

[10] M.Y. Wu , "On Runtime parallel scheduling for processor load balancing", IEEE Trans. Parallel Distrib. Systems , Vol. 8, No. 2, pp. 173-185, 1997.

[11] Kwangwan Nam, Jaewon Seo, "Synchronous Loadbalancing in Hypercube Multicomputers with Faulty Nodes", Journal of Parallel and Distributed Computing 58, pp. 26-43, 1999.

[12] Hwakyung Rim, Juwook Jang, "Method for Maximal Utilization of Idle links for Fast Load Balancing" , Journal of Korea Information Processing Society, Vol. 28, No. 12, 2001.

[13] Sajal K. Das, Daniel J. Harvey, and Rupak Biswas, "Adaptive Load-Balancing Algorithms Using Symmetric Broadcast Networks", NASA Ames Research Center, TR NAS-97-014, May. 1997.

[14] S.K. Das, D.J. Harvey, R. Biswas, "Parallel Processing of Adaptive Meshes with Load Balancing", Vol. 12, No. 12, 2001.

[15] Jae-Cheol Ryou , "A Load Balancing Algorithm in Distributed Computing System", Journal of Korea Information Science Society, Vol. 20, No. 3, 1993.

[16] C.L.Liu, Block designs in Introduction to Combinational Mathamatics, McGraw-Hill, pp359-383, 1968.

[17] M. K. Bennett, Affine and projective geometry, Wiley & Sons, 1995.

저자 소개

김성열(Seong-yeol Kim)

e-mail : green@mogent.net

1994년 조선대학교 전자계산학과 (이학사)

1996년 조선대학교대학원 전자계산학과 (이학석사)

2000년 조선대학교대학원 전자계산학과 (이학박사)

2002년 ~ 현재 울산과학대학 컴퓨터정보학부 전임강사

※ 관심분야 : 정보보안, 분산시스템, 전자상거래, 정보가전, 임베디드시스템