論 文

LAN을 이용한 고가용 시스템의 계층적 제어 알고 리즘

正會員 梁 志 好* 正會員 金 東 吉** 正會員 金 正 善**

Hierarchical Network Management Algorithm for Highly Available System with a LAN

Ji Ho YANG,* Dong Kil KIM,** Jeong Sun KIM** *Regular Members*

要 約 본 논문은 지능망 서비스 제어 시스템에서 시스템의 신뢰도와 고가용성을 증진시키기 위해 LAN을 이용하여 시스템의 중요부분을 부하분담 구조 또는 중복구조로 다중화 시켰을 경우 이를 효율적으로 운영관리하기 위한 망 상태관리 알고리즘인 계층적 제어 알고리즘을 제안한다.

제안된 알고리즘 검증을 위해 Petri-net을 이용하여 알고리즘을 모델링하고, reachabilaty tree를 구성하여 알고리즘이 특정 노드의 장애발생과 무관하게 항상 운영 가능하며 알고리즘이 무결함을 입증하였다.

또한 기존방식과 제안된 방식간의 비교를 위해 망 상태 관리를 위하여 요구되는 메세지 트래픽의 양이 집중제어방식과 거의 유사함을 알수 있었다.

ABSTRACT This paper proposes a hierarchical control algorithm for an efficient network management of a loosely-coupled system which consists of funcational division and duplicating structure with a LAN to enhance reliability and availability.

The algorithm is modeled using the Pertri-net and verified with the reachability tree.

Here, the proposed algorithm is compared with the purely centralized control algorithm and distributed control algorithm in terms of message traffic for the network management.

The result shows that the message traffic related to system performance is as low as centralized control algorithm and the system availability is independent of a specific processor failure.

I. 서 론

통신망 이용자의 요구수준이 높아짐에 따라 이를 수용하기 위하여 나타난 지능망은 새로운 서비스의 도입이 용이한 구조를 가지고 있으며, 지능망 서비스의 종류는 갈수록 다양해질 전망이다. 지능망 서비스를 위한 가장 핵심적 요소로 간주되는 서비스 제어시스템 (SCP : Service Control Point)은 다양한 서비스를 효과적으로

*韓國電子通信研究所 지능망研究室
Electronic and Telecomunication Research Institute
**韓國航空大學 航空電子工學科
Dept. of Avionics Hankuk Aviation University.
論文番號 : 90-53(接受1990. 4. 18)

제어하기 위하여 서비스 수행 교환기가 서비스 호처리를 수행할 수 있도록 서비스 제어 논리와 가입자 데이타 정보를 제공하는 데이타 베이스 시스템이다.[1]

망의 지능 중앙에 집중시킨 서비스 제어 시스템은 3분/년 이하의 down-time과 0.02% 이하의 호(call)서비스 손실을 요구하는 고가용 시스템이며 이와함께 평균 응답시간 500ms, 최대응답시간(99%) 1sec 이하를 요구하는 실시간 시스템이다.[2] 따라서, 가입자에 대한 서비스 제공의 중단을 최소로 하기 위하여 서비스 제어시스템은 시스템의 중요부분들은 부하분담 구조 또는 중복구조로 다중화시켜, 트래픽의 분담처리 및 상대 시스템의 고장시에는 대체시스템에서 서비스를 처리하도록 함으로써 전체 시스템의 신뢰도와 가용도를 높일 수 있는 구조를 갖도록 하는 것이 필수적이며 또한, 범용 실시간 컴퓨터 시스템에 LAN(Local Area Network) 및 각 서브시스템 이중화를 도입함으로써 고가용 구조를 구축하였을 경우 이를 효율적으로 운영 관리하기 위한 망 상태관리용 제어 알고리즘이 필요하다.

본 논문에서는 서비스 제어시스템의 구조와 같이 LAN을 이용하여 다중 프로세서를 연결한 시스템 구조하에서 망의 상태를 효율적으로 관리하기 위한 알고리즘의 연구내용을 기술한다.

Ⅱ. 계층적 제어 알고리즘

Ⅱ.1 알고리즘의 개요

망 상태를 관리하기 위한 방안으로는 한 프로세서에서 운용 관리 기능을 전담하는 집중제어 알고리즘(Centralized control algorithm)과 각 프로세서에서 자체적으로 상태관리를 수행하는 분산제어 알고리즘(Distributed Control Algorithm) 을 고려할 수 있다.

기존에 널리 사용되고 있는 기본적인 집중제어방식의 장점은 망 상태관리를 위해 필요한 통신메세지의 양이 적고, 상태관리를 전담하는 특정 프로세서를 제외한 나머지 프로세서에 부가되는 오버헤드가 주 업무에 비하여 무시할 수 있다는 점이나, 상태관리를 전담하는 프로세서에 장애가 발생하였을 경우 망의 상태를 유지할 수 없다는 문제점을 안고 있다. 또한 분산 제어방식의 경우 망 상태관리를 모든 프로세서에서 독자적으로 수행함으로써 특정 프로세서의 장애와 무관하게 상태 유지가 가능하나, LAN상의 트래픽이 $O(n^2)$으로 증가하고 각 프로세서에 부가되는 오버헤드가 주 업무에 영향을 미칠 수 있다는 단점이 있다.

이와 같은 각 방식들의 단점을 보완하면서 장점을 취한 알고리즘이 계층적 알고리즘이다. 계층적 제어 알고리즘은 몇개의 프로세서들을 그룹으로 묶고, 그룹들을 트리 구조로 구성하여 각 그룹내에서는집중 제어방식으로 상태를 관리하고, 트리구조상에서 차하위 프로세서들이 차상위프로세서를 감시하기 위해 차등 타임아웃 값을 할당받아 상태를 감시하는 분산 제어방식을 병행하여 운영하는 알고리즘이다.

LAN상에 프로세서들의 상태를 유지하기위한 계층적 제어 알고리즘은 다음의 두가지 룰(Rule)을 상황에 따라 적응시켜가며 논리적 상태 관리 구조인 트리구조를 재구성한다.

1.1 그룹단위 집중화 제어방식

Parent가 child의 장애를 탐지하고, 이를 처리하기위한 방식이다. 그룹내에서는 parent가 child들에게 child-test 메세지를 주기적으로 방송(broadcast)하고 child들로 부터 긍정 응답 메세지인 child-OK가 수신되기를 기다린다. 일정기간 경과후에도 메세지가 수신되지 않으면(acknowledgement가 없을 경우) 해당 child에서 장애가 발생하였다고 판단하고 그룹내에 장애 발생을 방송한다. 또한 망내의 방송을 위해 자신의 parent에게 이 사실을 통보하며, 장애 발생을 통보받은 parent는 그룹내의 child들과 parent에게 이를 방송한다. 트리 구조에 따라 이 과정을 진행함으로써 모든 노드에서 프로세서의 장애 발생을 전달받게 된다.

장애노드가 parent로서의 역할을 수행하고

있었을 경우 장애를 감지한 노드는 해당 그룹내의 특정 노드를 지정하여 그 그룹의 parent로 승격시키고, new parent는 승격 사실을 방송한다. new parent로 승격됨에 따라 자신의 parent 자리를 이탈하게 되면 위와 같은 방법으로 child들 중 new parent를 선정하는 작업을 계속한다.

1.2 Parent 장애 처리를 위한 분산 제어방식

Child들이 parent의 장애를 감지하고 처리할수 있도록 하는 방식이다.

루트 노드에 장애가 발생되면 위의 집중화 제어방식으로는 감지할 수 없다. 이 문제를 해결하기 위하여 그룹내의 노드들에게 parent를 감시하는 타임아웃 값을 서로 다르게 할당하며 이 값은 노드마다 고유하게 지정된다.

Child들이 parent로 부터 메세지 수신시 타이머를 reset 시키고, 긍정응답 메세지 송신한 후 타이머를 구동시킨다. 일정시간이 경과된 후에도 parent로 부터 메세지 전송이 없으면 acknowledgement가 없을 경우 가장 작은 값을 지정받은 child가 parent의 장애를 감지하고 new parent로 승격한다. 장애발생 상황을 그룹의 노드들과 child들에게 방송한 후 new parent 발생을 트리 구조를 따라 체계적으로 망내에 방송한다.

Ⅱ. 차등 타임아웃 분배 알고리즘

Child 프로세서가 parent 프로세서를 감시하기 위한 PMT(Parent Monitoring Timer)값을 결정하는 알고리즘은 트리 구조상에서 초기의 프로세서의 위치만 결정되면 프로세서의 레벨이 승격될 때마다 PMT의 값을 계속 추산할 수 있는 알고리즘이다. 트리구조로부터 프로세서의 위치는 레벨 식별번호, 그룹내 child의 순서 번호로서 구분된다. PMT의 값은 종국적으로 child의 순서번호에 의하여 결정된다.

모든 그룹내의 child들이 갖을 수 있는 PMT 값의 범위는 다음과 같이 나타낼 수 있다.

N : 한 그룹을 구성하는 child의 수
G_j : j번째 레벨에서의 그룹 순서번호
C_j : j번째 레벨에서의 child의 순서번호
t_0 : 1st child의 PMT 값
Δt : PMT 값들 사이의 간격 ($=t_n-t_{n-1}$)

t_0와 Δt는 네트워크의 규모와 요구되는 신뢰성에 의해 조정한다.

i번째 레벨에서 순서번호가 k인 노드의 PMT는

$$PMT(i, G_i, C_i) \quad 1 < C_i < N \tag{1}$$

와 같이 정의한다. 모든 그룹의 k번째 노드가 갖는 PMT 값은 동일하므로

$PMT(i, G_i, C_i)=PMT(C_i)$라고 할수 있다.

Child들이 갖는 PMT중 최소의 값을 t_0라 하면

$$t_0=PMT(1)=PMT(i,G_i,1) \tag{2}$$

또한, PMT들간의 시간 간격은

$$\Delta t=PMT(j)-PMT(j-1) \tag{3}$$

이므로 PMT의 최대값은

$$t_{max}=t_0+(N-1)\Delta t \tag{4}$$

가 된다.

프로세서가 leaf 노드에 존재할 경우 초기 위치를 레벨 1, 그룹 순서번호, j, child 순서번호 k인 노드라 할때

$$PMT(1,j,k)=PMT(k)=t_0+(k-1)*\Delta t \tag{5}$$

로 PMT를 구할 수 있다. 만일 이 노드의 parent가 장애가 발생하여 PMT 타임아웃이 발생하면 k번째 노드는 parent의 위치로 이동하게 된다. 이때 PMT의 값은 parent가 갖고 있던 PMT를 계승하게 된다.

이를 구하기 위해 먼저 해당 노드가 parent로 승격시 위치하게 되는 차상위 그룹의 그룹 순서번호를 구해야 한다.

$$G_2=j/n \quad (6)$$

$$G_3=[G/N]=[([j/N])/N]\simeq[j/N^2] \quad (7)$$

$$G_k=[G_{k-1}/N]\simeq[j/N^{k-1}] \quad (8)$$

현재 노드의 순서번호와 그룹 순서번호로 부터 노드가 레벨 k로 승격시 부여 받는 child 순서번호는

$$C_k= \begin{array}{l} \mathrm{mod}(G_{k-1},N)=\mathrm{mod}([j/n^{k-2}],N),\ \text{when} \\ G_{k-1}\neq aN,\ a \text{ is integer } N \\ \text{when } G=aN \end{array} \quad (9)$$

된다.

그러므로 k번째 레벨로 승격되었을 때 PMT는

$$PMT(k, G_k, C_k)$$
$$=PMT\ (K,[j/N^{k-1}],\ \mathrm{mod}([j/N^{k-2}],N)$$

$$=t_0+\{\mathrm{mod}\ (j/N^{k-2}],\ N)-1\}\ \Delta t \quad (10)$$

와 같다.

Ⅲ. 알고리즘 검증

1. 계층적 제어 알고리즘의 모델링

계층적 알고리즘을 Petri-net을 이용하여 표현하고 분석한다. Petri-net은 알고리즘을 기술하는데 유용하며, 또한 이를 통해 알고리즘의 검증이 가능하다.[3X4]

트리 구조상에서 노드들은 child 또는 parent의 상태로 존재할 수 있으며, 예외 상태로서 장애발생시 수리를 기다리는 장애상태가 있다. 그림 1은 Petri-net을 사용하여 알고리즘을 표현한 것으로 앞에서 기술한 바와 같이 다음의 4가지 경우에 대해 상태 동작 과정을 고려할 수 있다.

1.1 Parent가 child들로부터 ack.를 정상적으로 수신한 경우(t_1)

Ready to send가 set되면 (P_1) 메세지가 전송되고 CMT timer가 구동되며(t_2) parent는 wait for ack. 상태에 놓인다(P_2). 메세지가 전송되고 child가 ready to receive 상태(P_6)이면 child는 메세지를 수신하고 (t_6), ack.를 parent에게 전송한다(t_7). Ack를 기다리던 parent(P_2)가 ack.를 수신하면(t_3) parent는 다시 이 과정을 반복한다.

1.2 CMT의 타임아웃 발생(t_{10}) Child의 장애발생을 parent가 감지한 경우

Child에 장애가 발생(P_{11})하면 child로 부터ack.를 기다리며 CMT를 작동(P_2) 시키던 parent에서 time expire가 발생(t_{10})한다. Parent는 장애발생을 방송하고 장애가 발생한 노드의 child들중 정상 동작중인 한 노드를 선택하여 new parent로 승격시킨다.(t_{11})

1.3 PMT의 타임아웃 발생(t_8) : Parent의 장애발생을 child가 감지할 경우

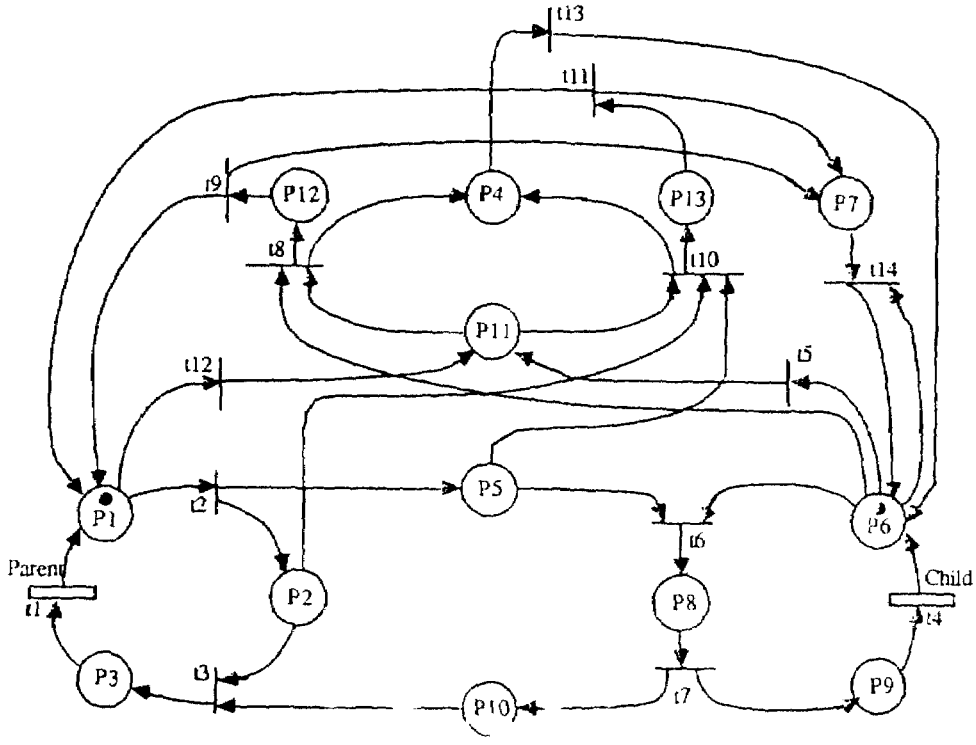
Parent로 부터 장애가 발생(P_{11})하면 parent로 부터 메세지를 기다리며 PMT를 작동시키던 child에서(P_6) time expire가 발생(t_8)한다. Child는 장애발생을 방송하고 자신이 new parent로 승격한다.(t_9) 또한 자신의 위치변경으로 인해 비워진 자리를 child들 중 하나를 선정하여 대치시킨다.(t_{14})

1.4 장애노드가 복구된 경우(t_{13})

장애가 발생하여 수리를 기다리고 있던 노드(P_4)가 복구되면 자신의 장애 복구 사실을 방송하고(t_{13}) leaf node로 연결된다.

2. Reachability Tree

Reachability tree는 Petri-net의 reachability



t_1 parent
t_2 CMT restart & send *msg.*
t_3 receive ack.
t_4 child
t_5 child fail
t_6 receive message
t_7 send ack.
t_8 PMT expire
t_9 broadcast failure & I'm new parent
t_{10} CMP expire
t_{11} broadcast failure
t_{12} parent fail
t_{13} recovery
t_{14} assign new child

p_1 ready to send
p_2 wait for ack. & CMT checking
p_3 ack. received & CMT reset
p_4 wait for recovery
p_5 *transmit msg.*
p_6 ready to msg. PMT checking
p_7 child absent
p_8 message received
p_9 ack. sended
p_{10} transmit ack.
p_{11} failure
p_{12} parent failure detection
p_{13} chlid failure detection

그림 1. 계층적 알고리즘의 모델

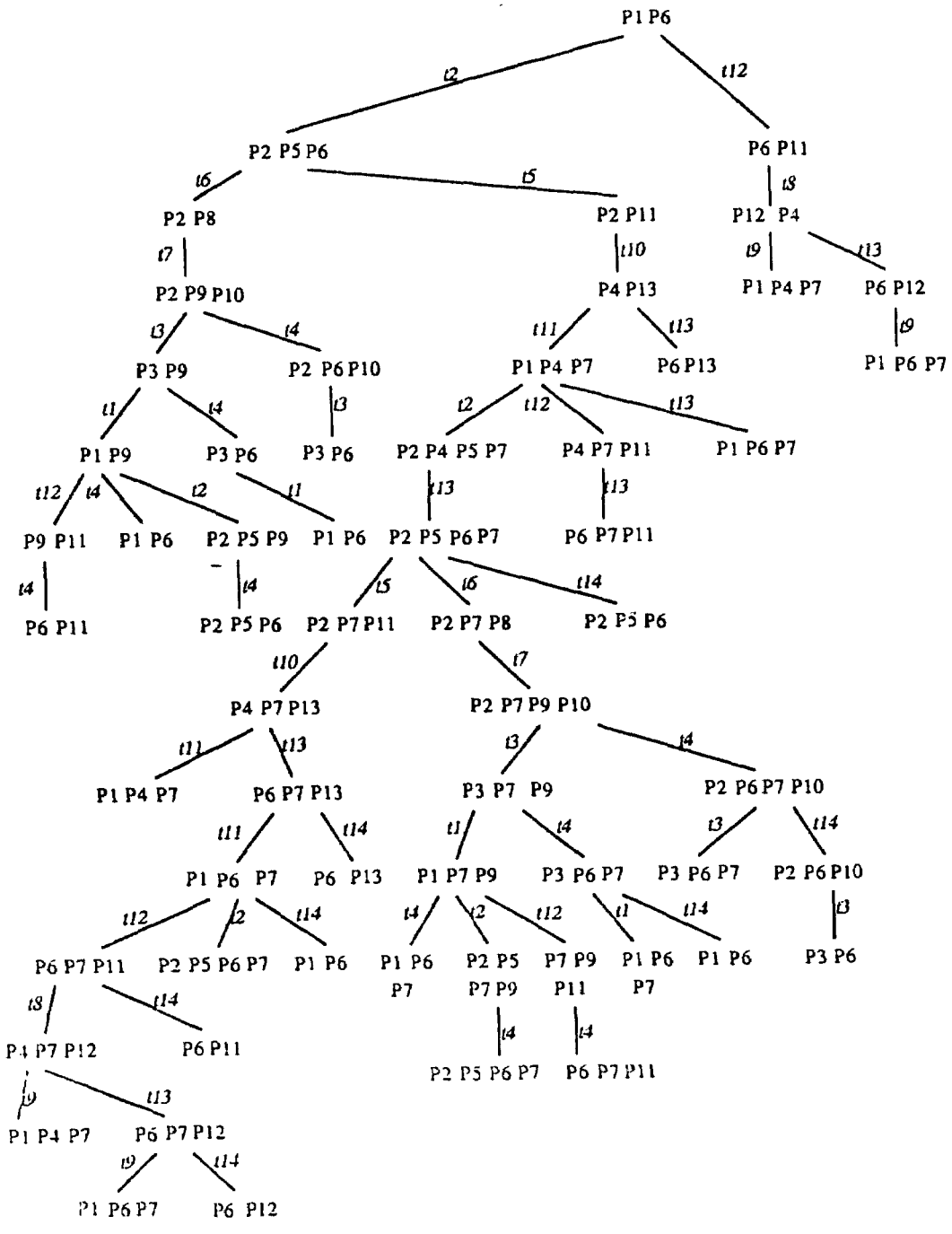


그림 2. 계층적 알고리즘의 Reachability Tree

set을 표현한다. 초기 마킹은 parent의 초기상태를 나타내는 P_1과 child의 초기상태인 P_6에 토큰을 하나씩 제공한 상태이다. 장소에 놓여진 토큰이 천이를 점화 가능하도록 할때마다 점화에 의해 토큰을 다른 장소에 이동해 가면서 토큰의 위치를 표시한 reachability tree는 그림 2와 같다.[5]

트리상의 모든 노드가 terminal, duplicate 또는 interior 노드로 분류될 때까지 reachability tree의 구성 작업이 계속된다.

3. 알고리즘 분석

계층적 제어 알고리즘을 Petri-net를 이용하여 모델링한 것은 분석하는 과정에서 다음과 같은 가정이 전개된다.

가정 1 : 장애가 발생한 프로세서들은 망 상태구조인 트리 구조로 부터 제거되며 이는 child 노드의 수를 감소시키는 결과를 초래한다. 이와 같은 상태는 Petri-net에서 한 장소로 표현 가능하며 제거된 child의 수가 추가될 때마다 이 장소에는 토큰이 증가하게 된다. 토큰의 수가 임계값 N을 초과할 때 전체 다운으로 간주할 수 있다.

가정 2 : Child의 자리가 비어있을 경우 복구에 의해 새로운 child가 추가되면 Petri-net는 망 상태 관리 구조인 트리 구조중 최하위 레벨에서의 알고리즘 운용 상태를 나타내게 된다.

가정 3 : Parent의 장애발생으로 인해 child가 parent의 위치로 이동하였을 경우 Petri-net 은 차상위 레벨의 동작을 나타낸다.

3.1 Liveness, Deadlock

Dead 마킹은 더 이상 점화될 천이가 없는 마킹으로 terminal node라고도 하는데 이와 같은 노드가 나타나면 알고리즘상에 deadlock이 존재함을 의미한다.

Reachability tree에서 모든 leaf node는 interior node나 duplicated node이며 duplicated node 역시 초기 마킹 상태에 도달할 수 있는 경로가 존재함으로 알고리즘 deadlock이 존재하지 않음을 입증할 수 있다.

Duplicated 마킹이란 트리의 앞단에서 나타났던 마킹이 다시 나타난 것을 말한다. 이 경우 더 이상의 트리의 확장은 불필요한 작업이므로 duplicated node는 leaf node가 된다.

3.2. Boundedness, Safeness

모든 장소에서 토큰의 수가 k-bound 되었는가의 여부를 살펴본다. 토큰이 P_1, P_6, P_7에 있을때 t_{12}가 점화된 후 t_8,t_9가 연속적으로 점화되면 P_7에 토큰이 누적된다. 이는 가정 1에서 언급한 바와 같이 장애가 발생한 child 노드들이 계속 증가되어 망 상태관리 구조인 트리 구조에서 제거되고 있는 노드의 수가 늘어나고 있음을 나타낸다. 그러므로 P_7에서 토큰이 증가하여 규정한 임계치 N을 초과하면, 즉 시스템을 구성하고 있는 프로세서중 N개에서 장애가 발생하면 시스템 다운으로 간주한다고 볼때 reachability tree가 k-bound 되어 있지 않더라도 논리적 해석에 의해 정상상태로 처리할 수 있다. 즉 Petri-net 의 reachability 검증 여부에 있어 Petri-net가 무결하다는 것과 알고리즘의 해석은 구분되어야 한다.

Ⅳ. 알고리즘의 비교

기존 방식과 제안한 방식간의 비교를 위해, 망상태 관리를 위하여 요구되는 메세지 트래픽 양의 측면에서 비교해본다. 계층적 제어방식에서 한 주기동안 망의 상태를 파악하기 위한 트래픽의 양은 전체 프로세서가 정상적인 경우 그룹내의 child들을 감시하기 위해 필요한 트래픽에서 장애가 발생하여 감시 메세지를 송출하지 않는 프로세서들의 트래픽을 제외시키고 또한 장애발생 사실을 망에 전달하기 위해 사용되는 트래픽과 트리구조를 유지하기 위해 사용되는 트래픽이

추가된다.

m : group 단위의 child 수
n : 전체 프로세서의 수
r : tree 구조의 layer 수
k_1 : k개 노드의 장애 발생시 leaf node에서 발생가능한 장애 노드의 추정치
k_n : k개 노드의 장애 발생시 2 r-1 layer에서 발생가능한 장애 노드의 추정치
T_n : 계층적 제어 방식으로 운용시 한 주기동안 발생하는 트래픽의 양

$$r=\mathrm{Log}_m(1-n-nm) \quad (11)$$

$$k_1=m^{r-1}\,k/n \quad (12)$$

$$k_n=(1-m^{r-1})k/n(1-m) \quad (13)$$

$$T_n=2(m+m^2+\cdots+m^{r-1})-2(k_nm+k)+2(n-k-1)k+(\sum_{i=1}^{r-1}(r-1)(n-k-1)m^{i-1})/n$$
$$=2\{(1-m^r)/(1-m)-1\}-2\{(1-m^{r-1})mk/(n(1-m)+k\}+2(n-k-1)k+(\sum_{i=1}^{r-1}(r-1)(n-k-1)m^{i-1})/n \quad (14)$$

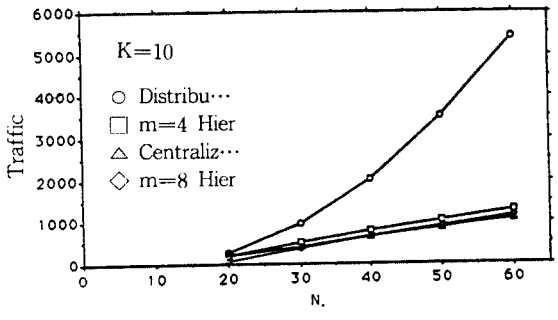


그림 3. 프로세서 수의 증가에 따른 트래픽 변화(방식간 비교)

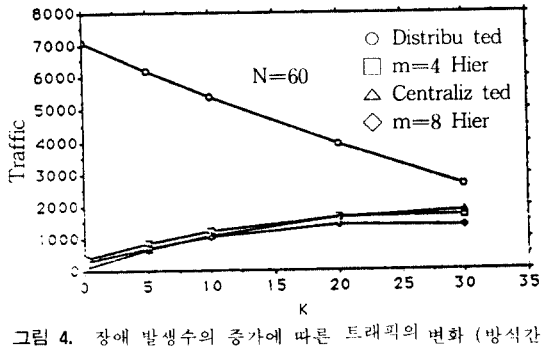


그림 4. 장애 발생수의 증가에 따른 트래픽의 변화 (방식간 비교)

LAN에 연결된 프로세서의 수가 증가함에 따라 변화하는 트래픽의 양을 각 방식간에 비교한 그래프가 그림 3과 같다. 장애 프로세서의 수를 10으로 가정하고 프로세서의 수를 20부터 60까지 증가 시키면서 변화하는 트래픽의 양을 추적한다.

집중 제어방식과 계층적 제어방식은 거의 동일하게 트래픽이 증가함을 알 수 있다. 또한 프로세서의 수를 60개로 고정하고, 장애의 수를 0부터 30까지 증가시키면서 변화하는 트래픽의 양을 그림 4에 나타내었다. 장애 프로세서의 수가 증가할 수록 분산 제어방식의 경우는 트래픽이 감소하고, 집중 제어방식과 계층적 제어방식에서는 안정된 증가를 보여준다. 표 1과 표 2는 그림 3,4에 대한 결과표이다.

표 1. 프로세서 수의 증가에 따른 트래픽 변화 (방식간 비교)

k	n	distri-traffic	central-traffic	hier-traffic	
				m= 4	m=8
10	20	280	208	159	-
10	30	960	428	409	181
10	40	2040	648	698	500
10	50	3520	868	964	779
10	60	5400	1088	1217	1038

표 2. 장애발생 수의 증가에 대한 트래픽 변화 (방식간 비교)

k	n	distri-traffic	central-traffic	hier-traffic	
				m=2	m=4
0	60	7080	118	481	375
5	60	6215	653	985	846
10	60	5400	1088	1389	1217
20	60	3920	1658	1898	1659
30	60	2640	1828	2006	1702

Ⅵ. 결 론

지금까지 LAN을 이용하여 다중연결된 프로세서들의 장애상태 관리를 위한 방안으로서 제안한

계층적 제어 알고리즘에 대하여 기술하였다.

기본적인 집중 제어방식과 분산 제어방식과의 비교를 통해 제안한 알고리즘이 네트워크의 성능과 직결되는 트래픽의 양의 분배에 있어 집중 제어방식과 유사한 유형을 갖음을 알 수 있다. 또한 계층적 제어 알고리즘을 페트리 네트를 이용하여 모델링하고, 무결성을 검증하였다.

알고리즘의 검증 방법으로서 reachability tree를 구성하여 liveness와 boundedness를 입증함으로서 알고리즘이 집중 제어방식과는 달리 특정 프로세서의 장애와 무관하게 항상 운용 가능하며, 알고리즘상에 루프가 존재하지 않음을 보였다.

그러나 이 알고리즘은 프로세서의 재원 할당(resource allocation) 측면은 배제시키고 단지 네트워크상의 프로세서들의 상태만을 관리하는 측면에서 고려되었다. 프로세서 단위로 망의 재원이 중복되어 있거나 한 프로세서에서 복수개의 재원을 처리할 수 있는 네트워크일 경우, 특정 재원을 관장하고 있는 모든 프로세서들에게서 장애가 발생한다면 대부분의 프로세서가 정상적이라고 할지라도 망 차원의 기능동작은 실패하였다고 보아야 한다.

이와 같은 관점을 고려할 때 프로세서의 장애 상태 관리 알고리즘에 추가적으로 재원을 관리하고 재 할당하는 알고리즘을 병행시켜 운용할때 좀더 효율적인 망의 관리가 수행될 수 있다. 추후 망의 재원 관리와 장애 관리를 병행한 알고리즘의 연구가 계속되어야 한다.

참 고 문 헌

1. Frank J. Weisser at el,"The Intelligent Network and Forward-Looking Technology", Bellsouth, IEEE Communications Magazine, 1988. 12.
2. Technical Advisory TA-TSY 000029, "Service Control Point Node Generic Requirements", Bellcore, 1986. 12.
3. Jean-Michel, "REBUS, A Fault-Tolerent Distributed System for Industial Realtime Control", IEEE Tran. Comp. C-31, No.7, July 1982.
4. Wenwey Heush, "A Network Architecture for Reliable Distributed Computing", IEEE Network, Vol.2, No. 4, July 1988.
5. Amir Herzberg, "Network Management in the Presence of Faults", Proc, ICCC, pp. 512-517, 1988.
6. James L. Petrson, "Petri Nets", Computer surveys, Vol.9, No.3, Sep. 1988.
7. Aimo A. Torn, "Simulation Graphs : A General Tool for Modeling Simulation Design", Simulation, Dec. 1981.
8. Kornel Terplan, "Communication Networks Management", Prentice-Hall, 1987.
9. Tadao Murata, "Modeling and Analysis of Concurrent System", Handbook of Software Engineering.
10. Nancy G.Leveson, "Safety Analysis Using Petri Nets", IEEE Trans. on S/W Engineering, Vol. SE-13, No.3, 1987. 5.
11. Tilak Agerwala, "Putting Petri Nets to Work", IEEE Computer, 1979. 12.

梁 志 好(Ji Ho YANG) 正會員

1961年 5月 7日生

1984年 2月：韓國航空大學 電子工學科 學士

1990年 2月：韓國航空大學 電子工學科 碩士

1984年 3月：韓國電子通信研究所 入所

1990年 6月現在：지능망시스템研究室研究員

金 東 吉(Dong Kil KIM) 正會員

1978年 3月～1982年 2月：韓國航空大學

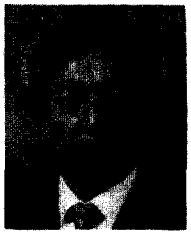
1982年 3月～1984年 2月：韓國航空大學 大學院(碩士)

1988年 3月～現在：韓國航空大學大學院(博士)

1982年12月～1986年10月：ETRI研究員

1986年10月～1989年10月：대성쎄타기계 技術理事

1989年10月～現在：진성전자 代表



金 正 善(Jeong Sun KIM) 正會員

1941年 5 月 5 日生

1965年～1990年 現在：航空大學 敎授

1988年～1990年：電子工學科 學科長

1990年 2 月～1990年 6 月：電子計算所長