論文 94-19-8-16

분산구조형 교환시스템의 과부하 제어

正會員 程 炫 必* 正會員 林 錫 鳩** 正會員 李 潤 鉉***

Overload Control of the Distributed Architecture Switching System

Hyon Pil Jung*, Seogku Lim**, Yun Hyun Lee*** *Regular Members*

要 約

본 논문에서는 축적프로그램 제어시스템이 분산구조형인 경우 이에 적합한 새로운 과부하 제어 방법을 제안한다. 분산구조에서 호 처리 기능의 대부분은 가입자 제어 프로세서(SP)에 분산되어 있지만, 번호 번역과 루팅제어와 같은 몇가지 기능들은 중앙 프로세서(CP)에 집중되어 있으므로 중앙 프로세서의 성능은 교환시스템에서 매우 중요하다. 그러므로 과부하시 CP의 적절한 동작을 보증하는 과부하 제어 방법이 필요하다. 제안한 과부하제어 방식의 성능을 평가하기 위하여 시뮬레이션을 수행하였으며, 이를 통하여 제안한 과부하 제어 방법이 매우 효율적임을 입증하였다.

ABSTRACT

In this paper, a new overload control scheme is proposed for a stored program control system. The proposed control scheme is fit for a distributed architecture. In a distributed architecture, most of the call control functions are distributed to many subscriber control processors(SP) and some functions as number translation and routing control are centralized to central processors(CP). Therefore the performance of the CP is critical to the switching system. We needed a control scheme that ensures the proper operation of the CP in overload condition. We conducted lots of simulation experiments to evaluate the performance of the proposed overload control scheme and concluded that the proposed overload control scheme is very effective for a distributed architecture.

* 조선대학교 병설 공업 전문대학 전자통신과
** 주성전문대학 전자과
*** 한국 항공대학교 통신정보공학과

論文番號 : 94110
接受日字 : 1994年 4月 18日

I. 서 론

교환기에 시도되는 호의 수가 계속적으로 교환기의 최대 호처리용량을 초과해서 입력되는 경우 교환

기는 과부하 상태에 이르게 되어 결국 교환기가 전혀 호를 처리하지 못하는 상태에까지 이를 수도 있다. 따라서 과부하 상태에 도달하더라도 위와 같은 현상의 발생을 사전에 방지함으로써 교환기가 지속적으로 호를 처리할 수 있도록 하는 과부하 제어기능이 필요하다.

축적프로그램 제어(SPC)시스템에서의 과부하 제어 방법들은 그 대상 시스템의 구조가 중앙집중구조인지 또는 분산구조인지에 따라 다르며, 또한 시도되는 호의 수나 큐의 길이 또는 프로세서 점유율 등과 같이 과부하 검출 및 제어 대상으로서 사용되는 파라메타들에 의해서도 다를 것이다.

과부하 제어 방법으로는 중앙집중구조와 분산구조에서의 방법등이 제시되어 있는 바[1][2][3][4], 분산구조형에서는 특히 분산구조를 고려한 효율적인 과부하 제어 방법이 필요하므로 이를 위해서 과부하시 시간구간을 제어구간과 조정구간으로 나누어서 조정구간에서는 제어파라메타 값을 측정하여 시스템의 상태를 파악한 후 이를 토대로 다음 조정구간내의 각 제어구간동안에 교환시스템에 입력될 수 있는 호의 수를 결정하는 방법을 사용한다.[2][3][5].

분산구조형 교환기는 기능 및 부하가 분산된 형태의 시스템이며, 호처리 기능의 대부분은 가입자 제어 프로세서(Subscriber control Processor, 이하 SP라 칭함)에 분산 수용되며, 시스템 구조상 중앙에 위치한 프로세서(Central Processor, 이하 CP라 칭함)에서는 번호번역, 루팅제어, 스위치 연결 및 해제 등과 같은 기능들을 수행하는데, 이러한 분산구조에서는 과부하 상태가 SP보다는 CP에서 발생할 확률이 높으며 시스템 전체에 미치는 영향도 크다. 따라서 본 논문에서는 모든 SP들이 과부하 상태가 아닌데도 CP가 과부하 상태가 되는 경우 제어 파라메타로서 입력되는 시도호의 수와 프로세서 점유율을 사용한 효율적인 과부하 제어 방법을 제안하며, 이의 타당성을 시뮬레이션에 의하여 입증하였다.

Ⅱ. 분산구조에서의 과부하 제어 알고리즘의 개념

과부하 제어란 교환시스템이 과부하 상태에 도달할 때 시스템에 들어오는 호중의 일부를 사전에 조절함으로서 시스템내의 프로세서 점유율을 원하는 수준(이하 목표치라 칭함)으로 유지시키고 또한 프로세서 입력 큐에서의 메시지 대기시간의 평균 및 분산을 최소화하여 일단 받아들여진 호들에 대해서는 적절한 지연시간을 제공함으로서 시스템의 호처리 용량을 일정 수준 유지하는데 있다.

분산 수용된 SP들은 과부하 상태가 아닌데도 CP가 과부하 상태인 경우 과부하 제어의 목적은 CP의 점유율을 목표치 수준으로 유지시키고 또한 프로세서 입력 큐에서의 대기시간의 평균및 분산을 최소화하는 데 있다. 프로세서 점유율의 목표치는 수율(throughput)과 서비스 기준(GOS) 및 지연시간 등을 고려해서 결정되어야 하지만 여기서는 90%를 CP 점유율의 목표치로 가정한다.

분산구조를 갖는 교환시스템에서는 그림 1에서 보는 바와 같이 호의 기각이 SP에서만 이루어 진다고 가정한다. 이는 CP에서 호를 기각하는 경우 이미 서비스를 수행한 SP의 자원이 낭비되기 때문이다[6].

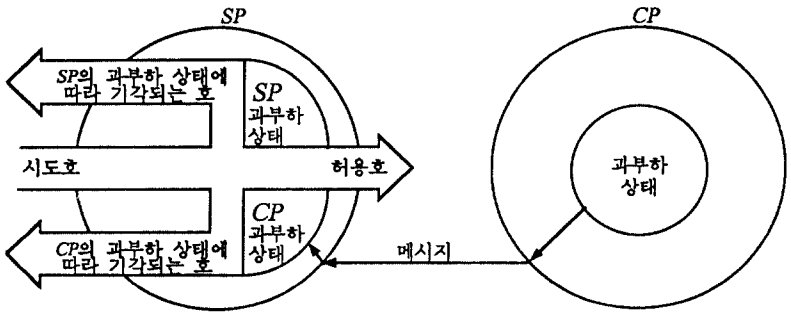

그림 1. 과부하 제어 알고리즘 개념
Fig. 1. Algorithm Concept of the Overload Control

따라서 CP는 과부하 상태가 되면 자신의 상태를 모든 SP에게 메시지로 통보하며, SP는 CP의 점유율이 90%에 가깝게 유지되도록 시도되는 호의 일부를 기각한다. 한편 CP는 과부하 상태가 아닌데도 SP가 과부하 상태가 되었을때는 SP가 독자적인 과부하 제어 방법에 따라 시도되는 호의 일부를 기각한다.

그러나 CP가 과부하 상태를 검출한 시점과 이를 SP에게 메시지로 통보한 후 SP의 과부하 제어 로직에 의해 제어된 호들이 CP에 도달하는 시점 사이에는 시간지연(time lag) 현상이 존재한다. 이러한 시간지연 효과를 감소시키기 위하여 시간축을 균등한 구간으로 분할하고 각 시간구간이 끝나는 시점에서 다음 시간구간에서의 바람직한 CP와 SP 점유율의 목표치를 예측한다.

본 논문에서 제안하는 과부하 제어방법은 교환시스템에 입력되는 시도호의 수와 프로세서 점유율에 기초하여 동작한다. 과부하시 제어방법은 교환시스템에 입력되는 시도호의 수와 프로세서 점유율에 기초하여 동작한다. 과부하시 프로세서 점유율을 목표치 수준으로 유지시키기 위하여 시스템에 입력되는 호들은 표.1과 같은 제어레벨(L)에 의하여 제어되며[3], 제어레벨 수를 12개로 정한 이유는 시뮬레이션 수행 결과 트래픽이 100%~500%에서 가장 효율적으로 제어가 이루어지기 때문이다. 제어레벨이 L=3이면 입력되는 호중 74.1%만이 시스템으로 들어오며 25.9%는 기각된다. 호의 기각은 SP에서만 이루어지므로 제어레벨은 SP에만 존재한다.

제어레벨은 제어구간은 끝나는 시점마다 변화되는데 제어구간 동안 실제로 프로세서에 들어온 호의 수(n)와 제어구간 동안 프로세서가 받아들이고자 하는 호의 수(n^*)를 비교하여 다음과 같이 결정한다.

$$L=\min[m,\ L+1] \quad \text{if } n>n^*$$
$$\quad =\max[0,\ L-1] \quad otherwise \qquad (1)$$

여기서 m은 가장 높은 제어레벨을 의미하는데 표.1에서는 m=11이다.

n^*은 조정구간이 끝나는 시점마다 변화되며, 조정구간 동안의 프로세서 점유율의 함수로 결정된다. 조정구간의 길이는 제어구간의 정수배로 이루어지는데 그 길이는 너무 길지도 또는 짧지도 않아야 한다. 따라서 본 논문에서는 프로세서 점유율을 예측하는 조정구간이 과부하 제어를 수행하는 제어구간의 4배가 되도록 설정하였다.

n이 과부하 시작 표시가 n_S보다 크면 과부하 제어가 시작되며 n^*는 초기치 n_I로 초기화 된다. 과부하 시작 표시기 n_S는 일시적인 호의 폭주로 인해 과부하 제어가 성급하게 시작될 확률이 0.01% 이하가 되도록 높은 값으로 설정하였다. 즉 호의 도착이 포아송 분포를 따른다고 가정하여 (2)식을 만족하는 n_S를 과부하 제어 시작 표시기로 정한다[3]. 여기서 t는 제어구간을 의미하며, k는 n_I의 값을 갖는다.

$$\sum_{k=n_S}^{\infty} \frac{(\lambda t)^k}{k!} e^{-\lambda t} \le 10^{-4} \qquad (2)$$

여기서 n_I는 프로세서 점유율이 90%일때 프로세서에 입력되는 호의 수이다.

또한 과부하 제어가 시작된 후 모든 SP의 제어레벨이 L=0인 상태가 6번 이상 계속되면 과부하 상태를 종료시킨다. 과부하 상태 종료후 n^*는 초기치 n_I로 다시 초기화 된다.

표 1. 과부하 제어 레벨
table. 1. Overload Control Level

	호시도 백분율(%)	
제어레벨	허용율	기각율
0	100.0	0.0
1	91.4	8.6
2	82.7	17.3
3	74.1	25.9
4	65.5	34.5
5	56.8	43.2
6	48.2	51.8
7	39.5	60.5
8	30.9	69.1
9	22.3	77.7
10	13.6	86.4
11	5.0	95.0

Ⅲ. 분산구조 교환시스템의 과부하 제어 알고리즘

그림 2는 본 논문에서 제안한 분산구조를 갖는 교환시스템의 과부하 제어 알고리즘의 흐름도를 나타낸다. 과부하 상태가 시작되면 모든 SP들의 제어레벨을 최대치 11로 설정하고, CP와 SP의 n의 값들도 각각 초기화된 후 과부하 제어가 시작된다. CP는 제어구간이 끝나는 시점에서 n_{cp}와 n^*_{cp}를 비교하여 SP

의 제어레벨 L을 조정함으로써 시도되는 호들을 제어한다. 여기서

n^*_{cp} : 제어구간 동안 CP가 받아들이고자 하는 호의 수

n_{cp} : 제어구간 동안 실제로 CP가 들어오는 호의 수

n^*_{sp} : 제어구간 동안 SP가 받아들이고자 하는 호의 수

n_{sp} : 제어구간 동안 실제로 SP가 들어오는 호의 수이다.

본 논문에서는 분산구조를 갖는 교환시스템에 적합한 제어 방법으로서 프로세서 점유율과 시도호 수를 제어 파라메타로 사용하는 2가지 방식을 제안하고 그 성능을 비교하고자 한다. 방식-1은 EWSD 스위칭 시스템에 적용된 방식[2] 을 분산구조에 적합하도록 변형한 방식으로써 과부하시 각 프로세서(CP, SP)의 점유율을 90% 수준으로 유지하는 방법이며, 방식-2는 과부하시 CP의 점유율은 90% 수준으로 유지하고, 각 SP의 점유율은 CP 점유율을 90%로 유지시키기 위한 바람직한 SP의 점유율 수준을 유지하고자 하는 방식으로써 현 조정구간에서의 CP와 SP의 점유율의 관계를 이용하는 방식이다.

본 논문에서 제안한 과부하 제어 방법에서의 제어레벨의 조정방법과 n^*의 결정방법은 다음과 같다.

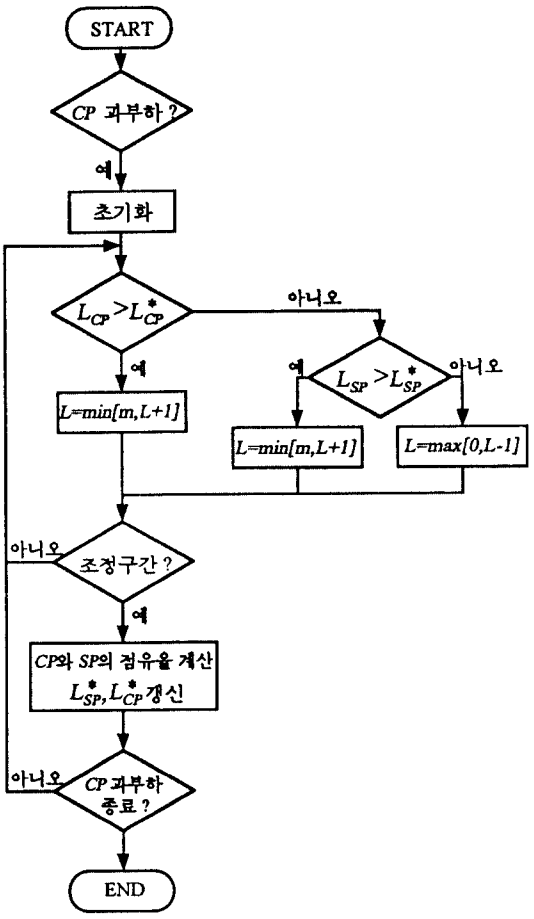

그림 2. 과부하 제어 알고리즘의 흐름도
Fig. 2. Flow Chart of the Overload Control Algorithm

1. 레벨(L)의 조정(제어구간은 끝나는 시점마다)

레벨 조정은 다음과 같은 동일한 방법을 이용한다.

1.1 $n_{CP}>n^*_{CP}$인 경우

n_{SP}와 관계없이 모든 SP의 레벨을 한 단계 높인다. 즉,

$$L=\min[m,\ L+1] \tag{3}$$

1.2 $n_{CP}\le n^*_{CP}$인 경우

n_{SP}가 n^*_{SP}보다 크면 해당 SP의 레벨을 한 단계 높이고, n_{SP}가 n^*_{SP}보다 작으면 해당 SP의 레벨을 한 단계 낮춘다. 즉,

$$\begin{aligned} L&=\min[m,\ L+1] \text{ if } n_{SP}>n^*_{SP}\\ &=\max[0,\ L-1] \text{ otherwise} \end{aligned} \tag{4}$$

2. n^*의 조정(조정구간이 끝나는 시점마다)

2.1 방식-1

방식-1에서는 CP와 SP의 점유율을 90% 수준으로 유지하기 위하여 $n^*(n^*_{CP}, n^*_{CP})$는 조정구간이 끝나는 시점마다 다음과 같은 방법에 의하여 갱신된다.

$$\begin{aligned} &\rho_M=0.9,\ \rho_L=0.89,\ \rho_H=0.91\\ &n^*_{i+1}=n^*_i+\Delta n \text{ if } \rho<\rho_L\\ &\quad=n^*_i-\Delta n \text{ if } \rho\ge\rho_L \end{aligned} \tag{5}$$

여기서 Δn는 프로세서 점유율이 90%일 때에 프로세서로 입력되는 호 n_M의 5%정도의 양이며, n^*의 초기값 n_I는 n_M의 1.2배가 되도록 정한다[2].

2-2 방식-2

방식-2는 CP의 프로세서 점유율은 90% 수준을 유지하지만 SP의 바람직한 점유율은 조정구간이 끝나는 시점마다 생신된다. SP의 바람직한 점유율은 CP와 SP의 현재 조정구간에서의 프로세서 점유율만을 이용하는 다음과 같은 간단한 방법을 제안한다.

$$\rho_{SP,i+1} = \frac{\rho_{M,CP} \times \rho_{SP,i}}{\rho_{CP,i}}$$

$$n_{SP,i+1} = \frac{\rho_{SP,i+1}}{T_{SP,proc}} \qquad (6)$$

여기서

$\rho_{SP,i}$: 현 조정구간에서의 SP 프로세서 점유율
$\rho_{CP,i}$: 현 조정구간에서의 CP 프로세서 점유율
$\rho_{M,CP}$: CP 프로세서 점유율의 목표치
$T_{SP,proc}$: 프로세서(SP)가 한호를 처리하는데 소요되는 시간이다.

또한 CP의 점유율을 90% 수준으로 유지하기 위하여 다음 조정구간내의 각 제어구간 동안 프로세서가 받아들이고자 하는 호의 수 $n^*_{CP,i+1}$는 현재 조정구간 동안의 프로세서 점유율의 함수로 설정된다. $n^*_{CP,i+1}$의 산출방법은 이미 과부하 상태인 프로세서에서 많은 부하가 가해지지 않도록 간단해야 하며, 현 조정구간에서의 CP 프로세서 점유율만을 이용하는 다음과 같은 방법을 제안한다.

$$n^*_{CP,i+1} = \min\left[n^*_{CP,i} + \frac{(\rho_{M,CP} - \rho_{CP,i})}{T_{CP,proc}},\ n_{MAX} \right] \quad \text{if } \rho_{CP,i} < \rho_H$$

$$n^*_{CP,i+1} = n^*_{CP,i} - \frac{(\rho_{M,CP} - \rho_{CP,i})}{T_{CP,proc}} \quad \text{if } \rho_{CP,i} > \rho_H \qquad (7)$$

여기서 $T_{CP,proc}$는 프로세서(CP)가 한호를 처리하는데 소요되는 시간을 의미하며, n^*_{CP}의 초기값은 프로세서 점유율이 $\rho_{M,CP}$가 될때의 호 도착율($n_{M,CP}$)의 1.2배 정도로 정한다. 식(7)에서 이전 조정구간에서의 프로세서 점유율이 작을 경우 다음 제어구간에 적용될 n^*값이 지나치게 커져서 알고리즘이 발산될 수 있는 상황을 방지하기 위하여 n_{MAX}를 사용하였으며, n_{MAX}는 n_M의 1.5배로 정하였다.

Ⅳ. 성능분석

1. 시뮬레이션 모델

분산구조를 갖는 교환시스템의 과부하 제어 알고리즘에 대한 성능 분석을 위하여 시뮬레이션을 수행하였다. 시뮬레이션은 그림 3에 나타낸 바와 같이 1개의 CP와 15개의 SP로 이루어지는 간략화된 모델을 대상으로 하였다. 사용한 시뮬레이션 언어는 SLAM Ⅱ를 이용하였고, 시뮬레이션은 기본적으로 call type 시뮬레이션이며, 프로세서(CP, SP)는 one queue one server로 간략하게 모델링하였다. CP와 SP 사이의 메시지 시나리오는 전형적인 전화호를 간략화하여 사용하였다. 시뮬레이션 모델에서 각 SP의 평균 점유율이 72%일 때, CP의 평균 점유율은 90%가 된다고 가정하였으며, CP는 점유율이 90%일 때 시간당 약 600,000(호)를 처리할 수 있다고 가정하였다. 이후에는 CP의 점유율이 90%일때에 시도되는 트래픽을 100% 부하로 표현한다.

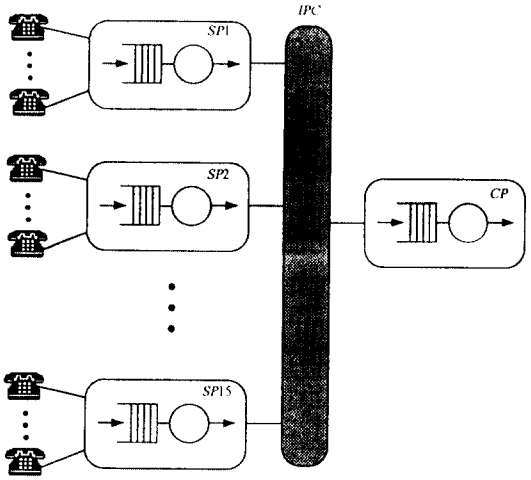

그림 3. 시뮬레이션 모델
Fig. 3. Simulation Model

통화시간은 [30, 120](초)인 일양분포(uniform distribution)를 따른다고 가정하였으며, 또한 모든 SP에 도착하는 호들은 포아송(Poisson) 분포를 따르는 것으로 가정하였다.

2. 시뮬레이션 파라메타

2.1 제어구간 및 조정구간

- 제어구간 = 1(초)
- 조정구간 = 4(초)

2.2 각 프로세서(CP, SP)에서의 호당 처리시간

CP에서는 호당 메시지 처리시간 5.4(msec)이고, SP에서는 64.8(msec)이다.

2.3 n*의 초기치(n_I)

초기치 n_I는 n_M의 1.2배 정도이므로 CP의 초기치 $n_{I,CP}$는 200(=0.9×1.2÷0.0054)(호/초), SP의 초기치 $n_{I,SP}$는 16.7(=0.9×1.2÷0.0648)(호/초)의 값을 갖는다.

2.4 Δn

Δn는 n_M의 5% 정도이므로 Δn_{CP}는 8.3(=0.9×0.05÷0.0054), Δn_{SP}는 0.7(=0.9×0.05÷0.0648)의 값을 갖는다.

2.5 과부하 시작 표시기(n_S)

호의 도착이 포아송 분포를 따르고 평균 도착률이 n_M일때 제어구간 동안 시도되는 호가 n_S를 넘을 확률이 0.01% 이하가 되도록 n_S의 값이 결정된다. 시뮬레이션 모델에서 $n_{S,CP}$는 216(호/초), $n_{S,SP}$는 29(호/초)의 값을 갖는다.

3 시뮬레이션 결과

과부하를 유발하는 입력 트래픽의 형태에 따라 과부하 제어 방법의 성능이 달라질 수도 있으므로 다양한 형태의 입력 트래픽을 고려할 필요가 있는데 이를 위하여 다음과 같은 3가지 형태의 입력 트래픽을 고려하였다.

1) 급진적 증가형의 트래픽
   CP의 프로세서 점유율이 0%에서 150%(200%, 300%, 400%)로 갑자기 증가하는 경우.
2) 선형적 증가형의 트래픽
   점유율이 60%에서 200%(300%, 400%, 500%)까지 선형적으로 증가하는 경우.
3) 주기적 변화형의 트래픽
   점유율이 100% 600% 사이를 주기적(sinusoidal)으로 변화하는데 그 주기가 200(초), 300(초), 400(초), 500(초)인 경우.

앞에서 설명한 입력 트래픽이 시뮬레이션 모델에 인가되는 경우에 시뮬레이션을 수행하여 프로세서(CP) 평균 점유율과 표준편차, 평균 큐길이와 표준편차, 평균 대기시간과 표준편차등으로 구분하여 방식-1과 방식-2의 결과를 표.2부터 표.4까지 나타냈다. 제안한 과부하 제어 방식은 일단 허용된 호들은 중간에 기각하지 않고 끝까지 처리를 완료하므로 시뮬레이션 결과에서 CPU 점유율에 관한 결과는 수율(throughput)에 대한 결과와 동일한 경향으로 변화될 것이므로 수율에 대한 결과는 제시하지 않았다.

표2는 급진적 증가형의 트래픽이 시뮬레이션 모델에 인가되었을 때의 결과를 나타내는 데, 트래픽이 인가되는 순간부터 시뮬레이션이 종료될 때까지의 평균값과 표준편차를 나타낸다. 표.3은 선형적 증가형의 트래픽이 인가되었을 때의 결과를 나타내는 데, 트래픽이 변화되는 순간부터 시뮬레이션이 종료될 때까지의 평균값과 표준편차를 나타낸다. 또한 표.4는 주기적 변화형의 트래픽이 시뮬레이션 모델에 인가되었을 때의 결과를 나타낸다.

이러한 결과로 부터 본 논문에서 제안한 방식-2의 CP의 점유율은 방식-1에 비하여 프로세서 점유율의 목표치 90%에 가깝게 유지되고 있으며, 또한 CP의 입력 큐에서의 평균 대기시간과 표준편차도 상대적으로 매우 작음을 알 수 있다. 방식-2에서 CP의 점유율이 원하는 수준(90%)에 가까이 근접한 이유

표 2. 급진적 증가형의 트래픽이 인가된 경우 프로세서의 성능

table 2. Performance of CP when the offered traffic is abruptly varied

(방식-1)

트래픽	점유율		큐길이		대기시간(초)	
	평균	표준편차	평균	표준편차	평균	표준편차
200%	0.8746	0.3307	34.458	55.717	0.1060	0.1560
300%	0.8473	0.3579	43.093	76.844	0.1370	0.2190
400%	0.8288	0.3741	74.495	100.300	0.2410	0.2800
500%	0.8312	0.3728	97.284	120.198	0.3140	0.3320

(방식-2)

트래픽	점유율		큐길이		대기시간(초)	
	평균	표준편차	평균	표준편차	평균	표준편차
200%	0.8919	0.3117	5.009	9.222	0.0151	0.0260
300%	0.8919	0.3115	5.263	9.706	0.0159	0.0274
400%	0.8918	0.3117	6.920	12.815	0.0209	0.0360
500%	0.8921	0.3113	8.090	15.959	0.0244	0.0450

표 3. 선형적 증가형의 트래픽이 인가된 경우 프로세서의 성능

table 3. Performance of CP when the offered traffic is linearly varied

(방식-1)

트래픽	점유율		큐길이		대기시간(초)	
	평균	표준편차	평균	표준편차	평균	표준편차
200%	0.8439	0.3626	37.248	55.294	0.1190	0.1550
300%	0.8242	0.3822	71.280	94.471	0.2330	0.2630
400%	0.8337	0.3727	97.880	119.942	0.3160	0.3310
500%	0.8737	0.3306	122.277	122.364	0.3750	0.3280

(방식-2)

트래픽	점유율		큐길이		대기시간(초)	
	평균	표준편차	평균	표준편차	평균	표준편차
200%	0.8955	0.3052	5.361	7.896	0.0161	0.0220
300%	0.8957	0.3054	6.991	12.928	0.0210	0.0362
400%	0.8954	0.3054	6.850	10.186	0.0206	0.0283
500%	0.8956	0.3051	9.821	18.483	0.0295	0.0518

표 4. 주기적 변화형의 트래픽이 인가된 경우 프로세서의 성능

table 4. Performance of CP when the offered traffic is sinusoidally varied

(방식-1)

트래픽	점유율		큐길이		대기시간(초)	
	평균	표준편차	평균	표준편차	평균	표준편차
200%	0.8461	0.3588	83.316	125.167	0.2650	0.3530
300%	0.8206	0.3637	74.924	118.858	0.2400	0.3370
400%	0.8433	0.3612	75.755	119.250	0.2410	0.3380
500%	0.8465	0.3590	76.327	113.408	0.2420	0.3190

(방식-2)

트래픽	점유율		큐길이		대기시간(초)	
	평균	표준편차	평균	표준편차	평균	표준편차
200%	0.8858	0.3171	9.847	20.447	0.0299	0.0579
300%	0.8876	0.3152	7.491	14.803	0.0227	0.0418
400%	0.8890	0.3134	6.716	12.915	0.0203	0.0364
500%	0.8899	0.3125	8.618	17.488	0.0260	0.0494

는 CP와 SP의 점유율 사이의 관계를 기초로 SP의 바람직한 점유율을 비교적 정확하게 예측한 결과이다. 한편 방식-1에서 CP의 점유율이 떨어지는 이유는 CP 점유율과 SP 점유율 사이의 관계를 고려하지 않았기 때문이다. 따라서 방식-2가 분산구조 교환시스템에서 효율적인 과부하 제어 방법임을 알 수 있다.

그림 4와 그림 5는 점진적 증가형의 트래픽(CP의 점유율이 0%에서 200%로 갑자기 증가한 경우)이 시뮬레이션 모델에 인가된 경우 각각의 과부하 제어 방법에서의 CP의 점유율의 궤적을 나타낸다. 점유율의 측정치가 유동적인 이유는 호의 도착이 random하기 때문이다. 방식-1에서는 점유율의 궤적이 50%와 100% 사이를 지속적으로 심하게 진동하게 있으며, 반면에 방식-2에서의 궤적은 과부하 상태가 시작된 시점부터 서서히 목표치(90%)로 증가하는 양상을 보여주면서 120(초) 이후부터 정상상태로 돌입함을 알 수 있다.

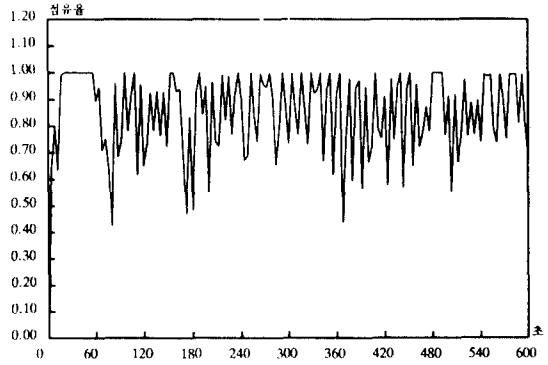

그림 4. CP 점유율의 궤적(방식-1)

Fig. 4. Locus of CP Utilization(method-1)

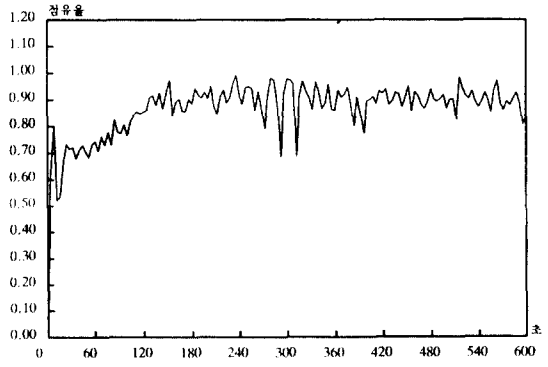

그림 5. CP 점유율의 궤적(방식-2)

Fig 5. Locus of CP Utilization(method-2)

따라서 방식-2는 방식-1에 비하여 CP의 프로세서 점유율의 변화가 심하게 나타나지 않으므로 안정된 알고리즘임을 알 수 있다.

표.5는 급진적 증가형의 트래픽이 시뮬레이션 모델에 인가된 경우 각각의 과부하 제어 방법에서의 제어레벨의 빈도수를 나타낸다. 제어레벨은 직접적으로 입력 트래픽을 제어하므로 제어레벨의 변이를 조사하는 것이 매우 중요하다.

따라서 표.5를 토대로 입력 트래픽의 변화에 대한 각 제어방식에서의 평균 호 허용율을 산출하여 표.6에 나타냈다. 비고란은 각 입력 트래픽에서의 이상적인 호 허용율이다. 표.6에서 보면 방식-2가 이상적인 호 허용율에 보다 가까이 근접하고 있으므로 방식-2가 방식-1에 비하여 효율적으로 입력 트래픽을 제어하는 알고리즘임을 알 수 있다.

표 5. 제어레벨의 빈도수(%)
table 5. Frequency of Control Level(%)

(방식-1)

트래픽	0	1	2	3	4	5	6	7	8	9	10	11
150%	1	5	14	19	21	19	14	6	1	0	0	0
200%	0	0	2	6	12	18	20	18	13	8	3	0
300%	0	0	0	0	0	5	15	20	20	20	15	5
400%	0	0	0	0	0	0	4	16	21	21	21	17

(방식-2)

트래픽	0	1	2	3	4	5	6	7	8	9	10	11
150%	0	3	12	24	28	21	10	2	0	0	0	0
200%	0	0	0	2	12	26	32	21	6	1	0	0
300%	0	0	0	0	0	1	10	29	37	20	3	0
400%	0	0	0	0	0	0	0	8	32	39	18	3

표 6. 트래픽 변화에 따른 평균 호 조절율
table 6. Average Call Acceptance Rate according to Traffic Variation

트래픽	방식-1	방식-2	방식-3
150%	0.652	0.663	0.666
200%	0.471	0.499	0.500
300%	0.309	0.331	0.333
400%	0.231	0.243	0.250

V. 결 론

본 논문에서는 분산구조를 갖는 교환시스템에 대한 효율적인 과부하 제어 방법을 제안하고 그 성능을 시뮬레이션에 의하여 평가하였다. 과부하시 프로세서 점유율을 목표치 수준으로 유지시키기 위하여 본 논문에서 제안한 과부하 제어 방법들은 시도되는 호 중 실제 교환기로 입력되는 호들을 조절하기 위한 제어레벨, 시간축을 균등한 간격으로 분할한 제어구간 및 4개의 제어구간으로 이루어지는 조정구간의 개념을 도입하였으며, 제어 변수로서 입력되는 시도호의 수와 프로세서 점유율을 사용하는데 프로세서 점유율을 목표치(90%) 수준으로 유지하기 위하여 제어구간 동안 프로세서가 받아들이고자 하는 호의 수 n^*를 계산하는 새로운 방법들을 제안하였다.

3가지 입력 트래픽 유형에 따른 시뮬레이션 결과는 CP 점유율과 SP 점유율 사이의 관계를 고려한 방식-2의 CP 점유율이 방식-1에 비하여 목표치 90%에 가깝게 유지하고 있으며, CP의 입력 큐에서의 평균 대기시간과 표준편차도 상대적으로 매우 작게 나타났다. 또한 방식-2는 참고 문헌[5] 에서의 방식보다 알고리즘이 간단하고 이상적인 호허용율에 매우 가까이 근접하고 있으므로 방식-1에 비하여 분산구조형 교환시스템에서 효율적이고 안정적인 과부하 제어 방법이라고 결론지을 수 있다.

참 고 문 헌

1. P. Somoza, A. Guerrero, "Dynamic Processor Overload Control for SPC Switching Systems," *Electrical Communication*, Vol. 55, No. 1, pp. 37-45, 1980.
2. G. Daisenberger, J. Oehlerich and G. Wegmann, "STATOR-STATistical Overload Regulation-and TAIL Time Account Input Limitation-Two Concepts for Overload Regulation in SPC Systemx," *11th ITC*, paper 2.1B-4, 1985.
3. P. Hanselka, J. Oehlerich and G. Wegmann, "Adaptation of the Overload Regulation Method STATOR to Multiprocessor Controls and Simulation Results," *12th ITC*, Torino, June 1988.
4. G.M. Andres, M.V. Altamirano, "System 12 Traffic Overload control," *Electrical Communication*, Vol. 59, No. 1/2, pp. 74-79, 1985.
5. H. Lee K.H. Kook, S.K. Lim, J.H. Baek, "Performance Analysis of Distributed Control Swit-

ching System in Traffic Overload Environments," *SICON '91*, Singapore, Sep., 1991
6. C.H. Yim and H.L. Hartmann, "Throughput behavior of switching systems under heavy load conditions" *11th ITC*, paper 4.3B 2, 1985
7. Donald Gross, Carl M. Harris, "*Fundamentals of Queueing Theory*," John Wiley & Sons.

程 炫 必(Hyon Pil Jung) 正會員

1948년 8월 15일생

1971년 2월 : 한국항공대학 항공통신공학과 졸업
1983년 8월 : 조선대학교 대학원(공학석사)
1991년 12월 : State University of New York at Buffalo 객원교수
1992년 2월 : 한국항공대학 항공전자공학과 박사과정 수료
1977년 3월 ~ 현재 : 조선대학교 멀티미디어공학부 전자통신과 교수

李 潤 鉉(Yun Hyun Lee) 正會員

1941년 8월 21일생

1965년 2월 : 한국항공대학 전자공학과 졸업
1985년 2월 : 경희대학교 대학원 전자공학과 공학박사
1985년 9월 : 통신기술사
1988년 2월 : State University of New York at buffalo 교환교수
1994년 ~ 현재 : 한국항공대학교 통신정보 공학과 교수

林 錫 鳩(Seog Ku Kim) 正會員

1959년 9월 20일생

1983년 2월 : 한국항공대학교 항공전자공학과 졸업
1985년 2월 : 서울대학교 대학원 전자공학과(공학석사)
1987년 1월 : 금성정보통신
1992년 2월 : 한국전자통신연구소
1994년 3월 : 순성 전문대학 전자과