論文93-30B-1-3

네트워크를 이용한 실시간 분산제어시스템에서 데이타 샘플링 주기 결정 알고리듬

(An Algorithm of Determining Data Sampling Times in the Network-Based Real-Time Distributed Control Systems)

洪 承 鎬*

(Seung Ho Hong)

要 約

분산제어시스템에서 제어관련 데이타들은 네트워크를 통하여 교환된다. 실시간 분산제어 시스템의 성능은 네트워크를 통한 센서에서 콘트롤러까지 및 콘트롤러에서 액튜에이터까지의 데이타 전송 지연시간에 영향을 받으며, 이러한 데이타의 전송 지연시간은 네트워크를 공유하는 각각의 제어요소에서 수행되는 데이타 샘플링 주기에 좌우된다. 본 논문에서는 실시간 분산제어시스템에서 데이타 전송지연시간으로 인한 제어시스템의 성능저하를 최소화하도록 하는 데이타 샘플링 주기를 결정하는 알고리듬을 제시한다. 이 알고리듬은 각각의 제어요소에서 샘플링되는 데이타들이 한정된 윈도우를 공유하는 "윈도우 개념"을 바탕으로 한다. 제시된 알고리듬은 이산사건/연속시간 시뮬레이션 기법을 통하여 검증되었다.

Abstract

Processes in the real-time distributed control systems share a network medium to exchange their data. Performance of feedback control loops in the real-time distributed control systems is subject to the network-induced delays from sensor to controller, and from controller to actuator. The network-induced delays are directly dependent upon the data sampling times of control components which share a network medium. In this study, an algorithm of determining data sampling times is developed using the "window concept", where the sampling data from the control components dynamically share a limited number of windows. The scheduling algorithm is validated through the simulation experiments.

Ⅰ. 서론

대형의 복잡한 제어시스템은 일반적으로 단일기능을 수행하는 여러개의 분산된 부시스템으로 나뉘어진다. 이러한 분산제어 시스템은 (1)시스템 설계의 변화 및 개선에 용이하게 대처하고, (2)시스템의 운용에 유연성을 제공하며, (3)시스템의 유지, 보수와 감시 기능을 용이하도록 한다. 여러개의 분산된 부시스템들에서 생성되는 센서 정보와 제어관련 정보는 네트워크를 통하여 교환된다. 네트워킹은 또한 실시간

*正會員, 漢陽大學校 制御計測工學科
(Dept. of Cont. & Inst. Eng., Hanyang Univ.)
接受日字:1992年 8月 10日

제어시스템의 백업 (back-up) 장비의 제공을 용이하게 하여 시스템의 신뢰도를 증가시키고, 고가의 희귀한 장비들을 부시스템들 간에 서로 공유할 수 있도록 한다. 따라서, 네트워크는 여러개의 복잡한 제어공정들로 구성된 대형의 복잡한 시스템에서 척추 (backbone) 역할을 수행한다. 이러한 시스템들의 예로는 자동화 공장, 전기 및 핵 발전소, 첨단의 항공기와 우주선 및 선박 등이 있다.[1-3] 최근에 와서는 자동차의 제어기능이 고도화 됨에 따라 자동차에서도 네트워킹 기술이 도입되고 있다.[4]

분산된 부시스템들을 네트워킹으로 연결하면 복잡한 시스템의 설계, 운용 및 관리에 신뢰도와 유연성을 부여할 수 있으나, 네트워크의 트래픽으로 인한 데이타의 지연시간으로 인하여 실시간 제어 시스템의 성능은 감소될 수 있다.[5] 따라서, 실시간 분산제어 시스템에서 네트워킹으로 인한 데이타의 지연시간은 지정된 한계치를 초과하지 않도록 설계되어야 한다. 데이타의 지연시간은 네트워크의 트래픽 상태에 따라 결정되며, 트래픽은 또한 각각의 분산된 제어요소에서의 데이타 샘플링 주기 (또는 네트워크의 관점에서 보면 데이타 도착 주기)에 직접적으로 영향을 받는다. 따라서 실시간 분산제어시스템에서 데이타 샘플링 주기는 다음과 같은 설계기준에 의하여 결정되어야 한다.

o 제어시스템의 성능 요구사항을 만족하여야 한다
o 네트워크의 이용도(utilization)를 최대화 하여야 한다.

시간에 제약을 받는(time-constrained) 시스템의 분석은 여러 문헌에 소개된 바 있으나,[6,7] 대부분이 큐의 용량이 무한정하고 데이타 도착주기가 프와송 (Poisson) 분포를 갖는다는 가정하에서 해석되었다. 그러나 이러한 해석들은 큐의 용량이 하나로 제한되고(이는 제어시스템이 가장 최근에 생성된 센서 또는 콘트롤러의 데이타를 필요로 하기 때문이다), 데이타들이 주기적으로 샘플링되는 실시간 분산제어시스템에는 적용할 수 없다. 본 논문의 목적은 실시간 분산제어시스템의 각각의 제어요소에서 데이타 샘플링 주기를 결정하는 알고리듬을 제시하는 것이다.

본 논문의 2장에서는 실시간 분산제어시스템의 특성을 간략히 기술한다. 샘플링 주기를 결정하는 알고리듬의 제시는 제 3 장에 기술되고, 제 4 장에서는 실시간 분산제어시스템의 설계의 예가 주어진다. 마지막으로 제 5 장에서는 본 논문의 결론이 기술된다.

Ⅱ. 실시간 분산제어시스템의 특성

실시간 분산제어시스템에서는 여러개의 제어 루프 (control loop)들이 하나의 네트워크 시스템을 공유한다. 그림 1에서 보는 바와 같이 각각의 제어 루프는 센서, 콘트롤러와 액튜에이터 등의 제어요소 (control component)들로 구성된다. 최근에 제안되고 있는 대부분의 산업용 네트워크시스템의 매체 접속 제어 (medium access control) 방식은 크게 다음의 두 가지로 구분될 수 있다.[11]

o 폴링(polling)방식에 의한 중앙 접속 제어 (예, FIP, MIL-STD-1553B 등)
o 토큰 전달 방식에 의한 분산 접속 제어 (예, IEEE 802.4 토큰버스, PROWAY 등)

폴링과 토큰 전달 방식들은 모두 순환 (cyclic) 서비스를 기본원칙으로 운용된다. 즉, 폴링 신호 또는 토큰은 제어요소가 위치한 노드를 차례로 방문하며, 각각의 제어노드의 전송 큐에서는 폴링 신호 또는 토큰이 도착되는 순간에 대기하고 있는 데이타를 전송한다. 본 논문에서 제시하는 알고리듬은 폴링 과 토큰 전달을 포함하는 모든 순환 서비스 매체 접속 제어 방식에 적용될 수 있다.

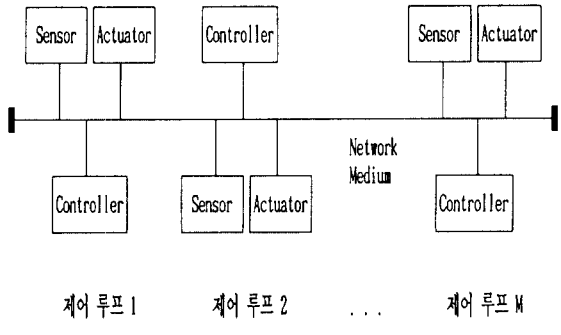
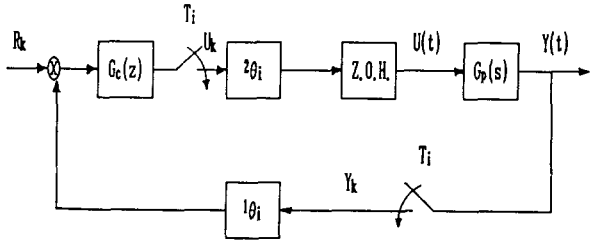


그림 1. 실시간 분산제어시스템
Fig. 1. A Real-Time Distributed Control System.

실시간 분산제어시스템의 각각의 제어요소에서 샘플링된 데이타는 토큰 또는 폴링 신호와 같은 서버 (server)가 도착할 때까지 전송 큐에서 대기하여야 하며, 이것이 네트워크로 인한 전송지연시간을 야기한다. 분산제어시스템의 i-번째 휘드백 (feedback) 제어 루프에서 네트워크로 인한 전송지연이 그림 2에 나타나 있다. 여기서 $^1\theta_i$은 센서-콘트롤러 지연시간을 나타내고, $^2\theta_i$는 콘트롤러-액튜에이터 지연시간을

나타낸다. 네트워킹에 의하여 야기되는 이러한 데이타 지연시간은 일반적으로 시간에 따라 불규칙하게 변하는(randomly time-varying) 특성을 가지고 있다.[5] 불행하게도 현재까지는 데이타 지연시간이 시간에 따라 불규칙하게 변하는 제어시스템을 해석할 수 있는 일반적인 방법이 발표되지 않고 있다. 본 논문에서 제시하는 실시간 분산제어시스템의 설계 기법은 각각의 제어요소에서 데이타 지연시간의 최대 허용치를 기준으로 하여 각각의 제어요소의 데이타 샘플링 시간을 결정하는 것이다.



$G_c(z)$: 콘트롤러 전달함수
$G_p(s)$: 플랜트 전달함수
Z.O.H.: Zero-Order Holder
$^1\theta_i$: 센서-콘트롤러 지연시간
$^2\theta_i$: 콘트롤러-액튜에이터 지연시간
T_i : i-번째 제어루프의 데이타 샘플링 시간
U_k : 샘플링된 콘트롤러 데이타
Y_k : 샘플링된 센서 데이타

그림 2. i-번째 제어 루프에서 네트워크 지연시간
Fig 2. Network-Induced Delay in the i-th Control Loop.

데이타 전송속도가 주어진 실시간 분산제어용 네트워크의 파라미터들로는 (1) 네트워크 내의 노드 수, (2) 데이타 길이 (또는 데이타 전송시간), (3) 데이타 도착 주기 등이 있다. 분산제어시스템에서 센서와 콘트롤러에서 생성되는 데이타의 길이는 일반적으로 일정한 길이로 패킷화(packetized)된다. 따라서 노드 수가 정해진 분산제어시스템의 설계자는 각노드에서의 데이타 도착 주기 (또는 제어시스템의 관점에서 보면 데이타 샘플링 주기)가 제어시스템의 성능요구사항을 만족하도록 설계하여야 한다. 휘드백 제어 루프의 제어성능은 "루프지연시간"에 직접적으로 영향을 받는다.[8] 루프지연시간은 센서 노드에서 데이타가 샘플링된 순간에서 부터 그 센서 데이타부터 생성된 콘트롤러 데이타가 액튜에이터에 전달되는 순간까지의 경과된 시간으로 정의된다. 센서, 콘트롤러 및 액튜에이터가 Ti의 같은 샘플링 주기를 가지는 일반적인 경우에서 제어루프 i에서의 루프지연시간 D_i가 그림 3에 나타나 있다. 그림에서부터 D_i는 다음과 같다.

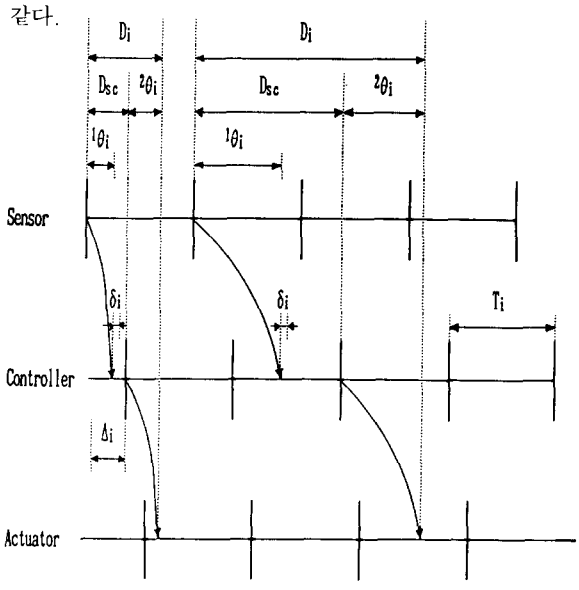


그림 3. i-번째 루프에서 루프지연시간
Fig. 3. Loop Delay of the i-th Control Loop

$$D_i = Dsc(T_i, \Delta_i) + {}^2\theta_i \quad (1)$$

여기서

$$Dsc(T_i, \Delta_i) = \Delta_i + n\,T_i \quad (2)$$

$$n = \begin{cases} 0, & {}^1\theta_i + \delta_i < \Delta_i \\ Int\,[({}^1\theta_i + \delta_i - \Delta_i)/T_i] + 1, & {}^1\theta_i + \delta_i \geq \Delta_i \end{cases} \quad (3)$$

Int [x] 는 x의 정수부분을 나타내고, δ_i는 콘트롤러 노드에서의 데이타처리 지연시간을 나타낸다. 또한 Δ_i는 i-번째 루프의 센서와 콘트롤러 노드에서 샘플링 순간의 차이를 나타낸다. 분산제어시스템에서 센서와 콘트롤러 노드는 서로 떨어져서 위치하게 되며, 따라서 센서와 콘트롤러 노드 내의 시계(clock)의 부정확성으로 인하여 두 노드에서 샘플링 순간이 동기화 되지 않을 수도 있다. Δ_i는 시간에 따라 값이 변화하며, 식 (1), (2), (3)으로부터 Δ_i는 루프지연시간 D_i에 영향을 미친다. 그러나 이러한 노드간의 비동기는 지정된 노드에서 네트워크를 통하여 주기적으로 동기화 메시지를 모든 노드에 전송함으로써 주어진 한계치를 초과하지 않도록 조정할 수 있다. 만일 Δ_i의 한계치가 샘플링주기 T_i에 비하여 매우 작고 ($\Delta_i \ll T_i$), 콘트롤러에서의 처리시간 δ_i 역시 T_i에 비하여 작은 값인 경우 ($\delta_i \ll T_i$), 루프지연시간은 그림 4에 주어진 것과 같이 단순화 되며, 여기서 제어루프 i의 루프지연시간은 다음과 같다.

$$Di = \{Int\,[{}^1\theta_i/T_i] + 1\}\,T_i + {}^2\theta_i \quad (2.4)$$

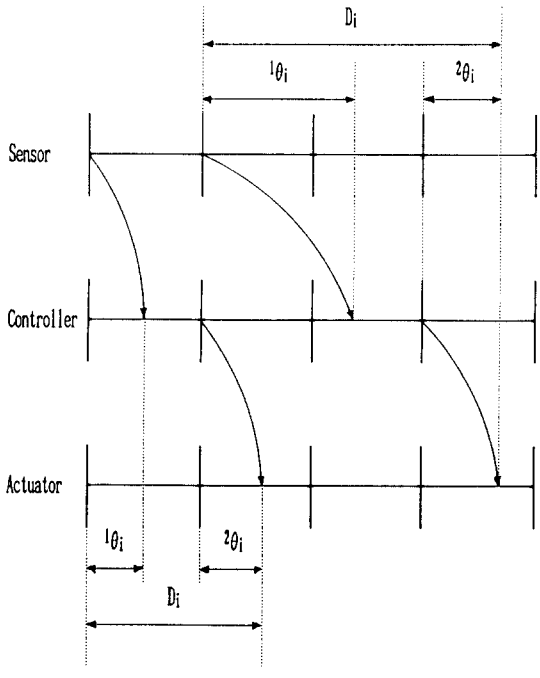


그림 4. i-번째 루프에서 단순화된 루프 지연 시간
Fig. 4. Simplified Loop Delay of the i-th Control Loop.

Ⅲ. 샘플링 주기 결정 알고리듬

i-번째 루프가 m_i개의 데이타 전송노드로 구성되고 (액튜에이터 노드는 일반적으로 제어 기능 관련 데이타를 전송하지 않음) M개의 제어 루프를 가지는 실시간 분산 제어 시스템에서 시스템 파라미터로는 (1) 네트워크 내의 전송 노드 수 ($N=\sum_{i=1}^{M} m_i$), (2) 패킷화된 데이타의 길이 (L'), (3) 네트워크의 데이타 전송속도 (B), (4) 각 제어루프에서의 데이타의 최대허용 지연시간 (Ψ_i, i=1에서 M) 등이 있다. 본 논문에서는 네트워크의 이용도를 크게 증가시키는 동시에 각각의 제어 루프의 제어성능 요구사항을 만족시키는 데이타 샘플링 주기 T_i(i=1 에서 M)를 결정하는 알고리듬을 제시하고자 한다. 모든 노드는 같은 크기의 패킷화된 데이타의 전송시간 ($L=L'/B$)를 가지며, 하나의 제어루프에 속해있는 모든 노드들은 같은 샘플링 주기를 갖는 것으로 한다.

네트워크로 인한 데이타의 지연시간은 불규칙적으로 변하는 속성이 있다. 따라서 그림 5에서 보는 바와 같이 지연시간이 데이타 샘플링 주기보다 길어지는 경우 두개 이상의 센서 데이타가 콘트롤러 노드의 샘플링 주기 동안에 도달하는 경우가 발생하고, 이 경우 마지막으로 도착한 (가장 최근에 생성된) 데이타 만이 콘트롤러 데이타를 생성하는데 사용된다. 이때 "데이타 손실" 현상이 발생한다. 반면에 다른 샘플링 주기에서는 센서 데이타가 도달하지 않는 "공허 샘플링" 현상이 나타난다. 데이타 손실과 공허 샘플링은 제어시스템의 성능을 저하할 뿐만 아니라 제어신호의 고주파수 잡음으로 인한 찌그러짐 현상을 야기하여 액튜에이터의 마모를 초래한다. 따라서, 모든 센서(또는 콘트롤러) 데이타는 다음 센서(또는 콘트롤러) 데이타가 샘플링되기 이전에 콘트롤러 (또는 액튜에이터)에 도달하여야 한다. 즉, m_i개의 데이타 전송노드로 구성된 i-번째 루프에서 다음의 조건을 만족하여야 한다.

$$ {}^{1}\theta_i < T_i,\ {}^{2}\theta_i < T_i,\ \ldots,\ {}^{m_i}\theta_i < T_i \qquad (5) $$

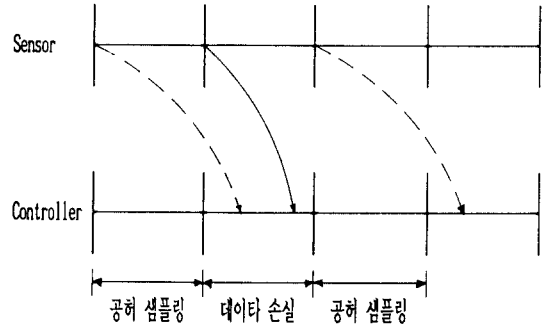


그림 5. 데이타 손실과 공허 샘플링의 예
Fig. 5. Example of Data Rejection and Vacant Sampling.

여기서 θ_{ij}는 i-번째 루프의 j-번째 노드에서의 데이타 지연시간이다. 이와 동시에 i번째 루프에서 루프지연시간은 최대허용치 Ψ_i를 초과하지 않아야 한다.

즉,

$$ D_i < m_i T_i \le \Psi_i \qquad (6) $$

(5)와 (6)에서 주어진 조건을 동시에 만족할 때의 샘플링 주기 결정 알고리듬을 "실행 가능" 알고리듬이라 하자.

M개의 제어루프에서 샘플링 주기벡터 T는 다음과 같이 주어진다.

$$ T = [T_1,\ T_2,\ \ldots,\ T_M] \qquad (7) $$

여기서 T_i는 i-번째 제어 루프에서의 샘플링 주기이며, 같거나 증가하는 순서를 갖는 원소배열이다. 즉, T는 모든 i에 대하여 $T_i \le T_{i+1}$를 만족한다. T_i의 시간 동안에 서버(토큰 또는 폴링신호)에 의하여 네트워크 내에서 전송될 수 있는 데이타의 최대수를 r_i라 하자. 즉,

$$r_i = Int\ [(T_i - N\sigma)/L] \qquad (8)$$

여기서 는 각 노드에서 서버의 처리시간을 나타내며 이는 한 노드에서 다른 노드로 서버의 전달시간도 포함한다. ($\sigma \ll L$)

보조정리 1: 샘플링 주기가 T_j인 임의의 노드 (즉, j-번째 제어루프에 속한 임의의 노드) j에서 T_i 동안 서버에 의하여 네트워크 내에서 전송된 데이타의 갯수가 r_j를 초과하지 않으면, 노드 j에서 데이타 손실이나 공허 샘플링 현상은 일어나지 않는다.

증명: T_j의 샘플링 주기를 갖는 임의의 노드 j에서 최악의 데이타 지연시간은 노드가 데이타를 샘플링하기 직전에 서버를 다음 노드에 전달했을 경우에 발생한다. 서버가 네트워크 내의 N개의 노드를 모두 방문하고 다시 노드 j로 돌아오는 동안 p개의 데이타를 전송하였다면 이때 노드 j에서의 데이타 지연시간은 $j = pL + N\sigma$ 이다. 만일 모든 T_j에서 p가 r_i를 초과하지 않는다면, (8)에서 $\theta_j < T_j$ 이며, 이 노드에서 데이타 손실 또는 공허 샘플링 현상이 발생하지 않는다.

네트워크 내에서 최소 샘플링 주기 T_1의 시간 동안에 서버에 의하여 전송될 수 있는 데이타의 최대수를 r이라 하자. 즉,

$$r = Int\ [(T_1 - N\sigma)/L] \qquad (9)$$

그림 6에서 보는 바와 같이 T_1은 r개의 윈도우로 구성되며, 각 윈도우의 크기는 패킷 전송시간 L과 일치한다.

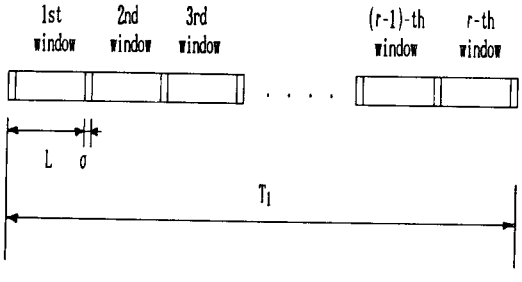


그림 6. 윈도우 개념의 예시
Fig. 6. Illustration of Window Concept.

정리 1: 실시간 분산제어시스템에서 만일 모든 샘플링 주기 T_1 동안 서버에 의하여 네트워크 내에서 전송된 데이타의 갯수가 r을 초과하지 않으면, 모든 노드에서 데이타 손실이나 공허 샘플링 현상은 일어나지 않는다.

증명: $T_j = k_j T_1$ ($k_j \ge 1$) 인 임의의 노드 j를 생각하자. 만일 모든 샘플링 주기 T_1 동안 서버에 의하여 네트워크 내에서 전송된 데이타의 갯수가 r을 초과하지 않으면 T_j의 시간동안 전송된 데이타의 수는 r_j를 초과할 수 없다. 보조정리 1로부터 임의의 노드 j에서 데이타 손실이나 공허 샘플링 현상은 일어나지 않는다.

참고사항 1: 정리 1에서 샘플링 주기를 결정하는 알고리듬은 기본적으로 "윈도우 개념"에 의하여 결정되어야 함을 알 수 있다. 즉, 모든 T_1의 샘플링 주기 동안 네트워크 내의 N개의 노드는 r개의 윈도우를 적절히 공유하여, T_1동안 서버에 의하여 전송되는 데이타의 갯수가 r개를 초과하지 않도록 하는 것이다.

(6)과 (9)로부터 r은 다음과 같이 결정된다.

$$r = Int\ [(\Psi_1/m_1 - N\sigma)/L] \qquad (10)$$

최소 샘플링 주기 T_1에 대한 다른 샘플링 주기들의 비율을 벡터 K로 표시하면 다음과 같다.

$$K = [k_1, k_2, \ldots, k_M],\ k_j = T_j/T_1,\ k_j \le k_{j+1},\ \forall j \qquad (11)$$

주어진 K에 대하여, T1의 시간동안 샘플링되는 데이타 수의 기대치 K는 다음의 식으로 표시된다.

$$\alpha_K = \sum_{i=1}^{M} (m_i/k_i) \qquad (12)$$

보조정리 2: 만일 (1) 모든 제어루프 i에 대하여 k_{i+1}이 k_i의 정수배이고 (즉, $k_i = 2^{n_i}$, 여기서 n_i는 0을 포함하는 양의 정수), (2) $\alpha_K \le r$의 조건을 만족하면, T_M의 주기 동안 샘플링된 모든 데이타는 같은 시간 동안 제공되는 윈도우에 의하여 처리된다. 여기서 T_M은 네트워크 내에서 최대 샘플링 주기이며, k_M개의 T_1 슬롯으로 구성된다.

증명: 샘플링 주기가 $T_j \le T_M$인 임의의 노드 j를 생각하자. 만일 $k_j = 2^{n_j}$ 이면, T_M 동안 노드 j에서 샘플링 되는 데이타의 갯수는 k_M/k_j의 " 정수"로 고정된다. 따라서, T_M의 시간 동안 N개의 노드에서 샘플링 되는 데이타의 갯수는

$$k_M = \sum_{j=1}^{N}(i/k_j) = \sum_{i=1}^{M}(m_i/k_i) = k_M\,\alpha_K$$

의 "정수"로 고정된다. T_M의 주기 동안 제공되는 윈도우의 갯수가 k_{Mr}이므로, $\alpha_K \le r$인 경우 각각의 T_M의 주기 동안 샘플링된 모든 데이타는 같은 시간 동안 제공되는 윈도우에 의하여 처리된다.

참고사항 2: $\alpha_K > r$은 입력되는 데이타 트래픽이 네트워크 용량을 초과하는 경우이다. 이 경우, 실시간 분산제어시스템 설계자는 네트워크 내의 노드 수를 줄이거나, 또는 전송속도가 빠른 고성능 네트워크 시스템으로 대체하여야 한다.

보조정리 2에서 각각의 T_M 동안 고정된 수의 데이타가 샘플링된다는 것은 각각의 T_i동안에도 r보다 작거나 같은 수의 데이타가 샘플링 되도록 하는 샘플링 주기 결정 알고리듬이 존재함을 의미한다.(정리 1 참조) 다음의 보조정리에서 이를 보여준다.

보조정리 3: $\alpha_K \le r$ 와 $k_i = 2^{n_i}$의 조건을 만족하는 경우, 데이타 손실과 공허 샘플링 현상이 일어나지 않는 샘플링 주기 결정 알고리듬이 존재한다. 즉,

$$S_j \le r,\ \forall j = 1, \cdots, K_M \qquad (13)$$

여기서 S_j는 T_M 내의 T_j 슬롯이 시작하는 순간인 A_j에 네트워크 내에서 샘플링되는 데이타의 갯수이다. (그림 7 참조)

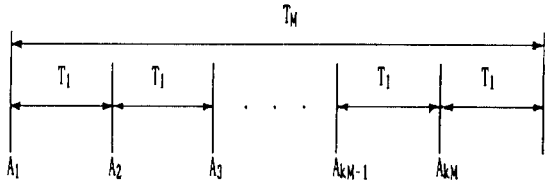


그림 7. T_M에서 A_i의 예시
Fig. 7. Illustration of A_i in T_M.

증명: 임의의 노드 j와 i에서 T_j는 T_i의 정수의 배수이다 ($j \ge i$). 작은 샘플링 주기를 갖는 노드부터 시작하여 $T_1, T_2, \cdots, T_i$의 샘플링 주기를 갖는 r개의 노드를 선정하여 이들이 A_1의 순간에 데이타를 샘플링 하도록 하면, 이들은 T_M의 시간 내에서 $A_i + nT_i$ ($n=1, 2, \cdots, k_M/k_i$)의 순간마다 동시에 샘플링 한다. 보조정리 2로부터 $\alpha_K \le r$를 만족하는 경우에 T_M의 시간 동안 생성된 모든 데이타는 같은 시간 동안 제공된 모든 윈도우들에 의하여 처리되며, 이는 A_1과 $A_1 + T_i$ 사이에 r보다 작은 수의 데이타가 샘플링되는 순간이 존재함을 의미한다. 이들 중 A_1에서 가장 가까운 순간을 선정하여 이를 A_k라 하자. 이때 각각의 $A_k + nT_i$ ($n=1, 2, \cdots, k_M/k_i$)의 순간들에 샘플링 되는 데이타의 갯수는 모두 같으며 r보다 작다. 샘플링 주기가 $T_{i+1} (\ge T_i)$ 인 (r+1)-번째 노드가 A_k의 순간에 샘플링하도록 하면, (r+1)-번째 노드는 $A_k + nT_{i+1}$ ($n=1, 2, \cdots, k_M/k_{i+1}$)의 순간마다 샘플링한다. T_{i+1}은 T_i의 정수배이기 때문에 이러한 순간들에 샘플링되는 데이타의 갯수 역시 r을 초과하지 않는다. (r+m)-번째 노드들 (m=2, 3, ···)에 대하여서도 A_k의 순간에 r개의 데이타가 샘플링될 때까지 위의 과정을 반복한다. 다음에 A_k에서 가장 가까우며 r보다 작은수의 데이타가 샘플링 되는 순간을 선정하여 N-번째 노드까지 샘플링 되도록 위의 과정을 되풀이 한다. 이러한 과정은 $S_i \le r,\ \forall j=1, \cdots, k_M$을 만족시킨다. T_M이 임의의 샘플링 주기 T_i에 대하여 정수배이기 때문에 만일 임의의 노드 i가 첫번째 T_M의 A_i에서 샘플링된다면, 노드 i는 다음의 모든 T_M의 A_i에서 샘플링된다. 이를 노드 1에서 N까지 반복하면 모든 T_M의 A_j에서 샘플링되는 데이타의 갯수는 S_i로 고정된다.

보조정리 2와 3에 의하여 "실행가능"한 샘플링 주기 결정 알고리듬은 다음의 정리로 요약된다.

정리 2: T_i에 대한 제어루프 i의 샘플링 주기의 비율 k_i가 다음의 조건을 만족하면, 샘플링 주기 결정 알고리듬은 "실행가능"하다.

$$(1)\ k_i = \| (\Psi_i/m_i)/(\Psi_1/m_1) \| \qquad (14)$$

$$(2)\ \alpha_K \le r \qquad (15)$$

$$(3)\ S_i \le r,\ \forall j = 1, \cdots, K_M \qquad (16)$$

여기서 $y = \| x \|$ 는 y는 x에서 가장 가까우며 x를 초과하지 않는 2^{n_i} (여기서 ni는 0을 포함하는 양의 정수) 의 정수이다.

증명: (14)는 $k_i = 2^{n_i}$이며, 제어루프 i의 루프지연시간 D_i가 한계치 Ψ_i를 초과하지 않음을 의미한다 (식 (6) 참조). (15)와 (10)은 보조정리 2와 3에 의하여 이미 증명되었다.

참고사항 3: 한 노드에서 데이타 전송시간이 L이므로 정리 2에서 여러개의 분산된 노드가 A_i의 순간에 동시에 데이타를 샘플링 한다는 조건 (3)은 이러한 노드들이 데이타 전송시간 L이내에 모두 샘플링을 완료하면 되는 조건으로 완화될 수 있다. 앞서 언급한 바와 같이 이러한 조건은 지정된 노드에서 네트워크를 통하여 동기화 메시지를 주기적으로 전송함으로써 쉽게 성취될 수 있다.

네트워크 상의 트래픽이 매우 적은 경우, 즉, $r \ge N$ 인 경우에는 다음과 같은 단순한 알고리듬이 적용

될 수 있다.

정리 3: 만일 r≥N인 경우 T_i(i=1, … , M)가 다음의 조건을 만족하면, 샘플링 주기 결정 알고리듬은 "실행가능"하다.

$$T_i = \Psi_i/m_i \tag{17}$$

증명: 임의의 노드 j (j=1, … , N)에서 $T_j \geq T_i$이므로 어느 T_i의 샘플링 주기에서든지 노드 j는 많아야 한번의 샘플링 밖에 수행할 수 없다. 따라서 T_i 동안 샘플링 될 수 있는 데이타의 최대수는 N이다. 만일 r (T_i 동안 제공되는 윈도우의 수) 이 N 보다 크면, 샘플링된 모든 데이타에서 데이타 손실 또는 공허 샘플링이 일어나지 않을 것이다 (정리 1 참조). 따라서 $T_i = \Psi_i/m_i$ 인 경우, 데이타 지연으로 인한 제어시스템의 성능 요구사항을 만족하게되고, 알고리듬은 "실행가능"하다.

참고사항 4: 네트워크의 "이용도" U는 단위시간에 대한 네트워크 사용시간의 비로 정의되며, 다음과 같이 나타낼 수 있다.

$$U = \sum_{j=1}^{N}(L/T_i) = (miL/T_i)\sum_{i=1}^{M}(1/k_i) \tag{18}$$

식(18)은 네트워크 이용도가 k_i값의 적절한 선택에 의하여 증가될 수 있음을 의미한다. 그러나 k_i의 증가는 (14)의 조건으로부터 Ψ_i 값에 제한을 받는다.

t_j를 첫번째 T_M 구간에서 노드 j가 데이타를 샘플링하는 순간이라 하면, 정리 2와 3에 의하여 실시간 분산제어시스템에서 데이타 샘플링 주기 결정 알고리듬은 다음과 같이 요약될 수 있다.

알고리듬
Given: N, L, σ, Ψ_i, m_i (i=1 to M)
Step 1: T_i 과 r 을 결정

$$T_i = \min_i [\Psi_i/m_i]$$

$$r = \mathrm{Int}\ [(T_i - N\sigma)/L]$$

Step 2: 각각의 제어루프에서 샘플링 주기를 결정
If r≥N (네트워크 부하가 작은경우), then
$T_i = \Psi_i/m_i$, ∀i=1 to M
Else

$$k_i = \left\| \frac{\Psi_i/m_i}{\Psi_1/m_1} \right\| \quad , \ \forall i=1 \text{ to } M$$

$$\alpha_K = \sum_{j=1}^{M}(m_j/k_j)$$

If α_K 〉 r, then
(네트워크는 과부하 상태 ⇒ 노드 수를 줄인다)
Else
$T_i = k_i T_1$
Endif
Endif

Step 3: 첫번째 T_M 구간에서 샘플링 순간 tj 결정.
For (i=1, j=1; i≤k_M, j≤N; i=i+1, j=j+1)
Do tj← A_i
while ($S_i \leq r$)
End

Ⅳ. 실시간 분산제어시스템 설계의 예

이 장에서는 5 개의 제어루프와 10 개의 데이타 전송노드 (5개의 콘트롤러와 5개의 프랜트) 로 구성된 실시간 분산제어시스템의 설계에 대한 예를 소개한다. 노드 (1, 2), (3, 4), (5, 6), (7, 8), (9, 10)들이 제어 루프 1, 2, 3, 4, 5 에 포함되는 것으로 한다. 패킷화된 데이타의 전송시간(L)과 각 노드에서 서버 처리시간(σ)은 각각 2 msec와 0.1 msec이며, 각각의 제어루프에서 데이타 지연 최대 허용시간은 다음과 같다.

[Ψ_1, Ψ_2, Ψ_3, Ψ_4, Ψ_5] = [20msec, 60msec, 100 msec, 200msec, 400msec]

알고리듬의 step 1으로부터 주어진 실시간 분산제어 시스템의 T_i과 r은 각각 10msec과 4로 주어진다. 따라서 step 2에서 벡터 K는 다음과 같이 결정된다.

[k_1, k_2, k_3, k_4, k_5] = [1, 2, 4, 8, 16]

주어진 K에서 α_K는 3.875로 정해지며 (α_K〈r), 따라서 샘플링 주기 벡터 T는 다음과 같이 결정된다.

[T_1, T_2, T_3, T_4, T_5] = [10msec, 20msec, 40 msec, 80msec, 160msec]

Step 3에서 노드 1에서 10의 첫번째 데이타 샘플

링 순간은 다음과 같이 결정된다.

$[t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}]$ =
[0msec, 0msec, 0msec, 0msec, 10msec, 10 msec, 30msec, 30msec, 70msec, 70msec]

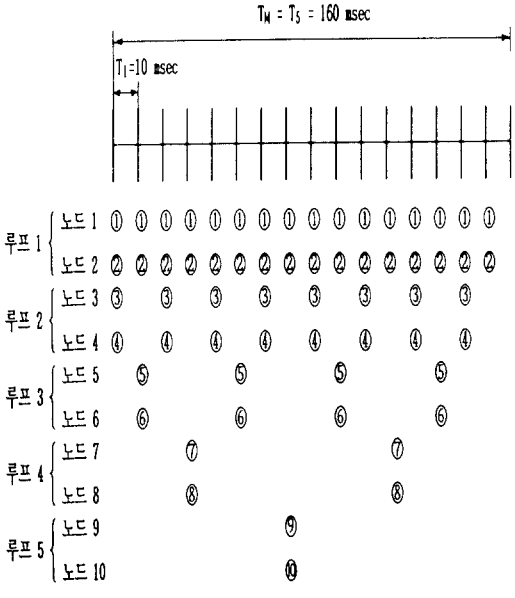


그림 8. 실시간 분산제어시스템에서 샘플링 순간
Fig. 8. Sampling Instants in the Real-Time Distributed Control Systems.

그림 8에 첫번째 T_M 구간에서 10개의 노드의 데이타 샘플링 순간이 주어졌다. 여기서 ①, ②, …, ⑨, ⑩은 각각 노드 1, 2, 3, 9, 10의 데이타 샘플링 순간을 나타낸다. 각각의 T_1 주기동안 네트워크에서 생성되는 데이타의 갯수가 r(=4)을 초과하지 않음을 주의하라. 그림 8에서 주어진 것과 정확히 같은 데이타 샘플링 형태가 다음의 모든 T_M의 주기에서 반복된다. (3.14)로부터 주어진 분산제어시스템에서 네트워크 이용도는 77.5% 이다 (네트워크 이용도 중 사용되지 않은 부분은 각 노드에서 서버의 처리시간 σ로 인한 불가피한 손실도 포함된다). 시스템 이용도 U_S를 T_M의 시간 동안 제공된 윈도우의 수에 대한 실제로 사용된 윈도우의 수의 비율로 정의하면 (즉, $U_S = \alpha_K/r$), 주어진 실시간 분산제어 시스템의 예에서 시스템 이용도 U_S는 96.875%이다.

실시간 분산제어시스템에서 제어성능은 이산사건/연속시간 시뮬레이션 모델에 의한 모의실험에 의하여 조사되었다. 이산사건/연속시간 시뮬레이션 모델은 (1) 네트워크 시스템을 모델링한 이산사건 시뮬레이션 모델과 (2) 프랜트의 연속시간 모델과 콘트롤러의 이산시간 모델로 구성된 제어시스템 모델을 통합한 것으로 이루어진다. 네트워크 시스템 모델은 다시 (1) 데이타를 생성하는 부모델과 (2) 순환 (cyclic) 서비스를 통하여 데이타를 전송하는 부모델의 서로 독립된, 그러나 상호 작용을 하는 두개의 부모델로 구성된다. 모의실험에 사용된 네트워크의 구성은 다음과 같다.

o 노드 1, 3, 5, 7 ,9는 각각 제어루프 1, 2, 3, 4, 5 의 센서/액튜에이터 노드이다. 즉, 이들 노드의 전송 큐는 센서와 연결되어 있고, 수신 큐는 액튜에이터와 연결되어 있다.

o 노드 2, 4, 6, 8 ,10은 각각 제어루프 1, 2, 3, 4, 5 의 콘트롤러 노드이다. 즉, 이들 노드의 전송 큐는 제어 신호를 처리하고, 수신 큐는 센서 데이타를 처리한다.

제어루프 1의 디지탈 콘트롤 시스템에서, 플랜트 전달함수는 다음과 같으며,

$$G_p(s) = \frac{1}{(0.3s+1)(0.03s+1)}$$

디지탈 콘트롤러에서의 전달함수를 이에 대응하는 아나로그 값으로 표시하면 다음과 같다.

$$G_c(s) = \frac{5(s+5)}{s}$$

여기서 s는 Laplace 변환 변수이다.

제어루프 1에서 단위 스텝의 기준입력에 대한 제어신호와 플랜트 출력의 시간에 대한 변화가 그림 9와 10에 나타나 있다.

그림에서 NORMAL은 제어루프 1에서의 데이타 샘플링 주기가 위에서 제시한 샘플링 주기 결정 알고리듬에 의하여 결정된 경우 (즉, 10 msec)이고, CASE 1과 CASE 2는 각각 제어루프 1에서의 샘플링 주기가 5msec과 13 msec인 경우 (즉, NORMAL의 경우보다 샘플링 주기가 작거나 큰 경우) 이다. 다른 제어 루프들에서의 샘플링 주기는 위에서 제시한 샘플링 주기 결정 알고리듬에 의하여 결정된 값들을 갖는다. 샘플링 주기가 NORMAL보다 큰 CASE 2에서 13 msec의 샘플링 주기는 당연히 제어시스템의 성능을 저하시킨다. CASE 1 (5 msec

의 샘플링 주기)에서 플랜트 출력에 대한 제어 시스템의 성능은 NORMAL의 경우에 비하여 향상된다. 그러나 5 msec의 샘플링주기는 데이타 손실과 공허 샘플링으로 인하여 그림 9에서 보는 바와 같이 제어 신호의 찌그러짐 현상을 야기시킨다. 이러한 제어 신호의 고주파수 잡음은 액튜에이터의 급속한 마모를 초래할 수도 있다.

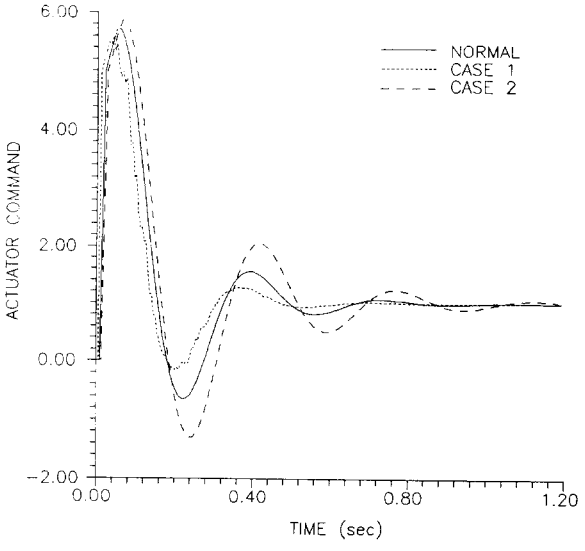


그림 9. 제어 신호: CASE 1과 2
Fig. 9. Transient Response of Controller Signal: CASE 1 and 2.

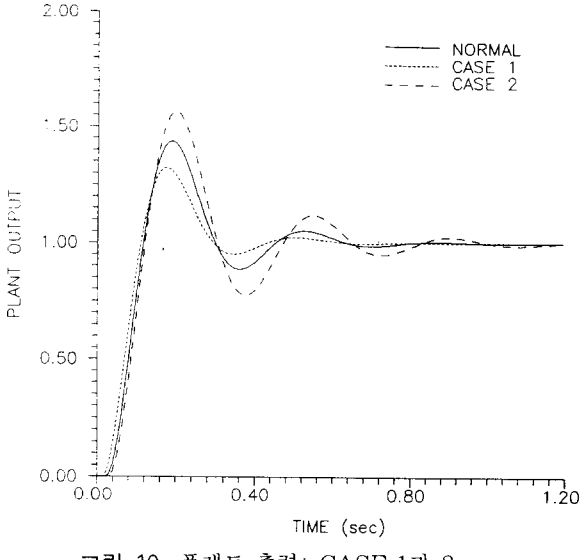


그림 10. 플랜트 출력: CASE 1과 2
Fig. 10. Transient Response of Plant Output: CASE 1 and 2.

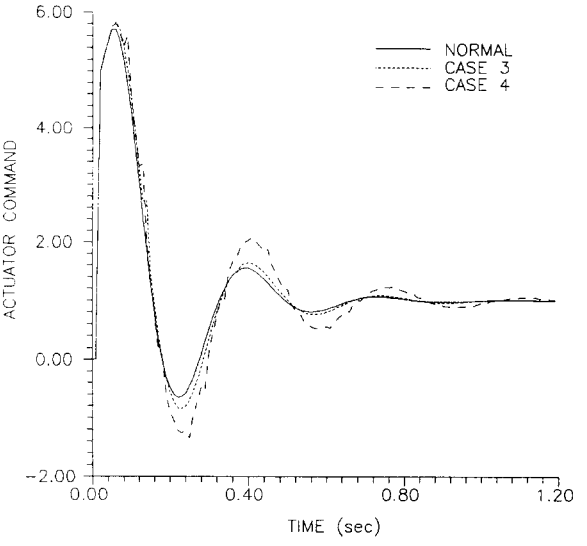


그림 11. 제어 신호: CASE 3과 4
Fig. 11. Transient Response of Controller Signal: CASE 3 and 4.

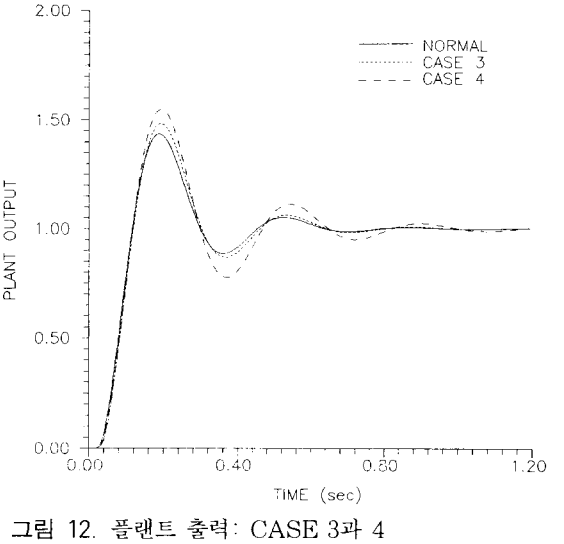


그림 12. 플랜트 출력: CASE 3과 4
Fig. 12. Transient Response of Plant Output: CASE 3 and 4.

다른 제어 루프들의 샘플링 주기가 제어루프 1에 미치는 효과가 그림 11과 12에 나타나 있다. CASE 3에서 제어루프 1, 2, 3, 4, 5에서의 샘플링 주기는 각각 10 msec, 17 msec, 55 msec, 107 msec, 135 msec이며, 이때 네트워크 이용도 U는 NORMAL의 경우와 동일하다. CASE 4에서 제어루프 1, 2, 3, 4, 5의 샘플링 주기는 각각 10 msec,

10 msec, 20 msec, 40 msec, 80 msec이며, 이때 네트워크는 과부하 상태이다. CASE 3과 4의 경우 모두 데이타 손실과 공허 샘플링으로 인한 제어신호의 찌그러짐 현상이 일어난다. 네트워크가 과부하 상태인 경우(CASE 4) 플랜트 출력의 성능은 당연히 감쇠한다. 네트워크가 과부하 상태가 아닌 경우에도 각각의 제어 루프에서의 데이타 샘플링 주기가 적절히 결정되지 않은 경우(CASE 3) 다른 제어 루프의 샘플링 주기에 의하여 제어루프 1의 성능이 감쇠됨을 알 수 있다.

V. 결론

실시간 분산제어시스템에서 제어 성능은 네트워크로 인한 데이타의 지연시간에 대하여 직접적으로 영향을 받는다. 시간에 따라 불규칙하게 변하는 특성을 가지는 네트워크 지연시간은 데이타 손실과 공허 샘플링 등의 현상을 초래할 수 있다. 본 논문에서 제시된 데이타 샘플링 주기 결정 알고리듬은 실시간 분산제어 시스템에서 (1) 루프 지연시간을 허용치 이하로 제한하고, (2) 데이타 손실과 공허 샘플링 현상을 제거하며, (3) 네트워크의 이용도를 크게 증가시킬 수 있다. 기존의 실시간 분산제어시스템에서는 제어 시스템의 실시간 성능 요구사항을 만족시키기 위하여 네트워크에 수용되는 노드의 수를 제한하였으며 (정리 3에서 $N < r$ 의 경우), 따라서 네트워크의 용량을 제대로 활용하지 못하는 경우가 많았다. 본 논문에서는 실시간 분산제어시스템의 제어성능 요구사항을 만족시키는 동시에 네트워크의 이용도를 크게 증가시키는 샘플링 주기 결정 알고리듬을 제시하였다.

參考文獻

[1] IEEE Network Magazine: Special Issue on Communication for Manufacturing, vol.2, no.3, May 1988.

[2] A. Ray, Y. Halevi, S. Lee, S. H. Hong, "Network Access Protocols for RealTime Distributed Digital Control Systems," *IEEE Industrial Application Society Annual Meeting*, Denver, CO, September 1986.

[3] J. W. Meyer, "SAE AE-9B Draft Standard High Speed Token Passing Data Bus for Avionic Applications," *IEEE/AIAA 7-th Digital Avionic Systems Conference*, Fort Worth, TX, pp. 234-241, October 1986.

[4] R. K. Jurgen, Senior Editor, "Coming from Detroit: networks on wheels," *IEEE Spectrum*, pp. 53-59, June 1986.

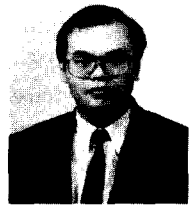
[5] A. Ray, S. H. Hong, S. Lee and P. J. Egbelu, "Discrete-Event/Continuous-Time Simulation of Distributed Data Communication and Control Systems," *Trans.* of the Society for Computer Siumlation, vol. 5, no.1, pp. 71-85, January 1988.

[6] J. F. Kurose, M. Schwartz and Y. Yemini, "Controlling Window Protocols for Time-Constrained Communication in Multiple Access Networks", : *IEEE Transactions on Communications*, vol.36, no.1, pp. 41-49, January 1988.

[7] K. W. Rose and B. Chen, "Optimal Scheduling of Interactive and Noninteractive Traffic in Telecommunication Systems," *IEEE Transactions on Automatic Control*, vol.33, no.3, pp.261-276, March 1988.

[8] A. Ray and Y. Halevi, "Integrated Com-munication and Control Systems: Part I- Analysis and PartII-Design Consider-ations," ASME Journal of Dynamic Systems, Measurement and Control, December 1988.

著 者 紹 介



洪 承 鎬 (正會員)

1956年 5月 31日 生. 1982年 2月 연세대 기계공학과. (공학사) 1985年 5月 Texas Tech University. Mechanical Engineering (M.S.) 1989年 8月 The Pennsylvania State University. Mechan-ical Engineering (Ph.D.) 1989年 5月 - 1992年 2月 한국전자통신연구소 선임연구원. 1992年 3月 - 현재 한양대학교 제어계측공학과 조교수. 주관심분야는 실시간분산제어. 자동화시스템. 컴퓨터 네트워크 등임.