# 실시간 통신을 위한 새로운 Timing 구조

김 경 재, 신 동 렬
성균관대학교 전기전자 및 컴퓨터공학부
전화 : (0331) 290-7232 / 팩스 : (0331) 290-7231

# A new timing structure for a realtime communication

Kyeung Jae Kim, Dong Ryeol Shin
School of Electrical & Computer Engineering Sungkyunkwan University
E-mail : kyeung@nature.skku.ac.kr
drshin@ece.skku.ac.kr

## Abstract

This paper presents a new timing structure for real time communications and its performance analysis. The cycle time consists of several "one time slot" which may be an interval defined by a synchronous traffic part followed by an asynchronous traffic part. If a station receives a token within a synchronous interval, it transmits its synchronous message if any, otherwise it may transmit an asynchronous message. This scheme is different from usual allocation schemes which divide one cycle into alternating synchronous and asynchronous subslots. This protocol is designed to prevent low priority messages from delaying too much due to lots of high priority messages. We propose the algorithm and show its justification by simulation

## Ⅰ. 서론

최근 수년간에 걸쳐 컴퓨터 및 통신 기술이 급속하게 발전하면서 통신망을 이용한 자동화 시스템의 보급이 크게 증가하고 있다[1,2]. 1980년대 후반이후에 센서 기술의 발달과 컴퓨터를 이용한 데이터 처리 능력의 발전으로 대형의 복잡한 제어시스템을 실시간으로 운용하는데 있어서 막대한 양의 정보를 적절히 처리할 수 있는 기술이 요구되었다. 예를 들어 화학공정이나 자동화 시스템에서는 무수히 많은 센서들이 설치되어 있을 것이다. 이러한 센서들에 의해 감지한 데이터는 제어 컴퓨터에 전송된다. 따라서 데이터를 적시에 받아들이고 적절하게 분배하는 데이터 전송기법이 필요하게 되었다. 이러한 문제점을 해결하기 위해 대형의 복잡한 제어시스템을 여러 개의 분산된 시스템으로 모듈화하고 각각의 부 시스템들을 제어하는 컴퓨터들을 네트워크로 연결하는 방법으로 확산되고 있다. 따라서 생산시스템분야에서는 이 기종의 장비들간의 통신을 위해 표준화가 필요했다[3]. 이에 필드버스가 대두되었다. 필드버스란 컴퓨터를 이용한 자동제어 시스템분야에서 필드 압출력 디바이스(센서, 엑츄에이터, 신호변환기, 컨트롤러)등의 데이터 전송에 대한 네트워크 프로토콜이다. 기존의 계측 기기는 4~20㎃의 아날로그 신호들을 그대로 사용하거나 RS232와 같은 통신장비를 통해 point-to-point 방식으로 연결되었다. 그러나 하나의 데이터를 여러 장비에 전송하기 위해서는 각각에 대해 따로 전송해야 함으로 배선의 양이 많아지고 그만큼 복잡한 배선에 대해 유지, 보수가 어렵게 되었다. 이에 모든 장비들이 하나의 버스를 공유하고 변조된 디지털 데이터 전송으로 구성된 필드버스가 나타났다. 필드버스는 실시간으로 필드 장비들과의 통신이 보장되어야 하기 때문에 효율적인 데이터관리를 위해 스케줄링 기법이 필요하다[4]. 본 논문은 모든 장비가 시간에 따라 모든 데이터를 실시간으로 처리될 수 있

도록 하기 위해 효율적인 메시지 관리에 대해 소개하고 그에 따라 각각의 장비에서 발생되는 데이터의 상태를 알아볼 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 제시한 timing 구조와 그것에 따라 데이터의 상태를 알아보고 3장에서는 시뮬레이션 방법 및 결과를 통해 기존의 성능과 비교할 것이다. 그리고 4장에서는 본 논문의 결론을 제시할 것이다.

## Ⅱ. 제안된 Timing 구조와 성능 분석

기존에 제안된 timing 구조는 토큰을 받은 노드는 먼저 high priority data를 처리한 후에 low priority data를 처리하였다[5-7]. 그러나 high priority data의 발생주기가 아주 짧다면 low priority data의 지연시간이 계속해서 커질 것이다. 본 논문에서 제시된 구조는 high priority data는 약간 지연되더라도 deadline이전에만 처리되도록 하고 low priority data를 처리하여 그것의 지연시간을 단축한다.

Time slot을 동기와 비동기 시간으로 구간을 나눠서 고려한다. 토큰을 받은 모든 노드들은 Token Holding Time(THT)이 유효하고 동기 시간일 경우에는 high priority data를 우선적으로 전송하고, 비동기 시간일 경우에는 low priority data를 우선적으로 전송한다. 그림1 은 제안된 timing 구조를 나타낸 것이다.

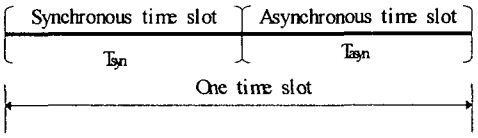


그림 1 Timing diagram

**Nomenclature**

M : 통신매체에 연결된 노드의 수

r : 토큰이 현재 노드에서 다음 노드로 이동할 때 걸리는 시간

$T_c$ : Token cycle time

$S_i^{s(a)}$ : i 노드에서 모든 high(low) priority data를 처리하는데 걸리는 시간

$T_{ov}$ : Token overhead time

$T_i^H$ : i 노드에서의 token holding time

$\lambda_i$ : i 노드에서의 inter-arrival rate

$L_i^{h(l)}(t)$ : t 시간에서 전체 시스템을 봤을 때 i 노드의 high(low) priority queue length

$\mu_{i,s}^{(h,l)}$ : $T_{syn}$ 에서의 high(low) priority data service rate

$\mu_{i,a}^{(h,l)}$ : $T_{asyn}$ 에서의 high(low) priority data service rate

$E_i^{(h,l)}$ : i 노드가 하나의 high(low) priority data를 실행하는데 걸리는 시간

$x_i^{(h,l)}$ : i 노드가 토큰을 받은 후에 $T_i^H$동안 처리할 수 있는 high(low) priority data 수

$A_i(t)$ : t 시간동안 i 노드에 들어온 데이터 수

$D_i^{h(l)}(t)$ : t 시간동안 i 노드에서 처리된 high(low) priority data 수

그림1 에서 one time slot에 할당할 수 있는 최소시간 min( $T_{slot}$)은 다음과 같다.

$$\min(T_{slot}) = \sum_i \max(S_i^s) + \sum_i \max(S_i^a) + 2*n*T_{ov}$$
$$= \sum_i (\max(S_i^s + S_i^a)) + 2*n*T_{ov}$$

One time slot이 할당되었을 때 어느 특정한 시간에 대해 시스템 성능을 분석해 보자. 먼저 token cycle time에 대해 알아보자. $T_c$동안에 메시지가 발생할 확률은 $\int_{i=0}^{T_c} \lambda_i e^{-\lambda_i t} dt$ 이므로 다음과 같은 식이 성립한다[8].

$$T_c = M*r + \sum_{i=1}^{M} S_i * \int_0^{T_c} \lambda_i e^{-\lambda_i t} dt$$
$$\therefore \sum_{i=1}^{M} S_i e^{-\lambda_i T_c} + T_c = \sum_{i=1}^{M} S_i + C \quad (1)$$

그리고 $T_c$의 경계선은 다음과 같다.

$$T_{ring} + T_{token} \le T_c \le T_{ring} + T_{token} + \sum_{i=1}^{M} S_i \quad (2)$$

Service time 분포만 알려지면 (1)식과 (2)식을 만족하는 $T_c$를 구할 수 있다. 특정 시간에서 특정 노드의 상태변화는 다음과 같다.

$$L_i^{h(l)}(t) = A_i(t) - D_i^{h(l)}(t) \quad (3)$$

T 시간동안 노드가 토큰을 받은 회수가 n 일 때,

$$A_i(t) = \lambda_i T_{c,1} + \lambda_i T_{c,2} + \cdots + \lambda_i T_{c,n}$$
$$= \lambda_i (T_{c,1} + T_{c,2} + \cdots + T_{c,n}) \quad (4)$$

이다. $D_i^{h(l)}(t)$는 측정하려는 시간이 전체 시스템의 동기 시간 구간에 있는지 아니면 비동기 시간 구간에 있는지 판별하여 그에 맞는 service rate를 적용하여 구할 수 있다.

노드가 토큰을 받은 후 THT동안에 처리할 수 있는 데이터 수, $x_i^{(h,l)}$는 다음과 같이 구할 수 있다.

$$x_i^{(h,l)} = \left\lfloor \frac{T_i^H}{E_i^{(h,l)}} \right\rfloor \quad (5)$$

이때 (5)식과 각 노드에서의 token cycle time에 token holding time을 더한 시간동안 처리될 수 있는 데이터 수의 비를 고려하면 각 노드의 service rate를 구할 수 있다. (6)은 동기, (7)는 비동기 시간 구간일 때의 high(low) priority data service rate를 나타낸 것이다.

$$\mu_{i,s}^{h}=\frac{x_i^h}{T_c+T_i^H}\ ,\ \mu_{i,s}^{l}=\frac{x_i^h-L_i^h}{T_c+T_i^H} \tag{6}$$

$$\mu_{i,a}^{h}=\frac{x_i^h-L_i^h}{T_c+T_i^H}\ ,\ \mu_{i,a}^{l}=\frac{x_i^l}{T_c+T_i^H} \tag{7}$$

단, $\mu_{i,s}^{l}$와 $\mu_{i,a}^{h}$은 각각 $x_i^h-L_i^h>0$ 와 $x_i^l-L_i^l>0$ 인 조건을 만족해야 한다. 그렇지 않으면 $\mu_{i,s}^{l}$ = $\mu_{i,a}^{h}$ = 0 이다.

결과적으로 (6),(7)식을 이용하면 $D_i^{h(l)}(t)$는 다음과 같이 계산된다.

$$\begin{cases} m*T_{slot} \le t < m*T_{slot}+T_{syn} \text{ 일때,} \\ \quad D_i^h(t)=t*\mu_{i,s}^{h} \quad , D_i^l(t)=t*\mu_{i,s}^{l} \\ m*T_{slot}+T_{syn} \le t < (m+1)*T_{slot} \text{ 일때,} \\ \quad D_i^l(t)=t*\mu_{i,a}^{l} \quad , D_i^h(t)=t*\mu_{i,s}^{h} \end{cases} \tag{8}$$

따라서 (4)식과 (8)식에 의해 (3)식은 다음과 같이 구할 수 있다.

$$\therefore L_i^{(h,l)}(t) =$$

$$\begin{cases} m*T_{slot} \le t < m*T_{slot}+T_{syn} \text{ 일때,} \\ \quad \lambda_i(T_{c,1}+T_{c,2}+\cdots+T_{c,n})-t*\mu_{i,s}^{(h,l)} \\ m*T_{slot}+T_{syn} \le t < (m+1)*T_{slot} \text{ 일때,} \\ \quad \lambda_i(T_{c,1}+T_{c,2}+\cdots+T_{c,n})-t*\mu_{i,a}^{(h,l)} \end{cases} \tag{9}$$

다음으로 분석할 성능은 생성된 데이터가 노드에서 대기하고 있는 시간이다. 특정 시간에 특정 노드를 측정했을 때 그에 대한 data waiting time, $\overline{w_i}(h,l)$,도 두 개의 시간 구간으로 나누어 계산된다.
동기 시간 구간일 때

$$\overline{w_i^h}=\overline{x_i^h}*\overline{T_c}\ ,\ \overline{w_i^l}=(\overline{x_i^l}+\overline{x_i^h})*\overline{T_c} \tag{10}$$

비동기 시간 구간일 때

$$\overline{w_i^l}=\overline{x_i^l}*\overline{T_c}\ ,\ \overline{w_i^h}=(\overline{x_i^l}+\overline{x_i^h})*\overline{T_c} \tag{11}$$

으로 간단히 정의할 수 있다.

## Ⅲ. 시뮬레이션

가정

1. 각 노드에 도착하는 (비)동기 데이터의 inter-arrival rate는 알고 있다.
2. 동기 데이터는 deadline을 갖는 high priority data이고 비동기 데이터는 지연시간이 허용되는 low priority data이다.
3. 각 노드는 high priority와 low priority data를 처리하기 위한 queue가 각각 존재한다.

Workload

각 노드에 발생한 high(low) priority data가 평균적으로 처리되기까지의 기다리는 시간(mean waiting time in each queue)과 각 노드에서 high(low) priority queue의 평균 길이(each mean queue length in node)를 측정한다.

입력변수를 기존의 timing 구조와 제안된 timing 구조에 일정하게 할당하고, inter-arrival rate를 노드1에만 크게 할당했을 때 두 timing 구조에 따라 변화하는 workload를 알아보자.

그림2와 그림3을 통해 기존의 timing 구조는 inter-arrival rate이 커지면 low priority data에 대한 성능이 계속해서 감소하고 있으나 제안된 timing 구조는 high priority data와 마찬가지로 안정적임을 알 수 있다. 또한 그림4와 그림5를 통해 제안된 timing 구조는 다른 노드에 대해서도 영향을 미치지 않음을 알 수 가 있다.

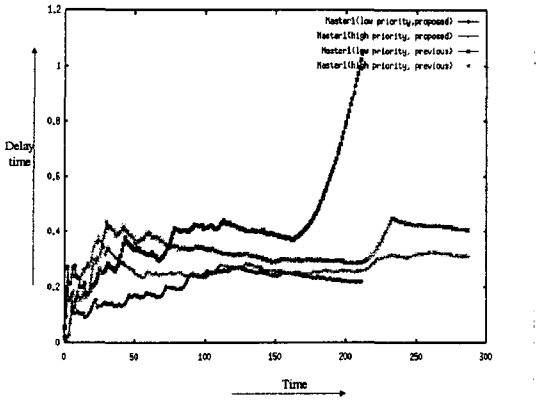


그림 2 각 노드1에서 기존의 timing 구조와 제안된 timing 구조의 지연시간 비교

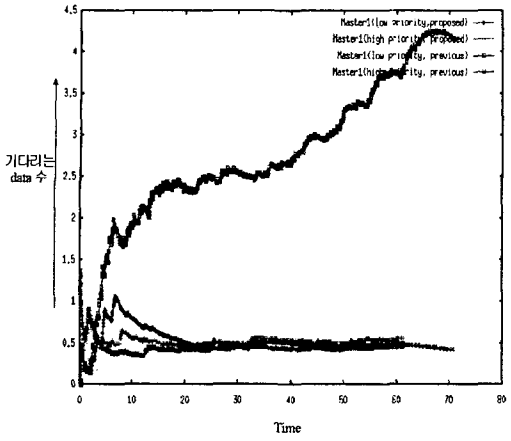


그림 3 노드1에서의 기존의 timing 구조와 제안된 timing 구조에 대해 노드에 기다리는 데이터의 수 비교

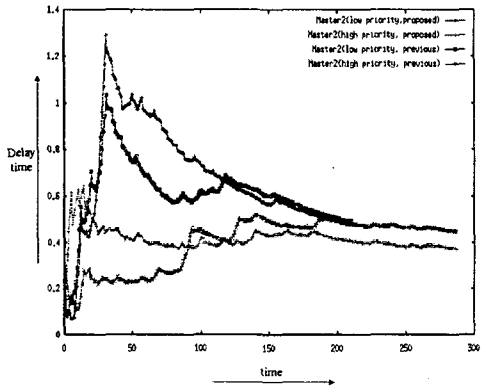


그림 4 각 노드2에서 기존의 timing 구조와 제안된 timing 구조의 지연시간 비교

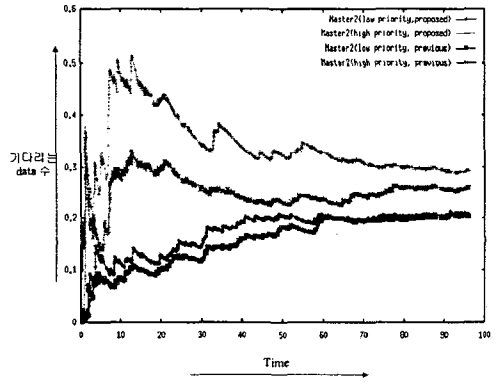


그림 5 노드2에서의 기존의 timing 구조와 제안된 timing 구조에 대해 노드에 기다리는 데이터의 수 비교

## Ⅳ. 결론

기존의 timing 구조와 제안된 timing 구조를 시뮬레이션을 통해 비교해 본 결과 각 노드의 inter-arrival rate이 작으면 기존의 timing 구조와 비슷하지만 inter-arrival rate이 커지면 기존의 timing 구조에 비해 제안된 timing 구조가 더 안정적임을 볼 수가 있다. 실시간 data뿐만 아니라 비 실시간 data도 보장할 수 있다.

현재, 실시간 통신에서 효율적인 메시지관리에 대한 중요성이 강조되고 있다. 이에 본 논문은 실시간 통신을 위한 새로운 timing 구조를 제시하고 그것에 따라 각 노드에 대한 성능을 분석하고 시뮬레이션을 통해 증명하였다.

## 참고문헌

[1] "Special Issue on Communication for Manufacturing", IEEE Network Magazine, Vol. 2, No. 3, 1988

[2] R. Piementel, "Communication Networks for Manufacturing", Prentice Hall, 1990.

[3] J. R. Jordan, "Serial Networked Field Instrumentation", John Willey & Sons, 1995.

[4] InTech, "Field Buses Special Issue", ISA Publication, Nov, 1996.

[5] P. Pleinevanx, J.D. Decotignic, "Time Critical Communication Networks", IEEE Network Vol. 2, No. 3, 1988

[6] G. Agrawal, B. Chen, W. Zhao, S. Davari, "Guaranteeing Synchronous Message Deadline with the Timed Token Medium Access Control Protocol", IEEE Trans. On Computers, Vol. 43, 1994

[7] Adel Ben Mnaouer 외 4명, "Asynchronous Bandwidth Allocation and Parameter Setting in the Feildbus Protocol", IEE Japan, Vol. 117-C, No. 7, 1997

[8] Kang G. Shin, "Real-Time Communications in a Computer-Controlled Workcell", IEEE Trans. on robotics and automation, Vol. 7, No. 1, Feb 1991