Proceedings of the 11th
KACC, October 1996

필드 버스 시스템을 위한 네트웍 스케쥴링 알고리즘

Network scheduling algorithm for field bus system

추 성 호, 김 일 환**

*강원대학교 제어계측공학과 (Tel: 0361-51-6501; Fax: 0361-242-2059; E-mail: somebody@cie.kangwon.ac.kr)
**강원대학교 제어계측공학과 (Tel: 0361-50-6347; Fax: 0361-242-2059; E-mail: ihkim@cc.kangwon.ac.kr)

Abstracts In field bus network, field device are connected with a medium. Because a medium must be shared for transmitting data, there are random delay time when data arrive destination station. It is difficult that all data packets are guaranteed synchronization and real-time restriction. In this paper, we show an algorithm that makes network utilization to maximum, guarantees real-time restriction, calculates sampling time at all control loop.

Keywords fieldbus network scheduling algorithm profibus FIP CAN

1. 서론

공정 제어를 위한 LAN은 최근 들어 폭발적으로 늘어났다. 또한 분산 아키텍처에 의해, 생산 라인들은 작업 부분 별로 독립되어있지만, 서로 연계하여 협력하는 시스템으로 변화하여 가고 있고, 지능화된 기기들로 대체하므로써 그런 작업이 가능하게 되어 가는 추세이다.

1.1 MAP (Manufacturing Automation Protocol)

생산 공정이 점차 자동화되어짐에 따라, 이미 설치된 로컬 네트웍 간의 비호환성은 설치·유지·보수·운용·재사용면에서 많은 비용 발생을 유발시키게 되었다. 생산되는 제품의 사양이 달라질 때마다 생산 라인의 기기들을 재배치하고 상호 연결에 만 들어가는 비용이 공장에 투자되는 전체 예산의 주요 부분을 차지하게 되었고, 그 비용의 축소를 위해 일반화된 연결 방식과 재배치에 따른 자원 소비 감축을 위해, 기기 간의 통일된 연결 규칙이 필요하게 되었다. 이에 따라 공장 플랜트 기기 간에 통신을 위해 MAP이 등장하게 되었다. OSI 계층 구조 모델을 따랐고, 각 계층 (layer) 에 해당하는 MMS, ASN.1, LLC, MAC[1] 등의 통신규약을 정의했다. [3]

하지만 MAP의 경우 다음과 같은 문제점이 있다.

- 제어시스템의 최하위 레벨의 기기까지 연결은 효율 면에서 비용 면에서 현실성이 없다.
- 비용이 많이 든다.
- 실시간 응용에서는 적당치 않다.
- 일반적으로 기간 (backbone) 네트웍이나 프랜트 또는 공정 단위 네트웍에 적당하다.

1.2 필드버스

OSI 모델에서도 기술되었듯이 생산 공정의 모든 상황에 7 계층 전부가 필요하지는 않다. 특히 실시간 환경과 근접한 공정 단위에서 같은 버스에 물린 기기들 간의 통신에서는 중간 계층을 생략하는 것이 효율이나 속도 면에서 많은 이득이 되었다. 분산 제어 시스템의 비교적 하위 공정의 컴퓨터(제어기)·센서·엑튜에이터끼리의 데이터 교환은, OSI Layer 1, 2, 7의 정의만으로도 충분했다. 또한, 필드에 위치한 기기 - 필드 기기 - 들이 점차 지능화되어 짐에 따라, 분산 처리를 지향되어, 데이터의 수집·처리는 기기들이 설치되어 있는 필드에서 처리되었고, 그에 따른 필드 기기간 통신을 위한 필드버스 (Fieldbus) 표준이 마련되었다. 필드버스는 아직 단일화된 규격은 없고, Profibus, FIP, CAN, Bitbus 등 여러 국가 표준 수준의 규약이 있다. [2, 9] 각 규약은 조금씩 다른 운용 방식을 택하고 있다.

필드버스의 일반적인 특징은 다음과 같다.

- 적은 비용으로 많은 기기를 연결할 수 있다.
- 패킷의 크기가 작아지므로 공정 호환성이 증가된다.
- fault를 검출하기 쉬우며, 그로 인해 시스템이 안정적이다.
- 트위스트 페어 선을 이용하며, 전류 변조를 한다.
- 전자기적 노이즈에 비교적 강하다.
- 1000 m 이상의 거리도 연결 가능하다.
- 연결구조가 간단하므로 유지·변경 작업이 용이하다.

1.3 네트웍 운용

여러 기기가 하나의 통신매체를 공유함에 따라, 제한된 자원의 바람직한 분배 방식을 고려하게 되었다. 필드에 있는 제어기, 센서, 엑튜에이터 등으로 구성된 각 제어 루프의 신뢰성을 보장하기 위해서는, 주기적 또는 비주기적으로 생성되어지는 센싱 데이터, 제어 신호, 네트웍 운용에 필요한 데이터 등 통신매체를 통해 전달되는 여러 종류의 데이터가 손상, 손실, 지연되는 일이 없게 하거나, 최소한으로 하는 것이 중요하다. 제한된 통신매체의 대역폭 (bandwidth) 안에 이런 모든 데이터들이 원활하게 교환되게 하기 위해서는, 네트웍 운용에 있어서 일단의 전략이 필요하게 되었고, 그로 인해 필드기기들과 통신매체를 포함한 전체 시스템 운용의 스케쥴, 네트웍 스케쥴링 문제가 대두되었다.

2. 기존의 연구

[8]에서는 일반적인 Timed-Token Protocol에서 각 스테이션의 Token Holing Time을 결정하는 알고리즘을 적용했다. 이 알고리즘 만으로는 주기적으로 발생하는 샘플링 데이터를 적정 시간내에 처리하는 데는 문제가 있다. [7]에서는 주기적 데이터가 생성되는 다중큐 시스템에서의 안정할 조건을 제시하였다. [10]에서는 우선도를 감안하여 필드버스 데이터 링크 레이어의 설계에 대해서 논의했고, [5]에서는 필드버스 시스템의 전체 샘플링 주기를 구하여, 네트웍 부하가 대역폭 내에 있는 지를 판

1) Manufacturing Message Specification, Abstract Syntax Notation no. 1, Logical Link Control, Medium Access Control

별하도록 했다. [11, 12]에서는 윈도우 프로토콜[6]을 이용하여 각 제어 루프에서 발생하는 데이터들을 일정한 주기로 배치하므로써 루프 지연 시간을 최소로 하는 방법을 제시하였고, [7]의 조건을 이용하여 그 방법이 안정하다고 주장하였다. 하지만, 실시간 비주기적인 데이터와 비실시간 비주기적 데이터에 대한 네트웍 사용 시간에 대한 언급 부족하고, 센서 노드가 하나인 제어 루프만 고려하였으므로 일반적인 시스템에 그 알고리즘을 적용함에 있어서 제약을 받는다.

3. 필드버스 시스템

필드버스 시스템에는 센서 노드, 제어 노드, 엑츄에이터 노드 이렇게 3가지 타입의 노드가 있다. 센서에서 샘플된 데이터는 측정 데이터라고 하고, 그 데이터를 토대로 제어기에서 계산된 값은 제어 데이터라고 하겠다.

3.1 구성

3.1.1 센서 노드

센서 노드는 필드버스 시스템에서 다음과 같은 역할을 한다.

- 플랜트로부터 데이터를 샘플링한다.
- 샘플된 데이터 - 측정 데이터 - 를 전송 큐에 삽입한다.
- 토큰을 받거나, 전송 요구 신호를 받으면 제어 노드로 측정 데이터를 보내고, 그렇지 않은 경우는 토큰 혹은 전송 요구 신호를 기다린다.

센서 노드는 필드 버스 프로토콜에 따라 토큰을 가질 수 있는 경우(예, FIP)와 일반적으로는 가질 수 없는 경우(예, Profibus에서 슬레이브 스테이션으로 설정된 경우)가 있다.

3.1.2 제어 노드

제어 노드에는 센서 노드로부터 주기적 데이터, 비주기적 데이터, 비실시간 데이터가 들어온다. 제어 노드의 역할은 다음과 같다.

- 각 센서 노드에서 측정 데이터를 취합한다.
- 입수된 데이터를 계산하여 제어 데이터를 생성시킨다.
- 계산된 제어 데이터를 전송 큐에 넣는다.
- 토큰을 가진 상태라면 전송 큐에 있는 제어 데이터를 엑츄에이터 노드로 전송하고 그렇지 않은 경우는 토큰을 기다린다.

3.3 엑츄에이터 노드

엑츄에이터 노드는 다음과 같은 역할을 한다.

- 제어 노드로부터 제어 데이터를 받는다.
- 플랜트로 제어 데이터를 출력한다.

3.2 운용 방식

3.2.1 CiT (Circulated Token)

토큰 패싱 타임 메커니즘으로, 여러 대의 마스터 스테이션들 사이에서 토큰이 순환된다. 토큰을 가진 스테이션 만이 네트웍 자원의 접근이 가능하다. 한 순환 주기 동안 각 스테이션들은 토큰을 한번 받을 수 있으며, 토큰 홀딩 타임이 끝나면 다음 스테이션에게 토큰을 넘겨준다. 하지만, 이 방식의 경우, 각 스테이션들은 자신에게 토큰이 올 때까지 기다려야하므로 데이터의 실시간성의 보장에 문제가 있다. 일반적으로 제어 노드로만 토큰이 전달되고, 센서·엑츄에이터 노드로는 전달이 되지 않는다. Profibus가 이 방식을 택하고 있다.

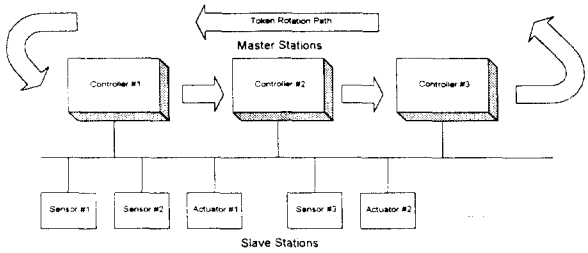

그림 1. CiT 운용
Fig. 1. CiT Operation

3.2.2 DeT (Delegated Token)

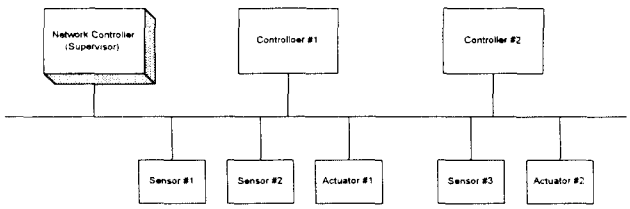

그림 2. DeT 운용
Fig. 2. DeT Operation

네트웍 상에 한 대의 수퍼바이저 스테이션이 있어서, 이 스테이션이 네트웍 접근 권한을 관리하는 형태이다. 각 스테이션들이 수퍼바이저에게 네트웍 접근 권한을 요청하면, 수퍼바이저는 요청의 우선도를 판별하여 토큰을 넘겨주고, 토큰을 받은 스테이션은 네트웍 접근 권한을 가지고 데이터 전송을 하게 된다. 데이터 전송이 끝나면 토큰은 다시 수퍼바이져에게 넘어가는 형태로 운용이 된다. FIP가 이 방식을 택하고 있다.

4. 네트웍 트래픽

4.1 가정

이 논문에서는 다음과 같은 가정을 한다.

- 하나의 제어 루프는 하나 이상의 센서 노드와 제어 노드, 엑츄에이터 노드로 구성된다.
- 각 센서 노드에서의 전송 큐의 크기는 1이며, 센서 노드에 지정된 샘플링 주기에 의해 계속 업데이트 되고, 전송 큐에 저장된다.
- 네트웍 상에 교환되어지는 데이터의 종류는 실시간 주기적 데이터, 실시간 비주기적 데이터, 비실시간 비주기적 데이터 세 종류가 있다.

4.2 표기

B : 네트웍 전송 속도

ι : 데이터 패킷의 크기

L : 패킷 하나가 네트웍 상에서 전송되는 속도

δ_i^j : 제어 루프 ι 에 있는 센서 노드 j 의 최대 허용 샘플링 지연 시간

o : 각 노드에서 토큰 관리를 위해 소비되어 지는 네트웍 오버헤드 시간

λ^{RA} : 실시간 비주기적 데이터 발생률

T_m^{RA} : 전체 네트웍 관점에서의 실시간 비주기적 데이터의 최소 전송 시간

U_m : 네트웍 이용도

T_i : 제어 루프 ι 의 샘플링 주기

A_{T_1} : T_1 동안 발생하는 데이터 패킷의 평균 수

$A^{RS}_{T_1}$: T_1 동안 발생하는 실시간 주기적 데이터 패킷의 평균 수

$A^{RA}_{T_1}$: T_1 동안 발생하는 실시간 비주기적 데이터 패킷의 평균 수

$A^{NRA}_{T_1}$: T_1 동안 발생하는 비실시간 비주기적 데이터 패킷의 평균 수

N^C : 제어 루프의 수 또는 네트웍 상의 제어 노드의 수

N^S : 네트웍 상의 센서 노드의 수

N^S_i : 제어 루프 i 에 포함된 센서 노드의 수

N : 토큰을 가질 수 있는 노드 수

4.3 스케쥴링 알고리즘

각 노드에서 발생하는 데이터는 일정한 크기 l 을 갖는 패킷으로 만들어진다고 가정한다. 네트웍의 전송 속도를 B 라고 하면, 하나의 패킷이 노드 간에 전송되어 지는 속도 L 은 다음과 같다.

$$L = \frac{l}{B} \tag{4.1}$$

일단 실시간 주기적 데이터만 고려하여, 제어 루프 i 에 속한 센서 노드 중, 최소 샘플링 타임를 T^{RS}_i 라고 하면,

$$T^{RS}_i = \min[\,\delta^j_i,\ j=1 \cdots N^S\,] \tag{4.2}$$

T^{RS}_i 중에서 최소를, T^{RS}_1 라고 하자.

$$T^{RS}_1 = \min[\,T^{RS}_i,\ i=0 \cdots N^C\,] \tag{4.3}$$

실시간 주기적, 실시간 비주기적, 비실시간 비주기적 데이터를 고려한 기본 샘플링 주기 T^B_i 는,

$$\begin{aligned} T^B_i &= T^{RS}_i + T^{RA}_i + T^{NRA}_i + T^O_i \\ &= T^{RS}_i + \lambda^{RA} \cdot T^{RA}_m + U_m \cdot T^B_i + \sigma \cdot T^B_i \\ &= \frac{T^{RS}_i + \lambda^{RA} \cdot T^{RA}_m}{1 - U_m - \sigma} \end{aligned} \tag{4.4}$$

$$T^B_1 = \frac{T^{RS}_1 + \lambda^{RA} \cdot T^{RA}_m}{1 - U_m - \sigma} \tag{4.5}$$

x 를 초과하지 않는 2의 제곱수를 y 라고 하는 연산자 $y = \langle x \rangle$ 를 이용하여, T^B_1 에 대한 T^B_i 의 비 k_i 를 정하면,

$$\begin{aligned} k_i &= \langle \frac{T^B_i}{T^B_1} \rangle \\ &= \langle \frac{T^{RS}_i + \lambda^{RA} \cdot T^{RA}_m}{T^{RS}_1 + \lambda^{RA} \cdot T^{RA}_m} \rangle \\ &(i = 1 \cdots N^C) \end{aligned} \tag{4.6}$$

k_i 를 이용하여 샘플링 주기를 변형하여 T_i 를 다시 기술하여,

$$\begin{aligned} T_i &= k_i T^B_1,\ i=1 \cdots N^C \\ &(T_1 = T^B_1) \end{aligned} \tag{4.7}$$

라 하면, T_i 는 항상 T_1 에 대해 2^n 배가 되며, 오름차순으로 배열된다고 가정한다. 즉,

$$T_{n+1} \geq T_n,\ n=0 \cdots N^C \tag{4.8}$$

T_i 가 k_i 의 배로 배열되므로써 실시간 주기적인 데이터가 어떤 한 T_1 주기에 몰리는 경우를 최소화할 수 있다.

한편, T_1 는 네트웍 관점에서 보면, 토큰이 네트웍 상의 모든 마스터 스테이션을 순환하는 데 걸리는 시간이라고 할 수 있다. T_1 동안에 네트웍 상에 전송되어질 수 있는 데이터 패킷의 수, 즉 패킷의 갯수 r 은 다음과 같다.

$$r = \lfloor \frac{T_1 - N \cdot \sigma}{L} \rfloor \tag{4.9}$$

단, CiT인 경우 제어 노드에만 토큰이 전달되므로 $N = N^C$, DeT인 경우 센서 노드에도 토큰이 전달이 되서 네트웍 오버헤드가 발생하므로 $N = N^C + N^S$ 이다. $N \cdot \sigma$ 항은, 토큰 운용을 위해 네트웍 자체에서 소비되어지는 오버해드 시간이다.

네트웍 상에 교환되어지는 전체 데이터 패킷의 수의 평균값이 r 보다 적으면 네트웍 부하가 무척 적은 경우가 되며, 이 경우, 한 노드로 수신 큐 크기 이상의 데이터가 한꺼번에 몰리는 경우를 제외하고는, 모든 데이터들이 시간 지연 없이 모두 처리가 된다고 하자.

또한, 주기 T_1 동안에 전송되어져야할 데이터 패킷 갯수의 평균값 A_{T_1} 는 다음과 같이 구해진다.

$$\frac{T^{NRA}}{T_1} = A^{NRA}_{T_1} \geq U_m \tag{4.10}$$

$$\begin{aligned} A_{T_1} &= A^{RS}_{T_1} + A^{RA}_{T_1} + A^{NRA}_{T_1} \\ &= 2 \sum_{i=1}^{N^C} \frac{1}{k_i} + \lambda^{RA} \cdot T^{RA}_m + U_m \end{aligned} \tag{4.11}$$

$A_{T_1} \leq r$ 인 경우, 네트웍 상에서 전송되어져야하는 패킷들을 시간대역에 맞춰 잘 배열하면, 네트웍 대역폭 내에서 수용이 되므로, 윈도우 프로토콜을 이용하여 스케쥴링 하는 데 적당한 조건이 된다. 하지만, $A_{T_1} > r$ 인 경우는, 네트웍 대역폭 안에서 모든 패킷을 관리할 수 없는 경우이므로, 네트웍 상의 노드 수를 줄이거나, 네트웍의 전송 속도를 높여야만 한다.

이상의 알고리즘을 정리하면 다음과 같다.

Given : N^S_i, N^C, δ^j_i, λ^{RA}, U_m, σ, B, l

Step 1 : 기본 샘플링 주기 T_1 을 구한다.

$$T^{RS}_1 = \min[\,\min[\,\delta^j_i,\ j=1 \cdots N^S_i\,],\ i=1 \cdots N^C\,]$$

$$T_1 = T^B_1 = \frac{T^{RS}_1 + \lambda^{NRA} \cdot T^{NRA}_m}{1 - U_m - \sigma}$$

Step 2 : T_1 동안 전송되어 질 수 있는 데이터 패킷 수 r 을 구한다.

if CiT,

$$N = N^C$$

if DeT,

$$N = N^C + N^S$$

$$r = \lfloor \frac{T_1 - N \cdot \sigma}{L} \rfloor$$

Step 3 : T_i 를 구한다.

$$N_P = \sum_{i=1}^{N_C}(N_i^S+1) + \left\lceil \frac{(U_m+\lambda^{RA}\cdot T_m^{RA})\cdot T_1}{L} \right\rceil \quad (4.12)$$

if $r > N_P$ (네트웍 부하가 적어서 기본 샘플링 주기 내에 모든 데이터 패킷이 처리되는 경우)

$$T_i = \frac{T_i^{RS}+\lambda^{NRA}\cdot T_m^{NRA}}{1-U_m-\sigma}$$

else

$$k_i = \left\langle \frac{T_i^{RS}+\lambda^{RA}\cdot T_m^{RA}}{T_1^{RS}+\lambda^{RA}\cdot T_m^{RA}} \right\rangle, \quad i=1 \cdots N^C$$

$$A_{T_1} = 2\cdot\sum_{i=1}^{N^C}\frac{1}{k_i}+\lambda^{RA}\cdot T_m^{RA}+U_m$$

if $A_{T_1} \le r$

$$T_i = k_i \cdot T_1$$

else (네트웍 대역폭 내에서 감당 못 함)

→ N, N^C, N^S, N_i^S 을 줄이거나, B 를 늘려서, **Step 1**을 반복

5. 결론

이 논문에서는 여러 개의 센서 노드와 제어 노드, 엑츄에이터 노드로 구성된 제어 루프를 감안하여, CiT와 DeT의 경우 각각에 대한 네트웍 특성을 고려하여, 네트웍 스케쥴링에 대한 알고리즘을 제시해 보았다.

참고문헌

[1] PTO, DIN 19 245 Profibus Standard Part 1. English Ver., Profibus Trade Organization, Apr. 1991.

[2] A. D. Stefano and O. Mirabella, "Evaluating the Field Bus Data Link Layer by a Petri Net-Based Simulation", *IEEE Trans. on Industrial Electronics*, Vol. 38, pp. 288-298, Aug. 1991.

[3] G. Olsson and G. Piani, "Computer Systems for Automation and Control", Prentice-Hall, pp. 354-363, 1992.

[4] H. S. Park, S. C. Ahn, and W. H. Kwon, "Performance and Parameter Region for Real-Time Use in IEEE 802.4 Token Bus Network", *IEEE Trans. on Industrial Electronics*, Vol. 40, pp. 412-420, Aug 1993.

[5] H. S. Park, Y. H. Kim, and W. H. Kwon, "Intelligent Networks for Mechatronics"

[6] J. F. Kurose, M. Schwartz, and Y. Yemini, "Controlling Window Protocols for Time-Constrained Communication in Multiple Access Networks", *IEEE Trans. on Communications*, Vol. 36, pp. 41-49, Jan. 1988.

[7] O. C. Ibe and X. Cheng, "Stability Conditions for Multiqueue Systems with Cyclic Service", *IEEE Trans. on Automatic Control*, Vol. 33, pp. 102-103, Jan. 1988.

[8] P. Montuschi, A. Valenzano, and L. Ciminiera, "Selection of Token Holding Times in Timed-Token Protocols", *IEEE Trans. on Industrial Electronics*, Vol. 37, pp. 442-451, Dec. 1990.

[9] P. Pleinevaux, and J. D. Decotignite, "Time Critical Communication Networks: Field Buses", *IEEE Network*, Vol. 2, pp. 55-63, May. 1988.

[10] S. Cavalieri, A. D. Stefano, and O. Mirabella, "Optimization of Acyclic Bandwidth Allocation Exploiting the Priority Mechanism in the Fieldbus Data Link Layer", *IEEE Trans. on Industrial Electronics*, Vol. 40, pp. 297-305, Jun. 1993.

[11] S. H. Hong, "Design of Fieldbus Networks by Bandwidth Allocation Scheme"

[12] S. H. Hong, "Scheduling Algorithm of Data Sampling Times in the Integrated Communication and Control Systems", *IEEE Trans. on Control Systems Tech*, Vol. 3, No. 2, pp. 225-230, Jun. 1995.