발전설비 분산제어 시스템에서 CAN 구축기술 연구

論 文
48A - 6 - 14

A Study on the Implementation of CAN in the Distributed Control System of Power Plant

金 昱 憲* · 洪 承 鎬**
(Wook-Heon Kim · Seung-Ho Hong)

Abstract - The CAN is a serial communication protocol for distributed real-time control and automation systems. Data generated from field devices in the distributed control of power plant are classified into three categories: real-time event data, real-time control data, non-real-time data. These data share a CAN medium. If the traffic of the CAN protocol is not efficiently controlled, performance requirements of the power plant system could not be satisfied. This paper proposes a bandwidth allocation algorithm that can be applicable to the CAN protocol. The bandwidth allocation algorithm not only satisfies the performance requirements of the real-time systems in the power plant but also fully utilizes the bandwidth of CAN. The bandwidth allocation algorithm introduced in this paper is validated using the integrated discrete-event/continuous-time simulation model which comprises the CAN network and distributed control system of power plant.

Key Words : CAN(Controller Area Network), power plant, real-time, bus utilization, bandwidth allocation

1. 서 론

발전 설비는 고품질의 전력을 최소한의 비용으로 생산하는 동시에 돌발 사고 등에 대비하여 충분한 안전성을 갖추어야 한다. 이러한 요구사항을 만족시키기 위하여 10여년 전부터 발전 설비의 제어 및 자동화 시스템에 컴퓨터를 이용한 분산제어 기술이 도입되기 시작하였다. 분산제어 시스템은 설비의 제어 기능을 비롯하여, 시스템의 감시 진단 기능과 각종 데이터의 저장, 분석 및 보고 기능들을 제공한다. 발전 설비의 제어 및 자동화 시스템에 분산제어 기술을 도입하기 위하여서는 각종 설비들 간에 데이터를 실시간으로 교환할 수 있는 기능이 부과되어야 한다. 따라서 발전설비에서 실시간 통신망의 도입은 시스템의 기능을 첨단화시키고 성능을 향상시키는데 있어서 핵심적인 역할을 수행한다[1].

최근에 와서 개방화된 분산제어 시스템의 요구가 증대되면서 필드에 설치된 각종 장비들 간에 제어 및 자동화 관련 데이터의 전송을 실시간으로 지원하는 필드버스라는 새로운 개념의 개방형 통신 프로토콜들이 개발되었다. 본 연구에서는 발전설비의 분산제어 시스템에 CAN(Controller Area Network) 프로토콜을 구축하기 위한 기초 연구를 수행한다. CAN은 원래 차량 내의 각종 계측 제어 장비들 간에 디지털 직렬 통신을 제공하기 위하여 개발된 차량용 네트워크 시스템으로, 1993년도에 ISO에서 국제 표준 규격으로 제정되었다[2]. CAN은 다른 자동화 통신망들에 비하여 가격 대 성능비가 우수하며, 지난 수년간 차량 내의 열악한 환경에서 성공적으로 동작되어 신뢰도가 검증된 통신망이다. 이러한 장점들로 인하여 최근에 와서 CAN은 공장자동화와 공정의 분산 제어 등의 각종 산업 설비에서 제어 및 자동화 관련 장비들 간에 데이터 교환을 위한 통신망으로 널리 사용되고 있다. CAN을 사용하는 대표적인 통신망으로는 DeviceNet, SDS, CAN Kingdom, CANopen/CAL 등이 있으며, 이들은 모두 데이터링크 계층으로 CAN을 사용하나 응용계층은 서로 다른 프로토콜을 채택하고 있다.

일반적인 산업 설비와 마찬가지로 발전 설비의 분산 제어 시스템에서 생성되는 데이터는 크게 산발적 또는 주기적으로 생성되는 실시간 데이터와, 제어 루프의 센서와 제어기에서 주기적으로 생성되는 제어 데이터 및 데이터 전송 지연시간에 크게 구애를 받지 않는 비실시간 데이터로 구분되며, 이러한 데이터들은 통신망이 제공하는 하나의 네트워크 미디엄을 공유한다. 실시간 처리를 요구하는 발전 설비의 분산 제어 시스템에서 네트워킹으로 인한 실시간 및 제어 데이터의 지연시간이 허용 한계치를 초과하는 경우에는 발전 설비 제어시스템의 기능과 성능에 악영향을 미칠 수 있다. 따라서 발전 설비의 분산 제어 시스템 설계시 데이터 지연시간과 단위 시간 당 데이터 처리량 등의 통신망에 대한 성능을 미리 분석할 필요가 있다.

CAN은 비교적 최근에 개발이 완료된 통신망으로 이에 대한 성능 평가 및 설계에 대한 연구 논문은 국내외적으로 아직 많이 발표되고 있지 않다[3-6]. Zuberi와 Shin[3]은 하나의 고성능의 CPU를 사용하는 기존의 중앙 제어 시스템을 여러 개의 작은 CPU를 사용하는 분산 제어 시스템으로 대

* 正 會 員 : 漢陽大 工大 制御計測工學科 碩士課程
** 正 會 員 : 漢陽大 工大 制御計測工學科 副教授
接受日字 : 1999年 1月 22日
最終完了 : 1999年 4月 14日

체하고 이를 CAN으로 통합하는 방법을 제시하였다. 분산된 시스템으로 변환하여 설계하기 위해서는 필드버스 상으로 통신을 하기 위한 실시간 네트워크 스케듈링 기법이 필요하며, 이를 위하여 Zuberi와 Shin은 CAN을 대상으로 MTS(Mixed Traffic Scheduler)라는 네트워크 스케듈링 기법을 제시하였다. MTS에서 메시지는 하드 데드라인(hard-deadline) 주기적 메시지, 하드 데드라인 비주기적 메시지 및 비실시간 비주기적 메시지로 분류하였다. 메시지의 우선 순위를 정하는 방법으로는 ED(earliest deadline)와 DM(deadline monotonic)의 두 가지 방법을 제시하였으며, 데드라인에 긴박한 메시지를 전송할 때는 ED를 사용하고, 상대적으로 실시간성을 요구하지 않는 메시지의 전송은 DM의 방법을 사용하도록 하였다.

Cena와 Valenzano[6]는 노드의 개수가 증가되어 네트워크의 트래픽이 높아지는 경우, 우선 순위가 낮은 메시지의 전송 지연 시간이 늘어나는 문제를 해결하기 위해 CAN의 확장 프레임의 우선 순위를 동적으로 할당하는 방법으로 DPQ(Distributed Priority Queue)와 PP(Priority Promotion)를 제시했다. DPQ는 전체 시스템을 하나의 크고 분산된 FIFO 큐라고 가정하고 29비트의 식별자에 우선 순위를 할당하는 방법이고, PP는 CAN 버스의 중재 과정에서 메시지의 전송에 실패한 횟수에 따라 메시지의 우선 순위를 높이는 방법이다. Tindell, Hansson, Wellings[4-5]은 CAN을 수학적으로 모델링하여 메시지의 응답 시간을 계산하는 방법을 제시하였다. 메시지 m에 대해 최악의 경우 응답 시간은 $R_m = t_m + C_m$으로 주어지며, 여기서 t_m은 블록킹 시간과 메시지간의 간섭 시간을 더한 큐잉 지연 시간이고 C_m은 전송 지연 시간이다.

최근에 필드버스에서 산발적 및 주기적 실시간 데이터와 비실시간 데이터가 하나의 통신망에서 공존하는 경우에 네트워크의 트래픽을 관리 및 제어하는 방법으로 대역폭 할당 기법[7]이 제시되었다. 대역폭 할당 기법은 필드버스의 대역폭을 충분히 활용하면서 산발적 및 주기적 실시간과 비실시간 데이터의 지연시간 요구사항을 만족시키는 필드버스 시스템 설계기법이다. 본 연구에서는 발전설비의 제어 및 자동화 시스템에 CAN을 도입하는 경우에 대역폭 할당 기법을 통하여 네트워크 시스템을 설계하는 방안을 제시하고자 한다. 본 연구를 통하여 제시되는 CAN 설계 기법의 타당성은 시뮬레이션 모델을 통하여 검증된다. CAN 프로토콜의 시뮬레이션 모델은 CAN의 이산 사건 시스템 모델과 발전설비 제어시스템의 연속 시간 모델을 통합한 통합 시뮬레이션 모델로 구성된다. 따라서, 시뮬레이션 모델은 CAN 프로토콜에서 트래픽 부하의 변화에 대한 데이터 지연 시간의 성능 관계뿐만이 아니라, 실시간 데이터의 지연 시간이 제어 시스템의 성능에 미치는 영향까지도 분석할 수 있다.

본 논문은 모두 6장으로 구성된다. 제 2 장에서는 CAN 프로토콜의 동작 원리에 대하여 간략히 기술하며, 제 3 장에서는 대역폭 할당 기법을 CAN 프로토콜에 적용하는 방안에 대하여 기술한다. 제 4 장에서는 CAN의 이산 사건 시뮬레이션 모델과 발전 설비 제어시스템의 연속 시간 모델에 대하여 기술하며, 제 5 장에서는 시뮬레이션 모델을 이용하여 측정한 CAN 프로토콜의 성능 해석 결과에 대하여 기술한다. 제 6 장에는 본 연구의 결론과 추후 연구 사항이 제시된다.

2. CAN 프로토콜

CAN은 CSMA/NBA(Carrier Sense Multiple Access with Non-destructive Bitwise Arbitration)라는 메시지 전송 메커니즘을 가지고 있다. CAN의 데이터 전송 메커니즘은 IEEE 802.3 CSMA/CD 프로토콜[8]과 유사하다. 즉, 각 노드는 데이터를 전송하기 이전에 버스의 상태를 감지하며, 버스의 상태가 비활성일때 준비된 메시지를 전송한다. CSMA/CD에서는 두 개 이상의 노드가 동시에 메시지를 전송하면 메시지 충돌이 일어나서 전송된 메시지가 모두 손실된다. 그러나 CAN에서는 전송되는 메시지가 11비트의 식별자(Identifier)를 가지고 있으며, 식별자를 통하여 우선 순위가 높은 메시지가 전송되도록 한다. 즉, 두 개 이상의 노드가 동시에 메시지를 전송하면 각 메시지는 서로 식별자를 1비트씩 비교하여 제일 높은 우선 순위의 메시지(즉, 가장 낮은 식별자 값을 가진 메시지)는 전송되고 낮은 우선 순위의 메시지들은 전송이 중단된다. 버스에서는 '0' 비트가 '1' 비트에 대해 우세한 성질을 가지고 있다. 즉 '0' 비트는 'dominant'한 값('d' 비트)이라고 하며 '1' 은 'recessive'한 값('r' 비트)이다. 전송하는 노드는 한 비트를 보낼 때마다 버스를 모니터링한다. 어떤 노드가 'r' 비트를 보냈는데 버스를 모니터링한 결과 'd' 비트가 검출되었다면 이는 버스 내의 다른 노드가 자기보다 높은 우선 순위의 메시지를 전송하고 있는 것이므로 그 노드는 메시지의 전송을 즉시 중단하고 수신모드로 전환한다. 전송을 중단한 노드는 버스의 상태를 계속 감지하여, 버스가 다시 비활성 상태가 되면 자동적으로 다시 메시지의 전송을 시작한다.

3. 대역폭 할당 기법 적용

CAN 버스에서는 실시간 데이터, 주기적 제어 데이터 및 비실시간 데이터들이 제한된 필드버스 대역폭을 공유한다. 여기서 실시간 데이터란 자동화 시스템에서 경고 신호와 같이 제한된 시간 내에 전송이 완료되어야 하는 데이터를 말하며, 제어 데이터는 제어시스템의 플랜트와 제어기를 포함하는 제어 루프의 센서와 제어기에서 주기적으로 생성되는 데이터로서, 이러한 데이터들은 데이터 샘플링 주기 이내에 전송이 완료되어야 한다. 비실시간 데이터는 파일의 전송과 같이 데이터 전송 지연 시간에 구애를 받지 않는 데이터를 말한다. 대역폭 할당 기법[7]은 필드버스의 데이터 링크 계층에서 실시간 및 주기적 제어 데이터의 실시간 데이터 전송 요구사항을 만족시키는 동시에, 필드버스에서 제공하는 대역폭을 충분히 활용하여 비실시간 데이터의 처리량을 극대화할 수 있는 일반적인 방법론을 제시하였다. 본 장에서는 대역폭 할당 기법에서 제시하는 일반적인 방법론을 CAN이란 특정 프로토콜에 적용하는 방법에 대하여 기술한다. 다음은 본 장에서 사용되는 기호들이다.

N : CAN에 접속되어 데이터를 생성하는 노드의 개수
M : CAN에 접속된 제어 루프 개수

N_p : 주기적 제어 데이터를 생성하는 전송큐의 개수

N_c : 실시간 데이터를 생성하는 전송큐의 개수

N_a : 비실시간 데이터를 생성하는 전송큐의 개수

L_p : 주기적 제어 데이터 패킷의 전송 시간

L_c : 실시간 데이터 패킷의 전송 시간

[Φ_i, $i=1$에서 M]: 제어 루프 i의 최대 허용 루프 지연 시간

[λ_c^i, $i=1$에서 N_c]: 실시간 데이터 전송큐 i에서 데이터 패킷의 단위시간당 평균 도착 빈도

[Λ_a^i, $i=1$에서 N_a]: 비실시간 데이터 전송큐 i에서 데이터의 평균 도착 빈도

[T_i, $i=1$에서 M]: 제어 루프 i의 데이터 샘플링 주기

본 논문에서 실시간 및 비실시간 데이터의 도착 빈도는 Poisson 분포를 가지는 것으로 가정한다. 그림 1에 대역폭 할당 기법의 기본 개념이 나타나 있다. 그림 1에 나타난 바와 같이 대역폭 할당 기법은 CAN의 대역폭을 주기적으로 발생하는 제어 데이터의 최소 생성 주기인 T_1의 기본 구간으로 나누며, T_1의 기본 구간은 다시 주기적 구간과 비실시간 구간으로 나뉘어진다. 주기적 구간에서는 제어 루프의 센서와 제어기에서 생성되는 제어 데이터들이 윈도우 스케듈링 기법[8]을 통하여 전송되고, 비실시간 구간에서는 비실시간 데이터들이 전송된다. 실시간 데이터는 가능한 한 최단시간 내에 전송이 완료되어야 하며, 따라서 데이터 전송 권한을 부여받은 노드는 구간에 관계없이 실시간 데이터를 제어 데이터와 비실시간 데이터에 우선하여 전송한다.

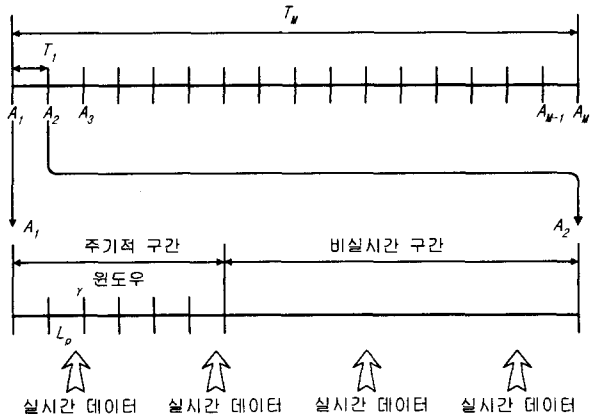


그림 1 CAN 버스의 대역폭 할당

Fig. 1 Bandwidth Allocation Scheme on CAN

CAN에 대역폭 할당 기법을 적용하기 위하여서는 대역폭을 주기적 구간과 비실시간 구간으로 구분할 수 있는 방법이 제시되어야 한다. CAN 프로토콜에서는 CAN 프레임의 식별자를 통하여 대역폭의 구간을 간단히 구분할 수 있다. 즉, 실시간 데이터는 프레임 식별자의 11비트중 첫 번째와 두 번째 비트를 '00...'으로, 제어 데이터는 '01...'로, 비실시간 데이터는 '10...'으로 세팅을 하여 전송한다. CAN 프레임의 식별자는 '0' 비트가 '1' 비트에 대해 우세하므로, 실시간 데이터는 제어 또는 비실시간 데이터에 비하여 우선적으로 전송된다. 실시간 데이터가 동시에 발생하는 경우에는 이들 가운데 우선 순위가 높은 데이터가 먼저 전송된다.

주기적 구간에서 사용되는 윈도우 스케듈링 기법의 기본 개념은 N_p개의 주기적 실시간 데이터를 생성하는 노드가 주기적 구간에서 제공하는 $\gamma(\gamma \le N_p)$개의 윈도우를 서로 동적으로 공유하도록 하는 것이다(여기서 윈도우란 하나의 주기적 제어 데이터가 전송되는데 소요되는 시간 슬롯으로 길이는 L_p이다). 이를 위하여서는 주기적 구간에서 생성되는 제어 데이터의 개수가 윈도우 개수인 γ을 초과하지 않아야 한다. 윈도우 스케듈링 기법은 필드버스에 접속된 제어 루프들에서 샘플링되는 데이터의 개수가 γ개의 윈도우 개수를 초과하지 않도록 각 제어 루프에서의 데이터 샘플링 주기와 샘플링 시작 시간을 적절히 스케듈링한다. 비실시간 구간에서 전송되는 비실시간 데이터는 주기적 구간을 침범하지 않아야 하며, 따라서 대역폭 할당 기법에서는 비실시간 구간에서 전송되는 비실시간 데이터의 패킷 길이가 필요에 따라 적절히 조정되어야 한다.

CAN에는 M개의 제어루프가 접속되는 것으로 가정하며, 각 제어 루프의 최대 허용 루프 지연 시간 Φ_i는 미리 알고 있는 것으로 가정한다. 여기서 루프 지연 시간은 센서 노드에서 데이터를 샘플링한 시점에서 시작하여, 새로운 센서 데이터를 통하여 제어기 노드에서 생성된 제어 신호가 구동기에 완전히 도달하는 시점까지의 경과 시간으로 정의된다. 각 제어 루프는 센서와 제어기의 두개의 데이터 전송 노드를 가지며, 센서와 제어기 노드는 동일 주기로 데이터를 샘플링한다. CAN에 접속된 제어 루프 i에서 생성되는 데이터의 샘플링주기 T_i ($i=1$에서 M)를 원소로 하는 샘플링 주기 벡터 $\boldsymbol{T}$와 최대 허용 루프 지연 시간 Φ_i($i=1$에서 M)를 원소로 하는 벡터 $\boldsymbol{\Phi}$는 다음과 같이 정의된다.

$$\boldsymbol{T}=[T_1, T_2, T_3, \ldots, T_M] \tag{1}$$

$$\boldsymbol{\Phi}=[\Phi_1, \Phi_2, \Phi_3, \ldots, \Phi_M] \tag{2}$$

벡터 $\boldsymbol{T}$와 $\boldsymbol{\Phi}$는 오름차순으로 정렬된다(즉, $T_i \le T_{i+1}$, $\Phi_i \le \Phi_{i+1}$). 그림 1에서 T_1은 M개의 제어 루프 가운데 최소 샘플링 주기이고 T_M은 최대 샘플링 주기이다.

주기적 구간에서 윈도우 스케듈링 기법은 각 제어 루프에서 데이터 샘플링 주기 T_i를 구하며, 이는 각 제어 루프의 최대 허용 루프 지연 시간 Φ_i에 의하여 결정된다. 즉, T_1은 각 제어 루프의 최대 허용 루프 지연 시간 Φ_i 중에서 최소값인 Φ_1에 의해서 결정되고, 나머지 제어루프들의 샘플링 주기는 최대 허용 루프 지연 시간을 초과하지 않는 조건에서 식 (3)의 관계를 만족시키는 최대값으로 지정된다. 이 경우에 모든 샘플링 주기는 서로 배수의 관계를 가지며,

따라서 네트워크 대역폭은 T_M주기로 반복된다[9].

$$\mathrm{Rem}\left[\frac{T_j}{T_i}\right]=0,\ \forall i,\ j\ (j\geq i) \tag{3}$$

여기서 각 제어 루프의 샘플링 주기 T_i는 T_1에 대하여 2의 멱(즉, 2^{n_i}, $n_i\in\{0,1,2,\ldots\}$)으로 지정된다. 주기적 데이터에 대하여 샘플링 주기가 서로 배수의 관계를 갖도록 하는 스케듈링 기법은 Profibus-PA[10]와 TCN[11]과 같은 프로토콜에서 이미 사용되고 있다. 본 논문에서 제시하는 대역폭 할당 기법에서는 T_M 주기 내에서 필드버스의 대역폭이 T_1 길이로 분할된다. 그림 1에서 A_l($l=1$ 에서 M)은 T_M의 구간에서 $l-$번째 T_1 슬롯이 시작되는 순간이다. 윈도우 스케듈링 기법은 A_l의 순간에 생성되는 데이터의 개수가 γ를 초과하지 않도록 하기 위하여 제어 데이터를 생성하는 노드 j($j=1$ 에서 N_p)에서 샘플링을 시작하는 순간인 t_j를 다음과 같이 설정한다[7].

$$t_j=\inf\left[A_l\geq A_{l-1}\ ;\ u^j(A_l)\leq\gamma\right],$$
$$j=1 \text{ 에서 } N_p,\ l=1 \text{ 에서 } T_M/T_1 \tag{4}$$

여기서 $u^j(A_l)$은 A_l의 순간에 생성되는 주기적 데이터의 개수이다.

CAN 프로토콜에 대역폭 할당 기법을 적용하기 위하여서는 노드들 간의 샘플링 시간을 동기화하는 기능을 필요로 한다. Rodd[13]는 CAN 프로토콜에서 H/W적으로 노드간에 동기화를 이룰 수 있는 방법을 제시하였다. 또한 Gergeleit[14]는 두 개의 동기화 메시지를 연속적으로 전송하는 방법을 통하여 S/W적으로 노드들간의 동기화를 구현하는 방법을 제시하였다. 동기화 메시지를 수신한 노드는 메시지를 전송한 시간, 메시지의 도착시간, 각 노드의 현재 시간 등을 비교하여 동기화를 이룬다. 본 논문에서는 노드들 간의 동기화가 H/W 또는 S/W적인 방법으로 구현되는 것으로 가정한다.

그림 1의 T_1 대역폭 구간에서 주기적 구간 τ_p를 제외한 나머지 대역폭은 비실시간 데이터의 전송을 위하여 할당된다. 주기적으로 생성되는 제어 데이터 전송에 적용되는 윈도우 스케듈링 기법에서는 $l-$번째 T_1 슬롯이 시작되는 A_l의 순간에 생성되는 데이터의 개수가 γ를 초과하지 않도록 스케듈링 되어 있어, 그림 1에서 A_l의 순간에 생성된 데이터는 주기적 구간동안 전송이 완료되고, 다음의 주기적 구간이 시작될 때까지 네트워크의 대역폭은 비실시간 구간에 할당된다. 비실시간 구간 τ_a에서는 전송 지연 시간에 크게 구애를 받지 않는 비실시간 데이터들만이 전송된다. 본 대역폭 할당 기법에서는 주기적 데이터의 식별자가 비실시간 데이터의 식별자에 비하여 높은 우선 순위로 할당되므로, 어느 노드에서든지 주기적 데이터가 생성되면 비실시간 구간 τ_a는 자동적으로 중지되고 네트워크 대역폭은 다시 주기적 데이터의 전송을 위한 주기적 구간 τ_p로 전환된다. 그러나 A_l의 순간에 한 노드에서 주기적 제어 데이터가 생성되었을 때 미디엄 내에 이미 비실시간 데이터 패킷이 전송되고 있는 중이라면, 이미 전송이 시작된 비실시간 데이터 패킷의 전송은 완료되어야 한다. 그림 1에서 비실시간 구간이 최소로 할당되는 극단적인 경우에 비실시간 패킷 전송시간 L_a가 $T_1-\tau_p$를 초과하게 되면 T_1의 구간 내에서 비실시간 구간이 주기적 구간을 침범하게 되며, 따라서 비실시간 데이터의 최대 패킷 전송 시간(즉, 111비트로 구성되는 최대 프레임의 전송시간)이 $T_1-\tau_p$보다 큰 경우에는 패킷 전송 시간 L_a가 다음과 같이 제한되어야 한다.

$$L_a< T_1-\tau_p \tag{5}$$

CAN 버스에서 네트워크의 트래픽이 안정화되기 위하여서는 데이터의 도착 빈도를 네트워크의 용량이 초과되지 않는 범위로 제한하여야 한다. 실시간 및 비실시간 데이터 트래픽의 안정화 조건은 Ibe와 Cheng[12]이 제시한 station backlog 기법을 이용하여 구할 수 있다. 먼저 실시간 데이터 트래픽에 대하여 고려하자. 실시간 데이터는 가장 높은 우선 순위로 전송되므로 제어 및 비실시간 데이터들의 전송에 영향을 받지 않는다. t 라는 매우 긴 시간 동안에 노드 i의 실시간 데이터 전송큐에 도착하는 데이터 패킷의 평균 개수를 생각하자. 실시간 데이터 전송큐가 안정화되기 위하여서는 도착된 모든 데이터 패킷을 전송하는데 소요되는 시간의 평균값이 t를 초과하지 않아야 한다. t의 시간 동안에 도착하는 실시간 데이터 패킷 개수의 평균값은 $\lambda_c^i t$이며, 따라서 t의 시간 동안에 노드 i에서 실시간 데이터 패킷을 전송하는데 소요되는 시간은 $(\lambda_c^i t)L_c$이다. 다음에는 t의 시간 동안에 네트워크 내의 다른 노드들에서 전송되는 실시간 데이터 패킷의 전송 시간을 고려하자. 노드 $j(\neq i)$에 대하여 만일에 $\lambda_c^j<\lambda_c^i$라면 t의 시간 동안에 노드 j에서 전송되는 실시간 데이터 패킷 개수의 평균값은 $\lambda_c^j t$개이며, 그렇지 않은 경우에는 노드 j에 도착된 실시간 데이터패킷들 가운데 $\lambda_c^i t$개의 데이터 패킷만이 평균적으로 전송될 것이다. 따라서, t의 시간 동안에 다른 노드들에서 전송되는 실시간 데이터 패킷의 전송 시간은 $L_c\sum_{j=1,j\neq i}^{N_c}\min[\lambda_c^i t,\ \lambda_c^j t]$가 된다. 노드 i에서 실시간 데이터 전송큐가 안정화되기 위하여서는 t의 시간 동안에 도착한 데이터 패킷을 전송하는데 소요되는 시간이 t보다 작아야 한다. 즉,

$$(\lambda_c^i t)L_c+L_c\sum_{j=1,j\neq i}^{N_c}\min[\lambda_c^i t,\ \lambda_c^j t]<t \tag{6}$$

네트워크 시스템이 안정화되기 위하여서는 네트워크 내의 모든 전송큐가 안정화되어야 하며, 따라서 가장 높은 데이터 도착 빈도를 갖는 실시간 데이터 전송큐가 안정화되면 네트워크 시스템은 안정화될 것이다. 따라서 실시간 데이터 트래픽의 안정화 조건은 다음과 같다.

$$L_c \sum_{j=1}^{N_c} \lambda_c^j < 1 \tag{7}$$

다음은 비실시간 데이터 트래픽의 안정화 조건에 대하여 고려하자. 비실시간 데이터는 가장 낮은 우선 순위로 전송되므로 제어 및 실시간 데이터들의 전송에 영향을 받는다. 역시 t 라는 매우 긴 시간 동안에 가장 낮은 우선 순위를 갖는 노드 i의 비실시간 데이터 전송큐에 도착하는 데이터 패킷의 평균 개수를 생각하자. 비실시간 데이터 전송큐가 안정화되기 위하여서는 도착된 모든 데이터 패킷을 전송하는데 소요되는 시간의 평균값이 t를 초과하지 않아야 한다. t의 시간 동안에 도착하는 비실시간 데이터 패킷 개수의 평균값은 $\lambda_a^i t$이며, 따라서 t 의 시간 동안에 노드 i에서 비실시간 데이터 패킷을 전송하는데 소요되는 시간은 $(\lambda_a^i t)L_a$이다. 다음에는 t의 시간 동안에 네트워크 내의 다른 노드들에서 전송되는 비실시간 데이터 패킷의 전송 시간을 고려하자. 노드 $j(\neq i)$에 대하여 만일에 $\lambda_a^j < \lambda_a^i$라면 t의 시간 동안에 노드 j에서 전송되는 비실시간 데이터 패킷 개수의 평균값은 $\lambda_a^j t$ 개이며, 그렇지 않은 경우에는 노드 j에 도착된 비실시간 데이터패킷들 가운데 $\lambda_a^i t$개의 데이터 패킷만이 평균적으로 전송될 것이다. 따라서, t의 시간 동안에 다른 노드들에서 전송되는 비실시간 데이터 패킷의 전송 시간은 $L_a \sum_{j=1,j\neq i}^{N_a} \min[\lambda_a^i t,\ \lambda_a^j t]$가 된다. 제어 데이터는 비실시간 데이터에 비하여 높은 우선순위로 전송되므로, 다음에는 t의 시간 동안에 전송되는 제어 데이터의 전송 시간을 고려하자. t의 시간 내에 존재할 수 있는 T_1 구간의 개수는 (t/T_1)이다. T_1 구간에서 생성되는 제어 데이터의 평균 개수는 $\alpha_K = \sum_{i=1}^{N_p} \frac{T_i}{T_1}$이다. 각각의 T_1 구간에서 제어 데이터를 전송하는데 소요되는 시간의 평균값은 $\alpha_K L_p$이므로, t의 시간 동안에 전송되는 제어 데이터의 전송 시간은 $(t/T_1)\alpha_K L_p$이다. 마지막으로 가장 높은 우선순위로 전송되는 실시간 데이터의 전송 시간을 고려하면, 노드 k에서 발생하는 실시간 데이터의 도착 빈도가 λ_c^k인 경우에 t의 시간 동안에 노드 k에서 전송되는 실시간 데이터 패킷 개수의 평균값은 $\lambda_c^k t$ 개이다. 따라서, t의 시간 동안에 전송되는 실시간 데이터 패킷의 전송 시간은 $L_c \sum_{k=1}^{N_c} \lambda_c^k t$가 된다. 노드 i에서 비실시간 데이터 전송큐가 안정화되기 위하여서는 t의 시간 동안에 도착한 데이터 패킷을 전송하는데 소요되는 시간이 t보다 작아야 한다. 즉,

$$(\lambda_a^i t)L_a + L_a \sum_{j=1,j\neq i}^{N_a} \min[\lambda_a^i t,\ \lambda_a^j t] + (t/T_1)\alpha_K L_p + L_c \sum_{k=1}^{N_c} \lambda_c^k t < t \tag{8}$$

네트워크 시스템이 안정화되기 위하여서는 네트워크 내의 모든 전송큐가 안정화되어야 하며, 따라서 가장 높은 데이터 도착 빈도를 갖는 비실시간 데이터 전송큐가 안정화되면 네트워크 시스템은 안정화될 것이다. 따라서 비실시간 데이터 트래픽의 안정화 조건은 다음과 같다.

$$L_a \sum_{j=1}^{N_a} \lambda_a^j + (\alpha_K L_p)/T_1 + L_c \sum_{k=1}^{N_c} \lambda_c^k < 1 \tag{9}$$

다음은 제어 데이터 트래픽의 안정화 조건에 대하여 고려하자. 주기적 구간에서는 우선 순위가 높은 실시간 데이터도 전송되므로 제어데이터의 전송은 실시간 데이터의 전송에 영향을 받는다. 분산제어 시스템의 실제 환경에서는 그러나 실시간 데이터의 발생 빈도가 제어 데이터의 발생 빈도에 비하여 상대적으로 매우 낮다[17]. 실시간 데이터가 Poisson 분포로 발생한다고 가정하는 경우에 T_1의 시간 이내에 노드 i에서 n개의 실시간 데이터가 발생할 확률은 $\frac{(\lambda_c^i T_1)^n}{n!} e^{-\lambda_c^i T_1}$이다. 예를 들어 $T_1 = 13.48$ msec이고, $\lambda_c^i = 0.01$ msec^{-1}인 경우에 T_1의 시간 내에 $n=10$개 보다 많은 실시간 메시지가 발생할 확률은 1.897×10^{-4}이다. 산발적으로 발생하는 실시간 데이터를 포함하는 경우 주기적 데이터의 안정화 조건은 확률적으로 결정된다. 본 논문에서는 T_1의 시간 이내에 발생하는 실시간 데이터의 최대 개수를 확률적으로 충분히 작은 값을 가지는 n_t개로 제한한다고 가정한다. 만일 $\gamma L_p + n_t L_c > T_1$인 경우에는 T_1의 주기 동안에 생성된 주기적 제어 데이터 가운데 일부는 같은 기간 동안에 전송이 되지 못하고 전송큐에서 제거되는 현상이 발생할 수 있다. 따라서 주기적 제어 데이터 트래픽의 확률적 안정화 조건은 다음과 같다.

$$\gamma L_p + n_t L_c \leq T_1 \tag{10}$$

실시간 데이터 트래픽의 안정화 조건인 (7)은 비실시간 데이터 트래픽의 안정화 조건인 (9)에 포함되므로, 실시간, 제어 및 비실시간 데이터 트래픽을 포함하는 CAN 버스가 안정화되기 위하여서는 식 (9)과 (10)을 동시에 만족하여야 한다. 본 논문에서 제시하는 대역폭 할당 기법을 요약하면 아래와 같다.

Given: N, M, N_p, N_c, N_a, L_p, L_c, n_t,

$[\Phi_i,\ i=1$에서 $M]$, $[\lambda_c^i,\ i=1$에서 $N_c]$,

$[\Lambda_a^i,\ i=1$에서 $N_a]$

Algorithm:

Step 1: 제어 루프 i에서 주기적 제어 데이터의 샘플링 주기 T_i를 구한다.

$$\Phi_1 = \min[\Phi_i,\ i=1 \text{ 에서 } M]$$

$$T_1 = \frac{\Phi_1 + L_p}{3}$$

$$k_i = \langle \frac{\Phi_i - (T_1 - L_p)}{2T_1} \rangle,\ \forall\ i=1 \text{ 에서 } M$$

(여기서, $y=\langle x \rangle$는 x를 초과하지 않는 범위에서 가장 큰 값을 갖는 2의 멱. 즉, 2^{n_i}, $n_i \in \{0, 1, 2, \ldots\}$)

$$\alpha_K = \sum_{i=1}^{N_t} \frac{1}{k_i}$$

$$\gamma = \lceil \alpha_K \rceil$$

If $\gamma L_p + n_t L_c > T_1$, then

(주기적 제어 데이터 트래픽의 과부하 $\Rightarrow M$ 또는 N_c 감소: goto Step 1)

else

$$T_i = k_i T_1$$

Step 2: 주기적 제어 데이터 생성 노드에서 첫 번째 데이터 샘플링 순간 t_j를 구한다.

$$t_1 = A_1 = 0$$

For ($j=2$; $l=1$; $j \le N_p$, $l \le k_M$; $j=j+1$, $l=l+1$)

$$t_j = \inf [A_l \ge A_{l-1}\ ;\ u^j(A_l) \le \gamma]$$

Step 3: 비실시간 데이터의 트래픽을 스케듈링한다.

$$L_a < T_1 - \tau_p$$

$$\lambda_a^j = \lceil L_a^j / L_a \rceil \Lambda_a^j$$

If $L_a \sum_{j=1}^{N_a} \lambda_a^j + (\alpha_K L_p)/T_1 + L_c \sum_{k=1}^{N_c} \lambda_c^k \ge 1$, then

(실시간 또는 비실시간 트래픽의 과부하 $\Rightarrow N_c$ 또는 N_a 감소: goto Step 3)

End

4. 발전설비 분산제어 시스템 시뮬레이션 모델

4.1 CAN 시뮬레이션 모델

본 연구에서는 CAN 프로토콜의 데이터 전송지연에 대한 성능을 분석할 수 있는 시뮬레이션 모델을 개발하였다. 시뮬레이션 모델에 입력되는 CAN 필드버스 파라미터에는 (i) 노드의 수, (ii) 각 노드에서 생성되는 메시지의 발생 시간 분포, (iii) 메시지 길이에 대한 통계적 정보, (iv) 미디엄의 전송 속도 등이 있으며, 이에 대한 시뮬레이션 모델의 출력으로 (i) 메시지 전송지연 시간의 평균값과 최대값 및 편차, (ii) 생성된 메시지의 개수, (iii) 전송된 메시지의 개수, (iv) 메시지 충돌 횟수, (v) 충돌한 메시지의 개수, (vi) 전송에 실패한 메시지의 개수 등이 측정된다.

네트워크 시스템은 사건이 발생함에 따라 시스템의 상태가 변하는 이산 사건 시스템으로 분류될 수 있으며, 본 연구에서 CAN 프로토콜의 시뮬레이션 모델은 이산 사건 시스템의 모델링을 위하여 개발된 시뮬레이션 전용 언어인 SIMAN/ARENA[15]를 사용하여 구현하였다. 그림 2에는 본 연구를 통하여 개발된 CAN 프로토콜 시뮬레이션 모델의 사건 순서도가 나타나 있다. 시뮬레이션 모델에서는 먼저 CAN 프로토콜의 파라미터를 초기화한다. 초기화가 종료되면 각 노드는 메시지 생성 사건을 통하여 메시지를 생성한다. 생성된 메시지에는 다음과 같은 메시지 속성들이 부여된다.

- 각 메시지의 식별자(Identifier) 값
- 메시지를 전송하는 스테이션 주소
- 메시지가 생성된 시간
- 메시지 길이
- 메시지에 실리는 데이터 값
- 메시지의 종류를 구분하는 인자

생성된 메시지가 전송큐에 삽입되면 다음에 생성될 메시지를 스케듈링한다. 각 노드에서 생성된 메시지의 전송은 2장에서 기술된 CAN 프로토콜의 동작과 동일하게 동작되도록 구현하였다. CAN 프로토콜에서 모든 메시지는 브로드캐스트 방식으로 전송된다. 각 노드에서는 메시지가 도착되면 메시지의 식별자를 검사하여 수신할 메시지인가를 판단하여 이를 받아들이거나 도착한 메시지를 버린다. 시뮬레이션 모델에서는 도착한 메시지의 수신이 허락되면 그 메시지의 생성 시간부터 수신시간까지의 시간인 전송 지연 시간을 비롯하여 각종 시뮬레이션 출력 값들을 측정한다.

본 연구에서는 CAN 프로토콜이 정상적으로 동작되는 경우에 대한 성능만을 고려하며, 따라서 시뮬레이션 모델에서는 메시지를 전송하거나 수신하는 일련의 과정에서 에러가 발생하지 않는다고 가정한다. 또한 비트 스터핑(bit stuffing)으로 인한 추가전송지연과 메시지 전파지연시간 등과 같이 메시지 전송지연 시간에 비하여 매우 작은 값을 갖는 항목들은 고려하지 않는다. 그림 2는 CAN 시뮬레이터를 위한 사건 순서도이다. 먼저 각 메시지의 파라미터를 초기화한다. 메시지의 우선순위, 메시지의 종류, 데이터의 길이 등 각종 네트워크 파라미터를 읽어 들이고 메시지에 속성을 부여한다. 첫 번째 메시지를 생성하여 큐에 삽입한 후, 다음에 생성될 메시지를 스케듈(schedule)한다.

버스의 상태가 비활성이면 'BusStatus'는 '0'이고, 활성이면 'BusStatus'는 '1'이다. 버스의 상태가 비활성이 될 때까지 계속 체크를 한다. 조건이 만족되면 다음 조건인 큐에 메시지가 있는지를 체크하여 전송할 메시지가 있으면 전송큐로부터 우선 순위가 가장 높은 메시지를 하나 꺼내어 복사하고, 복사한 메시지는 전송큐에 다시 삽입한다. 그리고 복사된 메시지는 버스로 보낸다. 서버로 모델링한 버스에 들어오는 메시지가 하나인지 둘 이상인지를 체크하여 메시지가 충돌했는지를 검사한다. 메시지의 충돌이 없으면 우선 순위를 비교할 필요가 없으므로 메시지를 전송하기 전에 복

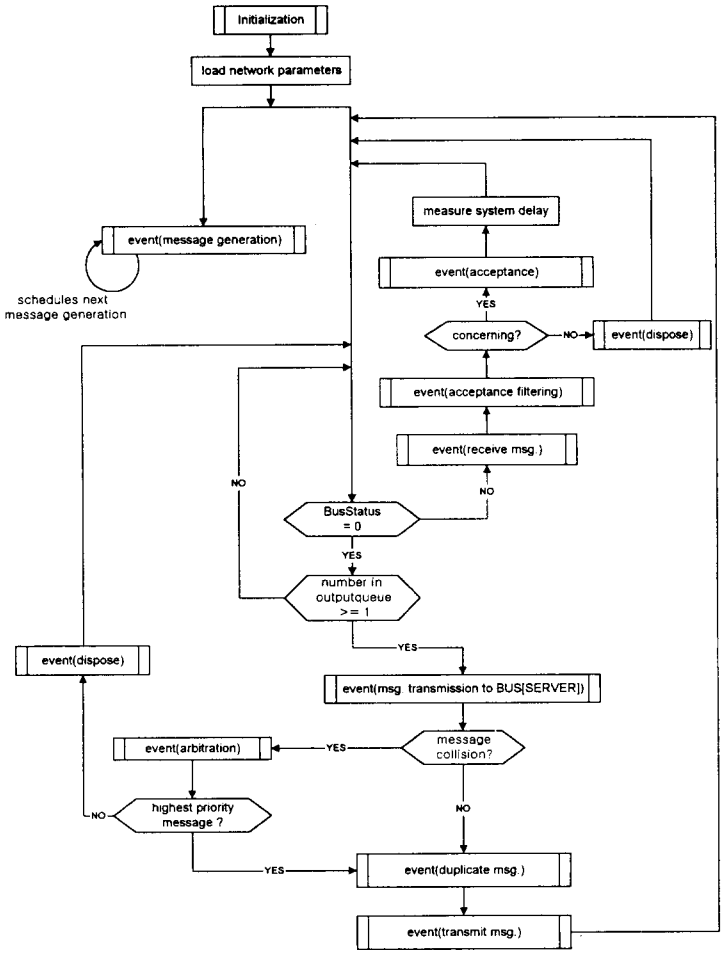


그림 2 CAN 시뮬레이션 모델의 사건 순서도
Fig. 2 Event flowchart for CAN simulation model

사해 둔 메시지를 삭제하고 다음 사건(transmit msg.)을 스케듈한다. 만약 메시지의 충돌이 발생하면 메시지의 우선순위에 의한 중재(arbitration)과정으로 들어간다. 중재 과정에서 이긴 메시지(우선순위가 가장 높은 메시지)는 전송 전에 복사해 둔 메시지를 삭제하고 다음 사건(transmit msg.)을 스케듈하며, 중재에서 진 메시지는 버스 상에서 없어진다. 전송된 메시지는 수신하는 스테이션의 수만큼 복사된 후 각 스테이션의 수신큐로 보낸다. 각 스테이션에서 메시지를 수신하면, 메시지의 식별자를 검사하여 수신되어야 하는 메시지인 경우 이를 받아들이고, 그렇지 않는 경우에는 수신된 메시지를 버린다(acceptance filtering). 메시지 수신 여부는 전체 네트워크 시스템이 어떻게 구성되느냐에 따라 다르며, 네트워크의 구현 목적에 따라 다르게 된다. 수신한 메시지의 수신이 허락되면 그 메시지의 생성시간부터 수신시간까지의 전송지연과 앞에서 언급한 다른 출력 값들을 측정한다.

4.2 발전설비 시뮬레이션 모델

본 절에서는 먼저 발전 설비 제어시스템에 CAN 버스를 도입하는 경우에 시스템의 구성 방안을 제시한다. 본 연구에서는 화력 발전소를 모델로 다룬다. 화력 발전소는 일반적으로 압력 제어, 온도 제어, 유량 제어, 수위 제어 등의 네 개의 제어루프를 가진다. 발전 설비에 CAN 버스를 도입하는 경우에는 압력, 온도, 유량, 수위 제어를 위한 제어기는 각각 따로 분산 설치하여 필드버스에 접속되도록 하고, 이러한 제어기들을 총괄 관리하는 supervisory 제어 컴퓨터를 설치한다. 발전 설비 시스템에 네트워킹을 도입하면 제어기들 간에 상호 보완 기능을 용이하게 구현할 수 있다. 즉, 하나의 제어기에 고장이 발생하면 다른 제어기가 이를 자동으로 감지하여 고장난 제어기의 기능을 보완하도록 하여 시스템의 신뢰도를 향상시킨다. 각각의 제어기들의 상태는 네트워크를 통하여 항상 모니터링될 수 있으므로 제어기의 고장은 쉽게 탐지될 수 있다.

발전 설비의 필드에는 많은 센서 및 구동기들이 설치되어 있다. 본 연구에서 제시하는 모델은 144개의 아날로그 입력점(AI)과 160개의 아날로그 출력점(AO)을 갖는 것으로 가정한다. 압력, 온도, 유량, 수위 제어 루프들은 각각 센서로부터 입력되는 36, 36, 36, 36개의 아날로그 입력점(AI)과 구동기로 출력되는 40, 40, 40, 40개의 아날로그 출력점을 가진다. Supervisory 제어기는 64개의 디지털 입력점(DI)과 64개의 디지털 출력점(DO)을 갖는다. 아날로그 및 디지털 입력 데이터는 시스템의 제어를 위하여 센서에서 해당 제어기로 전달되는 제어 데이터와 시스템 관리를 위하여 supervisory

컴퓨터로 전달되는 로깅 데이터로 구분된다. CAN은 네트워크 특성상 하나의 노드에서 전송되는 메시지가 모든 노드에서 수신되는 브로드캐스트 방식으로 전달되기 때문에 제어 데이터와 로깅 데이터의 생성주기를 적절히 조절하여 하나의 메시지로 보낼 수 있다. 압력, 온도, 유량, 수위 제어기와 supervisory 제어기 사이에는 시스템 관리와 관련된 프로그램과 제어기에 로딩되는 제어 프로그램 및 이와 관련된 파일데이터들이 CAN 버스를 통하여 교환되는 것으로 가정한다.

전력 설비의 필드에 설치되는 센서 및 구동기들이 CAN 접속 장치를 가지고 있지 않다면 이러한 장비들은 CAN 버스에 직접 연결될 수 없다. 이러한 장비들은 필드에 분산 설치되는 I/O(Input/Output) 노드를 통하여 네트워크에 접속되도록 해야 한다. I/O 노드는 아날로그 및 디지털 데이터를 처리할 수 있는 기능과 CAN접속 기능을 가지고 있다. 필드에 설치되는 여러 개의 센서 및 구동기들은 하나의 I/O 노드에 접속되어, I/O 노드가 이들에 대한 입출력 점들의 데이터를 관리하도록 한다. 이러한 경우에 각각의 센서와 구동기에 네트워크 접속 장비들을 설치할 필요가 없고, 기존의 센서 및 구동기들을 그대로 사용할 수가 있으며, 배선에 소요되는 비용을 절감하고, 여러 개의 중복 신호가 동시에 전송될 수 있다.

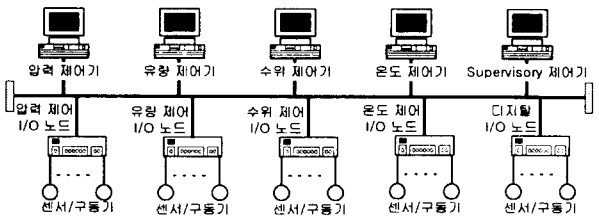


그림 3 발전 설비 시스템 구성도

Fig. 3 Schematic diagram of power generation system using CAN

그림 3에는 CAN 버스가 적용된 발전 설비의 구성도가 나타나 있다. CAN에 접속되는 노드의 개수는 5개의 제어기와 78개의 I/O 노드들을 포함하여 83개이며, 이들은 하나의 링크로 연결된다. 표 1에는 각각의 제어루프들에 대하여 접속되는 노드의 개수와 생성되는 데이터의 종류 및 입출력점의 개수가 주어져 있다. 표 1에서 노드 수는 해당 제어루프에서 데이터를 생성하는 노드의 수를 나타내며, 비고에는 생성된 데이터가 전송되는 방향이 나타나 있다. 표 1의 메시지 분류에서, '2'는 제어시스템의 플랜트와 제어기를 포함하는 제어루프의 센서와 제어기에서 생성되어 실시간으로 전송되어져야하는 주기적 상태 피드백(state feedback) 제어 데이터이다. '1'은 제어 데이터를 제외한 산발적으로 생성되는 실시간 데이터이며, '3'은 파일 전송에 사용되는 비실시간 데이터이다. 또한, 데이터 종류에서 DI-C, DI-L, DO, AI-C, AI-L, AO 는 각각 Digital Input-Control Data, Digital Input-Log Data, Digital Output Data, Analog Input-Control Data, Analog Input-Log Data, Analog Output Data를 나타내며, 데이터 프레임의 Identifier에서 'xx-'는 11비트의 Identifier 중에서 처음 두 비트를 나타낸다. 각각의 노드에서 생성되는 데이터의 identifier는 먼저 우선 순위에 따른 메시지 분류에 의하여 결정되고, 우선 순위가 같은 경우에는 노드 번호에 의하여 구분되었다. 제어루프별로는 온도 제어가 가장 높은 우선 순위를 가지며 수위 제어가 가장 낮은 우선 순위를 가지는 것으로 가정하였다. 데이터 종류별로는 실시간 데이터가 가장 높은 우선 순위를 가지며, 다음으로 제어 데이터, 비실시간 메시지의 순이다. 따라서, 표 1에서는 온도 제어의 실시간 메시지인 AI-L 데이터가 가장 높은 우선 순위를 가지며, 수위 제어의 비실시간 메시지인 파일 데이터가 가장 낮은 우선 순위를 가진다.

CAN에는 데이터 프레임, 원격 프레임, 에러 프레임, 과부하 프레임 등 4개의 프레임이 있다. 본 논문의 표 1에 나타난 각종 데이터는 데이터 프레임을 통하여 전송된다. 아날로그 센서에서 생성되는 데이터는 일반적으로 한 개당 2bytes의 길이를 가지고, 디지털 센서는 한 개당 1bit의 데이터 길이를 가진다. CAN에서는 하나의 프레임 당 최대 8bytes까지의 데이터를 전송 할 수 있다. 따라서 하나의 입출력 모듈에서는 최대 4개의 아날로그 센서 데이터, 또는 최대 64개의 디지털 센서 데이터를 수용할 수 있다. 즉, 아날로그 데이터일 경우에는 한 프레임당(한 노드당) 최대 4개의 센서 데이터(4개의 센서로부터 나온 값)를 보낼 수 있고, 디지털 데이터일 경우에는 한 프레임당(한 노드당) 최대 64개의 센서 데이터(64개의 센서로부터 나온 값)를 보낼 수 있다. 그러므로 메시지 프레임의 오버헤드를 포함하게 되면 주기적 제어 데이터는 최대 111비트가 된다. 본 연구에서 제어 데이터의 길이는 CAN에서 한 프레임이 가질 수 있는 최대 크기인 8바이트로 한다. 실시간 메시지는 경고 신호와 같은 시스템의 상태를 보고하게 된다. 이들 메시지는 가능한 한 매우 빨리 전송되어야 하는 긴박한 메시지들이다. 본 연구에서 실시간 데이터의 길이는 8바이트로 한다. 비실시간 메시지는 제어기 노드들 간의 파일의 전송 등을 담당하게 되는데, 이들 메시지는 전송 지연 시간에 크게 구애를 받지 않는다. 데이터량이 큰 파일일 경우에는 여러 개의 패킷으로 분리하여 전송해야 된다. 그러므로 하나의 프레임에서 6바이트는 파일 데이터를 할당하고 나머지 2바이트에는 분리된 패킷의 정보를 할당하여 전송하게 된다.

본 연구에서는 CAN의 데이터 지연시간이 제어 시스템의 성능에 미치는 영향을 조사하기 위하여, 제어 시스템으로 발전 설비 가운데 과열기의 온도 분무 제어 루프를 선정하였다[16]. 과열기의 분무 제어 루프는 PI 및 상태 제어 알고리듬을 사용한다. 그림 4에는 실제 화력 발전소에서 사용되는 과열기의 온도 분무 제어 루프의 구성도가 나타나 있다. 그림 4에서 과열기의 중간 온도 θ_1, θ_2, θ_3와 출력부의 스팀 온도 θ_o는 온도 센서에 의해 감지되어, CAN 네트워크를 통하여 온도 제어기 컴퓨터로 전달된다. 그림 4의 온도 제어기에서는 수신된 θ_o과 θ_r에 의해 에러를 측정하고, PI 및 상태 제어 알고리듬을 통하여 주기적으로 제어신호 U를 생성, 이를 CAN 네트워크를 통하여 온도 제어 I/O 노드로 전송한다. 온도 제어 I/O 노드에서는 이를 다시 온도 밸브 조정 장치로 보내며, 온도 밸브 조정 장치에서는 과열기에 온도 θ_i의 입력 스팀을 제공한다. 과열기는 외부로부터 교

표 1 발전 설비 노드 및 입출력점 구성

Table 1 Nodes and I/O points in the power generation facility

	노드 분류	노드수	데이터 종류	데이터 Identifier	입출력 점	메시지 분류*	데이터 크기 (bytes)	비고
supervisory 제어	센서(DI)	1	DI-C	00-	64	1	8	제어 데이터(DI 노드 → supervisory 제어기)
			DI-L	00-	64	1	8	로깅 데이터(DI 노드 → supervisory 제어기)
	구동기(DO)	1	-		64	-	-	-
	제어기	1	DO	00-	-	1	8	supervisory 제어기 → DO 노드
			file	10-	-	3	8(패킷)	supervisory 제어기 → 해당 제어기
온도 제어	센서(AI)	9	AI-C	01-	36	2	8	제어 데이터(AI 노드 → 온도 제어기)
			AI-L	00-	36	1	8	로깅 데이터(AI 노드 → supervisory 제어기)
	구동기(AO)	10	-		40	-	-	-
	제어기	1	AO	01-	-	2	8	온도 제어기 → AO 노드
			file	10-	-	3	8(패킷)	온도 제어기 → supervisory 제어기
압력 제어	센서(AI)	9	AI-C	01-	36	2	8	제어 데이터(AI 노드 → 압력 제어기)
			AI-L	00-	36	1	8	로깅 데이터(AI 노드 → supervisory 제어기)
	구동기(AO)	10	-		40	-	-	-
	제어기	1	AO	01-	-	2	8	압력 제어기 → AO 노드
			file	10-	-	3	8(패킷)	압력 제어기 → supervisory 제어기
유량 제어	센서(AI)	9	AI-C	01-	36	2	8	제어 데이터(AI 노드 → 유량 제어기)
			AI-L	00-	36	1	8	로깅 데이터(AI 노드 → supervisory 제어기)
	구동기(AO)	10	-		40	-	-	-
	제어기	1	AO	01-	-	2	8	유량 제어기 → AO 노드
			file	10-	-	3	8(패킷)	유량 제어기 → supervisory 제어기
수위 제어	센서(AI)	9	AI-C	01-	36	2	8	제어 데이터(AI 노드 → 수위 제어기)
			AI-L	00-	36	1	8	로깅 데이터(AI 노드 → supervisory 제어기)
	구동기(AO)	10	-		40	-	-	-
	제어기	1	AO	01-	-	2	8	수위 제어기 → AO 노드
			file	10-	-	3	8(패킷)	수위 제어기 → supervisory 제어기

란입력 D_1, D_2, D_3을 받는다. 그림 4에서 제어기 및 플랜트 모델의 전달함수는 다음과 같다.

$$G_c(s)=6\left(1+\frac{1}{0.508s}\right),$$
$$H(s)=\frac{1-e^{-T_0 s}}{s},$$
$$G_v(s)=\left(\frac{1}{1+0.018s}\right)^6, \qquad (11)$$
$$G(s)=\frac{1}{1+0.513s},$$
$$f_1=3.0,\ f_2=4.5,\ f_3=0.0,$$
$$D_1=D_2=D_3=\frac{1}{3}$$

그림 3에서 보는 바와 같이 CAN과 같은 필드버스 네트워크를 이용한 발전 설비 분산 제어 시스템에서는 여러 개의 제어 루프를 가지는 노드들과 모니터링 데이터 생성 노드가 하나의 네트워크 미디엄을 공유하며, 따라서 네트워크 내의 트래픽 부하가 증가하는 경우에 센서에서 측정된 계측 신호와 제어기에서 생성되는 제어 신호의 전송지연 시간이 증가하여 온도 제어 시스템을 포함한 다른 제어 시스템의 성능에 영향을 미칠 수 있다. 본 연구에서는 이러한 데이터 전송 지연 시간이 과열기의 온도 제어 시스템 제어 성능에 미치는 영향을 시뮬레이션을 통하여 조사한다.

5. 시뮬레이션 결과 및 고찰

본 연구에서 제시하는 전력 설비 시스템의 CAN 버스는 83개의 노드를 갖는다. 네트워크 내에는 supervisory 제어를 비롯하여 압력, 유량, 수위, 온도 제어 등 크게 다섯 개의 제어 루프군이 접속된다. 노드 그룹 (1(DI), 2(DO), 3(제어기))는 supervisory 제어, (4-12(AI), 13-22(AO), 23(제어기))는 온도 제어, (24-32(AI), 33-42(AO), 43(제어기))는 압력 제어, (44-52(AI), 53-62(AO), 63(제어기))는 유량 제어, (64-72(AI), 73-82(AO), 83(제어기))는 수위 제어 루프군에 속하여 있다. 하나의 제어 루프군에서 여러 개의 센서 및 구동기는 하나의 제어기를 공유한다. 그러므로 supervisory 제어군을 제외한 나머지 제어군에서 각각의 구동기마다 하나의 제어루프를 갖는 것으로 가정하면, 제어루프의 개수 $M=40$이다. 네트워크 내의 제어 데이터(AI-C, AO), 실시간(AI-L, DI-C, DI-L, DO) 및 비실시간(file) 데이터 전송큐의 개수는 각각 40, 39, 5개씩이다($N_p=40$, $N_c=39$, $N_a=5$). 제어 데이터 전송큐의 용량은 1이며, 실시간 및 비실시간 데이터 전송큐의 용량은 전송큐의 포화 상태로 인한 메시지 재거 현상이 발생하지 않도록 충분히 크다고 가정한다. 구동기 노드는 컨트롤러 노드의 데이터를 수신만 하고 메시지 발생은 없다고 가정한다. 따라서 노드는 83개이지만 메시지를 전송하는 노드가 42개이므로 $N=42$이다. CAN 버스의 길이는 500m로 가정하며, 이 경우 CAN 버스의 데이터 전송 속도 B 는 250 Kbps로 설정될 수 있다. 실시간 데이터와 주기적 실시간 제어 데이터의 길이는 프레임 간 공백기(Inter-Frame Space)를 포함하여 모두 111비트이므로, 패킷 전송 시간 L_c와 L_p가 모두 0.444msec이다. 본 시뮬레이션 실험에서 디지털 제어 입력 데이터(DI-C)와 디지털 출력 데이터(DO)를 포함하여 디지털과 아날로그 데이터의 로깅 데이터는 평균 1sec의 주기로 생성되는 것으로 가정한다. 따라서, 실시간 데이터 패킷의 도착 빈도 λ_c^i

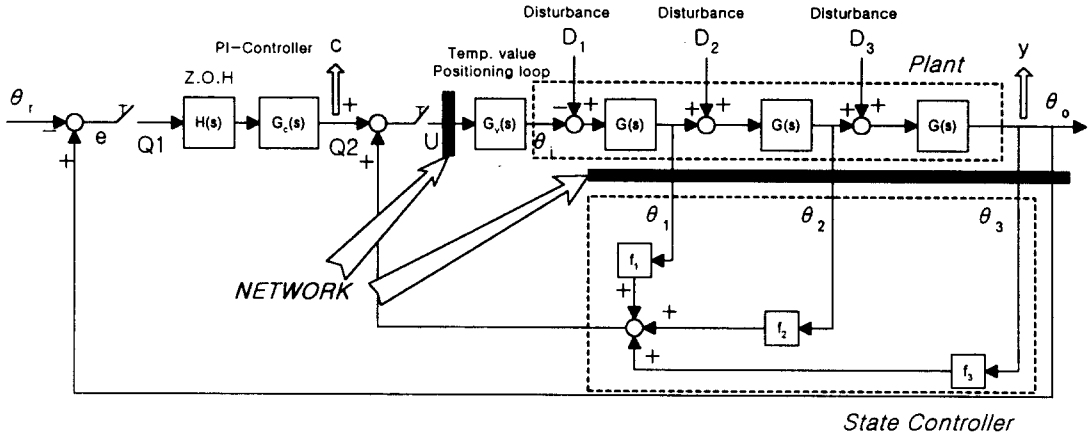


그림 4 과열기의 온도 분무 제어 루프 구성도

Fig. 4 Schematic diagram of superheater temperature-spray control loop

(j=1, 3, 4...12, 24...32, 44...52, 64...72)는 0.001 msec^{-1}의 지수분포(exponential distribution)로 메시지가 발생한다. 실시간 데이터의 최대 허용 데이터 지연 시간 Φ_c^m는 10 msec이며, 40개의 제어 루프에서 최대 허용 루프 지연 시간은 각각 [$\Phi_1^T, \ldots, \Phi_{10}^T, \Phi_1^P, \ldots, \Phi_{10}^P, \Phi_1^F, \ldots, \Phi_{10}^F, \Phi_1^L, \ldots, \Phi_{10}^L$] = [40 msec,..., 40 msec, 70 msec,..., 70 msec, 125 msec,..., 125 msec, 250 msec,..., 250 msec]로 주어지는 것으로 가정한다. 여기서 제어루프군 Φ_x^T, Φ_x^P, Φ_x^F, Φ_x^L는 각각 온도, 압력, 유량, 수위 제어루프군이다.

Algorithm의 Step 1으로부터 Φ_1^T와 T_1^T은 각각 40 msec와 13.48 msec로 주어지며, 샘플링 주기 비율 벡터 K는 [$k_1^T, \ldots, k_5^T, k_1^P, \ldots, k_{18}^P, k_1^F, \ldots, k_4^F, k_1^L, \ldots, k_4^L$] = [1,...,1, 2,...,2, 4,...,4, 8,...,8]이다. 최소 샘플링 주기 T_1^T의 시간 동안에 각 제어 루프군에서 생성되는 데이터 개수의 평균값은 α_K=18.75이며, 따라서 주기적 구간에서 윈도우의 개수는 γ= 19이다. 본 예에서 실시간 메시지의 발생으로 인하여 주기적 구간이 제어데이터의 최소 생성 주기인 T_1^T=13.48 msec을 초과할 확률을 4×10^{-4} 보다 작은 값으로 제한하는 것으로 가정하면, n_t는 11로 제한된다. 이 경우, $\gamma L_p + n_t L_c$(13.32 msec) < T_1^T(13.48 msec)이므로 주기적 구간에서 네트워크 시스템은 확률적으로 안정하다. 따라서, 각 제어루프군에서 샘플링 주기 벡터 T는 [$T_1^T, \ldots, T_5^T, T_1^P, \ldots, T_{18}^P, T_1^F, \ldots, T_4^F, T_1^L, \ldots, T_4^L$] = [13.48 msec,..., 13.48 msec, 26.96 msec,..., 26.96 msec, 53.92 msec,..., 53.92 msec, 107.84 msec..., 107.84 msec]로 결정된다. Step 2로부터 주기적 제어 데이터를 생성하는 노드의 첫 번째 주기적 실시간 제어 데이터 생성 순간은 각각 [{ $t_4, \ldots, t_{12}, t_{23}, t_{24}, \ldots, t_{32}$}, { $t_{43}, t_{44}, \ldots, t_{51}$}, { $t_{52}, t_{63}, t_{64}, \ldots, t_{68}$ }, { $t_{69}, \ldots, t_{72}, t_{83}$}] = [{0 msec,..., 0 msec}, {13.48 msec,..., 13.48 msec}, {40.44 msec,..., 40.44 msec}, {94.36 msec,..., 94.36 msec}]으로 주어진다.

Step 3으로부터 $T_1 - \tau_p$가 CAN 프로토콜의 최대 패킷 길이(111 bits)의 프레임 전송 시간을 초과하지 않으므로, 비실시간 데이터의 패킷 전송 시간은 CAN 프로토콜의 최대 패킷 전송 시간인 L_a=0.444 msec로 주어진다. 비실시간 구간에서 전송되는 비실시간 데이터의 안정화 조건은 (9)에 나타나 있다. 본 시뮬레이션 실험에서 비실시간 데이터 패킷의 도착 빈도 λ_a^j(j=3, 23, 43, 63, 83)가 0.16 msec^{-1}의 지수 분포로 주어진다면, $L_a \sum_{j=1}^{N_a} \lambda_a^j + (\alpha_K L_p)/T_1 + L_c \sum_{k=1}^{N_c} \lambda_c^k$ = 0.98로, 이는 트래픽 부하의 포화 상태인 1에 거의 육박하는 경우이다. 즉, 비실시간 데이터의 안정화를 유지하면서 비실시간 구간의 대역폭을 충분히 활용하는 경우이다.

표 2 데이터 전송 특성 비교

Table 2 Data latency and ratio of rejected data

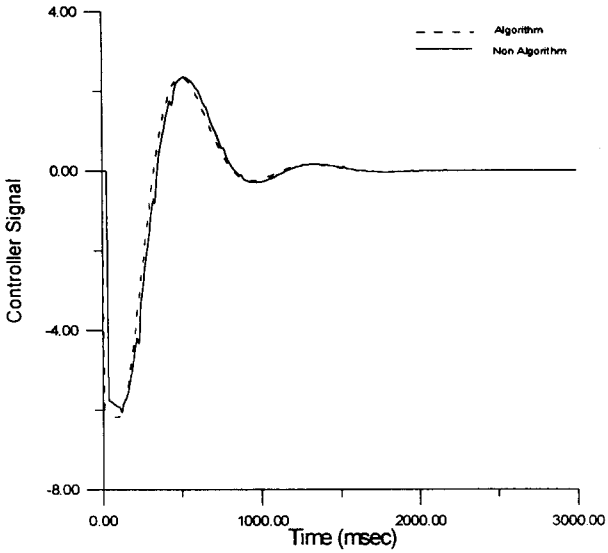
	평균 데이터 전송지연시간 (msec)		최대 데이터 전송지연시간 (msec)		실패 데이터 개수 비율 (실패데이터/생성 데이터)	
	비적용	적 용	비적용	적 용	비적용	적 용
실시간 데이터	0.73	0.73	2.16	1.68	0/181	0/183
주기적 제어 데이터	4.78	3.34	23.96	12.99	286/4190	0/4182
비실시간 데이터 패킷	10.19	25.82	125.17	262.03	0/2426	0/2417

표 2에는 대역폭 할당 기법을 적용하지 않았을 경우와 적용했을 경우에 데이터 종류별 평균 및 최대 지연 시간과 전송 실패 데이터의 개수가 나타나 있다. 알고리듬을 적용하

지 않았을 경우와 적용했을 경우에 네트워크의 트래픽 부하는 서로 같다. 알고리듬을 적용하지 않은 경우에서는 그러나, 주기적 제어 데이터의 스케듈링이 제대로 되지 않았다. 즉, 대역폭 할당 기법에서 제시된 주기적 제어 데이터의 첫 번째 생성 순간 등은 고려되지 않았으며, 따라서 $t_j=0$ msec($j=1, 2, 3, 4\sim12, 23, 24\sim32, 43, 44\sim52, 63, 64\sim72, 83$)에서 모든 노드가 첫 번째 주기적 제어 데이터를 생성한다. 표 2에서 비적용과 적용은 각각 대역폭 할당 알고리듬을 적용하지 않은 경우와 적용한 경우를 나타내며, 실패 데이터란 전송큐의 포화 상태로 인하여 생성된 데이터가 전송되지 못하고 전송큐에서 제거되는 경우를 말한다.

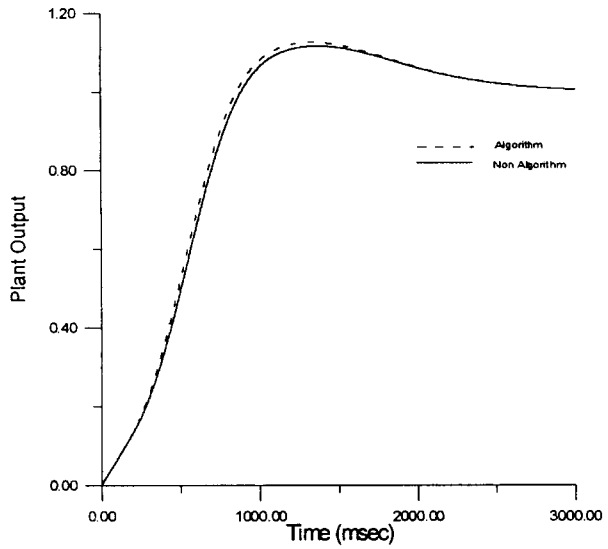
표 2에서 보면 가장 높은 우선 순위로 전송되는 실시간 데이터의 경우, 평균 데이터 전송 지연시간에서는 대역폭 할당 기법을 적용하지 않은 경우와 적용한 경우가 차이를 보이지 않는다. 그러나, 최대 지연시간은 대역폭 할당 기법을 적용한 경우 1.68 msec로 비적용 경우의 2.16 msec 비하여 감소하였다. 실시간 데이터는 두 경우에서 모두 최대 허용 지연시간 10 msec를 초과하지 않는다. 주기적 제어 메시지는 알고리듬을 적용하지 않는 경우 온도제어군의 최대 데이터 전송 지연 시간이 23.96 msec까지 증가하여 최대 지연시간에 대한 성능 요구 사항인 13.48 msec을 만족하지 못하며, 이로 인해 286개의 전송 실패 데이터가 발생하였다. 그러나 알고리듬을 적용한 경우에는 데이터 지연 시간이 12.99 msec로 최대 허용 전송 지연 시간을 초과하지 않으며, 따라서 전송에 실패한 메시지도 발생하지 않는다. 비실시간 데이터는 대역폭 할당 알고리듬을 적용하지 않은 경우에 비해 적용한 경우의 전송 지연시간이 오히려 늘어났다. 그러나 비실시간 데이터의 전송 지연시간은 응용 시스템의 성능에 영향을 미치지 않는다. 본 연구를 통하여 제시된 대역폭 할당 기법은 데이터의 전송 지연 시간이 응용 시스템의 성능에 영향을 미치지 않는 비실시간 메시지의 전송 지연시간이 증가 하는 반면, 전송 지연 시간이 응용 시스템의 성능에 민감하게 영향을 미치는 실시간 및 제어 데이터의 전송 지연시간을 허용 한계값 이내로 제한하여 응용 시스템의 성능 요구사항을 만족시킨다.

그림 5에는 대역폭 할당 기법을 적용했을 경우와 적용하지 않았을 경우에 5장에서 기술된 발전설비의 과열기의 온도 분무 제어 루프의 제어 성능이 비교되어 있다. 그림 5에서 알고리듬을 적용하지 않은 경우에 제어 신호에 지터(jitter)가 발생한다. 이는 전송큐에서 대기하고 있던 제어 데이터가 샘플링 주기 이내에 전송되지 못하고 전송에 실패하였기 때문이다. 이러한 제어 신호의 지터는 제어 성능을 저하시키는 동시에 제어 시스템에서 구동기의 마모를 가속화한다. 플랜트 출력을 보면, 알고리듬을 적용하지 않은 경우 시스템 응답이 늦어지고 따라서 제어 성능도 감소함을 알 수 있다. (a)와 (b)에서 대역폭 할당 기법을 적용했을 경우와 적용하지 않았을 경우의 정상 상태(steady-state)까지의 도달 시간이 거의 동일한데, 이는 본 시뮬레이션 실험에서 네트워크의 트래픽을 증가시키기 위하여 제어 시스템의 샘플링 주기를 실제 시스템에 비하여 매우 작게 설정하였기 때문이며, 따라서 제어 데이터의 전송이 실패되더라도 다음 번에 생성되어 도착한 제어 데이터가 바로 전에 전송 실패된 데이터의 제어 신호값을 보상해 주기 때문이다.



(a) 제어 신호

(a) Controller Signal



(b) 플랜트 출력

(b) Plant Output

그림 5 과열기 온도 분무 제어 루프의 제어 성능

Fig. 5 Performance of superheater temperature-spray control loop

본 장에서는 또한 3장에서 유도된 CAN 버스의 안정화 조건식 (9)의 타당성을 검증하기 위하여 우선 순위가 가장 낮은 비실시간 데이터 전송큐의 생성 주기 변화에 대한 전송 대기 메시지 개수를 시뮬레이션 모델을 통하여 알아보았다. 식 (9)에 의하면, 주어진 트래픽 부하에 대하여 시스템이 비안정화되는 순간은 λ_a^j=0.161, 즉 비실시간 데이터 패킷의 생성 주기가 6.21ms인 순간이다. 그림 6에는 메시지 생성 주기와 이에 따른 비실시간 데이터 전송큐에 대기하고 있는 메시지의 개수가 나타나 있다. 안정화 조건의 한계치

λ_a^i가 0.161이하(비실시간 메시지의 발생주기 6.21ms 이상)일 경우는 전송큐에서 대기하는 메시지의 개수가 어느 정도 증가 후 정상 상태에 도달하는 반면, 한계치를 초과한 직후부터는 전송큐에서 전송이 되지 못하고 대기하는 메시지의 수가 증가하여 CAN 버스 시스템이 불안정해짐을 알 수 있다.

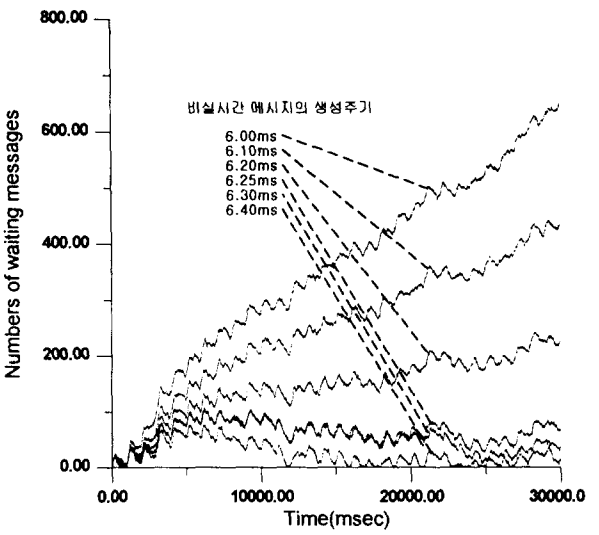


그림 6 전송 대기 메시지수

Fig. 6 Numbers of waiting messages for transmission

6. 결론 및 추후 연구 사항

CAN은 제어 및 자동화 장비들간에 데이터 통신을 담당하는 개방화된 네트워크 프로토콜이다. CAN에 접속된 각 장비들간에 발생하는 데이터는 크게 주기적으로 발생하는 제어 데이터와 주기적 또는 산발적으로 발생하는 실시간 데이터 및 파일 전송과 같은 비실시간 데이터로 분류된다. 비실시간 데이터는 전송 지연 시간에 크게 구애를 받지 않는 반면, 제어 데이터와 실시간 데이터의 전송지연 증가는 응용 시스템의 성능에 악영향을 줄 수 있다. 대역폭 할당 기법은 실시간 메시지의 성능 요구 사항을 만족시키기 위하여 필드버스의 주어진 대역폭에 제어 데이터와 실시간 및 비실시간 데이터 트래픽을 적절히 할당하는 방법이다. 본 논문에서는 대역폭 할당 기법을 CAN 버스에 적용하는 방법을 제시하였고, 이러한 대역폭 할당 기법을 통하여 CAN 버스에서 제어 데이터와 실시간 및 비실시간 데이터의 성능 요구 사항이 만족될 수 있음을 시뮬레이션 기법을 통하여 검증하였다. 본 논문에서 시뮬레이션 모델은 CAN의 이산 사건 시뮬레이션 모델과 발전설비 제어시스템의 연속 시간 모델이 통합된 형태로 구현되었다. 발전 설비에 CAN 버스를 도입하면, 시스템의 성능, 신뢰도, 유연성, 사용자 편의성 등을 향상시킬 수 있다. 본 연구의 결과는 국내의 발전설비에 CAN 버스를 도입하거나, 이를 운용 및 관리하는 단계에서 매우 효과적으로 활용될 수 있을 것이다.

최근에 기술선진국에서는 UCA(Utility Communications Architecture)를 이용하여 전력 관련 설비를 하나의 통신망으로 통합하려는 연구를 추진하고 있다[18]. UCA는 발전소를 비롯하여, 변전소, 송배전소의 모든 전력 설비들 간에 실시간 통신뿐만이 아니라, 전력 회사와 고객까지를 포함하여 전력과 관련된 모든 통신 요구 사항을 만족시키기 위하여 개발된 전력 설비용 개방형 통신 프로토콜이다. 본 연구의 후속 연구에서는 전력 설비의 필드에 설치된 각종 센서, 제어기 및 컴퓨터들이 CAN을 통하여 UCA와 같은 상위 통신망과 어떻게 접속시킬 것인가에 대한 연구를 수행할 필요가 있다.

감사의 글

본 연구는 1997년도 한국전력공사의 지원에 의하여 기초전력공학공동연구소 주관으로 수행된 연구로서, 관계부처에 감사 드립니다(과제번호 97-085).

참 고 문 헌

[1] 홍승호. "전력 설비의 분산제어를 위한 필드버스 네트워크 구축 기술 연구" 전기학회논문지, 제46권, 제4호, pp. 593-602, 1997. 4.

[2] International Standard 11898: Road Vehicles-Interchange of Digital Information-Controller Area Network(CAN) for High-Speed Communication, ISO, 1993.

[3] K. M. Zuberi and K. G. Shin, "Scheduling Messages on Controller Area Network for Real-Time CIM Applications", IEEE Trans. on Robotics and Automation, Vol. 13, No. 2, pp. 310-314, April 1997.

[4] K. W. Tindell, H. Hansson and A. J. Wellings, "Calculating Controller Area Network(CAN) Message Response Times", Control Engineering Practice, Vol. 3, pp.1163-1169, 1995.

[5] K. W. Tindell, H. Hansson and A. J. Wellings, "Analyzing Real-Time Communications: Controller Area Network(CAN)", Proc. Real-Time Systems Symposium, pp. 259-263, December 1994.

[6] Gianluca Cena, Adriano Valenzano, "An Improved CAN Fieldbus for Industrial Application", IEEE Trans. on Industrial Electronics, VOL. 44, August 1997

[7] 홍승호, "대역폭 할당 기법에 의한 필드버스 네트워크의 트래픽 관리 및 제어" 제어 · 자동화 · 시스템공학논문지, 제3권, 제1호, pp. 77-88, 1997. 2.

[8] IEEE, 802.3, Carrier Sense Multiple Access with Collision Detection, New York, IEEE, 1985a

[9] S. H. Hong, "Scheduling Algorithm of Data Sampling Times in the Integrated Communication and Control Systems", IEEE Trans. Contr. Syst. Technology., Vol. 3, pp. 225-230, June 1995.

[10] PROFIBUS-PA Profile for Process Control Devices Class B, PROFIBUS Technical Guideline, PROFIBUS

Nutzerorganisation e.V., Order No. 3.042, 1996
[11] International Electrotechnical Committee, IEC project 1375 reference 9/413/CDV, Train Communication Network, IEC, Geneva, 1996
[12] O. C. Ibe and X. Cheng, "Stability Conditions for Multiqueue Systems with Cyclic Service," IEEE Trans. on Automatic Control, Vol. 33, No. 1, pp. 102-103, January 1988.
[13] M G Rodd, K Dimyati and L Motus, "The design and analysis of low-cost real-time fieldbus systems", Proc. IFAC DCCS Workshop, 1997
[14] Martin Gergeleit, Hermann Streich "Implementing a Distributed High-Resolution Real-Time Clock using the CAN-Bus", Proc. of the 1st international CAN Conference, Mainz, Germany, September 1994, CiA.
[15] C. D. Pegden, R. E. Shannon and R. P. Sadowski, Introduction to Simulation Using SIMAN, McGraw-Hill, 1995.
[16] H. Olia and R. Herzog, Poryong 3-6 Dynamic Behavior, Analysis, Tuning and Optimization of the Control System, SULZER Technical Report # 2142, 1991. 5.
[17] K. Bender, PROFIBUS-The Fieldbus for Industrial Automation, Carl Hanser Verlag & Prentice Hall, 1993.
[18] Introduction to UCA Version 2.0, Electric Power Research Institute, 1997. 3.

저 자 소 개

김 욱 헌(金 昱 憲)
1971년 8월 17일 생. 1997년 한양대학교 제어계측공학과 졸업. 1997년~현재 동대학교 제어계측공학과 석사과정
TEL : 0345-400-4084
E-Mail: wookheon@fieldbus.hanyang.ac.kr

홍 승 호(洪 承 鎬)
1956년 5월 31일 생. 1982년 연세대학교 기계공학과 졸업. 1985년 Texas Tech. University 기계공학과(석사). 1989년 Pennsylvania State University 기계공학과 졸업(공박). 1989년~1992년 한국전자통신연구소 선임연구원. 1992년~현재 한양대학교 제어계측공학과 부교수
TEL : 0345-400-5213,
E-Mail : shhong@email.hanyang.ac.kr