필드버스 프로토콜 변환 게이트웨이 설계

박진원, 이준형, 이희찬, 김명균
울산대학교 컴퓨터.정보통신공학과

Design of fieldbus protocols conversion gateway

Zin-Won Park, Joon-Hyung Lee, Hee-Chan Lee, Myung-Kyun Kim
Dept. of Computer Engineering and Information Technology, University of Ulsan

ABSTRACT

오늘날 공장 자동화에 이용되는 필드버스 네트워크는 여러 가지가 있으며 이들은 각각의 미디어 접근 방식과 프로토콜을 사용하고 있다. 이러한 필드버스 네트워크는 완전히 분리되어 있기 때문에 네트워크 관리자는 이를 관리하는데 많은 어려움을 겪고 있다. 본 논문에서는 이기종의 필드버스 네트워크를 통합할 수 있는 필드버스 프로토콜 변환 게이트웨이를 설계하였다. 필드버스 프로토콜 변환 게이트웨이는 프로토콜 변환을 하여 다른 종류의 네트워크로 데이터를 전달할 수 있고 분리된 네트워크를 통합하고 통합된 환경으로 제어 및 모니터링을 가능하게 한다.

1. 서 론

공장 자동화 시스템에서 각 디바이스들은 필드버스라는 네트워크를 이용하여 연결되어 있다. 공장 자동화에 사용되는 네트워크는 여러 종류가 있지만 그 중 많이 사용되는 것으로는 CAN, Profibus, DeviceNet, Lonworks 등이 있으며 이들은 서로 다른 미디어 접근방식과 프로토콜을 사용하고 있다. 각 네트워크의 관리 시스템들은 네트워크에 독립적으로 존재하며 디바이스들에게 제어 명령을 내리거나 데이터를 수집하여 저장하고 분석한 다음, 다음 제어 명령을 내리는데 활용된다. 하지만 하나의 시스템에 서로 다른 네트워크가 구성되어 있는 경우도 있는데 이런 경우 각 네트워크를 관리하는 관리 시스템이 별도로 구성되어 있어 각 네트워크에서 수집한 정보를 통합하기 힘들뿐만 아니라 통합을 위한 새로운 네트워크를 구성해야하는 어려움이 있다. 따라서 본 논문에서는 서로 다른 네트워크로 구성된 필드버스를 이더넷을 이용하여 통합하는 필드버스 프로토콜 변환 게이트웨이를 설계하였다.

2. 필드버스 프로토콜

본 장에서는 여러 가지 필드버스의 종류 중에서 본 연구에서 고려한 CAN과 Profibus에 대해서만 다룬다. 또한 게이트웨이 설계에서 고려한 데이터 링크 계층에 대해서만 다룬다.

2.1 CAN 2.0x

CAN에는 몇 가지 버전이 있는데 본 절에서는 CAN 2.0A와 CAN 2.0B 버전에 대해서만 다룬다. CAN은 버스 구조의 토폴로지로 구성된다. CAN에는 노드의 주소를 표시하는 필드가 따로 없어 메시지 ID에 대한 필터링을 통하여 원하는 노드만이 수신하도록 하고 있다. 각 노드는 주기적으로 메시지를 발생시켜 전체 네트워크로 브로드캐스팅하고 메시지ID를 보고 이를 필요로 하는 노드만 읽어 들이는 통신 방식과 특정 노드가 다른 노드들로 메시지를 요청하고 요청을 받은 노드들이 이에 응답하는 통신 방식을 사용한다. CAN은 하나의 프레임에 전송할 수 있는 데이터는 0~8바이트로 제한되어 있다. CAN 2.0A 표준에서 프레임의 우선순위를 나타내는 아비트레이션 필드는 11비트의 길이를 가지고 있어 최대 2^{11}가지를 표현할 수 있으나 CAN 2.0B 표준에서는 29비트로 확장되어 있다. CAN의 메시지 전송 방식은 전송하고자 하는 메시지의 우선순위가 낮을 경우 경쟁에서 자주 뒤지게 되므로 오랫동안 전송할 수 없게 되는 문제점을 가지고 있으나 이를 해결하기 위한 스케줄링 방법 등이 많이 연구되고 있다[1,2].

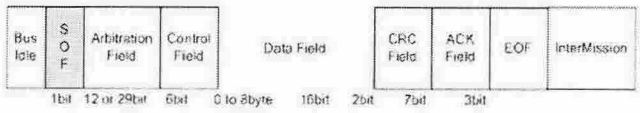


그림 1 CAN 프레임 구조

그림 1은 CAN 2.0x의 프레임 구조를 보여주고 있다.

2.2 Profibus

Profibus는 가상 토큰을 이용한 토큰 패싱(Token-passing) 방식을 사용하여 통신을 한다. 네트워크는 하나 이상의 마스터와 슬레이브들로 구성되어 있으며 토큰은 마스터들 사이에서만 이동한다. 그리고 통신은 토큰을 가지고 있는 마스터만이 시작할 수 있어 슬레이브들은 수동적인 통신을 한다. Profibus의 프레임은 시작 구분(Start delimiter) 필드를 가지고 있는데 이 필드를 통하여 프레임의 종류를 결정한다. Profibus 표준에 정의된 프레임 종류로는 no data 프레임, 고정 데이터 프레임, 가변 데이터 프레임, 토큰 프레임이 있으며 최대 246 바이트의 데이터를 전송할 수 있다. Profibus에 관한 많은 연구들은 토큰 보유 시간을 지정하는 스케줄링 방법과 실시간 데이터와 비실시간 데이터를 스케줄링 방법에 대해 이루어지고 있다[3].

아래의 그림 2는 Profibus의 가변 데이터 프레임 구조를 보여주고 있다. 각 필드는 데이터 필드를 제외하고는 모두 1바이트의 크기를 가진다.

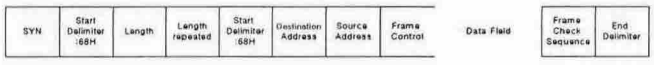


그림 2 Profibus 프레임 구조 : 가변 데이터 프레임

3. 필드버스 프로토콜 변환 게이트웨이 설계

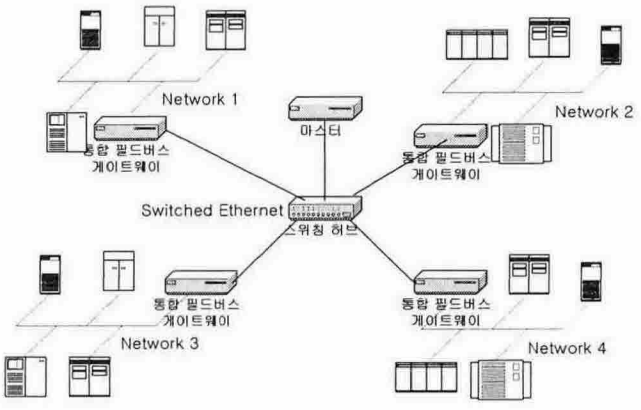
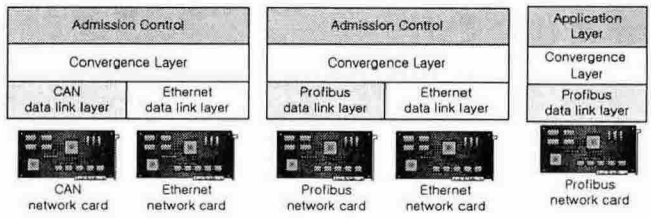


그림 3 필드버스 프로토콜 변환 게이트웨이를 이용한 구성

그림 3은 필드버스 프로토콜 변환 게이트웨이를 이용하여 네트워크를 스위치 이더넷으로 연결한 구성을 보여주고 있다. 본 논문에서 제안하는 통합 필드버스 게이트웨이는 CAN-Ethernet, Profibus-Ethernet 인터페이스를 가진 두 가지의 형태이다. 그림 4(a)와 (b)는 필드버스 프로토콜 변환 게이트웨이의 모듈 구조를 보여주고 있다. 본 절에서는 이더넷을 이용한 데이터 전달 방법과 프로토콜 변환과정을 위한 설계 방법에 대해 설명한다.



(a) CAN 게이트웨이 (b)Profibus 게이트웨이 (c)노드

그림 4 모듈 계층 구조

3.1 논리 주소와 물리 주소

필드버스에 존재하는 노드는 논리 주소와 물리 주소를 가진다. 논리 주소는 게이트웨이와 이더넷으로 연결된 전체 네트워크에서 각 노드가 가지는 유일한 주소를 나타내며 물리 주소는 각 필드버스에서의 노드 주소를 나타내며 기존의 필드버스 프로토콜의 주소이다. 논리 주소는 16비트의 길이를 가지며 상위 4비트는 네트워크 ID(게이트웨이 ID)를 의미하며 나머지 12비트는 노드 ID를 의미한다. 노드 ID는 다른 네트워크에 있는 노드와 통신을 하는 노드만이 가진다.

3.2 Convergence 계층

게이트웨이에 존재하는 Convergence 계층은 프레임을 분할하거나 재조립하는 역할을 한다. 한편 필드버스 노드에 존재하는 Convergence 계층은 노드의 논리 주소를 유지하는 계층이다. 그리고 새로운 제어 응용이 시작될 때 해당 응용에서 요구하는 메시지 요구사항을 게이트웨이에 전달하는 역할을 한다. 또한 새로운 메시지가 추가되거나 삭제되었을 때, 메시지의 요구사항이 변경되었을 때 이를 게이트웨이에게 알려주는 역할을 한다. 그림 4(c)는 필드버스 노드의 계층 구조를 보여주고 있다.

3.3 Admission Control 계층

게이트웨이에만 존재하는 Admission Control 계층은 동적인 메시지를 관리하는 역할을 수행한다. 노드로부터 전달받은 메시지의 추가, 변경, 삭제 정보를 바탕으로 필드버스와 이더넷 상의 스케줄링을 담당한다. 또한 목적지 논리 주소를 해당 게이트웨이의 이더넷 주소로 사상하는 역할과 브리지 테이블을 구성하고 관리를 한다.

3.4 프로토콜 변환

각 노드는 전송할 데이터가 생성될 경우 데이터를 필드버스로 전송할 때 해당 노드의 논리 주소를 송신 주소로 하고 목적지 노드의 논리 주소를 수신 주소로 하는 Convergence 계층 프레임을 구성한 후 해당 필드버스 프레임의 데이터 필드에 넣어 전송한다. 그림 5는 프레임 구조를 보여주고 있다. 여기에는 주소변환 테이블이 사용되며 표1(b)는 주소변환 테이블의 예를

보여주고 있다. 한편 이를 수신한 게이트웨이는 이더넷 프레임으로 변환하여 다른 게이트웨이로 전달하고 수신된 이더넷 프레임은 다시 목적지 필드버스 프레임으로 변환이 된다. 이 때 논리 주소와 이더넷 주소의 매핑 테이블이 사용되는데 그 예는 표1(a)에서 볼 수 있다.

표 1 테이블의 예

(a) 브리지 테이블 (b)주소 변환 테이블

브리지 테이블		주소 변환 테이블	
LogicalAddr.	Ether. Addr.	Node Addr.	Logical Addr
0x0A89	0x000001010789	0x789	0x0A89
0x0A8A	0x00000101058E	0x78A	0x0A8A
0x0A8B	0x0000010106A0	0x78B	0x0A8B
0x0A8C	0x0000010104EF	0x78C	0x0A8C

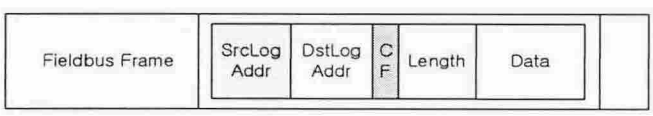


그림 5 필드버스 상에서의 프레임 구조

3.5 스위치 이더넷 스케줄링

CAN과 Profibus는 실시간 전송을 보장하는 네트워크이므로 노드의 데이터를 전달하는 이더넷 역시 실시간 전달을 보장해야만 한다[4]. Admission Control 모듈은 스위치 이더넷에서 실시간 전달을 보장하지 못할 수 있는 경우 스케줄링을 통하여 실시간 전달을 보장할 수 있도록 한다. 스위치 이더넷에서는 스위치와 노드 사이에는 충돌(collision)이 발생하지 않지만 다수의 데이터가 같은 목적지로 향하는 경우 스위치 버퍼의 한계를 초과하여 데이터 전달이 지연될 수 있다. 따라서 스위치 이더넷에 마스터/슬레이브 구조의 스케줄링을 적용하였다. 마스터는 게이트웨이들 사이에서 선출하여 정해지며 슬레이브들이 요청하는 폴링 메시지(polling message)를 수신하여 목적지가 같은 요청들간에 순서를 정하고 TM(Trigger Message)를 이용하여 이를 알려주는 역할을 한다. 슬레이브들은 TM에 따라 데이터를 전송한다. TM은 실시간 전송을 보장하기 위해 정해진 최대값을 넘지 않는 범위 내에서 가변적인 간격으로 전송된다. 또한 대역폭을 초과하는 요청이 수신될 경우 우선순위에 따라 스케줄링에서 제외시킬 수도 있다.

3.6 시뮬레이션

본 연구에서는 그림 3과 같은 환경을 가상으로 구성해 보았다. fieldbus gateway가 구현된 시스템을 스위치 이더넷으로 연결하고 각 시스템 내에는 각각 CAN과 Profibus 노드를 어플리케이션으로 구현하여 CAN의 한 노드가 Profibus로 전달되는 과정을 살펴보았다. 또한 같은 필드버스 네트워크에 있는 노드들 간의 데이터 전달과 다른 필드버스 네트워크의 노드에게 전달할 때 스케줄 알고리즘의 최적화를 위해 시뮬레이션을 하였고 이를 확인하였다. 시뮬레이션을 수행한 시스템의 구성은 아래 그림 6과 같다.

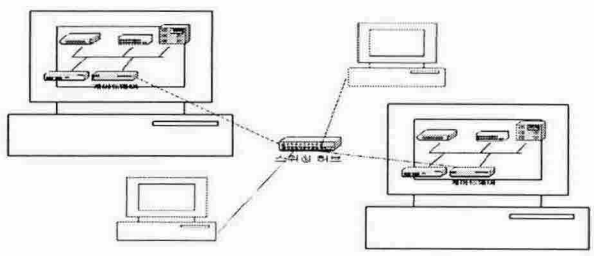


그림 6 통합 필드버스 네트워크 시뮬레이션 환경

4. 결론 및 향후 계획

본 논문에서는 서로 다른 필드버스 네트워크를 연결하는 필드버스 프로토콜 변환 게이트웨이에 대해서 알아보았다. 필드버스 프로토콜 변환 게이트웨이는 네트워크간의 데이터 전달을 위하여 프로토콜 변환을 수행하며 본 논문에서는 CAN과 Profibus간의 변환 과정에 대해서 알아보았다. 필드버스 노드는 논리 주소를 이용하여 전체 필드버스에 같은 주소 체계를 사용하며 이를 이용하여 프레임을 이더넷 프레임으로 변환하여 다른 게이트웨이로 전달하도록 하였고 수신측에서는 이를 다시 필드버스 프로토콜 프레임으로 변환을 한다. 또한 이더넷에서 실시간 전송을 위하여 스케줄링을 통한 전송을 한다. 따라서 동종의 혹은 이기종의 원격지 네트워크로 데이터를 실시간으로 전달하는 것이 가능할 뿐만 아니라 원격 제어 및 모니터링도 가능하다.

향후 프로토콜 변환 게이트웨이를 임베디드 리눅스로 구현하여 다양한 테스트를 수행할 것이다.

참 고 문 헌

[1]Cena, G.; Valenzano, A., "An improved CAN fieldbus for industrial applications", IEEE Transactions on , Volume: 44, 1997

[2] Almeida, L.; Pedreiras, P.; Fonseca, J.A.G., "The FTT-CAN protocol: why and how", IEEE Transactions on , Vol- ume: 49, 2002

[3] Tovar, E.; Vasques, F., "Real-time fieldbus communications using Profibus networks", IEEE Transactions on , Volume: 46, 1999

[4] Pedreiras, P.; Almeida, L.; Gai, P., "The FTT-ethernet protocol: merging flexibility, timeliness and efficiency", Real-Time Systems, 2002. Proceedings, 2002