객체지향기법이 도입된 분산 네트워크기반 시스템의 실시간 응답성능 평가

배덕진, 김홍렬, 김대원
명지대학교 정보제어공학과

Performance Evaluation of Distributed Network-based System Adopting an Object-oriented Method

Duck Jin Pae, Hong Ryeol Kim, Dae Won Kim
Department of Information Control Engineering, MyongJi University

Abstract - In this paper, we evaluate feasibility of an object-oriented method in a distributed real-time control environment through the prediction of delay expected. We adopt CAN as the distributed network, and the application layer of the CAN is composed of client/server communication model of COM and surroundings for the support of real-time capability of the COM. Mathematical models formalizing delays which are predicted to invoke in the COM architecture are proposed. Sensors and actuators which are widely used in distributed network-based systems are represented by COM objects in this paper. It is expected that the mathematical models can be used to protect distributed network-based systems from violation of real-time features by the COM.

1. 서 론

공장 자동화 환경에서 사용되어지는 CAN(Controller Area Network)을 포함한 필드버스 장비는 마스터/슬레이브(master/slave)의 형태로 구성된다. 공정제어(process control)를 위한 슬레이브 장비는 대부분 저가격의 단일프로세서로 구현되며, 수동적이고 단순한 동작을 하며 상위 계층의 네트워크와의 데이터 교환이나 제어에는 많은 제약을 갖는다[1,2]. 뿐만아니라 거의 모든 사무자동화 환경이나 공정자동화 환경, 그리고 MMI(Man Machine Interface) 환경은 MS Windows 기반의 PC 에서 객체지향시스템으로 구성되어 있으므로 공정제어환경에서 발생되는 하위 계층의 데이터를 상위계층과 공유하기위해서는 복잡한 절차를 거쳐야 한다. 이러한 이유로 공장 자동화 환경에서 사용자 인터페이스를 투명하게 하고, 상위 계층의 관리 시스템과 공정제어 시스템과의 유연한 데이터 교환을 위해 COM(Component Object Model), DCOM(Distributed COM) 등의 객체지향방법이 제시되고 있다[3,4]. 또한, PC의 Windows 환경에서 실시간성능을 보장받기 위해 Real-time COM[3], COM+[5] 및 OS Kernel의 확장으로 INTIME™[6], RTX™[7]등의 방법들이 제시되었다. 하지만 대부분의 기존 연구의 경우 실험 또는 모의 실험을 통한 벤치마킹(bench-making) 결과만을 제시할 뿐, 정량적인 분석이 이루어지지 않았기 때문에 지연시간에 영향을 주는 환경변수의 변화에 대해서는 그 결과를 예측할 수 없다는 단점을 갖는다.

공정제어를 위해 광범위하게 사용되고 있는 CAN의 경우에 데이터링크(data link) 계층과 물리 계층의 지연시간에 대해서는 많은 연구가 진행되어 왔으나[1,2,8,9], 보다 상위의 미들웨어(middle-ware) 및 어플리케이션 계층에서 사용되어지는 COM이나 DCOM에 대해서는 지연시간 분석이 수행되지 않았다.

이러한 이유로 본 논문에서는 CAN을 사용한 분산 환경의 공장자동화 시스템의 한 가지 예로서 반도체 제조 공정의 한 부분인 SMT(Surface Mount Technology) 공정을 대상으로 하여 하위계층과의 유연한 데이터 교환과 실시간 제어 성능을 보장하기 위해 Real-time COM 기술을 도입하여 시스템을 구성한다. 그리고 COM의 각 계층에서 발생 예측되는 지연시간을 수학적으로 모델링하여 모의실험을 수행함으로써 객체지향기법이 적용된 경우와 그렇지 않은 경우와의 성능비교를 통해 COM 도입에 따른 지연시간 발생을 정량적으로 평가하고, 객체지향기법을 도입하기 위한 일반적인 지표를 제시하고자 한다.

본 논문의 구성으로 2장에서는 CAN의 기존 시스템 구성 방식 및 Real-time COM의 개념을 기술하고, 3에서는 COM 계층의 지연시간 분석을 위한 수학적 모델링 및 모의실험 방법을 제안한다. 끝으로 4장에서는 앞으로의 연구계획 및 예상되는 결과를 기술하며 결론을 맺는다.

2. 본 론

2.1 CAN의 스케쥴링 방법 및 기존 시스템 구성

CAN 통신 프로토콜은 디바이스들간의 정보 교환 방식을 ISO의 OSI 참조 모델에 의거한 7개의 계층 중에 하위 2 개층인 데이터링크 계층과 물리 계층을 정의하고 있다[10]. 이더넷(ethernet)이 CSMA/CD 방식을 사용하는 것과는 달리, CAN 프로토콜은 데이터 링크 계층에서의 미디어 접근을 위해 충돌을 감지하고 중재할 수 있는 CSMA/AMP(Carrier Sense Multiple Access/Arbitration on Message Priority) 방식을 사용하고 있다. CAN 통신 프로토콜의 가장 큰 장점 중에 하나가 고유한 식별자를 통해 메시지에 우선순위를 부여할 수 있다는 것인데, 이 식별자를 통해 메시지간의 충돌을 방지하고 중재 역할을 한다. 메시지에 대한 식별자 부여를 위해 정적 스케쥴링 방식 또는 수행 시간 동적 스케쥴링 방식을 사용할 수 있다.

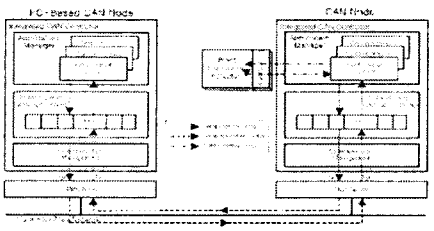

그림 1. CAN의 계층별 구조 및 지연시간

그림 1은 객체지향기법이 적용되지 않은 기존의 CAN을 이용한 분산 실시간 시스템의 구조를 계층별로 도식화한 것이다. 두 노드간에 발생하는 지연시간은 크

게 계산지연시간과 통신지연시간으로 구분할 수 있다. 계산지연시간은 로컬 노드에서의 CAN 컨트롤러, 또는 트랜시버와 같은 하드웨어의 수행으로 인한 하드웨어적인 지연요소와 메시지 입출력, 또는 하나의 칩에 구현된 제어프로세스의 처리를 위한 소프트웨어적인 지연요소로 구성된다. 또한, 통신지연시간은 메시지가 다른 노드에 전송되기 위해서, 미디어를 점유하기 위해 지연되는 미디어 접근 지연요소와 메시지가 네트워크 미디어를 통해 전송되는 전송 지연요소로 이루어진다.

2.2 Real-time COM의 구조 및 모델

COM은 프로그램의 컴포넌트 객체들을 개발하고 지원하기 위한 하부 기반구조로서, OSF/DCE(Open Software Foundation/Distributed Computing Environment)의 RPC(Remote Procedure Call) 규약[11]을 기반으로 하여 CORBA에서 정의된 수준의 기능 제공을 목표로 한다[12]. COM은 인터페이스 교섭, 생명주기 관리, 라이선스, 이벤트 서비스 등 하부 서비스를 제공한다. Real-time COM은 COM의 실시간 성능을 보장받기 위한 확장판으로, COM 오브젝트나 COM 서버 중 Real-time COM이 우선순위를 갖는다. Real-time COM의 구조는 그림 2와 같다[3].

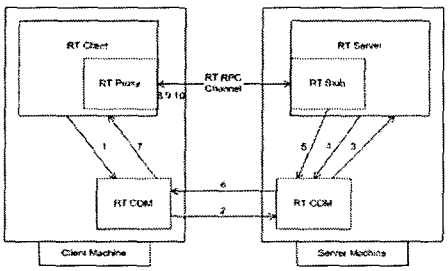

그림 2. Real-time COM의 구조 및 동작절차

Real-time COM 환경에서 클라이언트가 COM서버로부터 IReadData 인터페이스를 통해 데이터를 읽어오는 일반적인 과정은 다음과 같다:

1. 클라이언트는 서버의 UUID와 IReadData의 UUID를 통해 COCreateInstance()를 호출한다.
2. 클라이언트측의 COM은 서버 노드의 위치를 파악하고 서버측의 COM에게 호출을 전달한다.
3. 서버측의 COM은 서버를 구동한다.
4. 서버가 시작되면 스스로 레지스트리에 등록하고 COM 오브젝트를 생성하여 등록한다.
5. 서버측의 COM은 Stub을 생성하여 서버 공간에 삽입하고, Stub으로부터 IReadData의 표준 마샬링 패킷을 가져온다.
6. 서버측의 COM은 클라이언트측의 COM에게 필요한 정보를 반환한다.
7. 클라이언트측의 COM은 Proxy를 마샬링 패킷을 통해 클라이언트 공간에 삽입한다.
8. Proxy는 Stub과 RPC 채널을 설정하고, 인터페이스 포인터를 클라이언트에게 반환한다. 이 시점에서 클라이언트와 서버는 RPC 채널을 통해 직접 연결된다.
9. 클라이언트가 IReadData::Read()를 호출하면, Proxy는 호출을 패키징하여 Stub에게 전달하고, Stub은 패키징을 풀어 서버측의 실제 함수를 호출한다. 서버는 IReadData::Read()를 실행하고, Stub은 그 결과를 패키징하여 Proxy에게 전달하면, Proxy는 패키징을 풀어 클라이언트에게 결과를 반환한다.
10. 클라이언트가 서버와의 통신이 끝났을 때, 클라이언트는 IUnknown::Release()를 호출하고, Proxy는 과의 RPC 채널을 단절한다. 더 이상 필요 없는 Proxy와 Stub은 제거되기를 기다린다.

3. 모델링 및 모의실험 방법

3.1 모의실험 대상 시스템 정의

모의실험의 대상으로 반도체 제조장비간의 통신 시스템을 채택하였다. 반도체 제조장비간의 통신 프로토콜로서 이미 SECS/HSMS[13]가 널리 사용되고 있으나, 반체 제조 라인상의 각 장비의 상태를 실시간으로 모니터링하고 필요에 의해 제어 할 수도 있는 적합한 대상이기에 본 논문에서는 반도체 제조 과정 중 하나인 SMT 공정을 모의실험의 대상으로 선정하였다. 그림 3은 CAN으로 구성된 SMT 공정을 나타낸다. 각각의 장비는 Windows와 COM을 기반으로 구성되어 있으며, CAN으로 연결되어 있다.

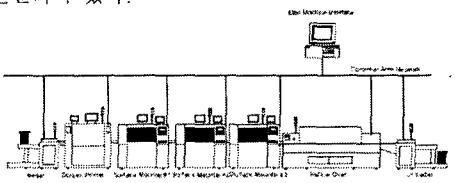

그림 3. CAN으로 구성된 SMT 공정

3.2 계층별 지연시간 모델링

본 논문에서 초점을 맞추고 있는 지연시간은 COM Server의 수행 시에 발생되는 지연시간과 OS의 RPC 채널에서의 지연시간이다. COM이라는 객체지향기법을 도입함으로써 발생되는 지연시간을 분석하는 것이 본 논문의 목적이므로 OS 계층에서 발생하는 지연시간과 네트워크 계층에서 발생되는 지연시간은 주어진 조건하에서 기존의 연구를 바탕으로 구한다[1,14].

COM 계층의 지연시간을 분석하기 위해 기존의 연구에서 제시한 방법은 COM 계층의 내부에서 메시지가 전달되는 과정별로 구분하여 그 지연시간을 측정하는 것이었다[14]. 하지만 계층별 지연시간을 측정하기만 할 뿐 수학적인 모델링을 수행하지 않았으므로 지연시간에 영향을 주는 환경변수의 변화에 대해서는 그 결과를 예측할 수 없다. 그러므로 본 논문에서는 메시지 전달 단위로 구분된 COM 내부의 지연시간에 대해 연산량과 전달 방식, 알고리즘 구현 방법 및 계층별 지연상수를 고려하여 수학적 모델링을 수행하고자 한다.

그림 4에 도식화되어 있는 각 계층별 지연시간을 하부에서부터 살펴보면 다음과 같다:

(1) Network Delay

송신측 CAN Board의 송신버퍼에서 전송미디어를 거쳐 수신측 CAN Board의 수신버퍼에 메시지가 도착하여 이벤트를 발생할 때까지의 지연시간으로 정의한다. 식(1)로 표현되는 Network Delay는 다시 세분화 하여 송신측 CAN Board가 전송을 위해 미디어를 점유하는데 까지 걸리는 대기행렬지연시간과 메시지가 미디어를 경유하여 수신측 CAN Board 의 수신 버퍼에까지 전송되는데 걸리는 전송지연시간으로 구분된다.

$$T_{net} = T_{Arbitration} + T_{Transmission} \quad (1)$$

(2) Delivery Delay

발생된 이벤트에 의해 CAN Board에서 수신된 메시지가 OS의 서비스인 IPC(Inter Process Communication)를 통해 RPC 서비스 중 RPC 통신 서비스로 전달되는 데까지의 지연시간으로 정의한다. 식(2)로 표현되는 Delivery Delay는 이벤트 처리를 위한 ISR(Interrupt Service Routine) 수행

하는데 걸리는 ISR지연시간과 IPC를 통해 RPC 서비스로 전달되는데 걸리는 IPC 지연시간으로 구분된다.

$$T_{deliv} = T_{ISR} + T_{IPC} \quad (2)$$

A

(3) RPC Runtime Delay

전달받은 메시지를 IDL(Interface Definit Language) 파일에 기술된 내용을 참조하여 UUID 확인하고, 원하는 Stub과 바인딩 되어있는 RPC 채널에 전달하는 과정에서 발생되는 지연시간으로 정의한다.

(4) Marshaling Delay

전달하고자 하는 호출이나 호출에 대한 파라미터를 서로 다른 아키텍쳐를 가진 시스템간의 통신을 위해 데이터 형태인 Byte Stream으로 변환하는 과정인 Marshaling 또는, Byte Stream을 원래의 데이터 형태로 환원하는 Unmarshaling 과정을 수행하여 실제 데이터나 호출을 서버나 클라이언트에게 전달할 때까지의 지연시간으로 정의한다.

(5) Computation Delay

서버 내부의 실제 함수가 호출되어 로컬 시스템에 연결된 디바이스에 접근하고 정해진 일련의 동작이 수행되고 그 결과 값을 다시 Marshaling 하기 위해 이벤트를 발생시킬 때까지의 지연시간으로 정의한다.

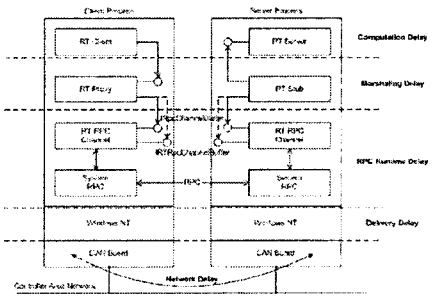

그림 4. 지연시간 모델링을 위한 계층구조

3.3 수학적 모델링을 기초로 하는 모의실험 수행

앞 장에서 제안한 방법으로 분산 네트워크 기반 시스템을 구현하기 위한 Real-time COM의 구조를 분석하고, 메시지 전달 단위로 구분된 시스템의 계층별 지연시간을 수학적으로 모델링한다. 수학적 모델링을 기초로 대상 시스템의 계층별 지연시간 및 실시간 응답특성을 알아보기 위해 모의실험을 수행한다. 모의실험을 위한 조건으로 네트워크의 부하 변화와 서버 및 클라이언트 시스템의 연산 부하, 그리고 전달 메시지의 크기를 변화시켜 각 부하의 변화에 따른 시스템 전체의 실시간 성능의 변화를 측정한다.

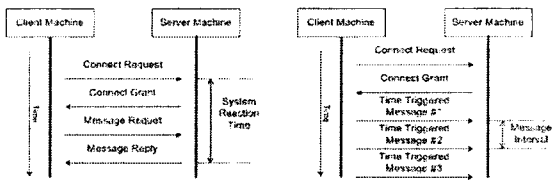

그림 5. 시스템 반응시간과 메시지 간격

대상시스템의 실시간성능을 평가하기 위한 지표[15]를 그림 5에서 설명한다. 비주기적 메시지의 경우 연결을 요청하고 연결이 설정되면 메시지 요구에 의해 원하는 메시지가 반환되는데 까지 걸리는 시간인 시스템 반응시간(System Reaction Time)과 주기적 메시지의 경우 정해진 주기에 의해 발생된 순차적인 메시지들 사이의 시간인 메시지 간격(Message Interval)을 적용하여 시스템의 실시간성능을 평가하고, 모의실험의 결과를 통해 어느 정도의 성능저하가 있는지를 분석한다.

4. 결 론

모의실험을 수행하여 얻어지는 결과는 객체지향기법을 도입하였기 때문에 필연적으로 발생하는 COM 계층의 지연시간이 분산 시스템의 전체 실시간 성능에 가시적인 성능저하를 가져올 것이라고 예상된다. 그러나 추가되는 COM 내부의 계층별 지연시간을 분석적으로 모델링하고, COM 계층에서 발생되는 지연시간을 예측할 수 있으므로 어느 정도의 성능저하를 초래할지를 정량적인 지표를 들어 예상할 수 있고, 실시간성이 요구되는 환경에서 CAN의 인터페이스를 투명하게 하는 객체지향기법을 도입할 수 있는 가능성을 제시한다. 따라서 공장 자동화 환경에서 실시간성을 보장하면서도 보다 유연한 계층간의 데이터 교환 및 제어를 수행할 수 있는 하나의 대안으로 많은 분야에 채택될 수 있으리라 예상된다.

[참 고 문 헌]

[1] Jeon, J. M., Kim, D. W., "An Analysis of Network-based Control System Using CAN(Controller Area Network) Protocol", Proceedings of the 2001 IEEE International Conference on Robotics & Automation, Vol. iv, pp. 3577-3581, May 2001.

[2] N. C. Audsley, A. Burns and M. F. Richardson, "Hard Real-Time Scheduling: The Deadline Monotonic Approach", Proc. of the IEEE Workshop on Real-Time Operation Systems and Software, 1991.

[3] Deji Chen, Aloysius Mok and Mark Nixon, "Real-time Support in COM", Proc. of the 32nd Hawaii International Conf. on System Sciences, 1999.

[4] N. Brown, C. Kindel, "Distributed Component Object Model Protocol-DCOM/1.0", http://www.microsoft.com/oledev/olecom/, 1998.

[5] Guy Eddon, "COM+: The Evolution of Component Services", Computer, Vol 33, pp. 104-106, July 1999.

[6] Martin Timmerman, "INTIME 1.20 Evaluation Executive Summary", Real-Time Magazine, 1999.

[7] Myron Zimmerman, "Windows XP with RTX - The off-the-shelf platform for Integrated Communication Equipment-White Paper", www.vci.com, March 2002.

[8] M. Farsi, K. Ratcliffm and Manuel Barosa, "An Overview of Controller Area Network", Computing & Control Engineering Journal, June 1999.

[9] J. Lehoczky, L. Sha and Y. Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior", Proc. of the IEEE Real-Time Systems Symposium, 1989.

[10] Bosch, "CAN Specification Version 2.0", ROBERT BOSCH GmbH, September 1991.

[11] DEC 1.1:Romote Procedure Call Specification.

[12] Christopher D. Gill, David L. Levine and Douglas C. Schmidt," The Design and Performance of a Real-Time CORBA Scheduling Service", May 2001.

[13] 김대원, 전종만, 이병훈, 김홍석, 이호길, "반도체 제조 공정에서 장비와 호스트간 SECS 프로토콜 통신을 위한 응용 프로그램 구현", 한국자동제어학술회의논문집, 2000.

[14] Alessandro Forin, Galen Hunt, Li Li and Yi-Min Wang, "High-Performance Distributed Objects over System Area Networks", Proc. of the 3rd USENIX Windows NT Symposium, July 1999.

[15] Thomas Handlich, Thorsten Sxcxepanski, "OPC-Making the Fieldbus Interface Transparenct", 1999.