CAN 기반 분산 내장형 시스템을 위한 CORBA의 설계

김기문 김세화[0] 홍성수
서울대학교 전기컴퓨터공학부
{kmkim, ksaehwa, sshong}@redwood.snu.ac.kr

Designing CORBA for CAN-based Distributed Embedded System

Kimoon Kim Saehwa Kim[0] Seongsoo Hong
School of Electrical Engineering and Computer Science

요 약

본 논문에서는 CAN 기반의 분산 제어 시스템을 위한 환경에 특화된 CORBA인 CAN-CORBA의 설계와 구현에 대하여 기술한다. CAN-CORBA는 예약 구독 기반의 그룹 통신과 CORBA의 전통적인 연결 기반 점대점 통신을 모두 지원한다. 또한 내장형 ORB간 통신 프로토콜인 EIOP를 사용하여 메시지 전송량을 획기적으로 줄였다. CAN-CORBA는 서울대학교에서 개발한 실시간 운영체제 mArx에 완전히 구현되었다. 다양한 경험과 실험 결과는 CAN-CORBA가 극심한 자원 제약 한계를 갖는 분산 내장형 제어 시스템을 개발하는 데 적합함을 보여주었다.

1. 서론

컴퓨터와 통신 기술의 융합이 급속하게 이루어지면서 컴퓨팅 환경과 프로그래밍의 스타일이 현저하게 변화하고 있다. 이러한 추세는 전형적인 컴퓨터 제어 시스템이 복잡한 분산 시스템으로 디자인되는 실시간 컴퓨터 제어 영역에서도 예외가 아니다. 제어 시스템은 종종 실시간 필드 네트워크에 의하여 연결된 많은 수의 값싼 마이크로 컨트롤러와 마이크로 프로세서들로 이루어진다. 한편 인터넷 관련 기술의 확산에 의해 제어 시스템조차 인터넷에 연결되도록 요구하고 있다.

네트워크화된 분산 컴퓨터 제어 시스템의 개발은 명백하게 심각한 소프트웨어 복잡성의 문제에 직면하게 되었다. 내장형 시스템은 조악한 환경과 이기종의 입출력 디바이스들에서 동작하고 엄격한 시간 제약과 고장 감내 요구를 가지고 있기 때문에 많은 경우에 있어서 극단적으로 복잡하다.

최근, CORBA [OMG 98], DCOM [Brown 98], JAVA RMI [Sun 99]와 같은 객체 지향 콤포넌트 기술이 이러한 소프트웨어 위기를 다루는 실행 가능한 해결책으로 믿어지고 있다. 그러나 CORBA와 같은 범용 미들웨어 시스템은 다음과 같은 몇 가지 이유에 의하여 내장형 실시간 제어 시스템에 직접 적용될 수 없다. (1) 과다한 자원을 요구한다. (2) 예측 가능한 동작을 보장 받기 힘들다. (3) CORBA의 연결 기반의 통신 모델은 제어 시스템 응용프로그램에 불충분하다.

이러한 표준 CORBA [OMG 98]의 문제점들을 해결하기 위하여 본 논문에서는 CAN [Bosch 91] 기반의 분산 제어 시스템을 위한 새로운 내장형 CORBA 설계를 제안한다. 이를 CAN-CORBA라고 부르기로 한다. 본 논문의 목적은 지난 [Kim 99]의 논문에서 설계한 내장형 ORB간 통신 프로토콜인 EIOP를 기반으로 하여 연결 기반의 점대점 통신과 예약 구독(subscription) 기반의 그룹 통신을 깔끔하게 동시에 지원하는 CAN-CORBA를 설계하는 것이다.

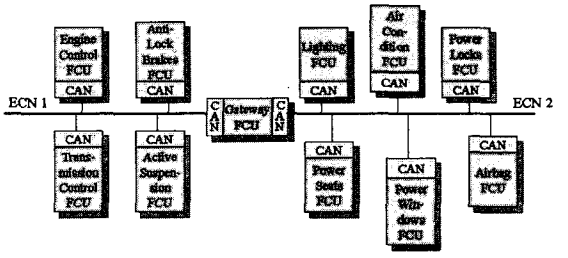

그림 1. 분산 내장 제어 시스템의 예: 차량 제어 시스템

2. 대상 하드웨어 시스템 모델

CAN-CORBA는 이전 논문 [Kim 98]과 같은 하드웨어 모델을 기반으로 한다. 그림 1은 분산 내장형 시스템의 예로서 차량 제어 시스템을 나타낸다. 대상 하드웨어 모델은 다수의 기능 제어 유닛(FCU)이 내장형 제어 네트워크(ECN)에 의하여 연결된 것으로 이루어진다. 각각의 FCU 는 하나 이상의 마이크로 컨트롤러와 마이크로 프로세서를 가지며, 센서와 구동기(actuator)와 접속하고, 규정된 제어 알고리즘을 수행함으로써 주어진 제어 임무를 수행한다. 시스템 구성에 따라 FCU는 데이터 생성자, 소비자, 또는 둘 다로서 동작한다.

본 논문에서는 의도적으로 CAN 2.0A만을 고려한다. 이는 실제로 DeviceNet [Allen 97]과 같은 상용 상위 레벨 프로토콜 제품에서 대부분이 2.0B를 지원하지 않으며, 따라서 이미 존재하는 네트워크와의 호환성을 해친다. 또한 2.0B의 여분의 메시지 비트는 버스 중재 부하를 가중시켜 메시지 전송 흐트러짐(jitter)을 크게 하여 예측 가능성을 떨어뜨린다.

3. CAN-CORBA의 트랜스포트 프로토콜

표준 CORBA는 TCP와 같은 표준 프로토콜에서 제공하는 점대점 통신만을 지원하는 반면, 분산 컴퓨터 제어시스템은 그룹 통신과 점대점 통신 모두를 요구한다. 이에 본 논문에서는 연결 기반 통신을 위한 연결 설정 프로토콜을 설계하였고 이를 예약 구독(subscription) 기반 통신 프로토콜의 채널 결합 프로토콜에 통합하였다.

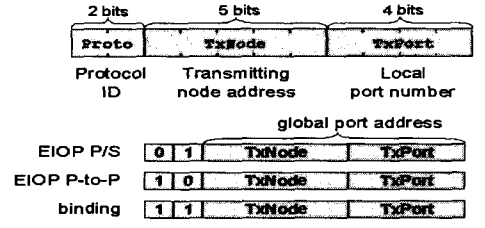

그림 2. CAN 식별자를 사용한 프로토콜 헤더 포맷

3.1 프로토콜 헤더의 정의

[Kim 99]에서와 같이 트랜스포트 레이어 프로토콜 헤더를 정의하기 위하여 CAN 식별자를 사용한다. 이에 있어서 다음과 같은 CAN에 대한 고려사항이 필요하다. (1) CAN 식별자는 단지 11비트 길이에 불과하므로 각각의 비트의 용도를 신중하게 결정해야 한다. (2) 서로 다른 CAN 노드들은 동시에 같은 식별자를 가지고 다른 메시지를 보내려는 시도를 해서는 안 된다. (3) CAN 식별자는 중재기간 동안 메시지 우선순위로 취급된다.

따라서 CAN 식별자의 효율적인 사용과 수행 부하와 프로토콜 스택의 코드 사이즈를 작게 하도록 프로토콜 디자인을 단순화하는데 초점을 맞추었다. 그림 2는 CAN-CORBA의 프로토콜 헤더 포맷을 보인다. CAN 식별자를 프로토콜 아이디 (Proto), 발송 노드 주소 (TxNode), 포트 번호 (TxPort)의 3개의 하부 필드로 나누었다. 프로토콜 아이디는 상위 레이어의 프로토콜 식별자로서 CAN 메시지의 식별자 뒤에 따라 나오는 데이터 필드는 이에 따라 해석되어진다. 01_2는 발행자/가입자 (publisher/subscriber) 프로토콜을, 10_2는 클라이언트/서버 프로토콜을 나타낸다. 11_2는 호출 채널 결합과 연결 확립을 나타내는 네트워크 관리 프로토콜에 사용한다. CAN 메시지 식별자는 숫자가 작을수록 높은 우선순위를 나타내기 때문에 발행자/가입자 프로토콜이 클라이언트/서버와 네트워크 관리 프로토콜에 비하여 높은 우선순위를 갖게 된다. 이렇게 설계한 이유는 발행자/가입자 메시지가 보통 시간에 민감한 센서 데이터를 가지고 있고 새로운 세션의 시작이 이미 허용된 실시간 메시지 수송을 끼어 들지 못하게 하기 위해서 이다. 남은 00_2는 가장 높은 우선순위의 프로토콜로서 사용자가 정의하여 긴급한 실시간 메시지를 전달하는 프로토콜로서 사용될 수 있다.

TxNode가 네트워크에 걸쳐서 전역적으로 식별 가능한 도메인 이름으로서 사용되기 때문에 TxPort와 TxNode는 합쳐서 전역 포트 식별자가 된다. 이는 서로 다른 노드에 있는 포트들이 같은 포트 번호를 가지게 하는 것을 가능하게 하여 소프트웨어 설계와 유지의 모듈성을 높이는 것을 돕는다.

3.2 예약구독 기반 통신을 위한 채널결합 프로토콜

예약 구독(subscription) 기반 통신을 위한 핵심 구성요소는 호출(invocation) 채널과 결합자(conjoiner)이다. 호출 채널은 발행자로부터 구독자 그룹으로의 가상 브로드캐스트 채널이다. 발행자는 자신의 포트들 중의 하나를 호출 채널에 밀어 넣은 후 구독자들에게 호출 채널을 알린다. 하나의 호출 채널은 하나 이상의 끼워진 포트를 가질 수 잇다. 발행자는 끼워진 포트로 호출 채널을 통해 메시지를 보내고 가입자는 호출 채널을 통해 메시지를 받는다.

결합자는 CAN의 모든 발행자와 가입자에게 잘 알려진 노드 식별자를 가진 한 노드에 존재한다. 이는 네트워크 초기화 직후에 시작되어야 하며 서비스 기간 동안 동작하고 있어야 한다.

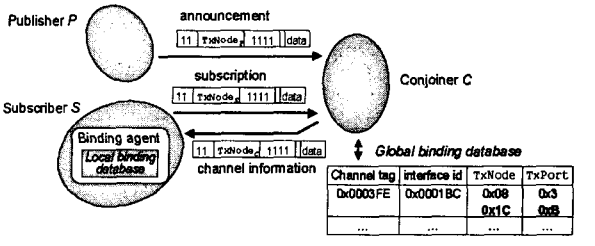

그림 3. 결합자(conjoiner)에 기반한 채널 결합 프로토콜

전역 결합 데이터베이스: 결합자는 각 호출 채널을 엔트리로 가지는 전역 결합 데이터베이스를 유지한다. 그림 3은 이를 토대로 결합자를 기반으로 한 채널 결합 프로토콜을 보여준다. 전역 결합 데이터베이스는 그림에서와 같은 4개의 항으로 구성된다. 채널 태그는 각 호출 채널에 연관된 유일한 기호 이름이다. 이것은 프로그래머가 정적으로 응용 프로그램 코드를 작성할 때 정의된다. 발행자와 가입자는 모두 이를 검색 키로 사용한다. OMG IDL 인터페이스 아이디는 각 IDL 인터페이스와 연관된 유일한 식별자로 IDL 컴파일러가 생성한다. 채널 태그와 인터페이스 아이디는 함께 각 호출 채널에 대하여 유일한 이름으로 작용한다. 각 호출 채널에 대하여 유일한 이름을 정의하는 것은 프로그래머의 책임이다.

채널 공지와 예약 구독: 그림 3에 나타나 있듯이, 발행자가 호출 채널에 붙고 싶을 때는 결합자에게 등록 메시지를 보낸다. 결합자는 이에 대하여 전역 결합 데이터베이스를 갱신하고 허가 메시지를 보낸다. 이와 유사하게 가입자는 호출 채널에 대하여 구독을 요청하는 메시지를 보내고 결합자는 이에 대하여 해당하는 결합 정보를 보내준다. 결합자는 비동기적으로 채널정보에 대한 정보를 받을 수 있다. 가입자의 지역 결합 대행자가 이러한 경우에 대한 지역 결합 데이터베이스를 관리한다. 결합자는 시스템의 모든 노드로부터 메시지를 받을 수 있어야 하기 때문에 지역 포트 번호 1111_2를 이러한 목적으로 예약되어 사용한다. 결과적으로 결합자는 이 포트 번호를 가지면서 11_2의 프로토콜을 가진 모든 메시지를 받는다.

3.3 점대점 통신을 위한 연결 확립 프로토콜

CAN-CORBA의 양방향 연결은 단방향 파이프의 한 쌍으로 이루어진다. 연결은 각 파이프의 근원지 노드에 속하는 두 개의 지역 포트를 사용한다. 목적지 노드가 근원지로부터 메시지를 받기 위해서는 근원지의 TxNode와 TxPort를 알아야 한다. 따라서 두 통신 끝 점 간의 연결 확립이 필요하다. 연결 확립 과정은 각 끝 점의 포트 주소를 교환함으로써 반대 방향의 파이프의 쌍을 결합하는 것이다. 이를 위해 소켓 프로그래밍에서 잘 알려진 귀 기울이는(listening) 포트의 개념을 사용한다. 귀 기울이는 포트는 서버에 속한 지역 포트로 서버에 접근하기를 바라는 클라이언트에게 알려진다(advertise). 채널

결합 프로토콜의 결합자와 마찬가지로 서버는 모든 CAN 노드로부터 요청 메시지를 받을 수 있어야 한다. 서버노드는 자신의 CAN 필터 레지스터를 식별자에 자신의 귀 기울이는 포트 숫자를 포함한 메시지를 받을 수 있도록 프로그램해야 한다. 그림 4는 5단계의 연결 확립 과정을 보여준다. 그림에서 연결 요청 메시지가 Proto 필드에서 11_2의 값을 가지는 것을 주목하자.

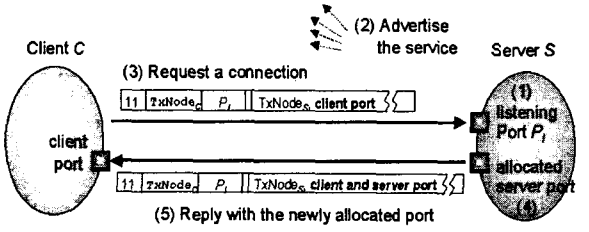

그림 4. 연결 확립 과정 (connection establishment process)

4. 두 통신 스킴을 위한 프로그래밍 모델

제안된 트랜스포트 프로토콜은 CORBA 응용 프로그램의 프로그래밍 스타일에 다음과 같이 영향을 준다.

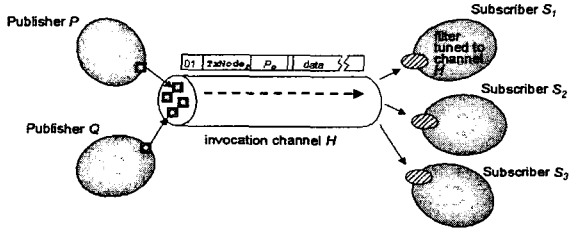

그림 5. 호출 채널을 통한 예약 구독에 기반한 통신

4.1 익명의 발행자/가입자 통신

그림 5는 발행자의 포트들이 채널에 끼워진 호출 채널을 보여준다. 호출 채널은 발행자의 코드를 작성하는 프로그래머에 의하여 OMG IDL로서 정의되며 이는 발행자가 구현해야 하는 메소드의 인터페이스를 나타낸다. 발행자는 가입자에게 메시지를 보내기 위하여 이 메소드를 호출한다. 이는 발행자로부터 가입자로의 익명의 통신을 제공한다. 가입자는 채널 태그와 호출 채널의 IDL 인터페이스만 알아도 되며 정확한 발행자는 알 필요가 없다. 또한 CORBA의 이벤트 서비스와 달리 메시지를 포워드 해주는 중간에 개입되는 오브젝트가 없이 발행자와 가입자간에 직접적인 통신이 이루어지므로 훨씬 효율적인 방법이다.

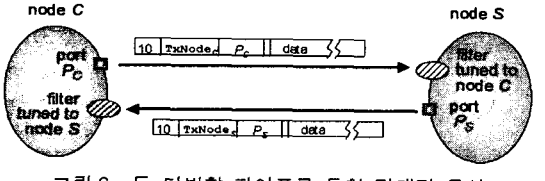

그림 6. 두 단방향 파이프를 통한 점대점 통신

4.2 클라이언트/서버 통신

그림 6은 두개의 단 방향 파이프를 사용한 연결 기반 통신의 모습을 보여준다. 그림에서와 같이 연결 상에서 전달되는 메시지는 Proto 필드로서 10_2의 값을 가진다.

5. EIOP 메시지

이전의 논문 [Kim 99]에서 제안한 내장형 ORB간 통신 프로토콜인 EIOP를 기반으로 하여, 본 논문에서 제안한 트랜스포트 프로토콜에 따라 메시지 타입을 새로이 정의한다. CORBA에서는 메소드 호출은 네트워크 위에서 전달되는 메시지로 변환된다. 표 1은 CORBA 2.2 GIOP에서 지원되는 8개의 메시지 타입과 EIOP에서만 지원되는 Publish 메시지 타입을 보여준다. 지원되지 않는 GIOP 메시지 타입인 LocateRequest와 LocateReply는 이동이 가능한 CORBA 오브젝트를 위해서만 지원되는 것이다.

Message type	Originator	GIOP	EIOP P-to-P protocol	EIOP P/S protocol
Publish				O
Request	Client	O	O	
Reply	Server	O	O	
CancelRequest	Client	O	O	
CloseConnection	Server	O	O	
MessageError	both	O	O	
LocateRequest	Client	O		
LocateReply	Server	O		
Fragment	both	O		

표 1. 지원되는 EIOP 메시지 타입

6. 결론

본 논문에서는 CAN 기반 분산 제어 시스템에 특화된 CAN-CORBA의 설계와 구현에 대하여 다루었다. CAN-CORBA의 ORB 코어는 [Kim 99]에서 설계한 EIOP를 기반으로 하여 네트워크상의 메시지 전송량을 줄였으며, 구독 예약 기반 그룹 통신과 표준 CORBA의 전통적인 연결 기반 점대점 통신을 모두 지원한다. CAN-CORBA는 서울대학교에서 개발한 실시간 운영체제인 mArx [Seo 98]에 완전히 구현되었다. 지면상의 부족으로 실험 결과는 생략하였다. 구현된 CAN-CORBA에 대하여 이루어진 다양한 실험과 경험은 CAN-CORBA가 심각한 자원 제약을 가지는 분산 내장형 제어 시스템에 적합함을 명백하게 보여주었다. 향후 연구로는 실시간 메시지의 시간 제약 요구사항을 만족하는 실시간 메시지를 지원하고 하나의 노드에 집중되어 있는 결합자를 위한 고장 감내성을 지원하도록 CAN-CORBA를 확장하는 것 등이 있다.

참고 문헌

[Allen 97] Allen-Bradley. DeviceNet speci fications, release 2.0, 1997.

[Bosch 91] Bosch. CAN specification, version 2.0, 1991.

[Brown 98] N. Brown and C. Kindel. Distributed component *object model protocol* DCOM 1.0, 1998.

[Kim 99] CAN 네트워크를 위한 내장형 ORB 프로토콜 설계, 김기문, 홍성수 외, 추계 정보과학회, 1999.

[OMG 98] Object Management Group. The Common Object Request Broker: Architecture and speci fication revision 2.2, February 1998.

[Seo 98] Y. Seo, J. Park, and S. Hong. Efficient user-level I/O *in the ARX real -time operating* system. In ACM LCTES Workshop, June 1998.

[Sun 99] Sun Microsystems. Java remote method invocation, http://java.sun.com/products/jdk/rmi, 1999.