분산 실시간 시스템에서의 네트워크 프로토콜

권 욱 현, 김 영 신

서울대학교 전기공학부

1. 서론

하나의 시스템에서 이루어지는 작업은 데이터 수집, 데이터 프로세싱, 그리고 결과 출력의 세 가지 단계로 구분할 수 있다. 이러한 시스템에 수집할 데이터의 양과 출력할 것이 많아지고 데이터 프로세싱이 증가하게 되면 단일 시스템으로는 한계가 발생하고, 이를 보완하기 위해 분산 구조가 요구된다. 특히, 이러한 구조가 정확한 시간 조건의 만족과 신뢰성의 확보라는 제한 조건이 있는 실시간 시스템에 적용되었을 때 이러한 시스템을 분산 실시간 시스템이라 한다.

분산 실시간 시스템은 모든 상황에서 종단간 데드라인(End-to-end deadline)을 만족해야 한다. 만일 종단간 데드라인이 만족되지 못하였을 경우 재앙적인 결과가 초래될 수 있기 때문이다. 이런 시스템으로는 비행기, 원자력 발전소 등 매우 다양하며 대부분의 경우 사람의 안전과 직접적인 관련을 가지고 있다. '종단간 데드라인'은 한 노드의 응용프로세스가 다른 노드의 응용프로세스에 어떠한 동작을 요구하는 시점부터, 요구받은 응용프로세스가 요구된 동작을 취하는 시점까지를 의미한다. 종단간 데드라인을 보장하기 위해서는 스케줄링(Scheduling)이 중요하다. 물론 네트워크(Network)로 연결된 분산 실시간 시스템 내에서 고려되어야 하므로 네트워크 프로토콜 또한 중요하다.

스케줄링은 하나의 공유자원을 여러 요구자에게 각 요구자의 시간 제한 조건(Timing constraints)을 만족시키면서 할당하는 것으로 정의할 수 있다. 대부분의 비 실시간 시스템의 경우에는 스케줄링의 주목적이 시스템의 모든 응용프로세스들을 수행시키는데 요구되는 전체 시간을 최소화하는 것이다. 반면에 실시간 시스템에 있어서는 주목적이 개개의 응용프로세스의 시간 제한 조건을 만족시키는 것이다. 특히, 분산 실시간 시스템의 경우에는 응용프로세스들이 네트워크를 통해서 영향을 주기 때문에 각 노드 내에서의 응용프로세스들에 대한 스케줄링 뿐만이 아니라 네트워크 자원에 대한 스케줄링이 매우 중요하게 고려되어야 한다. 분산 실시간 시스템의 응용프로세스와 네트워크에서 모두 시간 제한 조건을 만족시키기 위해서는 일반적으로 사전 스케줄링(Pre-run-time scheduling)이 필수적인 것으로 받아들여진다 [1, 2].

분산 실시간 시스템의 스케줄링에는 동기화가 매우 중요하다. 동기화는 두 개 이상의 관련된 동작이 동시에 발생하도록 만드는 것이다. 특히, 네크워크와 응용프로세스 간의 동기화가 보장되어야 종단간 데드라인을 보장할 수가 있다. 이는 네트워크와 각 노드의 응용프로세스가 별개로 스케줄링되어 동작하면 결국 분산 실시간 시스템 차원의 실시간성 요구를 만족시키는 것이 매우 어렵게 되기 때문이다. 응용프로세스 스케줄은 서로 다른 노드에 있는 응용프로세스들 사이의 통신에 걸리는 시간을 최소화하고 동시에 예측할 수 있도록 네트워크 스케줄에 동기 되어야 한다. 대부분의 동기 방법은 시계를 사용한다[3, 4]. 보다 수월한 동기화를 이루기 위해서 응용프로세스에 대한 스케줄은 네트워크에 대한 스케줄을 얻은 다음에 수행한다.

이 글에서는 분산 실시간 시스템과 관련하여 네트워크 프로토콜의 중요성을 중심으로 하여 종단간 데드라인을 만족시킬 수 있는 스케줄링 방법에 대해 소개하고자 한다. 2장에서는 분산 실시간 시스템에서 요구되는 네트워크 프로토콜의 특성에 대해 소개한다. 3장에서는 여러 가지 제안된 네트워크 프로토콜들의 특성을 소개하고, 4장에서는 분산 실시간 시스템의 스케줄링과 관련하여 네트워크 프로토콜들의 스케줄 특성을 논하였다. 5장에 결론이 있다.

2. 네트워크 프로토콜에 대한 요구사항

분산 실시간 시스템을 위한 네트워크 프로토콜의 요구사항은 매우 많으며, 또한 응용에 따라서 더 다양해 질 수도 있다. 여기서는 일반적으로 중요하게 요구되는 사항들에 대해 소개한다.

2.1 데이터 트래픽의 특성

하나의 네트워크 프로토콜은 보통 주기적인 데이터(Periodic data)나 스포래딕 데이터(Sporadic data) 중 한 가지를 효과적으로 전송한다. 주기적인

데이터를 전송하는 경우에는 네트워크에 발생하는 지터를 줄이는 것이 중요하다. 그리고 주기적인 데이터의 경우에는 미리 각 데이터의 주기를 알 수 있기 때문에 네트워크에 대한 경쟁을 미리 막도록 사전 스케줄링을 할 수 있다. 스포래딕 데이터를 전송하는 경우에는 네트워크에 의한 지연을 줄이는 것이 중요하다. 스포래딕 데이터의 경우에는 데이터가 사건에 의해 발생하기 때문에 네트워크의 서비스를 요구하는 시간을 미리 알 수 없다. 따라서 보통의 경우 데이터에 우선 순위를 지정하여 우선 순위에 의해 네트워크에 대한 경쟁을 해결한다. 이처럼 데이터의 특성에 따라 네트워크에 대한 요구사항이 다르기 때문에 하나의 네트워크 프로토콜이 두 가지를 모두 효과적으로 지원한다는 것은 매우 어려운 일이다.

2.2 시간 기반 네트워크 프로토콜과 사건 기반 네트워크 프로토콜

분산 실시간 시스템 응용은 크게 시간 기반 응용(Time based application)과 사건 기반 응용(Event based application)으로 구분할 수 있다. 보통 하나의 분산 실시간 시스템 내에서 이 두가지 특성이 모두 요구된다. 하지만 이 두가지 특성을 모두 만족하는 것은 어렵기 때문에 중요도에 따른 우선순위에 의해 선택을 하게 된다. 이런 응용의 특성에 따라 네트워크 프로토콜을 적당하게 선택해야 한다. 시간 기반 응용의 경우에는 보통 응용프로세스들의 수행시점, 선행관계, 중요도 등이 미리 결정된다. 따라서 주기적인 네트워크 요구가 대부분을 차지한다. 이런 응용에는 사전 스케줄 방법을 제공할 수 있는 네트워크 프로토콜을 사용하는 것이 바람직하다. 사건 기반 응용의 경우에는 응용 프로세스들에 대한 선행관계나 중요도 등은 미리 결정될 수 있으나 어느 시점에 사건이 발생하여 해당 응용프로세스가 네트워크 서비스를 요구할 지는 모른다. 따라서 이러한 응용에는 수행 시간 중 우선 순위에 의해 스케줄이 가능한 네트워크 프로토콜이 적당하다.

2.3 네트워크 프로토콜 지연과 지터

분산 실시간 시스템은 기본적으로 공유메모리가 없다. 따라서 분산된 각 노드들은 서로 데이터를 주고받으며, 이러한 데이터를 사용하여 동작을 한다. 따라서 데이터의 전송은 실시간 특성을 갖추어야 한다. 보통 데이터 전송은 통신 프로토콜에 따라 시간적 특성을 달리하는데 실시간 통신 프로토콜이 일반 프로토콜과 특히 구분되어야 할 특성으로는 전송시간에 대한 보장과, 작은 지터 등이다. 일반 네트워크에서는 평균 전송 시간이 작을수록 좋은 네트워크 프로토콜로 여겨진다. 하지만 분산 실시간 시스템에서는 최악의 상황을 전제로 분석 및 개발이 이루어지기 때문에 최대 전송시간이 작을수록 좋다. 물론 최대 전송시간은 최악의 경우에도 네트워크가 전송을 보장할 수 있는 시간을 의미한다. 그리고 전송시간의 변화 폭을 지터라 하는데, 지터는 센서 입력이나 구동기 출력과 관련된 데이터 전송의 경우에는 제어에 심각한 영향을 끼칠 수 있다. 따라서 작은 지터를 지원할 수 있는 네트워크 프로토콜이 좋은 프로토콜이라 할 수 있다.

2.4 물리적인 구조

분산 실시간 시스템에서는 기본적으로 멀티캐스팅(Multicasting) 기능이 지원이 되어야 하기 때문에 버스 형태의 네크워크나 링 형태의 네트워크가 물리 계층에서 적당하다. 버스 형태의 네트워크와 링 형태의 네트워크 사이에는 장점과 단점이 있기 때문에 어느 하나가 좋다고 단정할 수는 없으나 다음과 같은 특징을 가진다. 버스 형태의 네트워크의 경우에는 링 형태의 네트워크에 비해 멀티캐스팅과 브로드캐스팅 성능이 일반적으로 좋다. 또한 보다 단순한 접속(Interface) 방식을 필요로 한다. 그리고 네트워크 상의 한 노드가 고장이 나는 경우에 대한 고장 허용 특성과 한 노드에 고장이 났을 경우에 복구 능력에 있어 일반적으로 버스 형태가 링 형태보다 좋다. 하지만 요즘 많이 사용되고 있는 광섬유를 물리 매체로 사용하는 경우에는 점대점으로 연결(Point to point connection)해야 하는 광섬유의 특성상 링 형태가 버스 형태보다 좋다.

2.5 구성 가능성(Composability)

대형 분산 실시간 시스템의 경우에는 여러 작은 실시간 시스템을 통합하게 된다. 이 때, 작은 실시간 시스템들은 미리 지정된 대로 시험을 완료하고, 요구되는 실시간 특성들도 만족하는 것을 확인한 이후에 통합을 하게 된다. 그런데 일반적으로 통합을 하였을 때, 이미 만족되었던 요구되는 실시간 특성들이 통합 후에도 만족한다고 보장할 수는 없다. 이는 통합의 과정에서 기존에 가지고 있던 성질들이 깨지는 경우가 발생할 수 있기 때문이다. 하지만 분산 실시간 시스템에서는 작은 실시간 시스템에서 만족하던 실시간 특성들이 통합 후에도 작은 실시간 시스템에 특별한 수정 혹은 변화를 가하지 않고도 만족하는 것을 요구한다. 이 특성을 구성 가능성(Composability)라고 한다. 네트워크 프로토콜의 경우 사건 기반 네트워크 프로토콜은 구성 가능성을 지원하기 어렵고, 시간 기반 네트워크

프로토콜의 경우에는 구성 가능성을 지원하는 것으로 연구 결과가 나와있다[5].

2.6 클락 동기화

분산 실시간 시스템의 네트워크와 관련하여 중요한 것 중의 하나가 클락 동기화다. 공통의 클락이 없을 경우 즉, 기준 클락이 없으면 각 노드들은 제각기 자신의 클락에 따라 동작한다. 이로 인해 분산된 태스크들의 수행이나 데이터 전송과 같은 이벤트들의 순서가 뒤바뀌거나, 심지어는 시스템의 동작이 시간적 제약을 만족하는 지를 확인해 볼 도리가 없는 경우가 발생한다. 따라서 분산된 각 클락들은 기준 시간에 따라 동기화 할 수 있어야 한다. 클락 동기화는 여러 가지 방법이 있으나 Lamport[6]의 CNV나 COM이 대표적이다. 이 알고리즘들은 시스템의 각 클락들의 값을 반영하여 일정한 간격으로 분산된 클락들의 값을 갱신해준다.

3. 네트워크 프로토콜

분산 실시간 시스템에 있어 네트워크는 매우 중요한 부분이다. 여러 가지 형태의 네트워크 프로토콜이 제시되어 사용되고 있다. 분산 실시간 시스템의 통신망은 적용 범위에 따라서 계층구조를 가지는 경우가 많은데, 이런 경우 보통 IEEE 802.4 토큰 버스 규약[7], IEEE 802.5 토큰 링 규약[8], 그리고 FDDI 규약[9] 등의 통신 규약들은 중위권을 담당하고, 프랑스에서 제안된 FIP[10], 독일에서 제안된 Profibus[11], 그리고 자동차의 통신 규약으로 제안되어 국제 표준으로 자리잡은 CAN[12]등 많은 필드버스들은 하위 계층의 센서, 구동기 부분의 통신을 담당한다. 하지만 각 네트워크 프로토콜들의 특성으로 볼 때 중위 계층 하위 계층의 구분이 모호하다. 특히 필드 버스의 경우에는 중위 계층에 사용되는 경우가 종종 있어 하위 계층이라고 규정지을 수는 없다.

중위권 통신망 규약의 대부분은 논리적 혹은 물리적인 토큰 링 통신망이라고 할 수 있다. 동시에 대부분 광섬유를 통신매체로 지원하여 빠른 전송과 원거리 응용을 지원한다. IEEE 802.5 토큰 링 규약이나 FDDI는 물리적으로 통신망이 링 구조를 가지고 있으며, IEEE 802.4 토큰 버스 규약의 경우도 통신망 자체는 버스 형태를 가지지만 통신망에 연결된 스테이션들은 논리적 링을 구성하게 된다. 따라서 이들 통신망에서는 전송 권한을 나타내는 토큰이 통신망 전체를 회전하는 시간인 토큰 회전 시간이 중요한 성능 지표가 된다.

중위권 통신망과 관련된 네트워크에 대한 연구는 많이 이루어졌다. 동기(synchronous) 데이터 프레임의 전송한도 시간을 보장하는데 필요한 최악의 경우의 토큰 회전 시간에 대한 연구가 있다. FDDI 통신망에서의 최악의 경우의 토큰 회전 시간의 바운드에 대한 두 가지의 중요한 성질들이 증명되었다 [13, 14]. 실시간 데이터 프레임의 전송한도 시간을 보장하기 위해 타겟토큰 회전 시간, 토큰 보유시간, 버퍼의 수 등의 통신망 매개 변수를 정하는 방법이 제시되었다[15]. 토큰 보유시간을 할당하는 몇 가지 방법이 [16]에서 제시되었다. 이렇게 최악의 경우의 토큰 회전 시간에 대한 연구는 주로 동기 데이터 프레임을 대상으로 이루어졌다.

필드버스는 실시간 LAN(Real-time local area network)으로 분산 실시간 시스템의 통신 계층구조에서 가장 하위 부분을 담당한다. 필드버스사양들 대부분은 생산업자들이 어떤 특정한 요구사항들을 만족시키거나 주어진 계장에서의 신속한 해결책을 제시하기 위해 제안되었다. 대표적인 것으로는 원격 입출력을 위한 BITBUS (Intel), Interbus-S (Phoenix Contact), 프로세스 제어를 위한 HART (Rosemount), 자동차를 위한 CAN, VAN 또는 J1850이 있다. Batibus와 LON은 건물 자동화를 위한 것이며 CNC에 연결하는 구동기를 위해서 Sercos가 제안되었다. 범용 필드 버스인 MIL STD 1553, Profibus 그리고 FIP는 각각 미국, 독일, 프랑스에서의 필드버스 국가 표준이다.

필드버스가 본격적으로 관심을 끌게 된 것은 그리 오래되지 않지만 많은 연구가 이루어지고 있다. 예로 FIP에 관련된 연구들을 살펴보면 필드버스의 구조, FIP의 구조, FIP의 응용, 그리고 모델링 등이 있다. 우선 필드버스의 구조에 관한 연구 [17, 20]를 보면 이들 연구는 주로 대표적인 필드버스 중 Profibus와 FIP를 비교하였다. 이는 다양한 필드버스 사양에 대한 사용자의 결정을 돕기 위해 일종의 성능비교 형식으로 이루어져 있다. 내용은 주로 물리층이나 데이터링크 계층의 전송 방식, 효율 등을 비교하고 있다. FIP의 구조에 관한 연구 [21, 22]는 응용 계층에서의 시간적 공간적 요구사항을 기술하고 이들에 대해 문제점과 그에 대한 대안을 제시하였다. 이러한 연구에서는 데이터의 전송에 있어서 버스중재자의 스케줄링, 폴링 알고리즘 등에 대해서 논하고 있다. FIP의 응용에 대한 연구는 로보틱스 [23]나 NC [24]와 같은 부분에의 응용 사례를 소개하고 그 결과를 기술하고 있다. FIP의 특성인 방송 (broadcast), PDC구조의 특성을 어떻게 시스템에 응용할 수 있는가를 평가하고 있다. 모델링 관련 연구에서는 기존의 네트워크 시스템의 모델링과 마찬가지로 고전적인 Queueing system, Petri-Net [25] 등을 적용하여 프로토콜을 해석하고 시뮬

레이션하고 있다. 또한 소프트웨어적인 프로토콜의 모델을 제시한 연구[26]도 있다.

이 글에서는 중위 계층, 하위 계층에 구분없이 분산 실시간 시스템이 요구하는 네트워크 프로토콜 특성을 다룬다. 아래에서는 분산 실시간 시스템에서 현재 많이 연구가 되고 있는 CAN, TTP, FIP에 대해 소개한다.

3.1 CAN

CAN은 현재 국제 표준으로 확정된 네트워크 프로토콜이다. 기본 개념은 네트워크에서 주고받는 데이터에 우선 순위를 지정하고, 지정된 우선 순위를 하드웨어적으로 관리한다. 사전 실시간 스케줄링 과정은 따로 없으며, 데이터에 대한 우선 순위 지정이 스케줄링이라고 할 수 있다. 각 데이터에 대한 전송은 필요할 때 요구하면 우선 순위에 의해 자동적으로 순서가 결정되어 순서가 왔을 때 전송된다. 따라서 사건에 기반한 분산 실시간 시스템의 응용에 매우 좋은 성능을 낸다. 또한 모든 데이터에 대한 스케줄 가능성을 미리 검사할 수가 있기 때문에 네트워크에 신뢰성을 부여할 수 있다[27]. 하지만 CAN 프로토콜은 클락 동기화를 포함하지 않는다. 호스트 레벨에서의 분산 클락 동기화(Distributed clock synchronization)가 구현 가능하지만, 이것은 추가적인 상위 우선순위 데이터를 작은 지터로 전송해야 한다는 어려움이 있다.

3.2 TTP

TTP는 TDMA(Time Divsion Multiple Access) 방식의 네트워크 프로토콜이다. 데이터 별로 우선 순위를 지정하는 CAN과는 달리, 시간을 나누어서 사용하는 채널을 관리함으로써 실시간 통신을 지원한다. 채널에 대한 사전 스케줄링을 통해서 네트워크에 대한 요구들을 사전에 스케줄링할 수 있다. 따라서 시간 기반 분산 실시간 시스템의 응용에 적당하다. 하지만 분산된 노드들이 타임 슬롯으로 이루어진 채널을 통해 데이터를 전송하기 위해서는 클락 동기화가 매우 중요해진다. 이를 위해 TTP는 마이크로 초(usec) 정도의 정밀도로 전역 타임 베이스(Global time base)를 생성하기 위해 로컬 클락(Local clock)의 고장 허용 내부 동기화(Fault tolerant internal synchronization)를 제공한다. 각 노드가 각 데이터의 예상 도착 시간을 미리 알기 때문에, 미리 지정된 도착 시간(a priori specified arrival time)과 관측된 도착 시간(observed arrival time) 사이의 차이가 송신측 클락과 수신측 클락의 클락 차이의 지표가 된다. 따라서 동기화 데이터(synchronization messages)를 교환하거나, 데이터 안에 송신 시간을 포함시킬 필요가 없다. 즉, 클락 동기화가 데이터 길이에 대한 오버헤드없이 그리고 추가적인 데이터 없이, TTP 제어기가 하드웨어적으로 지원하는 고장 허용 클락 동기 알고리즘(Fault tolerant clock synchronization algorithm)에 의해 주기적으로 이루어진다.

3.3 FIP

FIP는 PDC(Producer-Distributor-Consumer) 형태의 네트워크 프로토콜이다. 네트워크에 분배자(Distributor)가 있어 네트워크에 대한 사용권을 관리한다. 즉, 중앙 집중식 네트워크 프로토콜이다. 주기적인 데이터의 전송에 대해서는 사전 실시간 스케줄링을 하고 비 주기적인 데이터의 전송에 대해서는 수행 중에 동적으로 스케줄링을 한다. 하지만 사전 스케줄링된 주기적인 데이터의 전송이 우선 순위가 높기 때문에 사건 기반 응용보다는 시간 기반 응용에 적당하다. 이러한 이유로 사건에 의한 데이터 전송이 주기적인 데이터의 전송보다 우선순위가 낮은 분산 실시간 시스템에 많이 사용되고 있다. 대부분의 네트워트 프로토콜이 시간, 공간적인 문제를 응용프로세스에 맡기는데, FIP는 이러한 문제들을 네트워크 소프트웨어 안에서 규정한다. 또한 FIP는 데이터링크 계층을 미리 스케줄하기 때문에 데이터링크 계층 아래에서는 스케줄링 방법에 따라서 쉽게 실시간성을 만족시킬 수 있다. 그리고 실제 전체 시스템 차원의 실시간 특성을 얻기 위해서는 메모리 크기, 매체의 이용률, 그리고 지터(jitter) 등, 여러 제한 조건들을 만족시키도록 한 스케줄링 방법과, 응용프로세스까지의 실시간 특성을 만족시키기 위하여 네트워크 스케줄을 데이터링크 계층과 응용 계층을 서로 연계해서 수행한 연구 결과가 있다[28].

4. 분산 실시간 시스템에서의 스케줄링

서로 다른 노드에서 동작하는 관련된 실시간 태스크들이 데드라인에 맞추어 수행되기 위해서는 네트워크에 대한 스케줄링과 이를 고려한 응용프로세스에 대한 스케줄링이 이루어져야 한다. 스케줄링 방법은 크게 하드 실시간 스케줄링(Hard real-time scheduling)과 소프트 실시간 스케줄링(Soft real-time scheduling)으로 나눌 수 있다. 하드 실시간 스케줄링은 하드 실시간 데드라인을 가진 프로세스들에 대한 스케줄링을 담당한다. 하드 실시간 데드라인은 지키지 못했을 때 심각한 위험과 손해를 줄 수 있는 데드라인을 말한다. 소프트 실시간 스케줄링은 소프트 데드라인을 가진 프로세스들에 대한 스케줄링을 담당한다. 소프트 실시간 데드

라인은 지키지 못했을 때 해당하는 작업, 데이터 등이 의미가 없어지는 데드라인을 말한다. 그리고 하드 실시간 스케줄링은 동적 스케줄링(Dynamic scheduling)과 사전 스케줄링(Pre-run-time scheduling) (혹은 정적 스케줄링(Static scheduling))으로 나눌 수 있다. 여기에서는 분산 실시간 시스템과 관련하여 하드 실시간 스케줄링만을 가정한다.

스케줄러가 스케줄 결정을 수행 시간 중(Run time)에 하면 동적(Dynamic)이라 한다. 현재 준비된(Ready) 태스크 중 하나를 선택하여 자원에 대한 사용권을 주게 된다. 동적 스케줄러는 현재 요구만을 고려하여 스케줄 결정을 하기 때문에 소요되는 노력이 실질적이다. 하지만 동적 스케줄링의 경우에는 단일 프로세서 다중 태스크 환경(Single processor multi-tasking)에서도 상호 배제나 선행 제한 조건이 있으면, 빡빡한 데드라인(Tight deadlines)을 보장하는 것이 어렵다. 따라서 분산 실시간 시스템의 경우에는 더욱 복잡해지기 때문에 하드 데드라인을 요구하는 경우에는 많이 쓰이지 않고 있으며, 최대한 노력(Best effort) 방식으로 실시간 요구사항들을 만족시킬 수 있는 응용에 주로 사용된다.

반면에 수행 이전에 스케줄 결정을 미리 하는 스케줄러를 사전(Pre-run-time) (혹은 정적(Static)) 스케줄러라 한다. 사전 스케줄러는 공유 자원 요구에 대한 모든 사항, 예를 들면, 최대 수행 시간(Maximum execution time), 선행 제한 조건(Precedence constraints), 상호 배제 제한 조건(Mutual exclusion constraints), 그리고 데드라인(Deadlines) 등에 대한 정보를 미리 필요로 한다. 이 정보들을 바탕으로 일종의 스케줄 테이블을 구성하여 실행 중에 이 스케줄 테이블에 기초하여 스케줄링을 한다. 따라서 수행 중 오버헤드가 작다. 사전에 스케줄을 결정하기 때문에 스케줄링 방식에 탐색 알고리즘(Search algorithm)을 사용하는 경우가 많이 있다.

아래에서는 분산 실시간 시스템의 네트워크 프로토콜과 관련하여 스케줄링을 할 때 고려해야 할 제한 조건들에 대해 소개하고, 각 네트워크 프로토콜들의 스케줄링 특성에 대해 기술한다.

4.1 네트워크 스케줄링에서의 제한 조건

분산 실시간 시스템에 적용할 네트워크를 선택할 때에도 위에서 언급한 스케줄링을 고려하여야 전체적인 실시간 특성을 얻을 수 있다. 우선 네트워크에서 공유 자원이라고 하면 통신 매체가 된다. 각 노드들은 통신 매체를 통하여 데이터를 송수신하여 해당 데이터를 공유하게 되며, 그 데이터에 기반하여 다음 동작을 취하게 된다. 따라서 네트워크 스케줄링은 분산 실시간 시스템에서 하드 실시간 특성을 만족시키기 위한 매우 중요한 부분을 차지한다. 네트워크 스케줄링은 네트워크에서 걸리는 지연에 예측 가능성(Predictability)을 부여하는 과정이다. 그러므로 앞에서 언급한 것처럼 사전 스케줄링이 필요하며, 사전 스케줄링은 다음과 같은 제한 조건들을 고려하여 수행되어야 한다.

통신 지터(Communication jitters)

분산 실시간 시스템에서는 네트워크를 통해 주고받는 데이터를 사용하여 제어를 하게 된다. 원하는 방향으로 시스템이 제어가 되기 위해서는 데이터에 가해지는 통신 지연이 일정해야 한다. 예를 들어 센서 입력 혹은 구동기 출력에서 통신 지연이 크게 차이가 발생하는 경우 전체 시스템의 성능과 안전에 큰 위협이 될 수도 있다. 따라서 통신 지터를 최소화할 수 있는 스케줄링 방법이 사용되어야 한다.

메모리 크기(Memory size)

분산 실시간 시스템에 있어서 사전 스케줄링은 매우 중요하다. 그런데 사전 스케줄링 방법은 미리 스케줄 정보를 오프 라인(Off line)으로 결정을 해서 그 정보를 미리 메모리에 가지고 있어야 한다. 하지만 대부분의 네트워크 시스템의 경우 제한된 메모리 크기가 제공된다. 따라서 사전 스케줄 정보가 크면 메모리에 그 정보를 모두 저장할 수 없게 되며, 결국 그 스케줄 정보는 사용할 수 없다. 이것은 스케줄은 가능하나 실제로 구현 가능성이 없는 경우이다. 이렇듯 스케줄 정보를 줄일 수 있는 스케줄링 방법이 중요한 경우가 종종 있다.

네트워크 이용율(Network utilization)

네트워크의 대역은 제한되어 있다. 따라서 대역에 대한 관리가 중요하며 네트워크 이용율이 네크워크의 스케줄 가능성에 영향을 많이 준다. 또한 미리 일부분만을 사전 스케줄링을 하고 일부분에 대해서는 동적 스케줄링을 수행하는 경우, 미리 스케줄된 데이터에 대한 네트워크 이용률이 지나치게 크면 미리 스케줄되지 않은 데이터의 전송이 불가능하게 된다. 또한 대부분의 네트워크 프로토콜의 경우 네트워크 이용율이 어느 정도를 넘게되면 제 성능을 발휘하지 못하는 경우도 있다. 따라서 네트워크 이용율을 줄일 수 있는 스케줄링 방법을 사용하는 것이 필요하다.

4.2 네트워크 프로토콜의 스케줄링 특성

위에서 언급한 여러 제한 조건들을 고려하면서

사전 스케줄링을 하는 것은 쉽지 않다. 또한 네트워크 프로토콜 마다 통신 매체 사용에 대한 방식이 다르기 때문에 사용할 수 있는 스케줄링 방법이 한정된다. 따라서 분산 실시간 시스템에 사용할 네트워크 프로토콜을 선택할 때는 신중하게 해야 한다. 현재 많은 종류의 네트워크 프로토콜이 제안되어 사용되고 있으나, 종단간 데드라인을 요구하는 분산 실시간 시스템에는 별로 사용되고 있지 못하다. 네트워크 프로토콜의 특징에 따라 살펴보자.

CSMA/CD

CSMA/CD(Carrier Sensle, Multiple Acces/ Collision Detect) 형태의 네트워크 프로토콜에는 대표적으로 이더넷(Ethernet)과 LON이 있다. 이 프토토콜은 네트워크에 대해 사전 스케줄 방법을 제공하지 못하며, 또한 무제한 지연(Unbounded elay)의 가능성으로 분산실시간시스템에적용하기에는 적당하지 않다.

CAMA/CA

CAMA/CA(Carrier Sense, Multiple Access/ Collision Avoidance) 형태의 네트워크 프로토콜에는 대표적으로 CAN(Control Area Network)이 있다. CAN과 같은 형태의 프로토콜은 데이터마다 미리 지정된 우선순위를 사용하여 네트워크에 접근하기 때문에 우선순위가 높은 데이터가 긴급하게 네트워크 서비스를 받을 수 있다. 모든 데이터의 우선 순위는 미리 결정된다. 하지만 현재 네트워크 서비스를 받고 있는 낮은 우선 순위 데이터에 의해 높은 우선 순위 데이터가 블록킹(Blocking)되는 경우가 발생할 수 있다.

Timed-token

토큰 버스와 (대표적으로 IEEE 802.4,Profibus) 토큰 링(IEEE 802.5)과 같이 시간토큰(Timed-token)을 사용하는 프로토콜의 경우에는 각 노드에 대해 토큰을 소유하였을 때 최대 전송시간을 보장해 준다. 하지만 토큰 회전 시간이 일정하지 않기 때문에 예측 가능한 사전 스케줄링을 하기가 매우 곤란하다. 또한 토큰 프로토콜에서는 토큰 분실이라는 심각한 오류 상태가 발생할 수 있다. 이 경우에 한 노드가 아닌 전체 노드에 영향을 주기 때문에 토큰을 재 생성하여 네트워크가 정상 상태로 돌아오는 시간이 매우 중요하다.

TDMA

TDMA(Time Division Multiple Access) 형태의 프토토콜에는 대표적으로 TTP(TimTriggered Protocol)가 있다. 통신 대역을 시간에 따라 분할해서 채널로 사용한다. 각각 데이터마다 미리 채널을 할당하여 통신이 이루어진다. 따라서 이 프로토콜에서는 사전 스케줄이 가능하다. 이 경우에 채널 관리와 클락 동기가 중요한 문제가 된다.

PDC

PDC(Producer-Distributer-Consumer)형태의 프로토콜에는 대표적으로 FIP(FactoryInstrumentation Protocol)가 있으며, 중앙 집중식 매체제어 방법을 사용한다. 네크워크에 분배자(Distributor)가 있어 미리 스케줄된 정보를 바탕으로 순서에 따라 데이터 전송을 관리한다. 중앙 집중식이기 때문에 관리와 운영이 쉬운 반면 통신 오버헤드가 상대적으로 크다. 또한 분배자가 고장이 났을 경우 전체 네트워크에 영향을 주기 때문에, 분배의 역할이 다음 분배자로 넘어가는 시간이 중요하다.

5. 결론

분산 실시간 시스템의 성능에 있어 네트워크 프로토콜은 매우 중요한 역할을 한다. 네트워크를 통해서 상호 동작하는 실시간 응용프로세스들이 시간 제한 조건들을 만족시키면서 원하는대로 실행하기 위해서는 네트워크를 고려한 스케줄링이 필수적이라는 것을 보였다. 아울러 하드리얼타임 특성을 요구하는 분산 실시간 시스템의 스케줄링은 사전 스케줄링 방법이 필수적임을 보였다.

이러한 이유로 분산 실시간 시스템에 적당한 네트워크 프로토콜을 선정하는 경우 스케줄링을 고려하여 결정하여야 한다. 분산 실시간 시스템이 사건에 의해 발생되는 데이터 처리와 이에 따른 동작과 관련된 응용인 경우에는 이에 알맞은 CAN과 같은 사건 기반 네트워크 프로토콜을 사용하는 것이 바람직하다. 그리고 주기적인 데이터의 교환 혹은 시간에 따른 동작과 관련된 응용인 경우에는 PDC 형태의 FIP나 TDMA 형태의 TTP같은 네트워크 프로토콜이 적당하다.

참고문헌

[1] J. Xu and D. Parnas, "On satisfying timing constraints in hard-real-time system s," acm sigsoft 91, pp. 132-146, 1991.

[2] K. Ramamritham and J. Stankovic, "Scheduling algorithms and operating systems support for real-time systems," *Proc. of the IEEE,* vol. 82, no. 1, pp. 55-67, January, 1994.

[3] N. Suri, M. Hugue and C. Walter,

"Synchronization issues in real-time systems," Proc. of the IEEE, vol. 82, no. 1, pp. 41-54, January, 1994.

[4] H. Kopetz and G. Grunsteidl, "TTP-A protocol for fault-tolerant real-time systems," *IEEE COMPUTER,* vol. 27, no. 1, pp. 14-23, January, 1994.

[5] H. Kopetz, "A comparison of CAN and TTP," *DCCS'98,* pp. 119-130, 1998.

[6] L. Lamport and P. Melliar-Smith, "Synchronizing clocks in the presence of faults," *Journal of the Association for Computing Machinery,* vol. 21, no. 1, pp. 52-78, 1985.

[7] *Token Passing Bus Access Method Physical Layer Specification,* ANSI/IEEE Standard 802.4, 1985.

[8] *Token Ring Access Method and Physical Layer Specification,* IEEE Standard 802.5, Y1983.

[9] *FDDI Token Ring Media Access Control,* ANSI Standard X3.139, 1987.

[10] *General Purpose Field Communication System,* prEN 50170, WorldFIP, 1995.

[11] *DIN 19 245 Profibus Standard, Profibus Trade Organization,* 1993.

[12] Road vehicles-Interchange of digital information - Controller area network (CAN) for high-speed communication, ISO 11898, 1993.

[13] M. J. Johnson, "Proof that timing requirements of the FDDI token ring protocol are satisfied," *IEEE Transactions on Communications,* vol. COM-35, no. 6, June, 1987.

[14] K. C. Sevick and M. J. Johnson, "Cycle time properties of the FDDI token ring protocol," *IEEE Transactions on software Engineering,* vol. SE-13, no. 3, March, 1987.

[15] N. Malcolm and W. Zhao, "The timed-token protocol for real-time communications," *IEEE Computer,* vol. 27, no. 1, January, 1994.

[16] G. Agrawal, B. Chen, W. Zhao, and S. Davari, "Guaranteeing synchronous message deadlines with the timed-token medium access control protocol," *IEEE Transactions on Computers,* vol. 43, no. 3, March, 1994.

[17] P. Raja, L. Ruiz, J. Hernandez, C. Fuhrman, G. Noubir, and J. D. Decotignie "Synchronous model for fieldbus applications," *IECON'93* pp. 525 - 529, IEEE 1993

[18] S. Cavalieri, A. Di stefano, and O. Mirabella "Assessment of the priority mechanism in the fieldbus data link layer," *IEEE IECON'91* pp 1673 - 1678, 1991

[19] P. F. Rosa, F. Vasques, and R. Valette "The conception of distribute application and the fieldbus configuration : A concurrent engineering vision," *IEEE IECON'94,* pp. 1135 - 1140.

[20] P. Noury "The fieldbus, a worldwide challenge, FIP, second generation technology," *COMPEURO 93,* Paris, Enry, IEEE Comp. Soc. Press, pp 446 - 451 June, 1993.

[21] Gonzalo Ulloa "Fieldbus application layer and real-time distributed systems" IEEE IECON'91, pp 1679 - 1683.

[22] J. Decotignie and P. Raja "Fulfilling temporal constraints in fieldbus," IEEE IECON'93, pp. 519 - 524,

[23] S. Cavalieri, A. Di Stefano and O. Mirabella "Meeting time requirements in robotics by a fieldbus communication system," IEEE IECON'93, pp. 1915 - 1920.

[24] G. Cena, C. Demartini, L. Durante and A. Valenzano "A FIP prototype network for numerical control applications," IEEE IECON'94 pp. 1195 - 1199.

[25] G. Noubir, P. Raja, and J. Decotignie "Simulating the fieldbus synchronous model by timed petri nets," IEEE IECON'94 pp. 1205 - 1209.

[26] G. Cena, C. Demartini and L. Durante "An object oriented model for the FIP protocol," IEEE *IECON'94,* pp. 1214 - 1219.

[27] K. Tindell, H. Hansson, and A. Wellings, "Analysing real-time communication: controller area network (CAN)," *IEEE*

Real-Time Systems Symposium, pp. 259-263, 1994.

[28] Y. Kim, S. Jeong and W.-H. Kwon, "A pre-run-time scheduling method for distributed real-time systems in an FIP environment," *Control Engineering Practice,* vol. 6, pp. 103-109, 1998.

저자소개

권 욱 현

1966년 서울대학교 전기공학 학사
1972년 서울대학교 전기공학 석사
1975년 미국 Brown 대학 제어이론으로 박사학위
1975년 - 1976년 Brown대학의 연구조교
1976년 - 1977년 Iowa대학의 조교수
1977년 - 현재 서울대학교 교수
1981년 - 1982년 Stanford 대학의 방문 교수
현재 제어계측 신기술 연구센터의 소장
연구분야는 다변수 강인제어, 예측제어, 이산현상 시스템, 네트워크 분석, 공장 자동화를 위한 컴퓨터 응용 등.

김 영 신

1993년 서울대학교 공과대학 제어계측공학과 졸업 (학사)
1995년 서울대학교 공과대학 제어계측공학과 졸업 (석사)
1997년 서울대학교 공과대학 전기공학부 박사수료 현재 서울대학교 공과대학 전기공학부 박사과정.

<관심분야>
- 분산 실시간 시스템Network,
- Hybrid System, Discrete Event System 등.