OLP 기반의 산업용 로봇 시뮬레이션 시스템의 분산화

김재욱[O]* 이정태* 류기열* 김종철* 범진환**
아주대학교 정보통신 전문대학원*
아주대학교 기계공학부**
{allah[O], jungtae, kryu, ehkjc, jhborm}@ajou.ac.kr

Distribution of an OLP based Industrial Robot Simulation System

JaeWook Kim[O]*, JungTae Lee*, Ki-Yeol Ryu*, Jong-Chul Kim*, Jinhwan Borm**
The Graduate School of Information and Communication, Ajou University*
Division of Mechanical Engineering, Ajou University**

요 약

최근 산업현장에서는 작업의 빈번한 변화에 따른 로봇의 재배치, 테스트로 인해 많은 비용이 소모되고 있다. 이러한 문제점의 해결방안으로 OLP(Off-line Programming) 시뮬레이션 시스템이 주목받고 있다. 현재 대부분의 OLP 시뮬레이션 시스템은 하나의 컴퓨터에서에서 실행되는 하나의 응용프로그램으로 되어있다. 따라서 로봇을 제어하는 시뮬레이션 프로그램이 복잡해지고 시뮬레이션 대상이 되는 작업장의 규모가 커지게 되면 이와 비례하여 시뮬레이션에 소모되는 시간이 늘어나게 되고 이로 인하여 시뮬레이션 진행상황을 실시간에 맞추어 출력하지 못하게 되는 경우가 발생하게 된다. 또한 하나 컴퓨터에서 실행되는 응용프로그램의 한계로 인해 시뮬레이션 상황을 출력하는 화면크기는 제한될 수밖에 없어지고 여러 작업장의 시뮬레이션 진행상황을 동시에 확인하기가 힘들어 진다. 본 논문에서는 이러한 문제점을 해결하기 위하여 하나의 PC에서 실행되는 OLP 시뮬레이션 시스템을 사용자가 지정하는 로봇의 집합 단위로 네트워크로 연결된 여러 개의 PC에서 실행할 수 있도록 하여 다수의 로봇 제어 프로그램을 실행하는데 걸리는 부하와 시뮬레이션 진행 상황을 출력하는 CAD시스템에 걸리는 부하를 분산 시키고 연결된 여러 PC를 통해 시뮬레이션 진행상황을 자유롭게 확인할 수 있는 구조를 제안하였다. 그리고 시뮬레이션 시스템의 분산화로 인해 발생하는 동기화 문제에 대한 해결 방안으로 기존 시스템에 사용되는 이벤트/사이클 혼합 방식의 로봇 시뮬레이션 시스템의 알고리즘을 분산시킨 방안을 제시하였다.

1. 서 론

최근 산업현장에서는 소비 시장의 다품종 소량생산요구에 의한 작업의 빈번한 변화에 따른 로봇의 재배치, 설계로 인한 테스트에 상당한 비용이 소모되고 있다. 이러한 문제점을 해결하기 위한 방법으로 실제 작업환경과 유사한 환경을 가진 시뮬레이션 시스템인 오프라인 프로그래밍 시뮬레이션 시스템(OLP :Off-line Programming Simulation System)이 주목받고 있다. OLP 시뮬레이션 시스템은 3차원 그래픽으로 모델링된 가상의 로봇을 이용하여 이를 제어하는 로봇 프로그램을 실행하게 하고 가상 로봇의 동작을 컴퓨터 화면상에서 확인할 수 있게 해주는 시스템을 지칭한다.[1]

현재 대부분의 OLP 시뮬레이션 시스템은 하나의 컴퓨터에서 실행되는 응용프로그램으로 되어있으며 시뮬레이션 대상이 되는 작업장의 규모에 대한 확장성이 좋지 않은 구조를 가지고 있다. 따라서 시뮬레이션 대상이 되는 작업장에 포함되는 로봇의 개수가 크게 증가하게 되면 이와 비례하여 시뮬레이션 시스템에 걸리는 부하가 증가하게 된다. 이로 인해 시뮬레이션 진행상황을 실시간에 맞추어 화면에 출력하지 못하게 되는 경우가 발생하게 되고 실시간 보다 많이 늦어지게 될 경우 화면을 통해 출력되는 로봇의 움직임을 제대로 확인하기가 어려워진다.

또한 하나의 컴퓨터에서 응용프로그램으로 동작하는 구조상의 한계로 인해 모니터를 통해 출력할 수 있는 화면크기의 제약될 수밖에 없다. 따라서 여러 개의 작업장을 시뮬레이션 하는 경우에는 각 작업장 간의 여러 로봇의 동작을 동시에 확인하기가 어려워지고 이로 인해 동일한 시뮬레이션을 여러 번 수행해야 하는 경우가 발생할 수 있다.

본 논문에서는 이러한 기존의 OLP 시뮬레이션 시스템의 한계로 인해 발생하는 문제점들을 해결하기 위한 방안으로 전체 시뮬레이션 시스템을 네트워크로 연결된 여러 개의 PC에서 시뮬레이션에 포함되는 로봇의 단위로 자유롭게 분산시켜 처리할 수 있는 구조를 제안한다. 로봇 단위의 시뮬레이션 시스템 분산화를 통해 시뮬레이션 진행상황을 화면에 출력하는 모듈과 시뮬레이션 언어로 작성된 프로그램의 처리에 걸리는 부하를 분산시켜 시뮬레이션의 규모와 복잡성에 대한 확장성을 제공하도록 한다. 그리고 OLP 시뮬레이션 시스템의 분산화에 의해 발생하는 동기화 문제에 대한 해결 방안을 제안한다.

2. OLP 기반의 로봇 시뮬레이션 시스템

OLP 기반의 로봇 시뮬레이션 시스템이란 로봇 프로그래밍 언어로 작성된 로봇 프로그램을 가상머신에서 실행시키고 이에 따른 실행결과를 CAD 시스템에 의하여 보여주는 시스템이다.[2]

이번 장에서는 일반적인 OLP 기반의 로봇 시뮬레이션

시스템의 구조와 구성요소에 대해서 알아본다. 기존의 일반적인 로봇 시뮬레이션 시스템의 구조와 순차적으로 시뮬레이션을 수행하는 방법인 이벤트/사이클 방식의 로봇 시뮬레이션 시스템을 살펴보면서 기존의 로봇 시뮬레이션 시스템이 분산화 될 경우 어떠한 문제점이 발생하는지에 대해 살펴보도록 한다.

2.1 OLP 기반의 로봇 시뮬레이션 시스템의 구조

OLP 기반의 로봇 시뮬레이션 시스템은 일반적으로 다음의 요소로 구성된다. 그림1은 일반적인 로봇 시뮬레이션 시스템의 구조를 나타낸 것이다.

Simulation Language File은 로봇 시뮬레이션 언어로 작성된 프로그램이 저장된 파일이다. 로봇 시뮬레이션 언어는 로봇을 제어하기 위해 설계된 언어이며 로봇 시뮬레이션 시스템은 로봇 시뮬레이션 언어로 작성된 로봇 프로그램을 실행하면서 로봇을 제어하며 시뮬레이션을 수행한다.

가상 머신(Virtual Machine)은 하나의 로봇과 대응되는 구성요소이다. 가상머신은 Simulation Language File로 구현된 로봇 프로그램을 실행하는 기능을 한다.

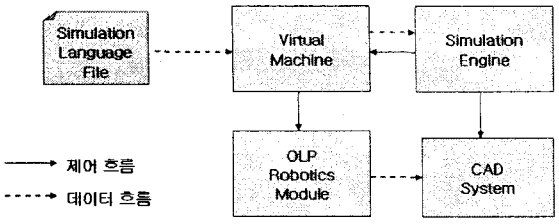

그림 1. 일반적인 로봇 시뮬레이션 시스템의 구조

시뮬레이션 엔진(Simulation Engine)은 전체 시뮬레이션을 조율하는 역할을 한다. 시뮬레이션에 포함되는 로봇이 하나 이상이므로 각각의 로봇에 대응되는 가상머신에 대한 제어와 동기화 그리고 시뮬레이션의 진행상황을 화면에 출력해주는 CAD 시스템을 제어하는 기능을 한다.

OLP Robotics Module은 시뮬레이션의 대상이 되는 워크셀과 로봇들을 가상환경에서 이용할 수 있도록 모델링한 것으로 로봇의 동작 위치, 걸리는 시간들을 계산하고 CAD 시스템에 화면 업데이트를 위한 데이터를 제공해 준다.

CAD 시스템은 3D로 모델링된 로봇과 로봇의 작업 환경을 화면으로 출력해주는 기능을 수행하며 시뮬레이션 상에서 로봇의 동작을 확인할 수 있도록 해준다.

일반적인 OLP 기반 시뮬레이션 시스템에서는 로봇과 가상머신이 일대일로 대응되게 된다. 따라서 시뮬레이션에 포함되는 로봇의 개수가 늘어날수록 가상머신의 개수가 늘어나게 되고 이로 인해 전체 시스템에 걸리는 부하는 점점 가중되게 된다. CAD 시스템 또한 로봇의 수가 증가할수록 처리해야할 데이터의 양이 증가하게 되고 화면 업데이트 작업으로 인한 부하가 늘어나게 된다. 본 논문에서는 이에 대한 해결 방안으로 시뮬레이션을 수행하는 여러 가상머신을 네트워크로 연결된 여러 PC로 나누어 병렬로 처리하는 방법을 제안한다.

2.2 이벤트/사이클 혼합 방식 로봇 시뮬레이션 시스템

OLP 기반의 로봇 시뮬레이션 시스템은 상호 신호를 주고받으며 실행되어야 하는 여러 개의 로봇 프로그램들의 실행결과가 실제 로봇이 동시에 작동하는 것처럼 보여줄 수 있도록 시뮬레이션 되어야 한다.

이벤트/사이클 혼합 방식의 로봇 시뮬레이션 시스템은 동시에 실행되어야 하는 로봇 프로그램을 정의된 이벤트를 이용하여 순차적으로 처리하는 방법이다.

이벤트/사이클 혼합방식의 로봇 시뮬레이션 시스템은 복수의 로봇을 동시에 동작시키는 것이 아니라 각각의 로봇을 정해진 짧은 시간 단위로 동작시킨 후 동기화를 수행함으로서 여러 개의 로봇의 순차적 실행이 동시 실행과 같은 결과를 나타내게 된다.[3][4]

이벤트/사이클 혼합 방식의 시뮬레이션 시스템에서 한 순간에 실행시키는 로봇(가상머신)은 하나이다. 따라서 여러 로봇프로그램을 순차적으로 처리를 하면서 모든 로봇이 동시에 실행되는 것처럼 처리하기 위해 운영체제가 프로세스 스케줄링을 수행하는 것처럼 각 가상머신이 실행하는 로봇 프로그램을 여러 실행 구간으로 나누어 이를 시뮬레이션 상의 시간 순으로 정렬한 후 순차적으로 실행해 나가는 방법을 사용한다. 실행구간은 다른 로봇(가상머신)에 영향을 주지 않고 로봇의 움직임에 변화가 없는 독립 실행 구간과 다른 로봇에 영향을 주거나 로봇의 움직임이 있는 연관 실행 구간으로 구분된다. 여기서 다른 실행구간에 영향을 주는 연관 실행 구간을 이벤트로 정의한다.

다른 로봇에 영향을 주는 연관 실행 구간의 예는 다음과 같다. 각각의 로봇프로그램이 공유하는 전역변수의 값을 변경 시킬 경우, 다른 로봇의 특정 프로그램을 실행 시킬 경우, 로봇의 동작이 시작되거나 끝날 경우, 가상머신이 동기화를 위해 정해진 시간만큼의 실행이 완료된 경우 등이 있다.

연관 실행 구간은 다른 로봇(가상머신에) 영향을 주기 때문에 동일한 시뮬레이션 시간 선상에 있는 다른 가상머신의 실행 구간 보다 먼저 실행되어야 한다. 따라서 각 가상머신의 다음 연관 실행 구간(이벤트) 정보를 얻어와 이를 시간 순서에 맞도록 실행을 해주어야 하는데 이 기능을 시뮬레이션 엔진에서 담당한다.

시뮬레이션엔진은 가상머신을 실행하게 되면 가상머신에서 이벤트 발생시점 전까지 실행 후 이벤트 생성하여 시뮬레이션 엔진에 알려준다. 시뮬레이션 엔진은 발생한 이벤트에 대한 처리를 수행한 후 각 가상머신이 발생한 이벤트 정보를 종합하여 어떠한 가상머신을 실행시킬 것인지 결정하여 시뮬레이션을 진행시킨다.

그림2는 시뮬레이션 엔진이 각 가상머신이 실행하는 프로그램을 이벤트에 의해 순차적으로 처리하는 것을 보여주는 예이다. 처음에는 각 가상머신이 발생시킨 이벤트 정보가 없으므로 각각의 가상머신(프로그램)을 이벤트가 발생하기 전까지 순차적으로 실행한다. 그 후에 각각의 가상머신이 발생시킨 이벤트(B2, A2, C3)를 비교하여 시

뮬레이션 상의 시간으로 가장 먼저 발생한 이벤트(B2)에 해당되는 가상머신을 실행시키게 된다. 해당 가상머신이 이벤트 발생 전까지 실행한 후 다시 이벤트 정보를 받아 시뮬레이션 시간상으로 가장 먼저 발생한 이벤트에 해당하는 가상 머신을 실행하면서 시뮬레이션이 진행 되게 된다.

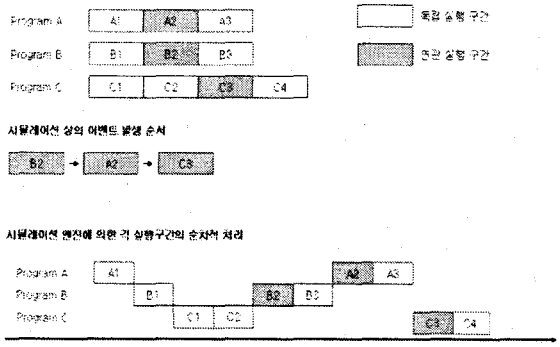

그림 2 이벤트에 의한 순차적인 처리

이러한 방법으로 시뮬레이션을 진행해 나가면서 각각의 가상머신의 시뮬레이션 상의 시간을 정해진 사이클 타임이 지날 때마다 같아지게 함으로서 동기화 문제를 해결하게 된다.

이러한 순차적인 처리방법의 로봇 시뮬레이션 시스템은 시뮬레이션의 대상이 되는 로봇의 개수가 늘어남에 따라 가상머신의 개수도 같이 증가하여 전체 시뮬레이션 시간이 이와 비례하게 증가하게 된다. 시뮬레이션 시스템이 분산되어 각 가상머신을 병렬로 처리하게 되면 로봇의 개수가 크게 증가하여도 전체 시뮬레이션 시간에 미치는 영향은 크지 않게 된다. 그러나 각 가상머신이 분산화 되어 병렬로 처리될 경우 다른 로봇에 영향을 주는 명령어에 대한 동기화 문제가 반드시 해결되어야 한다.

3. 로봇 시뮬레이션 시스템의 분산화

이벤트/사이클 방식의 로봇 시뮬레이션 시스템은 다른 로봇에 영향을 미치는 명령어에 대한 동기화 문제를 시뮬레이션 상의 시간에서 발생한 순서대로 처리함으로서 해결하였다. 그러나 이러한 순차적인 처리 방식의 로봇 시뮬레이션 시스템은 로봇의 개수가 늘어날수록 시뮬레이션 시간도 증가하게 되어 규모가 큰 여러 개의 작업장을 시뮬레이션하기에는 문제가 있다. 이번 장에서는 기존의 로봇 시뮬레이션 시스템의 문제를 해결하기 위해 분산된 로봇 시뮬레이션 시스템의 구조를 제안한다. 그리고 분산된 로봇 시뮬레이션 구조로 인해 발생하는 동기화 문제에 대한 해결 방안으로 기존의 이벤트/사이클 혼합 방식의 로봇 시뮬레이션 시스템의 동기화 방법을 분산된 시뮬레이션 시스템에 적용할 수 있도록 수정하였다.

3.1 분산된 로봇 시뮬레이션 시스템의 구조

분산된 로봇 시뮬레이션 시스템의 구조는 전체적으로 서버 클라이언트의 구조를 가진다. 서버와 클라이언트는 이벤트 정보를 주고받으며 시뮬레이션을 수행하게 된다. 전체 시스템은 그림3과 같이 서버와 메시지 채널, 여러 개의 클라이언트로 구성되며 각각의 역할은 다음과 같다.

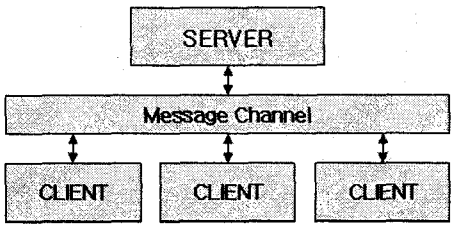

그림 3. 시스템의 전체 구조

메시지 채널은 서버와 클라이언트 사이에 존재하면서 클라이언트와 서버와의 연결, 메시지 패싱 등의 기능을 담당하는 미들웨어의 역할을 한다. 서버는 전체 시뮬레이션을 조율하며 각 클라이언트간의 동기화, 가상머신간에 공유되는 전역변수의 저장, 클라이언트가 발생하는 이벤트에 대한 처리를 담당한다. 클라이언트는 로봇 단위의 시뮬레이션을 수행하며 시뮬레이션하며 발생하는 이벤트를 서버로 전송하고 서버로부터 전달된 이벤트에 대한 처리를 한다. 시뮬레이션 설정 시에 각 클라이언트에서 실행할 로봇을 설정하며 클라이언트에서 처리하는 로봇은 여러 개가 될 수가 있다. [5][6]

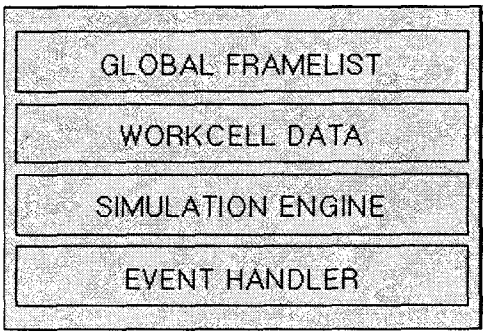

그림 4 서버 구조

그림 4은 서버의 구조를 나타내는 것으로 각각의 구성요소의 역할은 다음과 같다

WORKCELL DATA는 워크셀 단위로 시뮬레이션에 필요한 정보를 저장하는 데이터 구조이다. 워크셀은 로봇을 묶는 논리적인 요소로 서로 연관되어 있는 로봇을 그룹화 시킨다. WORKCELL DATA에 저장되는 정보는 워크셀에 포함되는 로봇의 정보, 로봇 간의 시그널 정보 등이다. 시그널은 로봇에서 포트를 통해 신호를 주고 받는

것을 말한다.

GLOBAL FRAMELIST는 가상머신 간에 공유되는 전역변수의 값을 저장하는 공간이다. 모든 클라이언트의 가상머신은 서버의 GLOBAL FRAMELIST를 통해 전역변수의 값을 얻어오거나 변경할 수 있다.

Simulation Engine(SE)는 전체 시뮬레이션에 포함된 가상머신 간의 동기화 기능을 제공한다. SE는 기본적으로 가상머신 간의 동기화를 위해 사이클 방식을 사용한다. 또한 전역변수나 시그널 같은 다른 가상머신에 영향을 미치는 명령어에 의한 동기화 문제도 처리한다.

이벤트 핸들러(EVENT HANDLER)는 클라이언트가 서버로 전송하는 이벤트를 처리하는 모듈이다. 동기화에 관련된 이벤트는 이벤트 핸들러가 직접 처리하지 않고 SE에게 전달하여 처리한다.

그림5 는 클라이언트의 구조를 나타낸 것이다. 가상머신(Virtual Machine)은 기존의 로봇 시뮬레이션 시스과 마찬가지로 로봇 시뮬레이션 언어로 작성된 프로그램을 실행하는 역할을 한다. 가상머신은 시뮬레이션 설정 시에 클라이언트에 포함된 디바이스의 개수만큼 생성된다. 이벤트 핸들러는 서버로부터 전달 받은 이벤트를 처리하거나 가상 머신이 시뮬레이션 언어로 작성된 프로그램을 실행하면서 발생한 이벤트를 처리하는 기능을 한다. OLP Robotics Module은 앞서 언급한바와 같이 디바이스의 동작에 관한 위치 계산이나 시간계산을 수행하는 모듈이며 화면에 디바이스의 동작 모습을 출력하는 CAD 시스템에 필요한 정보를 제공하는 기능을 한다.

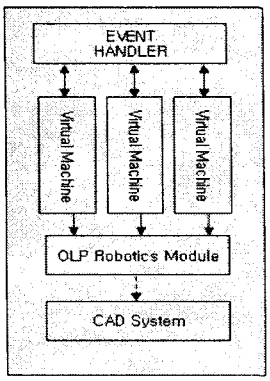

그림 5. 클라이언트 구조

기존의 로봇 시뮬레이션 시스템을 가상머신과 CAD 시스템을 여러 개의 클라이언트를 연결하여 분산시킨 구조에 대해 살펴보았다. 분산된 로봇 시뮬레이션 시스템에서는 가상머신의 처리와 시뮬레이션 진행상황을 화면에 출력하는 CAD 시스템이 여러 클라이언트로 분산되기 때문에 이에 집중되는 시뮬레이션의 대상이 되는 로봇의 개수가 늘어나게 되어도 전체 시뮬레이션 시스템에 미치는 영향은 크지 않게 된다. 이처럼 가상머신을 분산시켜 병렬로 처리할 경우 각 가상머신간의 동기화 문제가 중요시 된다. 이 문제에 대해서는 다음 장에서 자세히 살펴보도록 한다.

3.2 가상머신의 동기화

분산된 로봇 시뮬레이션 시스템에서는 가상머신이 동시에 실행되게 된다. 따라서 가상머신 간에 다른 로봇에 영향을 미치는 명령어에 대한 동기화 문제, 즉 전역변수의 사용이나 로봇 간의 시그널로 인한 발생하는 동기화 문제를 어떻게 해결하는 것이 중요한 문제가 되겠다.

본 논문에서는 이러한 분산된 구조에서 발생하는 동기화 문제를 해결하기 위해 전역변수나 시그널 값에 대한 접근 등, 전체 시뮬레이션에 영향을 주는 요소를 이벤트로 정의하고 이를 서버에서 처리하는 방법을 제안한다.

가상머신의 동기화는 기본적으로 기존의 이벤트/사이클 혼합 방식의 로봇 시뮬레이션 과 마찬가지로 사이클 타임을 통해 이루어지게 된다. 사이클 타임은 실행 시간이 긴 구간(로봇의 동작 명령어 처리 구간)을 정해진 짧은 시간단위로 나누어놓은 것으로 서버의 SE는 각각의 가상 머신을 사이클 타임 단위로 동기화 하며 모든 동기화가 이루어졌을 때 CAD시스템을 업데이트 시킨다.

서버의 SE에서 수행하는 사이클 타임을 이용한 기본적인 가상머신의 동기화 작업은 다음과 같다.

① SE가 모든 가상머신에 사이클 타임만큼 실행할 것을 명령

② 각각의 가상머신은 정해진 사이클 타임까지 실행이 끝나게 되면 서버로 사이클 타임 이벤트를 전송한 후 대기한다.

③ SE는 각 클라이언트들로 받은 사이클 타임 이벤트를 확인하여 모든 가상머신이 대기 상태이면 다음 사이클 타임만큼 실행할 것을 명령한다.

그러나 전역변수나 로봇의 시그널처리 같은 다른 로봇에 영향을 미치는 명령어로 인해 사이클 타임을 이용한 동기화만으로는 정확한 시뮬레이션이 이루어지기 어렵다. 서버에 저장된 전역변수나 시그널 값은 가상머신 간에 공유되어 if문과 같은 흐름제어문의 조건식으로 주로 사용되기 때문에 전역변수나 시그널의 값을 지정하는 시간과 사용하는 시간의 정확한 동기화가 이루어지지 않는다면 잘못된 전역변수의 값을 가져와 의도되지 않은 방향으로 시뮬레이션이 진행되는 경우가 발생하게 된다.

전역변수나 시그널의 경우 서버에 저장되는 값을 분산된 여러 가상머신들이 공유하는 형태이므로 서버에 저장된 값을 가져오고 읽어오는 시점의 순서와 시뮬레이션 상의 시간 순서와 일치하여야 한다. [7]

이러한 전역변수와 시그널로 인한 동기화 문제의 해결 방안으로 기존의 이벤트/사이클 혼합 방식의 로봇 시뮬레이션 시스템의 시뮬레이션 알고리즘을 적용하였다. 기존의 이벤트/사이클 혼합 방식의 로봇 시뮬레이션에서는 여러 가상머신이 생성하는 이벤트의 시뮬레이션 상의 발생 시간의 순서에 따라 순차적으로 가상머신을 실행하여 처리 하였으나 분산된 시뮬레이션 시스템에서는 이벤트를 발생시키고 가상머신은 대기한 후 서버는 발생한 이벤트에 대한 처리만 하고 그 다음 이벤트까지의 실행 구간은 분산된 가상머신이 동시에 처리할 수 있도록 하였다.

SE가 수행하는 전역변수나 시그널에 의한 동기화 문제

의 처리방법은 다음과 같다.

① 가상머신에 전역변수나 시그널의 값을 변경하거나 서버로부터 가져오는 명령어를 실행하게 되면 우선 가상머신에서의 로봇 프로그램의 실행을 멈추고 서버로 전역변수의 값이나 시그널 값의 읽기나 쓰기에 대한 이벤트를 전송한다. 이때 서버로 전송하는 이벤트에는 전역변수의 읽기나 쓰기를 수행하는 시뮬레이션상의 시간이 포함되어야 한다.

② 모든 가상 머신이 대기 상태이거나 사이클 타임까지의 실행을 끝내게 되면 SE는 전달 받은 이벤트 정보를 시뮬레이션상의 시간 순으로 가장 빠른 이벤트를 처리하고 해당 가상 머신은 다음 대기 상태까지 실행하게 된다.

③ 다시 가상머신이 모두 대기상태가 되면 서버는 저장된 이벤트 정보를 통해 가장 빠른 이벤트를 실행시키고 모든 가상머신이 사이클 타임까지 실행하게 되면 이벤트 정보를 초기화 시키고 모든 가상머신을 다시 동시에 실행시킨다.

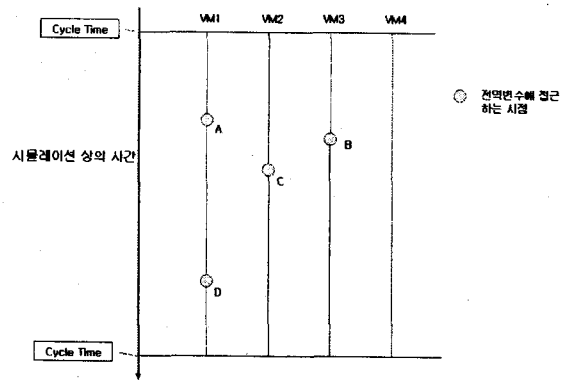

그림 6 전역변수에 의한 동기화의 예

그림6은 전역변수나 시그널에 의한 동기화 처리방법을 설명하기 위한 그림으로 4개의 가상머신(VM)이 한 사이클 타임 동안 전역변수에 접근하는 시점을 표현한 것이다. 앞서 설명한 방법에 따라 VM1, VM2, VM3는 프로그램을 실행하다가 전역변수에 접근하는 시점인 A, B, C에서 서버로 이벤트를 전송한 후 대기하고 VM4는 사이클 타임까지 실행을 완료하게 된다. SE 는 모든 가상머신이 대기 상태가 되었기 때문에 가상머신으로 받은 이벤트 중 시뮬레이션 상의 시간 순서에 따라 가장 빠른 이벤트인 A를 처리하게 된다. VM1은 대기상태가 해제되어 다음 전역변수의 접근 시점인 D까지 실행을 완료하게 된다. 그러면 다시 모든 가상머신이 대기 상태가 되므로 저장된 이벤트 중 가장 빠른 B를 대기상태에서 해제하여 실행하게 되고 다시 대시 상태가 되면 C->D순으로 대기 상태를 해제 하여 실행하도록 한다.

위와 같이 전역변수나 시그널로 인한 동기화를 위해 가상머신을 대기시키는 방법은 구현된 로봇 프로그램에 따라 가상머신의 대기시간이 달라지기 때문에 정확한 성능의 향상 정도를 평가하기가 어렵다. 그러나 기존의 순차적인 이벤트/사이클 혼합방식의 로봇 시뮬레이션 시스템에서는 가상머신의 개수와 비례하여 전체 시뮬레이션 시간이 증가 하였으나 분산된 시스템에서는 가상 머신의 개수와는 무관하게 전역변수의 접근 빈도에 따라 전체 성능이 결정되게 된다. 일반적으로 한 사이클 타임 내에서 여러 가상머신이 전역변수에 접근하는 경우가 많지 않고 대부분의 사이클이 로봇을 동작 시키는 동작명령어로 이루어지기 때문에 성능향상을 기대할 수 있다.[8]

4. 결론 및 향후 연구

본 논문에서는 기존의 한 컴퓨터에서 응용프로그램으로 실행되는 OLP 기반의 로봇 시뮬레이션 시스템의 문제를 해결하기 위해 로봇 단위로 자유롭게 분산시켜 처리할 수 있는 구조를 제안하였다. 분산된 구조에서는 시뮬레이션 언어로 작성된 프로그램을 실행하며 로봇을 제어하는 가상머신이 병렬로 동작하기 때문에 각 가상머신 간의 공유되는 전역변수에 의한 동기화 문제를 기존의 이벤트/사이클 혼합방식의 시뮬레이션 알고리즘을 분산 시스템에 적용하여 해결하였다.

본 논문에서 제안된 분산 로봇 시뮬레이션 시스템은 아직 구현 중에 있다. 앞으로는 성능 분석을 통하여 성능의 향상 정도와 구조의 효율성을 검증하는 연구가 필요하며 동기화에 소모되는 비용을 최소화할 수 있는 방안이 추가로 연구되어야 하겠다. 그리고 사용자의 편의성을 위해 분산된 시뮬레이션 구조에 알맞도록 시뮬레이션 시스템의 UI를 개선하는 방안도 같이 연구되어야 할 것이다.

5. 참고 문헌

[1]Eberhard Roos, Arno Behrens, "Off-line programming of industrial robots - Adaptation of simulated user programs to the real environment", Computers in Industry, Vol.33, pp.139-150, 1997

[2]Maynard A. Holliday, "Simulation and Off-line Programming Systems Comparison", European Simulation Multiconference 1994

[3]한정욱, OLP 기반 이벤트/사이클 혼합 방식 산업용 로봇 시뮬레이션 엔진 구현, 한국컴퓨터종합학술대회 2006

[4]박찬진, "이벤트/사이클 혼합 방식을 통한 분산시뮬레이션의 성능향상", 부산대학교 대학원 컴퓨터공학과 석사학위논문, 2006

[5]Poolsak Koseeyaporn, " Windows-based Robot Simulation Tools", Seventh International Conference on Control, Antomation, Robotics And Vision , 2002

[6]Pools Koseeyaporn, " Integratable Robot Simulation Tools ", IEEE Southeastcon 2002

[7]DANIEL A. REED, ALLEN D. MALONY, "Parallel Discrete Event Simulation Using Shared Memory", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. VOL. 14, NO. 4. APRIL 1988

[8]JinshengXu,"Predicting the performance of synchronous discrete event simulation systems", ICCAD2001