실시간 분산 객체에 기반한 시뮬레이션 모델의 설계 및 구현+

오영배* · 김강호* · 정연대*

Design and Implmentation of a Simulation Model Based on Real-Time Distributed Object

Young-Bae Oh · Kang-Ho Kim · Yeon-Dae Chung

〈요 약〉

실시간 분산 시뮬레이션 응용을 개발할 때 시뮬레이션 대상의 시간적 행동 및 분산 노드간 상호작용의 복잡성 때문에 모델 개발에 모델 개발에 어려움이 있다. 그러나 실시간 분산 객체(RTO)를 기반으로 하여 시뮬레이션 모델을 설계할 때 모델의 시간적 행동의 표현이 자연스러워지고 설계의 명확성을 가져다 준다.

본 연구에서는 RTO 모델을 이용하여 압연공정 제어시스템 실시간 시뮬레이션 모델을 설계 구현 하였다. 구현 과정에서 RTO 접근방법이 설계의 자연스러움, 설계 명세의 단순 명확화, 시간적 행동 표현의 복잡성 제거, 객체의 노드 분산 용이성등 많은 장점을 가지고 있음을 확인 하였다. 이 방법은 보다 복잡하고 규모가 큰 실시간 분산 시뮬레이션에 효과적으로 적용할 수 있다.

+ 본 논문은 과학기술처 1996년도 핵심 소프트웨어 기술개발 사업의 연구비에 의해 연구되었음.

* 시스템공학연구소 소프트웨어공학연구부

1. 서 론

실시간 시뮬레이션은 실시간 시스템을 구성하는 프로세서들의 기능적 행위(Functional Behaviors)와 시간적 행위(Temporal Behaviors)를 정밀하게 모형화 하고 이를 고성능의 컴퓨터를 이용하여 실시간으로 실행하여 봄으로써 실시간 시스템의 구성 요소간의 유기적인 관련성과 영향을 분석 평가하는 기술이다.

일반적인 실시간 시스템은 수십, 또는 수백개의 프로세서의 집합으로 구성되어 있다. 기존의 시뮬레이션은 단일 프로세서 상에서 시뮬레이션을 실행하고 있기 때문에 실시간성의 검증이 약화되고 시뮬레이터의 성능도 저하되는 경향이 있다. 최근에는 분산 및 병렬처리 컴퓨팅이 실용화 되어 분산 병렬 컴퓨터 환경을 이용한 고성능의 실시간 시뮬레이션이 가능하고, 그 활용이 중요시되고 있다. 실시간 시뮬레이션 기술은 이러한 사실을 근거로 개발 되고 있다. 이러한 고성능, 고정밀의 실시간 시뮬레이션을 통하여 신뢰성 있는 실시간 시스템의 개발이 이루어지게 되고, 대규모의 복잡한 실시간 시스템의 최적 설계와 효율적인 운영에 긴요한 의사결정 정보가 제공되므로 실시간 시뮬레이터 기술 개발의 필요성이 부각되고 있다.

시뮬레이션 모델링은 컴퓨터 시스템에서 쓰이는 용어로써 시스템을 표현할 수 있다고 가정한다. 이점에 관해서 중요한 것은 시스템 상태 기술의 개념이다. 시스템이 변수의 집합 즉 시스템의 고유한 상태 또는 조건을 나타내는 변수값의 조합으로 특성화 될 수 있다면 변수값의 조작은 시스템의 움직임을 하나의 상태에서 다른 상태로 시뮬레이션 한다. 시뮬레이션 실험은 모델내에 설계된 잘 정의된 운영 규칙을 따라서 상태에서 상태로 움직이는 모델의 동적인 행동을 관찰하는 것과 관련된다[9]. 시스템 상태내의 변화는 시간에 따라 연속적으로 발생할 수도 있고 이산적인 시간 순간에 발생할 수도 있다. 이산적인 순간들은 모델 입력의 특성에 따라 결정적으로 성립될 수도 있고 추계적(stochastic)으로 성립될 수도 있다. 이산적 변화 모델과 연속적 변화 모델의 동적인 행동을 기술하는 절차는 달라도 시간에 따라 변하는 시스템의 상태를 표현하여 시스템을 시뮬레이션하는 기본적 개념은 같다.

최근의 컴퓨터 기술은 각각의 컴퓨터 또는 프로세서들을 서로 연결하여 병렬 또는 분산 컴퓨팅 환경을 가능하게 하였다. 예를 들면 비교적 값이 싼 여러 개의 미니 컴퓨터 또는 마이크로 컴퓨터는 네트워크로 연결 될 수 있고 더 큰 컴퓨터는 다수의 프로세서들을 묶어서 자신의 일을 할 뿐만 아니라 서로 간의 통신도 가능하게 한다. 이러한 환경에서는 컴퓨팅 과업의 각기 다른 부분들을 각각의 프로세서에 분산하여 동시에 또는 병렬로 실행하여 과업을 완료하여 전체 수행시간을 줄일 수 있다. 이러한 능력은 컴퓨팅 과업의 특성과 가용한 하드웨어와 소프트웨어에 좌우된다.

대부분의 실시간 시스템은 그 규모가 크고 복잡하기 때문에 사용자의 요구 사항을 정확하게 규명하기가 어렵고, 개발하면서 당면하는 의사결정 사항들에 대해서 결정을 내리기가 매우 어렵다. 특히 시간적 행위에 대한 예측과 증명을 할 필요가 많은데 이를 해결하는 방법으로 실시간 시뮬레이션 기법이 필요하다.

실시간 시뮬레이션을 수행할 때 다음의 두 가지 필수 조건을 만족시켜야 한다.

(1) 실제 응용 환경에서 발생하는 여러 사건들의 상대적 시간성이 정확하게 시뮬레이터에 의하여 모방되어야 한다. 따라서 사건들의 순서가 실제 환경에서나 시뮬레이터 상에서 같아야 함은 물론이고 실제 환경에서 일어난 두 사건 간의 시간 간격과 상응하는 시뮬레이터 상에서의 시간 간격과의 비율이 어느 두 사건을 고려하더라도 항상 일정하거나 미세한 변동폭을 보

여야 한다.

(2) 시뮬레이션 속도는 실제 응용 시스템이나 환경이 진행하는 속도만큼 빨라야 한다. 따라서 이러한 실시간 시뮬레이션은 응용 환경에 존재하는 여러 구성 요소간의 상호 작용의 시간적인 면을 정확하게 보여줄 수 있는 것이다.

Zeigler는 실시간 시뮬레이션을 다음과 같이 정의하였다[5]. "실시간 시뮬레이션은 시뮬레이션의 시간적 기초가 컴퓨터 시스템의 시계(Clock Time)와 매우 밀접히 연관되는 시뮬레이션이다." 이상적인 시스템에서는 모델에 의해 계획되는 모든 사건들이 실제 시간에 계획된 시간과 정확하게 일치되는 시간에 발생한다. 그러나 한정된 처리 시간 때문에 실제 사건 시각은 모델에 규정된 값과 달라질 수 있다. 따라서 경성 마감시간(Hard Deadline)을 가진 응용에서는 충분히 빠른 컴퓨팅 플랫폼(platform)이 필요하고, 계획된 시간의 허용된 한계(tolerance) 내에서 사건이 발생하도록 하여야 한다.

이 분야의 기본 기술 확보가 매우 미흡한 국내 현실을 극복하기 위하여 시스템공학연구소와 포항공대는 미국 UCI와 공동으로 과학기술처와 (주)포스콘의 지원을 받아 실시간 시뮬레이션 기술을 개발하고 있다. 본 연구과제에서는 거시적 및 미시적 시뮬레이터 와 O/S 지원 기능을 개발하고 있다. 본 논문에서는 미시적 시뮬레이터 개발의 중간 연구 결과를 노하고자 하며, 본 연구에서 개발한 시뮬레이션 모델링의 접근방법을 소개하고 압연공정 시뮬레이터의 설계 및 구현을 통해 이 방법의 적용을 논한다.

2. 접근 방법

2.1 시뮬레이션 모델링

본 연구의 공동연구 파트너인 Kane Kim은 전통적인 객체 지향 모델을 확장하여 실시간 특성을 보다 효과적으로 표현할 수 있는 모델을 제안하였는데 이를 RTO.k 모델로 명명하였다[3]. 이 모델은 전통적인 객체 지향 모델[2][8]과 구분되는 다음과 같은 특징이 있다.

(1) SpM(spontaneous method)으로 불리는 시간 구동 객체 메소드와 SvM(service method)으로 불리는 메시지 구동 메소드 간의 명확한 분리 : 실시간 객체(Real-time Object, RTO)의 일부 메소드(method)에 대하여 실시간 시계(Real-time clock)는 시계가 설계시에 규정된 어떤 값에 도달할 때 메소드 실행을 구동하는 장치가 되며, 이러한 메소드를 자발적 메소드(SpM) 라고 하며 클라이언트(client)로부터 메시지(message)에 의해 구동되는 고전적인 서비스 메소드(SvM)와 명확히 구분된다. 실시간 시계가 설계 시에 결정될 수 있는 어떤 값에 도달할 때 취해지는 행동은 SpM에만 나타난다.

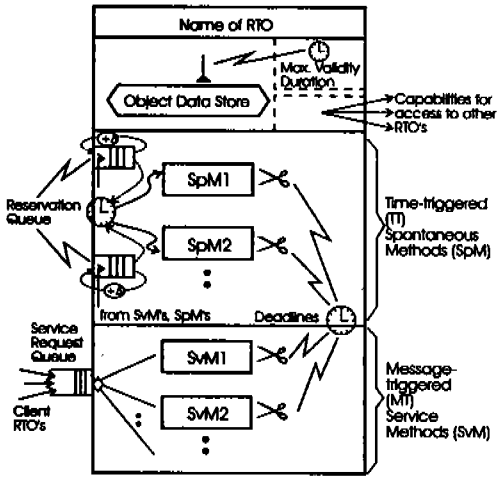
(2) SvM에 대하여 SpM 에게 항상 우선 순위를 부여하는 기본 동시성 제한 조건(Basic Concurrency Constraint): 기본적으로 외부 클라이언트로부터 발생되는 메시지에 의해 구동 되는 서비스 메소드는 충돌 가능한 시간 구동 메소드의 수행이 일어나지 않을 때에만 활동할 수 있다. 정확하게 표현하면 SvM이 SpM과 객체 데이터 공간(Object Data Space, ODS)의 동일한 부분을 접근하는데 충돌이 배제되지 않는다면 SvM은 SpM의 시간 구동에 대해 규정된 시간 영역 내에서는 허용되지 않는다. 이러한 제한 조건을 기본 동시성 제한 조건이라고 부른다. 따라서 SpM은 SvM에 비해 더 높은 우선 순위가 부여된다. 이러한 동시성 제한 조건은 SpM간의 동시적 수행이나 SvM간의 동시적 수행에는 어떠한 제한 조건도 부과되지 않는다는 것에 주의해야 한다.

(3) 실시간 객체의 각각의 메소드 수행에 대하여 마감시간(deadline)이 부과된다.

(4) 실시간 객체 내에 포함된 실시간 데이터는 최대 유효 기간(Maximum Validity Duration)이라 불리는 시간 간격 이후에는 유효성을 상실한다.

<그림 1>에 객체 데이터 공간(Object Data Space, ODS)상에서 작동되는 두 개의 구별되는 유형의 메소드 즉, SpM과 SvM이 표시되어 있다. SvM으로부터 SpM을 명확하게 분리한 것이 RTO.k 객체 모델의 중요한 특징이다.

객체 모델은 원래 시뮬레이션 응용에 적용하기 위하여 사용되었다. 그러므로 환경 객체를 모델링 하는데 있어서 객체 모델, 즉 시간이 변하는 내부 상태를 가진 환경 내의 모듈 개체를 사용하는 것은 오랫동안 활용되었다. RTO.k 객체 모델은 실시간 제어 시스템 설계의 가변적인 추상화뿐만 아니라 응용 환경을 정확하게 표현하는데 효과적인 표현 구조를 가지고 있다. 따라서 이러한 모델링 구조를 가지고 있는 RTO.k 모델을 본 연구의 시뮬레이션 모델로서 적용하였다.



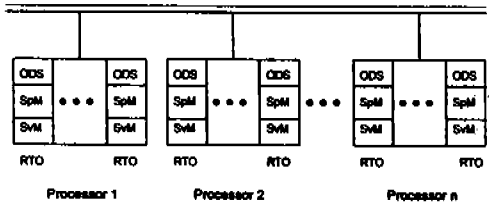
〈그림 1〉 RTO.k 모델[3]

2.2 실시간 분산 시뮬레이션 모델

동적 시뮬레이션을 분할하여 여러 작업을 다수의 프로세서들에 분산하는 방법은 여러 가지가 있다. 가장 직접적인 접근 방법은 각각의 지원 기능들을 서로 다른 프로세서들에 할당하는 것이다. 시뮬레이션의 논리적인 실행은 아직까지 순차적이며 다만 시뮬레이션 주프로그램은 지원 기능들을 다른 프로세서들에게 보내고 작업을 같이 한다. 분리된 프로세서들에 시뮬레이션을 분산하는 또 다른 방법은 모델 그 자체를 분리하여 여러 개의 하위모델(submodel)로 나누고 이것들을 다른 프로세서에게 할당하여 수행시키는 것이다. 각각의 하위모델들은 다른 프로세서들에 할당되고 각 프로세서는 모델의 한 부분을 시뮬레이션한다. 프로세서들은 하위모델 간의 적절한 논리적 관계를 유지하기 위하여 필요할 때 마다 서로 통신해야 한다. 활동의 시간 순서를 올바르게 유지하기 위해서 주의를 기울여야 하는데 즉, 모델의 전체 활동을 올바르게 나타내기 위해서는 다른 프로세서상의 하위모델의 운영을 동기화(synchronize)하여야 한다. 이런 유형의 분산 시뮬레이션의 중요한 장점은 통합 시뮬레이션 시계(Global Simulation Clock)와 완전한 사건 리스트(Event List)가 필요치 않다는 것이다. 시계와 사건 리스트를 대신하는 것은 프로세서간에 메시지를 전달하는 시스템이며 여기에서 각 메시지는 시간 스탬프(Time Stamp)를 전달한다[10].

시뮬레이션 RTO는 각각 자신의 시계를 가지고 각자의 시뮬레이션 주기로 수행하며 RTO 간에는 메시지 전달 시스템에 의해 동기화를 이루어 전체 시뮬레이션 모델의 일관적인 운영을 기하게 된다. 시뮬레이션 RTO를 하드웨어 자원에 구동 시킬 때 네트워크상의 분산을 고려하여 세가지 모델을 구성할 수 있다. 첫번째 모델은 다수의 RTO를 각각의 하드웨어 프로세서에 할당하는 것이다. <그림 2>는 이 모델을 보여주고 있으며 각 노드(node)의 프로세서와 메모리를 다중 RTO의 구동을 책임진다. 이 모델은 가장 보편적으로 사용된다. 두 번째 모델

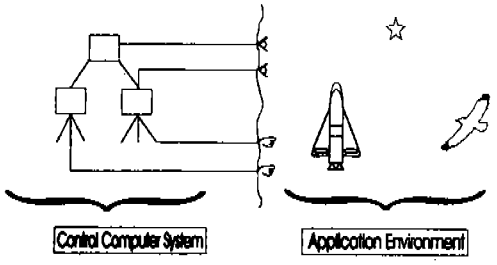
은 하나의 RTO가 하나의 프로세서에 대응되는 모델이다. 하나의 RTO가 전용 프로세서를 가지므로 첫 번째 모델에 비해 하드웨어 자원을 더 많이 갖게 되고 따라서 보다 정밀한 시뮬레이션을 가능케 한다. 그러나 노드 분산에 따른 네트워크 트래픽(traffic)이 증가하게 된다. 세 번째 모델은 RTO내의 각각의 메소드가 하나의 프로세서에 대응되는 모델이다. 이 모델에서는 하나의 메소드가 자신의 프로세서를 가지게 되므로 하드웨어 자원을 최고로 이용할 수 있다. 따라서 이 모델은 매우 복잡한 시뮬레이션 또는 매우 짧은 시뮬레이션 주기(Simulation Time)를 가지는 응용에 적합하다. 이 모델은 각각의 SpM 및 SvM들이 네트워크상의 노드에 넓게 분포되어 있으므로 통신 트래픽이 가장 많이 일어나게 된다.



〈그림 2〉 RTO 분산 시스템 모델

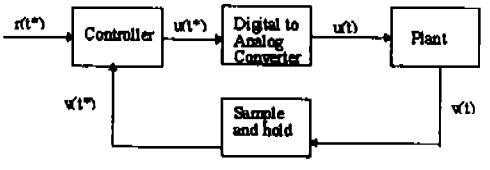
2.3 제어기와 시뮬레이션

컴퓨터 제어 시스템의 전형적인 예를 <그림 3>이 보여주고 있다. 여기에서 컴퓨터 제어 시스템은 실시스템과 상호작용을 하게 되는데 센서(sensor)는 시스템의 프로세스를 감시하고 프로세스 정보를 제어 컴퓨터에 보내준다. 이 정보는 제어 컴퓨터의 제어 논리(logic)에 따라 액츄에이터(actuator)를 작동하여 설계자가 원하는 방향으로 프로세스를 운영한다. 이런 실세계의 장비들을 제어하기 위해 컴퓨터는 일정 간격으로 측정 장비에서 데이타를 샘플링(sampling)할 필요가 있다. 따라서 실시간 시계가 필요하다. 또한 장비들을 정확한 시간과 순서로 가동시키기 위하여 액츄에이터의 실시간 구동이 필요하다[1].



〈그림 3〉 제어시스템 모델

많은 실시간 시스템은 여러 가지 공학적 활동의 제어에 사용된다. <그림 4>는 피드백(feedback) 제어기를 나타낸다. 이 제어기는 입력 변수를 어떻게 변화시켜야 원하는 출력벡터를 발생시킬 수 있는지를 계산하기 위한 수학적 모델을 가져야 한다. 이러한 모델을 찾는 것은 제어공학의 관심 분야이다. 많은 경우 플랜트 1차 미분 방정식으로 표현되어 플랜트의 내부 상태와 입력 변수와 시스템의 출력간의 관계를 모형화 한다. 플랜트의 출력을 변경하는 것은 이러한 방정식을 풀어서 요구하는 입력치를 주는 것이다.



〈그림 4〉 피드백 제어 시스템 모델

정해진 시간 내에 새로운 설정값으로 변경하여야 하는 실시간 시스템의 요구 조건은 수학적 모델 또는 실제 시스템이 필요로 하는 조작 내용을 훨씬 복잡하게 만든다. 이런 어려움 때문에 대부분의 제어기들은 컴퓨터로서 구현된다. 컴퓨터로 구현하게 되면 시스템에 디지털(digital) 요소를 도입하게 되고, 제어 값을 이산치로 나타내게 되어 제어 사이클의 특성이 변화

한다. <그림 4>에서 *로 표시한 것은 이산치를 나타내는데 센서가 플랜트 정보를 일정주기로 샘플링하고 액츄에이터가 이산치 데이터를 아날로그 신호로 변환하여 플랜트에 전송한다.

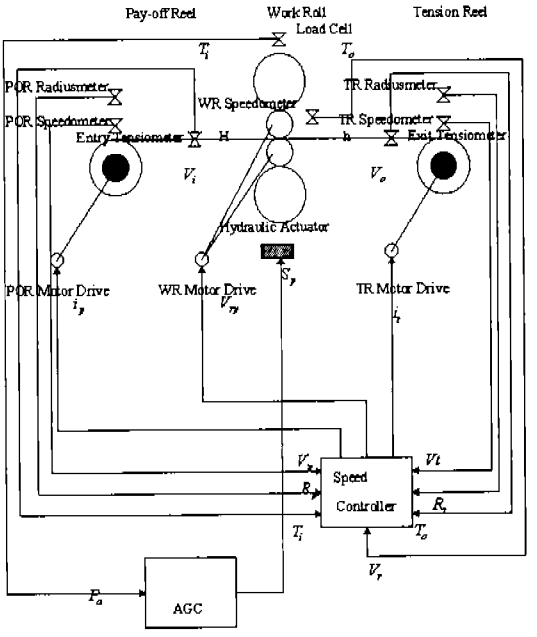
이러한 제어 시스템을 컴퓨터 시뮬레이션 하기 위하여서는 플랜트 역시 컴퓨터상에서 구현하고 따라서 플랜트도 제어기와 마찬가지로 수학적 모델을 만들고 이를 이산치로 구현하여야 한다. 플랜트를 시뮬레이션 하는 목적은 대부분의 실시간 시스템이 대규모이고 고가의 장비이므로 제어 시스템을 개발할 때 플랜트 시뮬레이터를 만들어 이를 시험해 보는 것이 개발 비용 및 기간에 있어서 큰 효과를 주기 때문이다. 시뮬레이션의 정밀도는 시뮬레이션 주기에 비례한다. 본 연구에서는 압연공정 및 이를 제어하는 AGC(Automatic Gauge Controller)의 수학적 모델을 바탕으로 압연공정 시뮬레이터 및 AGC 제어기 프로토타입(prototype)을 개발하였다.

3. 압연공정 시뮬레이션 모델 설계

3.1 압연공정 및 AGC의 개요

AGC(Automatic Gauge Controll)는 자동 판두께 제어 방법이며 BISRA 방법을 비롯한 여러가지 방법이 이용되고 있다[11]. 그러나 대부분의 AGC 시스템이 압연공정의 수학적 모델링에 기초한 제어 알고리즘을 이용하고 있기 때문에 모델링 오차의 존재 시에는 응답 특성이 크게 변화될 뿐만 아니라 전체 제어 시스템에도 영향을 미치게 된다.

연구에서 개발한 AGC 시뮬레이터는 단일 스탠드밀(Single Stand Mill)을 그 대상으로 하고 있다. 단일 스탠드 AGC 시스템은 크게 두 부분으로 나누어지는데 하나는 Roll Stand, Pay-Off Reel, Tension Reel 및 각종 액츄에이터로 이루어지는 압연공정이고 다른 하나는 압연공정을 효과적으로 동작시키기 위한 제어 시스템이다 [12]. <그림 5>는 단일 스탠드 밀을 위한 압연공정의 개요도이다.



〈그림 5〉 단일 스탠드 밀 압연공정 개요도

여기서 전체 시스템은 압연공정부와 AGC 시스템부로 나누어지는데, 압연 공정부는 압연현상이 발생하는 롤 탠드부, 소재의 풀림 및 감김이 일어나는 Pay-off reel부와 Tension reel부로 이루어지고, AGC부는 압연 스케줄 및 압연공정부의 궤환 신호를 이용하여 원하는 두께의 소재를 얻을 수 있도록 각종 제어 신호를 생성하는 제어시스템으로 구성된다. 압연공정부에 대한 입력신호에는 작업 롤 간격 기준신호(*SP*), 롤 속도 기준신호(V_{rp}), Pay-off reel 구동모터의 기준 전류신호(i_p) 및 Tension reel 구동모터의 기준 전류신호(i_t)등이 있으며, 출력신호에는 입출력 소재두께(H, h), 입출측 장력(T_i, T_o), 압하력(P) 및 롤 간격 (S)등이 있다. AGC부의 입력신호에는 원하는 소재의 출측두께, 입축장력 기준신호, 출측장력 기준신호 및 압연공정부로부터의 여러 가지 궤환 신호들이 있으며, 출력

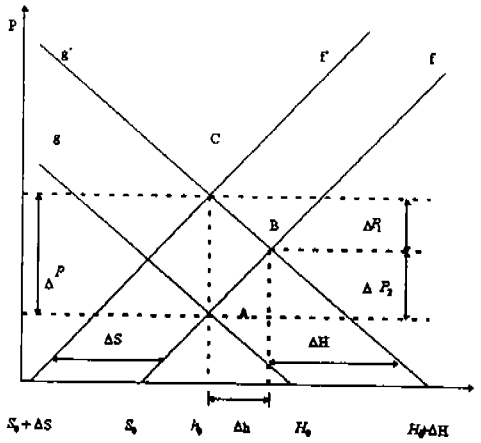
신호들은 압연공정부의 입력신호들과 동일하다.

3.1.1 AGC 시스템의 제어방법

가. 장력 제어

압연공정에서 압연하중은 장력에 의해 영향을 받는다. 장력은 각 reel과 스탠드 roll의 속도 차이와 소재두께에 의해서 변하게 되는데, reel의 속도는 소재가 감겨있는 정도에 따라 즉, reel의 반경에 따라 속도가 증가하거나 감소하게 된다. 장력은 압연하중에 영향을 미치고 있고 이로 인해 개루프 특성 및 압연공정에서의 제어에 의한 효과도 또한 줄어들게 된다. 따라서 장력을 일정하게 유지하는 작업이 필요하며, 이를 성취하기 위해 장력 제어기를 설치하여야 한다.

나. 압연원리



〈그림 6〉 압연 특성 곡선

압연 소재의 소성특성과 압연기의 탄성특성을 간략하게 나타내면 <그림 6>과 같다. 소재두께에 따른 특성곡성 중 f는 압연기의 탄성곡선이고 g는 소재의 소성곡성이다. f'은 f에서 롤 간격이 변화되었을 때 이동된 탄성곡선이고, g'은 g에서 입측 소재두께가 변화되었을 때 이동된 소성곡선이다. 압연점은 이들 곡선의 교점에서 결정되며 그 교점에서의 소재두께가 출측 두께로 결정된다. 따라서 어떤 요인에 의해 이들 곡선에 변화가 생긴다면 출측 두께는 변하게 된다. <그림 6>의 압연특성 그래프에서 볼 수 있듯이 입측 소재두께 변동이 있을 때 소성곡선 g는 g'로 이동된다. 이에 따라 압연점은 변동되어 A에서 B로 이동되고 압연하중과 출측 소재두께도 각각 △P, △h 만큼씩 이동된다. 따라서 원하는 두께의 소재를 얻을 수 없다. 이 출측 두께 변화 △h를 제거하기 위해 롤 간격을 △S 만큼 이동시켜 압연점을 B에서 C로 이동시켜 원하는 출측 소재두께 h_0를 얻는 것이 본 제어의 목적이다.

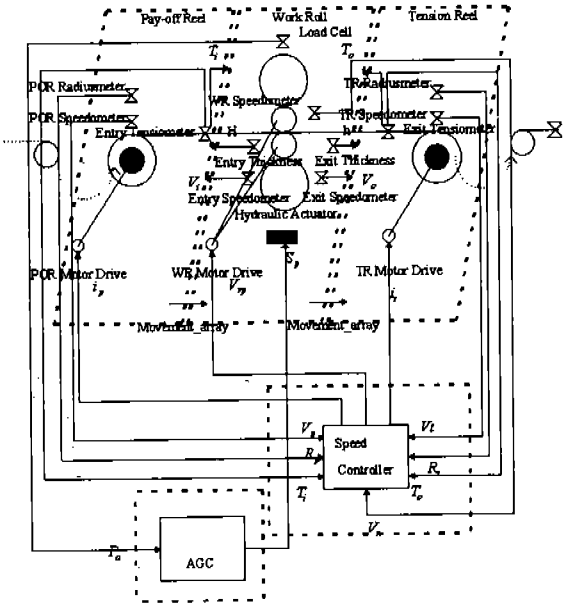
다. 두께 제어

단일 스탠드의 압연 제어는 압연기의 탄성특성과 소재의 소성특성에 기반을 두고 있는 압연원리를 이용하고 제어구조를 사용함으로써 여러 가지 방법이 사용되고 있다. 본 연구의 시뮬레이션 모델은 BISRA AGC 방법을 사용하고 있는데, 외란과 롤 간격 변화에 의한 압연하중 변화를 제어량에 이용하는 방법이다. 입측 소재두께등의 변동이나 작업 롤 간격 등이 변화하였을 때 압연하중이 변화되고, 압연원리에 따라 압연점이 이동하게 되는데, Load Cell로 측정되는 압연하중 신호를 궤환하고 이를 통해 적당한 제어입력을 생성함으로써 롤 간격을 동작시키는 원리로 되어 있다.

3.2 시뮬레이션 모델 설계

단일 스탠드 밀 압연공정은 시뮬레이션 환경인 압연공정과 AGC 제어기로 구분할 수 있고 다시 압연공정은 Pay-off reel, Work roll 및 Tension reel로 구분할 수 있으며 AGC 제어기는 속도 제어기와 AGC로 구분할 수 있다. 즉 전체 시스템은 5개의 객체로 분리할 수 있다. 이들 객체들은 모두 실시간 객체로서 각각을

RTO로 대응시킬수 있고 각각의 RTO는 각각 고유한 주기로 구동되는 SpM과 외부로 부터의 메시지에 의해 구동되는 SvM을 갖는다.



〈그림 7〉 압연공정 AGC 시뮬레이션 모델링의 개요도

<그림 7>에 압연공정 AGC 시뮬레이션 모델링의 개요도를 나타내었다. Pay-off reel 객체는 POR Radiusmeter, POR Speedometer 및 Entry Tensiometer등 3개의 센서로부터 POR 상태정보를 수집하며 이들을 주기적으로 Speed Controller 객체로 전송한다.

Speed Controller에서 계산한 전류신호는 POR Motor 가 받아서 적절한 속도를 생성하며 이를 통해 일정한 장력을 유지하게 된다. Work Roll 객체는 Load Cell, WR Speedometer, Entry Thickness Sensor, Exit Thickness Sensor, Entry Speedometer 및 Exit Speedometer를 통해 WR 상태정보를 수집하여 Speed Controller, AGC, POR 및 TR 객체등에 전송한다. Load Cell로 부터는 Work roll의 압하력을 읽어들이고 이를 AGC가 받아서 적절한 Roll Gap를 생성하여 Work Roll에 전송한다.

Tension Reel 객체는 TR Radiusmeter, TR Speedometer 및 Exit Tensiometer 등의 3개의 센서로부터 TR 상태정보를 수집하며 이들을 주기적으로 Speed Controller 객체로 전송한다.

Speed Controller에서 계산한 전류신호를 TR Motor가 받아서 적절한 속도를 생성하며 이를 통해 일정한 장력을 유지하게 된다. Speed Controller 객체는 POR, WR, TR 객체들의 센서로부터 메시지를 수집하여 각 객체의 소재장력을 일정하게 유지하기 위한 계산을 통해 전류값을 생성하며 이를 각각의 객체에 전송한다.

AGC 객체는 Work roll의 Load Cell로부터 압하력 정보를 수집하여 Target Thickness을 제공하기 위한 Hydraulic Actuator 이송량을 계산한다. 이 계산의 결과에 의해 Roll Gap 명령치를 Work Roll에 송신한다.

Rolling Process
Access Capability None
Object Data Store • Rolling Mill • Rolling Mill Controller
SpM Update the states of the physical objects in the rolling process • Update the states of Rolling Mill • Update the states of Rolling Mill Controller
SvM Load Material

〈그림 8〉 압연공정의 RTO

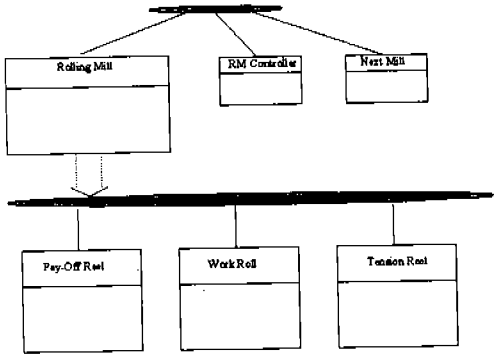
Rolling Mill
Access • AGC • Speed Controller
Object Data Store • Pay-Off Reel • Work Roll • Tension Reel
SpM • Update the states of Pay-Off Reel • Update the states of Work Roll • Update the states of Tension Reel
SvM • Receive data from AGC • Receive data from Speed Controller • Load Material

〈그림 9〉 Rolling Mill의 RTO

3.3 압연공정 RTO의 분할

RTO 모델링 방법은 우선 전체 시스템을 하나의 RTO로 정의하면서 출발한다. 대상공정을 거시적으로 보게 되면 Data Space는 Rolling Mill과 제어기로 구성되며 제어기는 AGC 및 Speed Controller 구성된다. SpM 은 Data Space 내의 Data가 자신의 상태를 주기적으로 update하는 것을 표현할 수 있고, SvM은 소재의 장입으로 기술된다. 기본적으로 이것이 가장 간단히 표현한 시스템의 개략 명세이다. <그림 8>에 Top level의 RTO 명세가 나타나 있다.

다음은 Top level RTO 의 Rolling Mill을 RTO 명세하면 <그림 9>와 같이 되고 Data Space에 POR, WR 및 TR이 나타난다. 이 RTO를 분할하면 각 Data는 자신의 RTO로 분리되고 이를 RTO 분할 네트워크로 나타내면 <그림 10>과 같이 된다.



〈그림 10〉 압연공정의 RTO 분할

이와 같이 하여 압연공정은 5개의 RTO로 분할되고 각각의 RTO는 RTO 명세 방법에 따라 상세 설계에 들어가게 된다.

3.4 RTO 명세

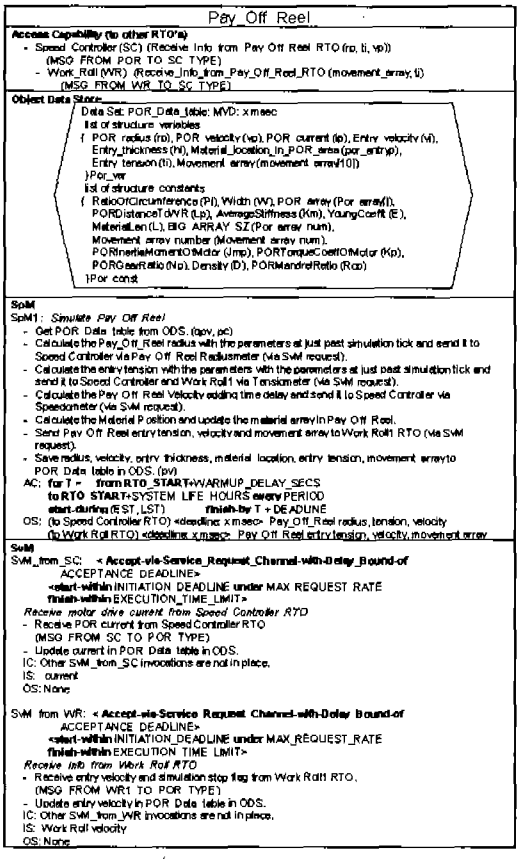
RTO를 분할하여 나오는 단말 노드의 RTO를 대상으로 설계 명세를 작성한다. RTO는 4개의 부분으로 구분된다. <그림 11>에 POR의 RTO 명세가 나타나 있다. 먼저 Access Capability는 RTO 자신이 호출하는 타 RTO들을 기술한다. 여기에는 RTO 자신이 호출하는 타 RTO 들을 나열하고 각 RTO로 전송하는 메시지 타입(type)을 기술한다. 이것은 프로세서 노드 내 및 프로세서 노드간에 분산되어 있는 RTO 간의 전송 메시지 채널(channel)을 선언함으로써 분산 네트워크상의 RTO간 메시지 전달을 가능케 하기 위한 것이다. 물론 분산 네트워크를 지원하는 통신 프로토콜(protocol)의 지원이 필요하다.

두 번째 부분은 Object Data Store 인데 여기에는 RTO가 다루는 데이터들을 기술한다. 이들 데이터는 RTO의 속성 상수(Attribute Constant)도 있고 SpM과 SvM이 수시로 update 하는 변수(variable)도 있는데 노드 내의 RTO들 및 RTO내의 메소드들이 한 노드에서 Concurrent 하게 구동되므로 이 공유 데이터 구조(Shared Data Structure)를 update할 때 이를 모니터(monitor)에 주는 장치가 필요하다. 본 연구에서는 Dream Kernel[4]의 CREW (Concurrent Read Exclusive Write) 모니터가 이 역할을 하고 있다. 또한 데이터 선언 시 MVD(Maximum Validity Duration)를 기술해 주게 되는데 이는 데이터의 최대 유효 시간으로서 데이터가 어떤 시점으로부터 의미를 가질 수 있는 시간을 의미하며 비실시간 시스템에서는 이것이 무한대이다.

세 번째 부분은 SpM 이며 여기서는 시간 구동 메소드를 기술한다. 각 SpM은 각각의 구동 조건(Activation Condition)을 가지며 여기에는 RTO의 시작시간(start time), 구동주기(period) 및 마감시간(deadline)을 정의한다. 이와 같은 실시간 속성을 가진 메소드를 구동하기 위해서는 실시간 실행 지원 장치가 필요하다.

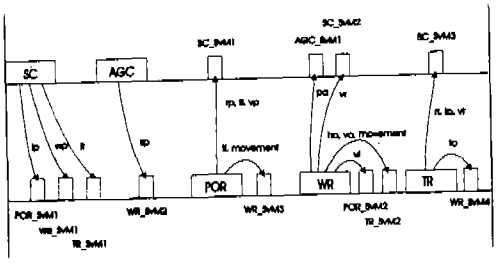
또한 SpM의 우선순위(priority)를 기술하는데 SpM은 SvM에 대해 항상 높은 우선순위로 가지나 SpM 간에는 동일한 우선순위에서는 선착순이며 서로 다른 우선순위를 가지는 SpM이 경쟁하게 되면 우선순위에 따른 실행순서를 가지게 된다. 출력 명세(Output Specification)는 SpM이 출력하는 메시지의 명세를 기술하며 목

적지 RTO 및 서버(sever) SvM과 메시지 타입 및 마감시간을 기술한다.



〈그림 11〉 Pay-Off Reel 의 RTO명세

네 번째는 SvM이며 메시지에 의해 구동되는 메소드를 기술한다. 각각의 SvM은 마감시간을 가지며, Client RTO로부터 메시지를 받고 필요한 일을 수행하며 ODS를 update 하거나 다른 RTO로 메시지를 출력한다. 여기서 입력 명세(Input Specification)는 전달 받는 메시지의 명세이며 출력 명세는 다른 RTO로 전송하는 메시지의 명세이다. RTO가 구동되면 실시간 시계의 호출에 따라 계속 살아 움직이는 SpM과는 달리 SvM은 메시지에 의해 호출될 때에만 살아서 움직이고 일을 끝내고 나면 죽어버리는(sleep) 버리는 수동적 프로세스(Passive Process)이다.



〈그림 12〉 RTO메소드의 타임라인 분석

3.5 타임라인 분석

압연공정 시뮬레이션 모델을 두 노드에 할당하여 한 노드는 제어기 프로토타입, 다른 노드를 환경 시뮬레이터로 하면 제어기 노드에서는 Speed Controller 및 AGC RTO가 할당되고 환경 시뮬레이터에는 Pay-Off Reel, Work Roll 및 Tension Reel이 할당된다. 이들 각 RTO를 각 노드별로 분리하고 각 RTO의 메소드를 한 시뮬레이션 주기에서 구동 시간에 따라 표시하면 <그림 12>와 같다.

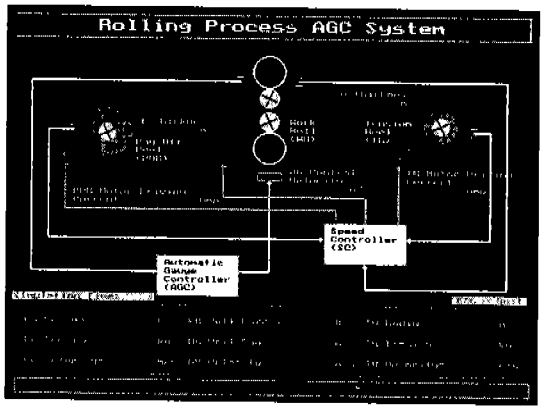
두 개의 수평선은 두 개의 노드를 나타내고 두 노드상에 위치한 네모들이 각 RTO의 메소드들이다. 네모의 폭은 마감시간내의 실행시간을 의미하며 큰 네모는 SpM을 작은 네모는 SvM을 나타낸다. 각각의 SpM들은 SvM을 호출하여 메시지를 전달하는 구조를 보여주고 있으며 SvM의 구동 시각은 다른 SpM이 실행되고 있지않을 때 자신이 호출된 시점임을 보여준다. 다만 POR SpM이 WR_SvM3를 호출하는 것과 같이 한 노드상에서 호출이 일어날 때는 SpM이 종료된 후에 SvM이 실행된다.

4. 시뮬레이션 모델의 구현

4.1 구현 환경

본 시뮬레이터는 실시간 시뮬레이션 엔진으

로서 Dream Kernel 상에서 구현되었고, 시뮬레이션 언어는 C++ 언어를 사용하였으며 RTO 모델을 지원하는 Dream Library를 이용하였다. Ethernet LAN 상의 3대의 PC에 각각 제어기 프로토타입과 환경 시뮬레이터 및 그래픽 노트가 탑재된다. 제어기 및 시뮬레이터는 각각 실시간 시뮬레이션 주기에 따라 각 RTO를 수행하며, 각 노드 및 RTO간의 분산 네트워킹은 Dream Net의 Logical Multicast Protocol(LMP)을 이용하여 분산 노드 및 RTO 간의 노드 투명성(transparency)을 제공하고 있다. 그래픽 노드는 제어기와 시뮬레이터가 네트워크상에 multicast하는 메시지를 수신하여 이를 그래픽으로 보여준다. <그림 13>은 그래픽 노드의 실행 예를 보여주고 있다.



〈그림 13〉 그래픽 노드

4.2 RTO의 분산

Dream Net의 LMP는 LAN상의 multicasting 프로토콜로서 네트워크상의 노드 투명성을 제공해 준다. 구현된 시뮬레이터의 정밀도를 향상시키기 위하여는 각 RTO에 대한 컴퓨팅 파워를 증가시켜야 하고 따라서 각 RTO를 각각의 프로세서에 할당할 필요가 생긴다. 이런 경우에 기존 노드의 RTO들을 간단히 분리하여 각 프로세서에 쉽게 이식(porting)할 수 있다. PC 프로세서의 파워, RTO의 복잡도 및 시뮬레이션 주기 등에 따라 노드 분리의 수준을 고려할 수 있다. 본 연구에서 구현한 시뮬레이션 모델의 변수를 더욱 늘리고 시뮬레이션 주기를 더 짧게 하면 보다 현실에 가까운 시뮬레이터가 될 수 있는데, 이렇게 하려면 RTO의 복잡도는 증가하게 되고 RTO 수행에 따른 컴퓨팅 파워의 요구량은 더욱 커지게 되며 따라서 RTO의 노드별 분리가 필요하게 된다.

RTO의 복잡도가 증가하여 하나의 프로세서가 처리하기에 충분치 않을 정도가 되면 RTO 자체를 분리하여 메소드별로 노드를 할당하여야 한다. 이것 역시 LMP 상에서 쉽게 구현할 수 있다.

5. 결 론

전통적인 시뮬레이션[10]은 가상 시간(Virtual Time)에 의해 시뮬레이션이 진행되며 사건(event) 처리를 위해 사건 리스트를 필요로 한다. 실시간 시뮬레이션은 실시간에 의해 시뮬레이션이 진행되며 사건 리스트를 필요로 하지 않는다. 실시간 시계를 이용함으로써 시뮬레이션 대상의 시간적 행위를 모의할 수 있다는 것은 전통적 시뮬레이션과 대비되는 특징이다.

실시간 분산 시뮬레이션을 구현함에 있어서 실시간 객체 지향(RTO) 방법은 설계 및 구현에 있어서 많은 장점이 있음을 확인하였다.

시간 구동 메소드와 메시지 구동 메소드를 구분함으로써 실시간 행위의 설계 명확성을 가져다주었고 시뮬레이션 모델을 실시간 객체로 분할함으로써 구현의 용이성을 가져왔다. 또한 RTO 방법으로 구현함에 따라 시뮬레이션 모델의 노드 분산이 매우 용이함을 경험하였다.

실시간 객체에 기반한 시뮬레이터의 개발은 많은 장점을 가지고 있으며 이는 보다 복잡한 대규모 실시간 응용에 적합한 접근 방법으로 판단된다. 예를 들면 복잡한 로직을 가지고 있

으며 매우 많은 분산 노드로부터 데이터를 실시간으로 수집, 통제해야 하는 교통제어 시스템 모델에 본 연구의 접근 방법이 유용할 것으로 기대된다. 또한 안전이 최우선시 되는 원자력 발전소는 시스템 개발 시 설계의 명확성이 매우 필요한 분야인데 이러한 분야에도 필요할 것으로 생각되며 공장 및 물류자동화 시스템의 시뮬레이션 또한 본 연구의 접근 방법이 유용할 것으로 생각된다.

본 연구의 압연공정 시뮬레이터는 단일 스탠드를 대상으로 하였다. 현재 본 연구팀은 이를 다 스탠드로 확장하고 있으며 시뮬레이션의 정밀도를 높이기 위한 노드의 분산에 관한 연구를 하고 있다. 또한 시뮬레이션 결과를 다양하게 제공하기 위한 그래픽 인터페이스도 연구하고 있고 3차원 VR(Virtual Reality) 노드 개발이 진행되고 있으며, 향후 이 연구를 토대로 하여 다양한 시뮬레이션 응용에 적용할 예정이다.

〈참고문헌〉

[1] Alan Burns, Andy Wellings, "Real-Time Systems and Their Programming Languages", Addison-Wesley, 1990.

[2] G. Booch, Object-Oriented Design, Benjamin Cummings, CA.

[3] K.H. Kim et al., "Distinguishing Features and Potential Roles of the RTO.k Object Model", Proc. IEEE Workshop on Object-oriented Real-time Dependable Systems, Dana Point, pp.36-45, Oct. 1994.

[4] K.H. Kim et al., "A Timeliness-Guaranteed Kernel Model-DREAM Kernel and Implementation Techniques", Proc. Int'1 Workshop on Real-Time Computing Systems and Applications, Tokyo, Japan, pp.80-87, Oct. 1995.

[5] Bernard P. Zeigler, Jinwoo Kim, "Extending the DEVS-Scheme Knowlege-Based Simulation Environment for Real-Time Event-Based Control", IEEE Tras. on Robotics and Automations, pp.351-359, June 1993.

[6] K.H. Kim, Y.B. Oh, "Real-Time Object Based High Assurance Design of a Control Computer System and an Environment Simulator: A Case Study", Proc. ISSAT Int'1 Conference on Reliability and Quality in Design", Anaheim, pp.126-134, Mar. 1997.

[7] Alan Finn, Randall Decker, Chris McClurg, Dayle Harmon, "Simulation of Multiple Access Protocols for Real-Time Control", Simulation, pp.123-130, Feb. 1992.

[8] J. Rumbaugh et al., Object-Oriented Modeling and Design, Prentice Hall, New Jersey.

[9] A. Alan B. Pritsker, "Introduction to Simulation and SLAMII", John Wiley & Sons, 1993.

[10] Averill M. Law, W.David Kelton, Simulation Modeling & Analysis, McGraw-Hill, 1991.

[11] Vladimir B. Ginzburg, High-Quality Steel Rolling, Marcel Dekker, 1993.

[12] 김광배외, 압연용 Automatic Gauge Controller의 개발, 한국과학기술연구원, 1995.