HLA/RTI 기반의 페트리 네트 분산 시뮬레이션

임동순, 오현승
한남대학교 산업시스템공학 전공
대전시 대덕구 오정동 133

Distributed Simulation of Petri Net Models with HLA/RTI

Dong-Soon Yim and Hyun-Seung Oh
Department of Industrial and Systems Engineering, Hannam University

요 약

A distributed simulation with HLA/RTI provides stable and satisfactory results. In this study, a distributed simulation of Petri net models under the HLA/RTI framework is considered. Throughout our experiences, it is recognized that the proper use of interface specification and time management services are important in order to achieve successful implementation of RTI. The interfacing tokens that are delivered to other models are distinguished as information entity and physical entity. Both entities are modeled as Interaction Class in order to send and receive messages. In synchronizing local simulation clocks, a conservative method with NERA service is considered. A cel manufacturing system is modeled and implemented with RTI to illustrate the distributed simulation of Petri net models.

I. 서 론

이산사건 시뮬레이션에서의 실행속도를 향상시키고, 거대한 모델의 메모리 문제를 해결하기 위한 방법으로 병렬 시뮬레이션 기법이 연구되어 왔다. 기존의 방법인 순차적 시뮬레이션은 하나의 시뮬레이션 모델을 한 컴퓨터에서 시간에 따른 순차적 사건 순으로 실행시키는 것에 비하여 병렬 시뮬레이션은 여러 대의 컴퓨터 또는 다중 프로세스를 갖는 한 컴퓨터에 시뮬레이션 모델을 분산시켜 실행시킴으로써 그 속도를 향상시키는데 목적이 있다[1,3].

병렬 시뮬레이션에서 추구하는 시뮬레이션 모델의 분산은 모델링 관점에서도 많은 장점을 제공한다. 특히, 복잡한 시스템을 시뮬레이션 하는 경우에 모델이 거대해져 전체 시스템을 하나의 모델로 묘사하는데 어려움이 존재한다. 만약, 모델링 대상이 되는 시스템이 각각의 모듈화된 단위 모델로 쉽게 작성되고, 각 단위 모델간의 인터페이스가 쉽게 구현될 수 있다면 분산 모델링은 좋은 장점을 제공한다.

HLA (High level architecture)는 미 국방성에의 주도하에 개발된 것으로서 여러 단위 모델로부터 거대한 시뮬레이션 모델을 생성하기 위한 소프

트웨어 구조이다[5]. HLA 시뮬레이션은 RTI (Run time infrastructure)라는 소프트웨어를 통하여 연결된 여러 시뮬레이터들로 구성되어 있다. 각 시뮬레이터들을 페더레이트(federate)라고 부르고, RTI를 통하여 연결된 페더레이트의 집합을 페더레이션(federation)이라고 부른다.

RTI는 다수의 시뮬레이션 모델들을 연결시키기 위한 서비스를 제공하는 특별한 목적의 분산 운영체제라고 할 수 있다. HLA의 인터페이스 규격은 페더레이션 실행 중에 RTI 또는 각 시뮬레이션 모델이 수행하는 서비스의 집합을 정의하고 있다[2,5]. RTI를 이용한 분산 시뮬레이션은 이러한 인터페이스 서비스를 이용하여 모델 간에 메세지를 주고 받음으로써 가능하다. RTI 소프트웨어는 CRC (Central RTI component)와 LRC (Local RTI component)로 구성되어 있다. 각 페더레이트는 자신의 시뮬레이션 모델과 엔진을 갖고 있고, LRC를 이용하여 RTI와 교신하여 다른 모델로 메시지를 보내거나 받기 위하여는 RTI를 경유하여야 한다. 이러한 중앙집중적인 RTI의 구조는 진정한 의미의 분산 시뮬레이션에 위배된다. 그러나, 각 시뮬레이션 모델의 시뮬레이션 시각을 동기화 하기 위하여는 분산 시뮬레이션에서의 시간 동기화 방법을 사용하여야 한다. RTI 소프트웨어는 분산 시뮬레이션에서의 시각 동기화를 위하여 보수적 방법[1]과 낙관적 방법[4]을 포함하고 있다.

본 연구는 Petri net의 분산된 모델들을 RTI 기반으로 통합하여 시뮬레이션을 수행한 결과를 제시한다. 특히, 시뮬레이션 모델 간의 인터페이스 방법과 시뮬레이션 시각을 동기화하기 위한 보수적 방법의 RTI 서비스 이용 방법을 적용한 사례를 소개한다.

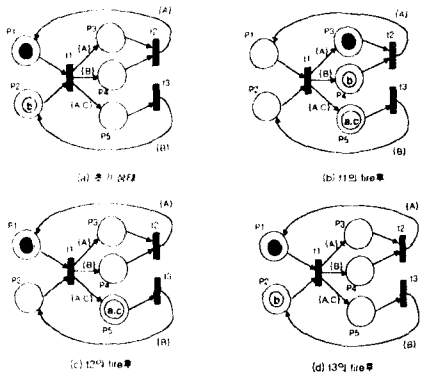
Ⅱ. JAVA 기반의 페트리 네트 시뮬레이션

본 연구에서는 생산시스템에 적용될 수 있는 페트리 네트의 모델링 및 시뮬레이션을 위해 JAVA 언어를 이용하여 시뮬레이션 도구를 작성하였다. 모델링 측면에서는 시스템의 상태 또는 행위를 플레이스(place)로 나타내고, 사건을 트랜지션(transition)으로 묘사하였다. 그러나, 행위 만을 하나의 플레이스로 표현할 경우 많은 플레이스를 야기하므로 하나의 행위와 이에 수반되는 행위 종료 사건, 그리고, 대기 상태를 하나의 플레이스로 모델링 가능토록 하였다. 또한, 시스템내의 자원 그리고, 흐름 객체들은 토큰(token)으로 모델링하도록 하였다. 사용된 페트리 네트네트는 기본적으로 일반적인 페트리 네트에 따르나, 생산 시스템의 특성을 고려하고, 시뮬레이션 실행을 위해 다음과 같은 모델링 요소를 추가하였다.

1) 플레이스는 시간 지연을 갖는 플레이스로 토큰이 도착하면 주어진 지연시간을 갖는다.
2) 플레이스에 있을 수 있는 최대 토큰의 수인 용량이 설정되어 있다.
3) 기본 토큰은 토큰 클래스들의 인스턴스이다.
4) 플레이스에 있는 한 토큰은 기본 토큰들의 집합이다.
5) 트랜지션의 출력 아크에는 생성되는 토큰 클래스의 집합이 정의된다.

다음의 예제를 통해 본 연구의 페트리 네트 실행을 설명하기로 한다. 그림 1의 페트리네트 모델에서 각 플레이스의 용량은 1이다. 현재 플레이스 P1과 P2 에는 각각 클래스 A와 B의 인스턴스인 토큰 a와 b가 존재한다. 따라서, 트랜지션 t1이 발사조건을 만족하여 즉시 발사되어 입력 플레이스에 있는 두 토큰은 t1으로 이동된다. 출력 플레이스 P3로의 아크에는 {A}가 정의되어 있어 이동된 토큰 중 토큰 클래스 A의 인스턴스인 a가 복사되어 P3로 이동된다. 출력 플레이스 P4로의 아크에는 {B}가 정의되어 토큰 클래스 B의 인스턴스인 b가

복사 이동된다. 플레이스 P5로의 아크에는 (A, C)가 정의되어 있다. 따라서, 토큰 a가 복사되고, C 클래스의 인스턴스인 새로운 토큰 c가 생성되고, 이들 두 토큰이 하나로 합쳐져 P3로 이동된다. 이 새로이 합쳐진 토큰은 플레이스에서 하나의 토큰으로 간주된다. 그러나 트랜지션이 발사 되면 다시 본래의 토큰 인스턴스들로 분리되어 출력 아크의 토큰집합에 따라 복사되고 제거 된다. 결국 t1의 발사 결과는 그림 4-(b)와 같다. 그림 4-(b)에서 t2가 발사 조건을 만족하여 우선적으로 발사된다고 하자. 트랜지션 t2의 발사 결과 토큰 a가 복사되어 P1으로 이동하고, P4에 있던 토큰 b는 삭제된다. 따라서 그 결과 그림 4-(c)와 같이 된다. 그림 4-(c)에서 t3가 발사되면 그림 4-(d)와 같이 되어 P5에 있던 토큰 (a, c)는 없어지고, 새로운 토큰 b가 P2에 생성된다.



[그림 1] 페트리 네트 실행

Ⅲ. 페트리네트 모델 간의 메시지 교환

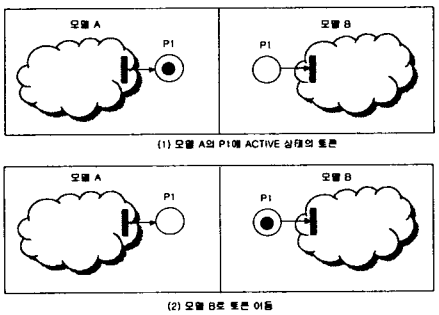
HLA/RTI 하에서 모델간의 교신을 위한 메시지의 형태는 Interaction class와 Object으로 나뉜다. Interaction class는 메시지가 교신을 위한 일시적인 수명을 가지는 특성이 있으나, Object은 메시지의 내용이 교신에 관계 없이 계속적으로 유효하는 경우에 한한다. 예를 들어, 탱크가 페더레이트 A에서 페더레이트 B로 이동할 때 탱크는 계속 유효하여 Object이 된다. 그러나, 미사일을 A에서 B로 쐈을 경우 미사일은 A에서 생성되어 B로 보내지고, B에서 미사일을 받아 처리된 후 즉시 없어져 Interaction class에 해당된다. Object은 특성치(attribute)를 포함하고, Interaction class는 파라미터(parameter)를 포함한다. 이들은 FOM(Federate Object Model)에 포함되어 FED 파일에 정의되어 있어야 한다.

본 연구에서 사용하는 페트리 네트 이산 사건 시스템에서는 모델 간에 송수신되는 엔터티는 토큰이다. 토큰은 송수신할 메시지의 내용에 따라 크게 물리적인 엔터티와 정보형 엔터티로 나뉠 수 있다. 정보형 엔터티는 생산 시스템의 경우 셀 콘트롤러와 기계 간에 주고 받는 명령 또는 상태 정보가 그 예로서 Interaction Class의 정의에 맞는다. A모델에서 생성된 정보형 엔터티는 즉시 B모델로 이동되어 처리된 후 제거된다. 물리적인 엔터티는 생산 시스템의 경우 작업물, 공구, 차량 등과 같이 물리적인 실체를 갖는 것으로 Object에 해당하나 Interaction Class로 정의하여 토큰의 복사 및 제거 절차에 따르도록 하였다. 모델 간의 엔터티를 송수신하기 위하여 페트리 네트의 일반적인 플레이스외에 정보형 엔터티 입, 출력 플레이스와 물리적 엔터티 입, 출력 플레이스를 추가하였고, FED 파일에 정의된 송수신 메시지의 구조는 다음과 같다.

```
(class Token reliable timestamp ToModel
        (parameter TokenID)
        (parameter TokenContainer)
        (parameter TokenName)
        (parameter TokenClass)
        (parameter TokenPlace)
        (parameter TokenCreateTime)
        (parameter TokenEventType))
```

모델 간에 정보형 엔터티를 송수신하는 절차는

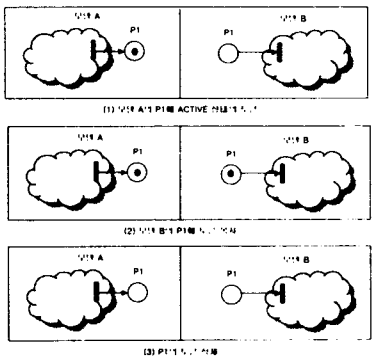
그림 2와 같이 한 토큰이 모델 A의 정보형 엔터티 출력 플레이스인 P1에 도착하면 토큰에 포함된 정보를 참조하여 토큰 메시지를 B 모델로 보내고, P1의 토큰을 삭제 한다(그림 2-1). 정보형 토큰을 보내는 메시지의 파라미터 EventType의 값은 토큰이 P1에 도착했음을 알리는 ARRIVAL로 한다. 반면, 토큰 메시지를 받은 모델 B는 파라미터에 정의된 TokenPlace의 이름을 갖는 정보형 엔터티 입력 플레이스에 해당 토큰을 생성한다(그림 2-2).



[그림 2] 정보형 엔터티 송수신

그림 3의 모델 간에 물리적 엔터티를 송수신하는 절차는 다음과 같다.

1) 모델 A에 토큰이 물리적 엔터티 출력 플레이스인 P1에 도착하면 토큰에 포함된 정보를 참조하여 토큰 메시지를 모델 B로 보낸다. 이때 파라미터 EventType의 값은 ARRIVAL로 하고, TokenPlace의 이름은 플레이스의 이름인 P1으로 한다(그림 3-1).
2) 토큰 메시지를 받은 모델 B는 파라미터에 정의된 TokenPlace의 이름을 가진 물리적 엔터티 입력 플레이스인 P1에 해당 토큰을 생성한다(그림 3-2).
3) 모델 B의 물리적 엔터티 입력 플레이스인 P1에 있는 토큰이 제거되면 해당 토큰 정보를 참조하여 토큰 메시지를 모델 A로 보낸다. 이때 파라미터 EventType의 값은 DEPART로 하고, TokenPlace의 값은 P1으로 한다(그림 3-3).
4) 토큰 메시지를 받은 모델 A는 파라미터 TokenPlace에 정의된 물리적 엔터티 출력 플레이스에 있는 해당 토큰을 삭제한다(그림 3-4).



[그림 3] 물리적 엔터티 송수신

RTI는 수신한 메시지의 클래스를 참조하여 이들을 구독 신청한 모든 페더레이트에게 메시지를 송신하기 때문에 각 페더레이트에서 보내는 메시지의 수신자를 정의하여야 한다. 이는 RTI의 데이터 분배 서비스를 이용하여 송수신할 메시지의 전달 범위(region)을 정의할 수 있다. 예를 들어, 3개의 페더레이트가 있고, 3 종류의 메시지를 표 1과 같이 보낸다고 하자. 3개의 전달 범위를 생성하여 다음과 같이 서로 중첩되지 않는 범위를 할당하고 각 페더레이트의 수신범위를 표 1의 마지막 열에 나타낸 바와 같이 설정한다.

Region 1의 범위: 0 이상 1 미만
Region 2의 범위: 1 이상 2 미만
Region 3의 범위: 2 이상 3 미만

각 federate는 표 1의 수신 범위에 해당하는 메시지 만을 구독 신청하기 위하여 subscribeIntertactionClassWithRegion 서비스를 요청한다.

또한, 각 메시지의 송신 시

sendInteractionWithRegion 서비스를 이용하여 송신 범위를 설정한다. 예를 들어, MSG12의 수신자는 페더레이트 2이므로 송신범위를 Region 2로 한다. 구체적으로 InteractionClass에 정의된 Token의 transmission 범위인 ToModel의 값을 Region2로 한다.

[표 1] Federate 간의 메시지 송수신

페더레이트	송신	수신	수신범위
Federate1	MSG12	MSG31	Region1
Federate2	MSG23	MSG12	Region2
Federate3	MSG31	MSG23	Region3

Ⅳ. 시뮬레이션 시각 동기화

시뮬레이션 시각 동기화 방법을 위해 RTI의 시간 관리 서비스는 크게 두 가지로 TSO(Time stamped order) 메시지 전달 서비스와 페더레이트가 자신의 시뮬레이션 시각(local time)의 진전을 요청하고 요청된 시간 이전의 외부 사건이 더 이상 없을 때 시간 진전을 허락하는 프로토콜을 제공한다. 시각 진전 프로토콜은 크게 보수적 방법[1]과 낙관적 방법[4]으로 나뉜다.

보수적 방법하에서 다음 사건 시뮬레이션(next event simulation) 엔진을 갖는 페더레이트가 시각 T에서의 내부 사건을 처리 한 후 외부 사건을 요청하는 서비스는 NER(Next event request) 또는 NERA(Next event request available)에 의한다. NERA의 경우 시각 T에서 Time Advance Grant가 주어졌을 경우 T와 동일한 시각의 TSO를 다 받았다는 것을 보장하지 않지만 시각 T에서 메시지를 보낼 수 있다는 것이다(lookahead가 0일 경우). 역으로 NER의 경우 T에서의 Time Advance Grant는 동일한 시각에서의 메시지를 모두 받았다는 것을 보장하지만 같은 시각에서의 메시지 송신은 불가능하다는 것이다. 따라서 NER(T)는 외부로의 메시지 송신이 항상 T 이후에 발생한다는 상황에서 사용할 수 있다. 만약, 한 페더레이트에서 다음 내부 사건의 시각이 T이고, 이 사건의 발생은 동일한 시각에서의 외부 사건을 발생시킨다면 NERA(T)를 사용하여야 한다. 그러나, 동일한 시각에서 메시지를 수신한 페더레이트가 메시지를 그 시각에서 다시 보낸다면, 이는 문제를 발생시킨다. NERA에서는 시각 T에서의 외부 메시지를 모두 받을 수 있다는 것을 보장하지 못하기 때문이다. 이를 해결하는 방법으로 페더레이트가 외부로부터의 메시지를 받았을 때 동일한 시각에서 즉각적인 외부로의 메시지를 보내지 말아야 한다.

본 연구에서는 외부에서 입력되는 메시지에 의해 페트리 네트의 플레이스에 토큰이 생성되면 약간의 지연시간을 갖도록 하였다. 또한 미래사건 리스트는 외부로의 메시지 송신 사건을 포함하기 때문에 lookahead를 0으로 하고, NERA 서비스를 이용하였다.

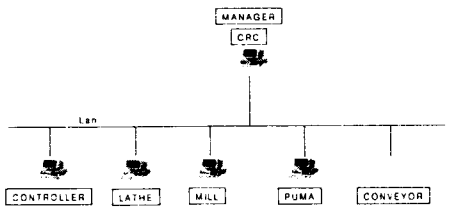
Ⅴ. 예제: 셀 콘트롤 시뮬레이션

분산된 페트리네트 모델들의 RTI에 의한 실행을 예제를 통하여 설명하기로 한다. 자동화 생산 시스템의 중요한 요소인 셀은 두 대의 기계와 로봇, 그리고, 작업물의 입출력 콘베이어로 구성되어 있다. 기계는 밀링기계(MILL)와 선반기계(LATHE)로 구성되어 있고, PUMA 로봇은 셀 내의 작업물 이동을 담당한다. 작업물은 콘베이어를 통하여 유입되고, 처리 된 후 방출된다.

시뮬레이션 모델은 두 기계(LATHE, MILL)와 로봇(PUMA), 콘베이어(CONVEYOR), 그리고, 셀 콘트롤러(CONTROLLER)로 구성된다. 셀 콘트롤러는 각 기기에게 적절한 명령을 하달하고, 기기의 상태를 모니터링 한다. 콘트롤러와 기기 사이에 전달되는 이러한 메시지를 정보형 엔터티로 모델링하였다. 기기사이에는 작업물이 이동된다. 예를 들

어, 로봇과 기계 사이에 작업물이 전달된다. 이러한 작업물의 이동은 물리적 엔터티로 모델링되었다.

페트리 네트 모델들은 각각의 컴퓨터에서 실행되어 컴퓨터 네트워크를 통하여 분산 시뮬레이션을 수행하였다[그림 4]. 부가적으로 모든 모델을 관리하는 MANAGER 페더레이트가 추가되어 CRC가 실행되는 컴퓨터에 있도록 하였다. MANAGER 페더레이트는 각 페트리 네트 모델들의 RTI상에서의 페더레이트 생성, 조인, 제거, 실행 초기화 등을 위한 관리자 역할을 한다. RTI 소프트웨어는 Pitch Portable RTI (pRti^{TM}) [6]를 이용하였다.



[그림 4] 페트리 네트 모형의 분산 시뮬레이션

Ⅵ. 결 론

본 연구에서는 RTI를 이용하여 페트리 네트 모델의 분산 시뮬레이션을 수행한 경험을 설명하였다. 특히, 각 단위 모델 간의 인터페이스 방법과 메시지 송수신 방법, 그리고, 시뮬레이션 시각의 진전을 위한 RTI 서비스를 어떻게 사용하는지에 대한 연구를 수행하였다.

셀 생산 시스템을 대상으로 분산 시뮬레이션을 수행한 결과 모델 간의 인터페이스는 RTI를 이용하여 용이하게 구현할 수 있었고, 안정적인 실행을 가능케 하였다. 본 연구는 페트리네트 모델 만을 대상으로 하였으나, 앞으로 다양한 모델링 방법에 따른 모델들을 통합 실행할 수 있는 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] Chandy, K. M. and Misra J., Asynchronous Distributed Simulation via a Sequence of Parallel Computations , Communications of the ACM, 24(4), pp. 198-205, 1981.

[2] Chiola, G. and Fersha A., Distributed Simulation of Petri Nets ,

[3] Department of Defense, High Leval Architecture Run-Time Infrastructure Programmers Guide (Version 1.3), 1998.

[4] Fujimoto, R. M., Parallel and Distributed Simulation Systems, Wiley Interscience, 1999.

[5] Jefferson, D., Virtual Time , ACM Transactions on Programming Languages and Systems, 7(3), pp. 404-425, 1985.

[6] Object Management Group, Inc., Distributed Simulation Systems Specification, 2000.

[7] Pitch AB, pRTI, http://www.pitch.se