'88 추계학술대회
'88 — K — 4

확장된 페트리 네트와 분산형 공정 제어의 모델링

임 성 호°, 김 현 기, 우 광 방
연세 대학교 공과대학 전기공학과

Extended Petri Nets and Distributed Processor Systems Modeling

Sung-Ho Lim, Hyun-Ki Kim, Kwang-Bang Woo
Dept. of Electrical Eng. , Yonsei University

Abstract

In order to represent and analyze distributed system design, the model based on an extended form of Petri nets, which enables one to represent both the structure and the behavior of a distributed system, is predented. Behavioral properties of the design representation are verified by translating the extended Petri net into an equivalent ordinary Petri net. The model emphasizes the unified representation of control flows, hierarchical structure, and distributed system state. Modeling technique isemployed for the performance and function analysis of flexible manufacturing system with a set of processors.

I. 서 론

현대의 기술은 급격한 기술발달과 더불어 다품종 소량생산의 추이에 의하여, 고정적이었던 제어 시스템을 가변적으로 하는 제어 방식에 대해 유연성(flexibility)을 갖도록 하는 공장 자동화에 이르게 되었다. 이는 특히 상호 호환성이 없는 PLC, 로보트, 수치제어기 등의 언어가 상이할 뿐 아니라 서로 다른 수준의 도구(tool)를 하나로 결합해야 하므로 이들을 총괄적으로 감시하고 제어하기 위한 새로운 소프트웨어의 개발이 요구된다. 시스템을 모델링함에 있어 비동기성 및 병렬성을가지는 시스템은 그래픽적인 관점에서 많은 시도가 있어왔다. 이 중 대표적인 것은 GRAFCET[1], R-NET[2], C-NET[3] 등이 있다.

본 연구에서는 공장 자동화를 모델링함에 있어 분산시스템으로 착안을 두었으며, 여러 프로세서 간의 메시지를 매체로 하는 통신 프로토콜을 두어 동기화 및 병렬성을 확장된 페트리 네트로 표시함으로써 계층적 접근에 의한 분산시스템의 모델링 방법을 제시하고자 한다.

II. 분산시스템의 정의

1970년대와 1980년대에 걸쳐 분산시스템이 발전하기 시작하여 컴퓨터 네트워크, 다중 프로세서(multiprocessor)그리고 통신등에 있어 기초가 되어왔다. 1980년에 이르러 마이크로 프로세서와 VLSI의 유용성때문에 여러가지 네트워크 기술이 개발되었다. 원거리 네트워크(WAN: Wide Area Net - work)및 근거리 네트워크(LAN: Local Area Network) 그리고 다중 프로세서등은 일종의 분산 시스템의 일종이다. 분산시스템은 다수의 처리요소(PE : Processing Elements)로 구성되어 있으며 서로간에 통신이 가능하다. 분산시스템에서의 처리계획(process scheduling) 문제는 분산 시스템에서 가장 복잡한 문제중의 하나이다. 일반적으로 이것은 다음의 두가지 범주[4]로 나누어 진다.

- 임의 처리(Random Process) — 주기적이지 않고 어떤 임의의 사건에 의한 하드웨어 인터럽트에 의해 처리된다.
- 주기적 처리(Periodic Process) — 주기적이며 하드웨어 및 소프트웨어의 클럭 인터럽트 혹은 주기적인 사건에 의해 처리된다.

본 논문에서는 주기적인 처리 형태를 다룰것이며 스케쥴링 기술은 모든 처리요소가 전체 시스템에서 대기중인 프로세

서를 받아들일수 있는 대기 큐(Ready Queue)를 가질수 있도록 하였다. 분산 시스템을 모델링하기 위해 계층적인 원리를 이용해서 모델을 하위모델(submodels)로 분해하여 표현한다. 일반적인 분산된 시스템의 시뮬레이션은 그림 II-1과 같다.

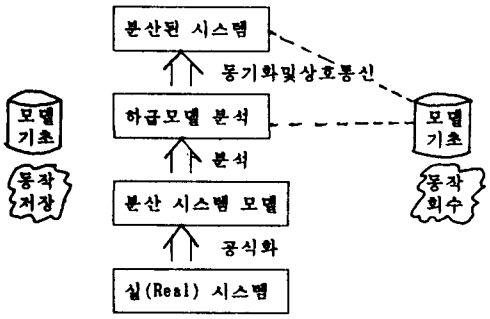

그림 2. 분산된 시스템 시뮬레이션

III. 페트리 네트의 확장

III-1. 페트리 네트

페트리 네트는 비동기적이고, 동시적인 시스템의 동작을 묘사하고 해석하기 위한 간편하고도 강력한 그래픽적인 도구(tool)이다[5]. 이 그래픽 모델은 플레이스(place), 트랜지션(transition), 그리고 플레이스와 트랜지션을 연결해 주는 호(arc)로서 구성되며, 페트리 네트의 실행은 토큰(to- ken)의 이동에 의한다. 토큰을 갖는 페트리 네트를 표시된 페트리 네트라고 하며, 그 플레이스의 마킹(marking)은 모델화된 시스템의 상태(state)를 나타낸다. 트랜지션의 작동은 어떤 사건의 발생을 의미하는데 페트리 네트에 대한 작동규칙(firing rule)은 다음과 같다.

- 트랜지션은 그의 모든 입력 플레이스들이 토큰을 갖고 있어야만 작동가능(enable)하다.
- 트랜지션은 작동가능하여야만 작동할 수 있다.
- 트랜지션이 작동하게되면, 트랜지션의 모든 입력 플레이스들로부터 토큰이 제거되고, 모든 출력 플레이스에 토큰이 들어간다.

III-2. 안전한 페트리 네트

각 플레이스들이 많아야 한개의 토큰을 가지는 네트를 안전한 페트리 네트(Safe Petri Nets)라 한다. 안전한 페트리 네트의 모델은 SPN = (P, T, I, O, M)의 5개의 요소로 정의된다. 여기서 P는 플레이스의 집합, T는 트랜지션의 집합, I, O, M은 각기 트랜지션의 입력 플레이스, 출력 플레이스, 그리고 마킹 M을 나타낸다. 안전한 페트리 네트의 작동규칙에서 추가해야할 중요한 사실은 트랜지션은 모든 출력 플레이스들이 토큰을 갖지 않아야만 작동가능하다는것이다.

III-3. 확장된 페트리 네트의 제시 배경

순차의 비동기적이고 동시수행적인 작업은 토큰들의 이동으로 구현되나, 안전한 페트리 네트로 구성하게되면 분산 시스템과 같은 수많은 기계동작들을 포함한 복잡한 제어를 기술하기에는 많은 네트가 필요하므로 시스템을 올바르게 기술하기 어렵게 된다. 이러한 문제점을 피하고 병렬 및 분산 시스템의 구조 및 동작을 기술하기에 적합한 확장된 페트리 네트를 제시하게된다. 본 논문에서 제시한 확장된 페트리 네트 이외에도 수정된 페트리 네트들이 제시되어왔다. 평가 네트(Evaluation nets)[6], 프로 네트(pro-nets)[7], 매크로 E 네트(macro E-nets)[8]등이 트랜지션의 작동규칙을 수정하면서 제시되었다.

III-4. 확장된 페트리 네트의 구성

확장된 페트리 네트는 다음의 4개의 요소로 정의된다.

EPNs = (S, C, R, M)

S: 제어상태 변수의 집합

C: 하위네트 호출 집합

R: 상호 연결관계

M: 초기 마킹 함수

제어상태 변수는 원래의 페트리 네트에 있어 플레이스에 해당되며, 입력 및 출력 제어상태 변수로 나뉘어진다. 제어상태 변수의 마킹 이동은 시스템의 제어 흐름을 나타낸다. 하위네트 호출 집합 C는 원래의 페트리 네트에 있어 트랜지션에 해당되며, 다음의 4개의 요소로 정의된다.

Ci = (Pi, Qi, Ti, Fi)

Pi: 입력 제어상태 변수

Qi: 출력 제어상태 변수

Ti: 트랜지션

Fi: 제어전달 정의

하위네트 호출 집합 C및 트랜지션의 작동규칙은 안전한 페트리 네트와 같으나, 상호 연결관계 R에 따라서 달라진다. 확장된 페트리 네트의 예가 그림 2에 나타나 있다.

III-5. 상호 연결관계및 작동규칙

상호 연결관계 R은 제어상태 변수와 트랜지션및 하위네트 호출과의 논리적인 관계를 말한다. 그 논리적인 표현에는 AND, OR, Exclusive-OR, 우선 순위(Priority)등이 있으며 그림 3및 그림 4에 잘 나타나 있다.

확장된 페트리 네트의 작동규칙은 상호 연결관계에 의해 달라지는데 표1및 표2에 잘 나타나 있다.

표1. 입력 제어상태 변수에 의한 작동규칙

상호 연결 관계	실행전 제어 상태		작동가능 시험	실행후 상태 변수의 값	
	p1	p2	p1 * p2	p1	p2
A N D	D	D	D	-	-
	D	E	D	-	-
	E	D	D	-	-
	E	E	E	M(p1)-1	M(p2)-1
	P1	P2	P1 + P2	P1	P2
O R	D	D	D	-	-
	D	E	E	-	M(p2)-1
	E	D	E	M(p1)-1	-
	E	E	E	M(p1)-1	M(p2)-1
X I O R	P1	P2	P1 ⊕ P2	P1	P2
	D	D	D	-	-
	D	E	E	-	M(p2)-1
	E	D	E	M(p1)-1	-
	E	E	D	-	-
우 선 순 위	p1	p2	$p1^{(i)} ++ p2^{(j)}$	p1	p2
	D	D	D	-	-
	D	E	E	-	M(p2)-1
	E	D	E	M(p1)-1	-
	E	E	E	if i > j M(p1)-1	-
				if i < j -	M(p2)-1

** D는 작동 불능(Disable)을 나타낸다.
E는 작동 가능(Enable)을 나타낸다.

표2. 출력 제어상태 변수에 의한 작동규칙

상호 연결관계	작동 결과 값	
	p1	p2
p1 * p2	M(p1)+1	M(p2)+1
p1 + p2	M(p1)+1	—
	r —	M(p2)+1
	r M(p1)+1	M(p2)+1
p1 ⊕ p2	M(p1)+1	—
	or —	M(p2)+1
$p1^{(i)} ++ p2^{(j)}$	if i > j M(p1)+1	—
	if i > j —	M(p2)+1

IV. 시스템 시뮬레이션

IV-1. 확장된 페트리 네트에 의한 메시지 통신 설치

여러개의 프로세서로 구성된 분산 시스템에 있어서 프로세서 상호간의 통신이 이루어져야할 필요성이 있다. 본 논문에서 제시한 통신모델은 메시지를 통해 자료나 도구를 프로세서간에 주고 받는 모델이며 그림 5에 잘 나타나 있다.

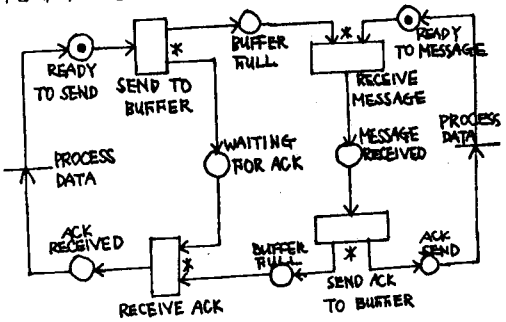

그림 5. 메시지를 매체로 한 통신 프로토콜

IV-2. 자료의 구조

확장된 페트리 네트를 컴퓨터상에서 수행하기 위하여, 각 노드들은 메모리내에 구조적인 형태로 기억된다. 본 연구에서는 구조적 언어인 C언어로 연결 리스트(linked list)형식[9]을 구성하여, 그래프 형태의 자료구조를 갖도록 하였다. 각 노드들은 자신의 입출력 노드들을 지정하는 포인터(pointer)를 갖는다. 여기서 하위 네트를 갖고있는 노드는 자신의 하위 네트에 대한 선두 포인터(head pointer)를 갖고 있어야 한다. 각 플레이스들은 시스템의 상태에 대한 정보와 토큰의 수를 갖고있으며, 트랜지션은 시스템의 동작에 대한 정보와 동작이 행해지는 시간을 갖고있다.

IV-3. 확장된 페트리 네트를 이용한 시뮬레이션의 예

현대의 생산기술은 다변화하여 다품종 소량생산에 의한 유연성 있는 생산시스템(FMS: Flexible Manufacturing System)의 기술에 이르게 되었다. 이것은 여러개의 프로세서가 순차및 동시처리되는 분산 시스템으로 볼 수 있다. 본 연구에서 예로 들은 시스템은 기계(Machine)3, 도구(Tool)2, 도구 이동기(Tool Conveyor)2을 가진 시스템이다. 본 공정은 두단계의 공정으로 이루어 진다. 첫번째 단계는 기계1이 도구1을 쓰고 난뒤 도구2를 쓴다. 두번째 단계는 기계2가 도구1을 쓰거나 기계3이 도구2를 쓰면 제품에 대한 공정은 끝나게 된다. 단, 도구 이동기1은 도구1을 이동시키고, 도구

이동기2는 도구2를 이동시킨다. 여기서 M1이 T1을 사용하는 처리동작과 M3이 T2를 사용하는 처리동작, 그리고 M1이 T2을 사용하는 처리동작과 M2가 T1을 사용하는 처리동작들은 동시에 처리하게 된다. 또한 T1을 사용하는 M1과 M2, 그리고 T2을 사용하는 M1과 M3사이에는 메시지에 의한 상호 통신이 필요하게된다. 이 시스템을 확장된 페트리 네트로 구성한 모델이 그림 6및 그림 7에 나타나 있다.

그림 7에서 하위네트 호출 집합 C1은 도구 이동기가 기계의 버퍼에 도구를 공급해 주는 네트이다. 그림 6에서 볼때 C3는 기계 M1이 도구 T1을 쓰고난뒤 도구 T2를 쓰게되는 우선 순위를 나타낸다. C4, C5, C9, C10은 기계들의 처리동작을 나타낸다. 여기서 C4와 C10 그리고 C5와 C9는 동시처리의 동작이 일어나게 된다.

V. 결 론

본 연구에서는 지금까지 제시되었던 안전한 페트리 네트의 문제점을 수정하여 확장된 페트리 네트 모델을 제시하였으며, 여러 프로세서간의 갈등(Conflict)현상을 없애기 위하여 메시지에 의한 통신 프로토콜을 두어 원활한 동시처리가 이루어 지도록 하였다. 또한 자료 구조에서 시스템의 실행과정을 알수 있도록 초기 마킹에서 나오는 성취가능한 마킹의 집합을 추출할 수 있다.

이상에서 볼때 확장된 페트리 네트를 이용해 더욱더 복잡한 네트워크, 공장 자동화및 컴퓨터의 집적생산시스템(CIM: Computer Integrated Manufacturing System)등의 구현에 이용 가능하리라 본다.

VI. 참고 문헌

[1]. Kichie Matsuzaki, et. al :Petri Net structured sequence - control language with GEAFCET like graphical expression for programmable controllers, Proc. IECON '85, pp 423/438, 1985.

[2]. Tomohiro Murata and Norihisa Komoda : A Petri Net-Based Controller for flexible and Maintainable Sequence Control and its application in Factory Automation, IEEE Trans. on Industrial Electronics, vol IE-33, no1, pp 1/8, Feb., 1986.

[3]. Kensuke Hasegawa and Paulo Eigi Miyagi : Application of the Mark Flow Graph to Represent Discrete Event Production system and system Control, Tran. of SICE, 24-1, 69/75, Jan. 1988.

[4]. Doo-Kwon Baik : Distributed Simulation on Multiprocessors, Kyung Moon Sa, vol I, pp2/16, 1985.

[5]. Peterson, J. L. : Petri Net Theory and the Modeling of Systems, Prentice-Hall Inc., Englewood Cliffs, NJ, 1981.

[6]. Nutt, G. I. : Evaluation nets for computer system performance analysis, in Proc. 1972 Fall Joint Comput. Conf., vol 41, pp 279/236.

[7]. Noe, J. D. and Nutt, G. I. : Macro E - nets for representation of parallel systems, IEEE Trans. Comput., vol C-22, pp 718/727, Aug., 1973.

[8]. Noe, J. D. : Hierachical modeling with pro - nets, in Proc. Nat. Electron. Conf., vol 32, Oct. 1978.

[9]. Suck-Ho Lee : Data Structure, Hong rung Inc., 1987.

[10]. Ramamoorthy, C. V. : Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets, IEEE Trans. on Software Eng., vol SE-6, no5, pp 733/745, SPET., 1980.

[11]. Giorgio Bruno and Giuseppe Marchetto : Process - Translatable Petri Nets for the Rapid Prototyping of Process Control Systems, IEEE Trans. on Software Eng., vol SE-12, no2, pp 346/357, FEB., 1986.

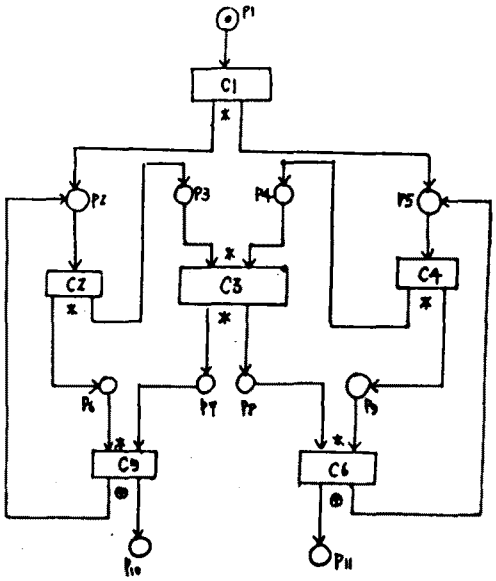

그림 3. 확장된 페트리 네트에 대한 모델 예

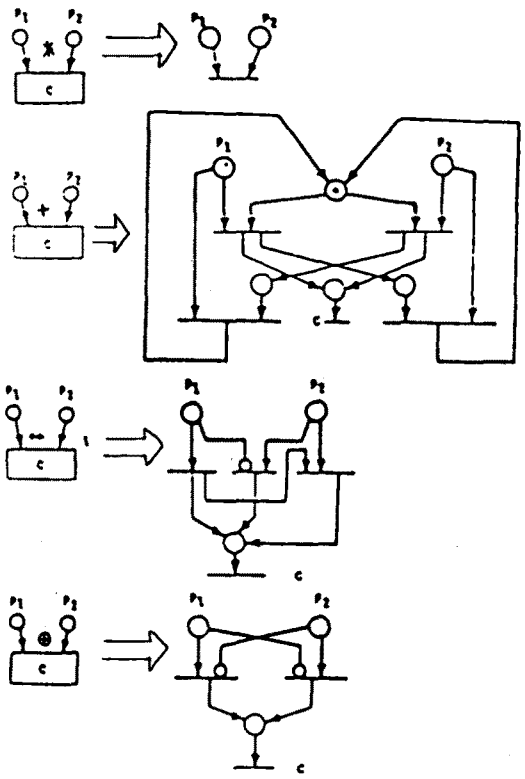

그림 4. 입력 제어상태 변수의 상호 연결관계 표시

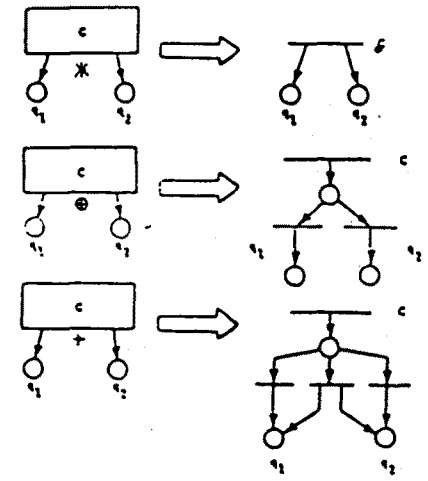

그림 5. 출력 제어상태 변수의 상호 연결관계 표시

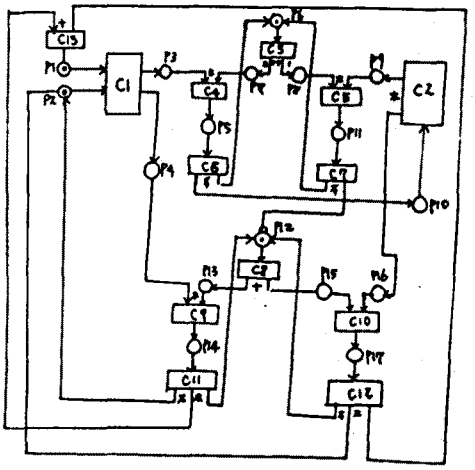

그림 6. 시스템의 예에 대한 모델링

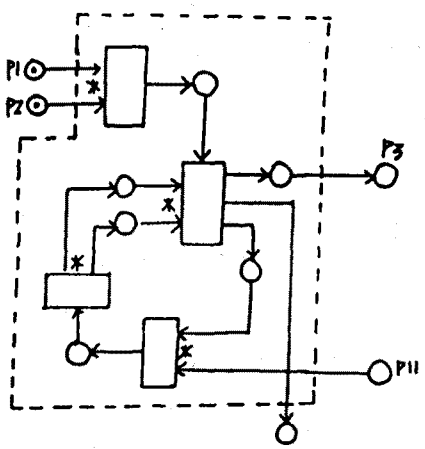

그림 7. 하위네트 호출 C1의 내부구조 표시