컴퓨터 통신 프로토콜의 모델링과 시뮬레이션에 관한 연구

손 진 곤　　백 두 권
고 려 대 학 교　　전 산 과 학 과

A Study on Modelling and Simulation of Computer Communication Protocols

Jin-Gon Shon and Doo-Kwon Baik
Dept. of Computer Science, Korea Univ.

< 요　　약 >

본 논문에서는 컴퓨터 통신 프로토콜의 모델링과 시뮬레이션에 관한 이론적인 연구를 하였다. 먼저 통신 프로토콜의 정의와 기능을 설명하였고 통신 프로토콜을 위한 모델을 분류하였다. 또한, 통신 프로토콜의 모델 중에 Timed Petri Net (TPN) 모델에 시간함수 τ를 부여함으로써 TPN을 구조적으로 정의하였으며 PN Based모델의 문제점을 보완하기 위하여 시스템 시뮬레이션 분야의 이산사건 시스템 명세(DEVS:Discrete EVent Simulation) 개념을 도입하였다. 중요한 연구 결과로서 TPN모델이 DEVS모델이라는 정리를 제시하고 증명하였다. 이 정리에 따르면,TPN모델을 시뮬레이션 할 때 시뮬레이션을 위해 모델을 설계할 필요없이 DEVS모델로 변환하여 사용함으로써 시뮬레이션을 수행할 수 있다.

ABSRACT

In this paper, we have studied modelling and simulation of computer communication protocols theoretically. After describing a definition and functions of communication protocols, we have classified models for protocol design. And, in those protocol models, by endowing Timed Petri Net(TPN) models with a time function τ, we have proposed a structual definition of TPN models.

Furthermore, in order to complement Petri Net Based models with some problems, we have introduced the Discrete EVent system Specification(DEVS) concept in system simulation field. As an important result of our study, we have presented a theorem, which says that a TPN model becomes a DEVS model, and proved it.

According to the theorem, We can perform efficient simulation by using the DEVS model transformed from a TPN model when we intend the TPN model to be simulated, otherwise we design another simulation model for it.

1. 서론

컴퓨터 하드웨어의 가격이 저렴해지고 처리능력도 두드러지게 향상됨에 따라 지리적으로 분산된 컴퓨터 자원들을 서로 연결시켜 자원의 공유 및 처리의 분산화 등을 위해 네트워크의 필요성이 크게 대두하게 되었고, 컴퓨터 네트워크내에서 분산된 프로세스 상호간의 통신을 제어하기 위해서는 통신 프로토콜이 반드시 필요하게 되었다.

통신 프로토콜의 모델링과 시뮬레이션에 관해서는 프로토콜의

서술(description), 성능평가, 유효성확인(validation) 및 검증(verification) 등을 효과적으로 수행하기 위해 연구되어 왔으며 특히 국제 표준기구(ISO)에서 OSI 참조모델을 발표한 1970년 이후 이에 관한 다양한 연구가 활발하게 진행되었다.

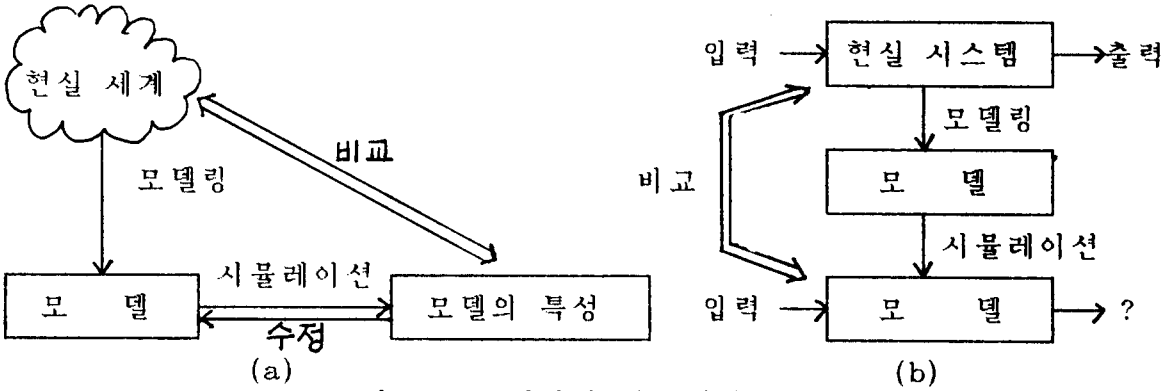
통신 프로토콜의 모델 가운데 Petri Net Based 모델의 경우 시스템의 세부적인 행위 (behavior)는 상세히 표현할 수 있으나 시스템의 전체적인 행위를 조감하고자 할 경우에는 사용되기 어렵다는 단점을 갖는다. 그런 반면 모델링과 시뮬레이션 분야의 시스템 설계 및 분석방법의 하나인 DEVS 모델의 계층적 분석방법은 시스템의 세부적인 행위를 표현하고 분석하는데는 문제점을 내포하고 있으나 시스템을 전체적으로 통합하여 분석하는 경우에는 상당히 훌륭한 방법으로 평가된다. 뿐만아니라 Petri Net Based 모델에서 분석할 수 없는 시스템의 행위를 DEVS 명세로는 시뮬레이션 방법을 통해 분석이 가능하다는 장점이 있다.

따라서 본 논문에서는 통신 프로토콜의 모델링에 관한 이론 중 Petri Net Based 모델의 단점인 계층성 표현 능력의 부족과 상태 폭발(state explosion) 문제를 보완하기 위하여 시스템 시뮬레이션 분야의 이산사건 시스템명세(DEVS : Discrete EVent system Specification) 개념을 도입하고, 시간조건이 첨가된 Timed Petri Net(TPN) 모델이 DEVS 모델로 변환될 수 있음을 밝힘으로써 TPN 모델에 의한 해석적 분석과 DEVS 모델에 의한 시뮬레이션적 분석을 효율적으로 통합분석할 수 있는 가능성을 제공한다.

2. 모델링과 시뮬레이션

2.1 모델링과 시뮬레이션

일반적으로 사람은 현실 세계를 사실 그대로 또한 세부적으로 정확하게 이해할 수는 없다. 따라서 현실 세계를 개념적으로 이해하기 위하여 사람들은 머리속에 현실 세계를 이러할 것이다라고 추정하는데 이러한 작업을 모델링이라 하고 그 결과로서 추정된 현실 세계를 모델이라 한다. 또한 작성된 모델에 대해 어떤 목적을 가지고 실험하는 작업을 시뮬레이션이라 하며 그 결과로서 얻어지는 모델의 특성을 현실 세계의 특성으로 간주한다. 현실 세계의 행위와 모델의 행위가 같지 않으면 모델이 잘못 작성되었다고 보고 모델을 수정하는 작업이 필요하다 (그림 1).



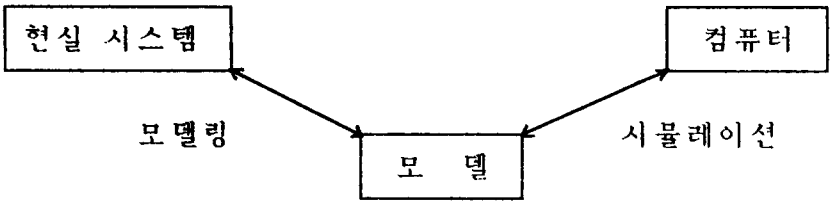
< 그림 1 > 모델링과 시뮬레이션

이러한 과정은 반복적으로 계속될 수 있고 대부분 컴퓨터(기계)로 시뮬레이션 할 수 있는 수준까지 모델은 계속 변환된다. 예를 들어 통신 프로토콜이라는 현실 시스템을 개념적으로 머리속에

모델링하고 그것을 Petri Net 으로 모델링하고 그다음 Petri Net 모델의 실행을 위해 시뮬레이션 모델을 작성하고 프로그램을 작성하나 결국은 컴퓨터(기계)가 처리할 수 있는 어떤 프로그램을 작성하게 되는데 이 때 이 프로그램 역시 모델이다.

2.2 구성요소와 관계

이러한 모델링과 시뮬레이션 작업은 현실세계, 모델, 컴퓨터의 세가지 구성요소와 현실세계와 모델간의 모델링 관계, 모델과 컴퓨터 간의 시뮬레이션 관계등 두가지의 관계로 구성된다(그림 2) [Zeig 84].



< 그림 2 > 모델링과 시뮬레이션의 구성요소 및 관계

(1) 현실 시스템

현실 시스템은 실제 행위데이타를 만들어 내는 근원지로서 모델링을 하는 사람이 관심을 갖는 현실 세계의 한부분이다. 이것은 자연적 시스템이거나 인공적인 시스템일 수 있으며 또한 현재 존재하는 시스템이거나 장래에 만들어 질 시스템일 수도 있다.

(2) 모 델

모델은 현실시스템처럼 행위데이타를 생성할 수 있도록 작성되는 현실 시스템의 수학적 표현이다. 따라서 모델은 그 시스템에 대한 특성을 파악하고 다른 사람과 의사전달을 할 수 있는 수단이 된다. 모델의 행위데이타란 모델이 무엇을 하는가를 설명하는 것인데 이 행위데이타를 생성할 수 있도록 모델은 어떤 구조를 갖게 된다. 그러기 위해서는 집합, 관계 및 함수등과 같은 수학적 개념이 모델을 작성하는데 기초적인 도구가 된다.

(3) 컴퓨터

작성된 모델에 따라 행위데이타를 생성할 수 있도록 해주는 사람이나 기계등의 도구를 컴퓨터라 한다. 즉, 적합한 모델이 주어졌을 때 행위데이타를 생성할 수 있는 처리기로서 만일 모델이 프로그램으로 작성되었을 때 그 프로그램을 실행시킬 수 있는 디지탈 컴퓨터 또는 모델이 미분방정식으로 표현되었을 때 그 해를 구하는 사람등이 컴퓨터가 된다.

(4) 모델링 관계

현실 시스템을 모델로 작성하는 과정인 모델링에서는 모델이 현실 시스템을 얼마나 잘 표현하는가라는 모델의 유효성(validity) 을 고려한다. 즉, 현실 시스템의 행위데이타와 모델이 컴퓨터를 이용하여 생성하는 행위데이타와 같은가를 확인해야 하는데 모델의 유효성에는 복제적유효성, 예측적유효성, 구조적유효성 등의 종류가 있다.

(5) 시뮬레이션 관계

시뮬레이션 과정에서는 컴퓨터가 얼마나 신뢰성있게 모델이 작성된 바대로 수행하는가라는 정확성에 관한 검증(verification) 을 고려해야 한다. 예를 들어 모델이 프로그램으로 작성되었을 때 컴퓨터가 정확히 프로그램대로 실행되었는가를 확인한다.

2.3 모델링과 시뮬레이션의 목적

시스템을 모델링하고 시뮬레이션하는 목적으로 첫째는 어떻게 현실시스템이 작동하는가를 이해하기 위해서이다: 이런 경우는 현실시스템의 행위나 구조에 대한 가정(hypothesis)을 포함하는 모델을 작성하고 시뮬레이션하게 된다. 두번째 목적은 현실시스템의 효율적인 작동을 가능하도록 매개변수연구(parameter study)를 하기 위해서이다. 즉, 현실시스템의 작동중 어떤 측면을 최적화할 수 있도록 모델의 매개변수를 제어하게 되는데 이런 경우는 현실시스템에 대한 대체시스템으로서의 모델을 이용한다.

모델을 현실시스템에 대체하는 이유는 현실시스템의 실행이 비용이나 시간이 많이 요구되고, 경우에 따라서는 불가능할 경우에 대체 효과를 높일 수 있으며 컴퓨터를 이용하는 시뮬레이션은 반복할 수 있으며 통계적, 도식적 요약정보들을 쉽게 이해할 수 있도록 생성해주기 때문이다.

3. 통신 프로토콜과 모델

3.1 통신 프로토콜의 정의

우리가 일반적으로 두 개체가 성공적으로 통신하려면 서로 같은 언어로 말해야 한다. 무엇을 통신하고, 어떻게 통신하고, 언제 통신할 것인가는 관련된 개체들간에 서로 용납되는 규약에 따라야 하는데 이러한 규약을 통신 프로토콜이라고 한다. 즉, 하나의 데이터/컴퓨터 통신 시스템에서 서로 떨어져 있는 통신 실체간에 여러 가지 유형의 매체를 통하여 확실한 통신을 달성할 수 있도록 해 주는 일련의 절차나 규범을 말한다. 프로토콜이 가지는 기본적인 요소는 구문(syntax), 의미(semantic) 및 타이밍(timing) 등이 있다.

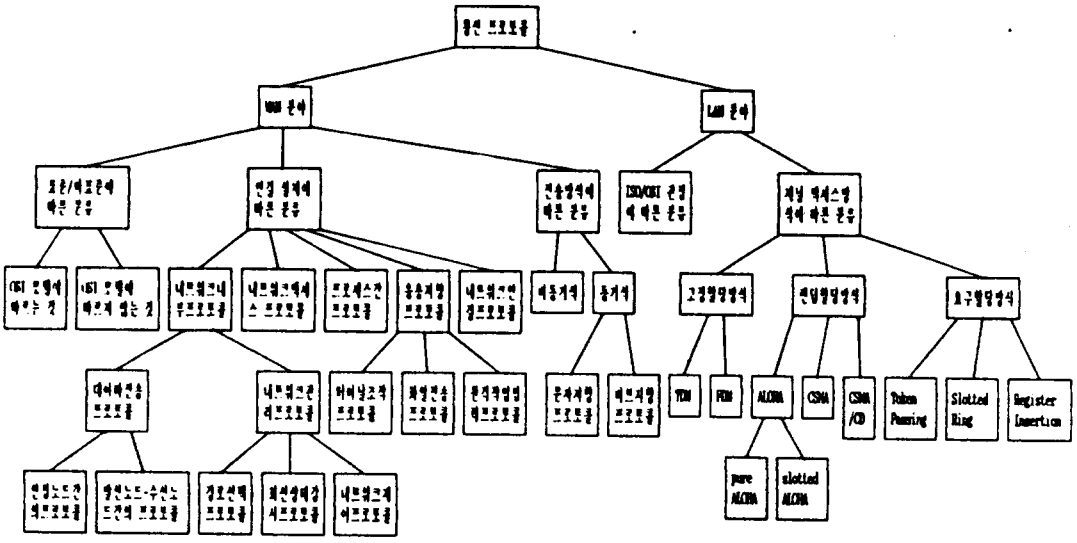
3.2 통신 프로토콜의 기능

통신 프로토콜이 갖는 기능은 다음과 같다 [STAL 85].

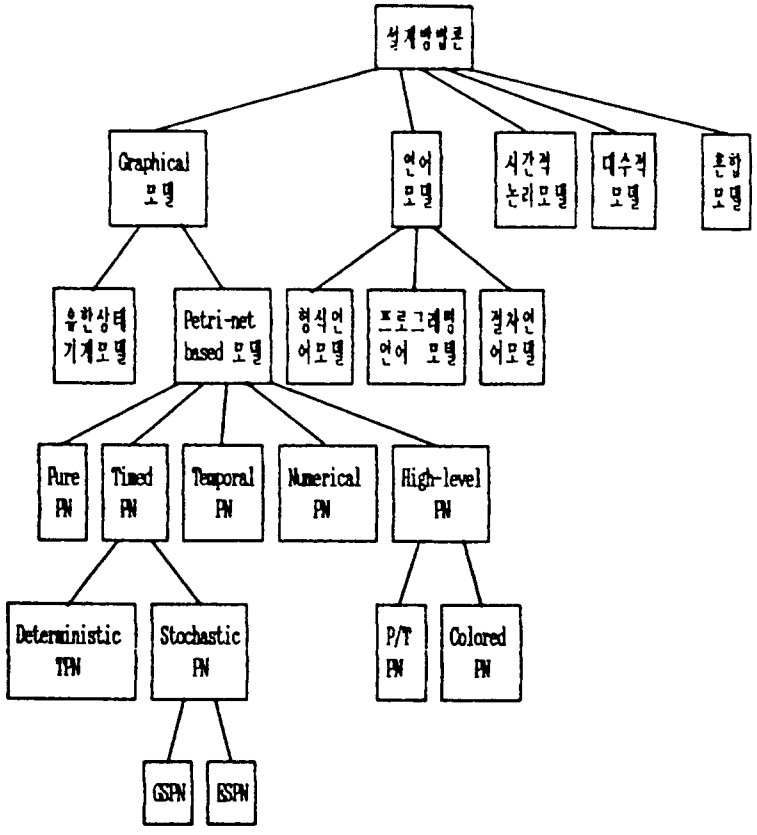
- 분할과 재결합(fragmentation and reassembly)
- 캡슐화(encapsulation)
- 연결제어(connection control)
- 흐름제어(flow control)
- 에러제어(error control)
- 순서제어(sequencing)
- 주소기법(addressing)

3.3 통신 프로토콜의 분류

통신 프로토콜의 특성을 기준으로 크게 4 가지로 프로토콜을 나누어 볼 수 있는데, 첫째로, 두 개체 사이의 통신이 직접적(direct) 또는 간접적(indirect)일 수 있는데, 공유 링크를 가진 두 개체간에는 직접적인 통신이 가능하나, 두 개의 개체 사이에 교환망이 구성되어 있지 않을 경우 에는 여러 네트워크를 통해 간접적으로 통신해야 한다. 둘째로, 구조면에서 살펴보면 단일체(monolithic) 또는 구조적(structured)으로 구성될 수 있는데,개체간의 통신이 하나의 단위로 처리하기에는 너무 복잡하므로, 하나의 프로토콜 대신 일련의 프로토콜들이 계층을 이루는 구조를 형성한다. 세째로, 프로토콜은 대칭(symmetric) 또는 비대칭(asymmetric) 일 수 있는데, 대칭적이란, 동등한 개체들간에 통신이 이루어진다는 것이다. 네째로, 표준(standard) 또는 비표준(nonstandard) 일 수 있는데, 표준화 되지 않은 프로토콜은



< 그림 3 > 통신 프로토콜의 분류



< 그림 4 > 통신 프로토콜 모델의 분류

특정한 통신 상황에서 특정한 모델의 컴퓨터에 사용된다.
또한 통신 프로토콜은 크게 LAN 분야와 WAN 분야로 나눌 수 있는데 그림 3에 나타내었다 [TANE 81, STAL 85].

3.4. 통신 프로토콜 모델의 분류

통신 프로토콜을 모델링하는데 이용되는 모델들을 분류하면 그래픽한 모델, 언어 지향적인 모델, 시간적 논리 모델, 대수적 모델, 그리고 혼합 모델로 나눌 수 있다(그림 4) [PETE 81, MURA 89].

4. TPN모델과 DEVS모델

4.1 Timed Petri Net (TPN) 모델

(1) TPN모델의 시간조건

Transition(또는 place)에 시간적인 제약을 부가한 TPN에 대한 연구는 Merlin의 박사학위논문[MERL 74]에서 시작하여 고정적 TPN, 확률적 TPN등 많은 연구가 발표되고 있으며, 특히 통신 프로토콜 분야에서는 Zuberek, Razouk, Phelphs와 그리고 Garg 등이 TPN을 적용하고 있다.

Merlin이 제시한 TPN의 정의에 따르면 TPN은 각각의 transition t_i에 대해 두 개의 시간, t_i^*와 t_i^{**}를 첨가시킨 PN이다. 여기서 시간 t_i^*는 t_i가 점화가능한 시점에서부터 점화될 때까지 반드시 기다려야 하는 최소한의 시간이며, 시간 t_i^{**}는 t_i가 점화가능한 시점에서부터 점화될 때까지의 시간중 최대한의 시간이다(즉, t_i는 늦어도 t_i^{**}이전에 점화되어야 한다).

Merlin의 TPN에서 고려하지 않은 시간적 제약으로 Carlier등 [CARL 84]이 제시한 점화실행시간(firing execution time)이 있는데 이것은 점화시작시점(starting firing time)에서 점화종료시점(ending firing time)까지의 시간을 나타낸다(본 논문에서는 각 transition t_i에 대해 점화실행시간을 f_i로 표시하기로 한다). 이때 t_i는 t(점화시작시점)와 t+f_i(점화종료시점) 사이에서 실행된다고 말한다. t_i의 입력 place들로부터의 token 제거는 점화시작시점에 발생하고, 출력 place들에게로의 token 첨가는 점화종료시점에 발생한다. Merlin의 시간조건과 Carlier등의 시간조건을 비교해 보면 그림 5와 같으나 두 경우 모두 시간 조건을 함수로 표시하지 않았기 때문에 TPN을 구조적으로 정의하지 못했다.

(2) TPN모델의 구조적 정의

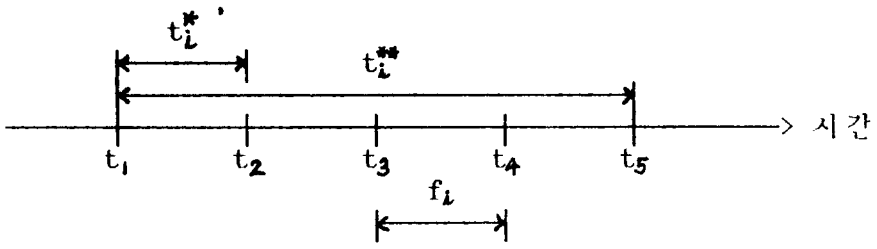
이제 Petri Net의 각 transition에 시간조건을 제공하는 시간 함수를 정의하고 그것을 이용하여 TPN을 구조적으로 정의한다.

< 정의 1 > Petri Net C = < P, T, I, O >에 대한 시간함수 τ는

$$\tau : T \longrightarrow R,$$
$$\tau(t_i) = (t_{1i}, t_{2i}, t_{3i}, t_{4i})$$

로 정의한다. 여기서
t_{1i}는 최소대기시간(Merlin의 t_i^*)
t_{2i}는 최대허용시간(Merlin의 t_i^{**})
t_{3i}는 점화시점
t_{4i}는 점화실행시간(Carlier의 f_i)이다.

< 정의 2 > Timed Petri net D = < C, τ >는 Petri net C = < P, T, I, O >와 C에 대한 시간함수 τ로 정의된다.



- t_1 : PN에 의한 논리적 점화가능시작시점
- t_2 : 시스템에 의한 실제적 점화가능시작시점
- t_3 : 점화시작시점
- t_4 : 점화종료시점
- t_5 : 시스템에 의해 허용된 점화종료한계시점
- $t_i^* = t_2 - t_1$: 최소대기시간
- $t_i^{**} = t_5 - t_1$: 최대허용시간
- $f_i = t_4 - t_3$: 점화실행시간

< 그림 5 > TPN 모델에서의 시간조건

편의상 D = < P, T, I, O, τ >로 표시할 수 있다.

필요에 의해 시간함수 $\tau(t_i) = (t_{1i}, t_{2i}, t_{3i}, t_{4i})$는 $\tau(t_i) = (t_{1i}, t_{2i})$(Merlin의 경우) 및 $\tau(t_i) = t_{4i}$(Carlier의 경우) 등으로 제한하여 사용할 수 있다.

(3) Petri Net Based 모델의 문제점

Petri Net을 이용하여 통신 프로토콜을 설계하는데에는 유한상태기계 모델에서 처럼 상태폭발(state explosion)문제가 가장 큰 문제점으로 부각되며 그밖에도 Petri Net의 실행에 따르는 문제와 modular 설계를 지원하기 어려운 문제등이 Petri Net 모델링의 취약점으로 남아 있다.

4.2 DEVS 모델

시뮬레이션 분야에서는 모델을 행위(behavioral) 데이타를 생성해주는 instruction들의 집합으로 보고 있다[ZEIG 84]). 그러한 모델이 컴퓨터 프로그램으로 구현될 때 프로그래밍 언어에 의존성이 생기므로 프로그래밍 언어에 무관하게 형식론(formalism)에 맞춰 모델을 표현한 것을 시스템 명세(system specification)라 하는데 본 논문에서 이것 역시 모델로 간주하여 서술한다.

DEVS (Discrete EVent system Specification) 모델은 연속적인 시간에서 사건이 이산적(discrete)으로 발생하는 모델로서 이산시간 모델(discrete time model)보다 더 현실적인 시스템을 모델링할 수 있는데 주로 이산사건 시뮬레이션을 하기 위한 모델이다.

DEVS 모델(시스템 명세)은 입력집합 X, 출력집합 Y, 상태집합 S, 상태전이함수 δ, 출력함수 λ 및 시간진행함수 t_a로 구성되며 구조적으로 표시하면 다음과 같다[ZEIG 84]. 즉, DEVS 모델 M은 M = < X, Y, S, δ, λ, t_a> 의 구조를 갖으며 M의 각 구성원소들은 다음의 조건을 만족한다.

(1) 입력집합 X
입력변수에 대한 치역(range)들의 곱집합

(2) 출력집합 Y
출력변수에 대한 치역들의 곱집합

(3) 상태집합 S
상태변수에 대한 치역들의 곱집합

(4) 시간진행함수 t_a

$t_a : S \longrightarrow R_o$, (단, R_o는 음수를 제외한 실수집합)

$s \longmapsto t_a(s)$

여기서, $t_a(s)$는 시스템이 외부사건을 입력받지 않는 한 상태 s에 머무를 수 있도록 허용한 시간으로 한다.

(5) 전체상태집합(total state set) Q

전체상태집합이란 M에 의해 모델링(명세)되는 시스템의 상태의 총괄집합으로 다음과 같이 정의된다.

$Q = \{(s, e) \mid s \in S, 0 \leq e \leq t_a(s)\}$

여기서 e는 상태 s에서의 진행시간(elapsed time)을 의미한다.

(6) 상태전이함수 δ

상태전이함수 δ는 시스템 외부에서 내부로 사건(external event)이 입력되었을 때 모델의 상태변화를 설명하는 외부상태전이함수 δ_{ext}과 외부사건이 없을 때 상태변화를 설명하는 내부 상태전이함수 δ_{int}로 나뉜다.

1) 외부 상태전이함수 δ_{ext}

$\delta_{ext} : Q \times X \longrightarrow Q$

$(s,e,x) \longmapsto \delta_{ext}(s,e,x)$

여기서 $\delta_{ext}(s, e, x)$는 외부사건 x가 입력되었을 때 즉시 전이되는 상태를 의미하며 그 상태를 s'라 하면 $\delta_{ext}(s,e,x) = (s',0)$ 처럼 진행시간 e를 0으로 reset한다.

2) 내부 상태전이함수 δ_{int}

$\delta_{int} : Q \longrightarrow Q$

$(s,e) \longmapsto \delta_{int}(s,e)$

여기서 $\delta_{int}(s,e)$는 외부사건이 입력되지 않는 상태 (s,e)에서 $e = t_a(s)$가 되었을 때 전이되는 상태를 의미하며 그 상태를 s'라 하면 $\delta_{int}(s,e)=(s',0)$로 진행시간이 0으로 reset된다.

(7) 출력함수 λ

$\lambda : Q \longrightarrow Y$

$(s,e) \longmapsto \lambda(s,e)$

여기서 $\lambda(s,e)$는 상태 s에 진행시간 e동안 머무르고 있을 때 출력되는 값이다.

4.3 모델의 변환

Timed Petri net D = < P, T, I, O, τ >에 marking μ가 주어졌을 때 marked TPN D = < P, T, I, O, μ, τ >으로 표현할 수 있다. TPN D = < P, T, I, O, μ, τ >에서 $|P| = n$이고 초기 marking을 μ_o라 하면 도달성 집합 $R(D, \mu_o)$을 $\overline{\mu_o}$로 나타낸다. 그러면 $\overline{\mu_o} \subset N^n$이다. 또한 marking $\mu_i \in \overline{\mu_o}$에 대하여 점화가능한 모든 transition들의 집합을 $E(\mu_i)$로 표시한다. 한편, TPN모델의 시간함수 τ(4.1절의 정의 1)를 marking들의 집합인 $\overline{\mu_o}$에 대해서도 다음과 같이 정의할 수 있다.

< 정의 3 > marked TPN D = < P, T, I, O, μ, τ >에 초기 marking μ_o가 주어졌을 때 시간함수 τ를 다음과 같이 확장시켜 정의한다. 단, transition $t_i \in T$에 대해 $\tau(t_i)$는 $\tau(t_i) = t_{4i} = f_i$(Carlier의 점화실행시간)으로 한다(4.1절 정의 1참조).

$\tau : \overline{\mu_o} \longrightarrow R_o$

$\mu_i \longmapsto \tau(\mu_i)$

여기서 $\tau(\mu_i)$는 $\tau(t_k) = \min\{\tau(t_j) \mid t_j \in E(\mu_i)\}$를 만족하는 transition t_k에 대한 $\tau(t_k)$ 값으로 정의한다.

TPN 모델을 DEVS 모델로 변환하기 위해서는 TPN 모델에 입력과 출력 개념을 명시해야 한다. 즉, TPN의 place중에 외부로부터 입력을 받는 place를 p_{in}으로, 외부로 출력을 내보내는 place를 p_{out}로 표시하자. 또한 외부의 입력집합을 X로 표시하되, 입력이 없는 경우(즉, X=ϕ인 경우)를 포함한 입력집합은 X^{ϕ}로 표시하자. TPN의 출력집합은 Y로 표시하되 출력 place p_{out}에 token의 유무만 고려하여 Y={0, 1}로 제한시킨다. 이상과 같이 TPN 모델에 입출력 개념을 도입하면 다음의 정리가 성립한다.

[정리] Marked TPN 모델은 DEVS 모델이다.

[증명] Marked TPN 모델 D = <P, T, I, O, μ_0, τ>가 DEVS 모델임을 보이기 위해서는 D가 DEVS의 모델구조 M = <X,Y,S,δ,λ,t_a>을 갖는다는 것을 보여야 한다.

먼저, TPN 모델의 입력집합 X와 출력집합 Y는 DEVS 모델에서의 입력집합 X와 출력집합 Y이 된다.

한편 TPN에서의 상태는 marking으로 표현되므로, 초기 marking μ_0으로부터의 도달성집합 $R(D,\mu_0) = \overline{\mu_0}$는 TPN 모델의 모든 상태를 표현하게 되고 따라서 DEVS에서의 상태집합 S가 된다. 그리고 marking에 대해 확장된 시간함수 τ(정의 3)를 도입하면 집합 $G = \{ (\mu_i, e_i) \mid \mu_i \in \overline{\mu_0}, 0 \le e_i \le \tau(\mu_i) \}$를 TPN 모델에서 고려할 수 있는데 이 G는 시스템의 상태를 진행시간과 함께 총괄적으로 표현하므로 DEVS 모델에서의 전체상태집합 Q에 대응한다.

TPN 모델에서의 상태 μ_k는 모든 transition $t_j \in E(\mu_k)$ 중에서 점화시간이 가장 작은 transition t_i가 제일 먼저 점화가 종료됨으로써 다음 상태(next state) μ_l로 전이된다. 이것을 함수 δ_{int}로 표시하면, δ_{int}는 G에서 G로의 함수로서 $e_k = \tau(\mu_k)$가 될 때 $\delta_{int}(\mu_k, e_k) = (\mu_l, 0)$로 표현된다. 이 δ_{int}는 DEVS 모델에서의 내부 상태전이함수가 된다. 만일 TPN의 외부로부터 입력 place p_{in}로 x개의 token(외부사건)이 입력될 경우에는 원래의 μ_k에 μ_{in}(p_{in}에만 x개의 token이 존재하고 나머지 모든 place에는 token이 없는 marking)을 더해 준 μ_l로 상태가 전이된다. 이것을 함수 δ_{ext}로 표시하면, δ_{ext}는 G x X 에서 G로의 함수로서 외부사건 x가 발생하였을 때 $\delta(\mu_k, e_k, x) = (\mu_l, 0)$로 표현되며 이 δ_{ext}는 DEVS 모델의 외부 상태전이함수가 된다.

출력 place p_{out}에 token이 도착하면 출력집합의 한 값을 할당받게 되는데 token의 유무에만 관심을 가지고 모델링을 한 경우라면 출력집합을 Y = {0,1} 로 선택할 수 있다. 즉, p_{out}에 token이 하나이상 존재하는 상태(marking)이라면 그 상태에 대한 출력함수값을 1로 할당하고 그렇지 않은 경우 출력함수값을 0으로 할당한다. 이것을 함수λ로 표시하면, λ는 G에서 Y로의 함수로서 $\lambda(\mu_k, e_k) = \begin{cases} 1 & (\mu_k(p_{out}) > 0) \\ 0 & (\mu_k(p_{out}) = 0) \end{cases}$ 로 표현되는데 이 λ는 DEVS 모델에서의 출력함수가 된다. 이상에서 marked TPN D는 DEVS 모델 M_D= <X, Y, $\overline{\mu_0}$, δ, λ, τ> 로 표현된다. ∎

5. 결론

본 논문에서는 먼저 통신 프로토콜의 정의와 기능에 대하여 설명하였고 통신 프로토콜의 종류를 체계적으로 분류, 설명하였다. 또한 기존에 나와 있는 프로토콜 모델들을 분류하였으며 특히 통신 프로토콜의 내적 행위를 상태의 변화로 잘 표현하는 Petri Net 모델

가운데 timeout과 같은 시간적인 제약조건을 첨가한 Timed Petri Net(TPN) 모델에서 시간함수를 정의하고 확장성을 부여하였다.

그런데 Petri Net 모델에서는 상태폭발문제, 실행상의 문제, modular 설계 능력 미비의 문제등이 있어서 분석 능력이 효과를 잃게 될 경우가 있다. 일반적으로 해석적인 분석이 어려운 경우 시뮬레이션 기법을 활용하는데 시뮬레이션 분야에서 가장 이론정립이 체계적인 DEVS(Discrete EVent system Specificatio) 모델링을 본 연구에서 대안으로 삼았다. 그 이유는 우선 DEVS 모델이 Petri Net의 문제점을 해결하며 일단 Petri Net 모델이 DEVS 모델로 변환되면 DEVS 모델링 분야의 좋은 장점을 Petri Net 모델에 적용하는 가능성을 만들기 때문이다.

따라서 본 논문에서는 통신 프로토콜의 모델링과 시뮬레이션을 보다 효율적으로 수행할 수 있도록 TPN 모델이 DEVS 모델이 됨을 정리로서 제시하였는데 이 정리는 TPN 모델링 따른 해석적 분석과 DEVS 모델링에 따른 시뮬레이션적 분석을 가능하게 해준다. 즉, TPN 모델로 설계한 통신 프로토콜을 시뮬레이션 하기 위해 또 다른 모델을 설계할 필요없이 곧바로 DEVS 모델로 변환시킴으로써 시뮬레이션을 효율적으로 수행할 수 있다.

< 참고 문헌 >

[CARL 84] J. Carlier, Ph. Chretienne, and C. Girault, "Modelling Scheduling Problems with Timed Petri Nets", Advances in Petri Nets 1984, Lecture Notes in Computer Science (Eds. G. Goos and J. Hartmanis), Springer-Verlag, 1985, pp. 62-82.

[MERL 74] P. M. Merlin, "A Study of the Recoverability of Computing Systems", Ph. D. dissertation, Dept. of Information and Computer Science, Univ. of California, Irvine, California, 1974.

[MURA 89] Tadao Murata, "Petri Nets:Properties, Analysis and Applications," Proceedings of the IEEE, April, 1989.

[PETE 81] James L. Peterson, Petri Net theory and the modeling of systems, Prentice-Hall Inc., 1981.

[STAL 85] William Stallings, Data and Computer Communications, Macmillan Publishing Company, 1985.

[TANE 81] Andrew S. Tanenbaum, "Network Protocols," ACM Computing Surveys, Vol. 13, No. 4, DEC. 1981.

[ZEIG 84] B. P. Zeigler, Multifacetted Modelling and Discrete Event Simulation, Academic Press Inc., London, 1984.