TCPN을 이용한 실시간 시스템의 스케줄가능성 분석 및 회복기법

김 춘 배† · 박 흥 복††

요 약

TCPN은 시간 요소가 첨가된 petri nets의 확장으로, 실시간 시스템의 분석과 모델링을 위한 형식적인 방법을 제공한다. 실시간 스케줄가능성 분석은 일련의 병행 프로세스들이 제한된 마감시간 내에 종료할 수 있는지를 판단하는 것이다. 본 논문에서는 실시간 시스템에 주어진 시간명세를 TCPN으로 모델링하고, 이러한 TCPN 모델을 기반으로 실시간 시스템이 마감시간 내에 스케줄가능한지를 분석하는 알고리즘을 제안한다. 또한, TCPN으로 모델링된 실시간 시스템이 스케줄가능하지 않을때, 시간명세를 변형하거나 스케줄가능한 최적의 시간으로 시간제한을 완화함으로써 스케줄을 가능하게 하는 회복기법을 제안한다.

A Schedulability Analysis and Recovery Technique of Real-Time System using TCPN

Choon-Bae Kim† · Hung-Bog Park††

ABSTRACT

Timing Constraint Petri Nets(TCPN) extend petri nets by adding timing constraints, it is to provide a formal method for the modeling and analysis of real-time systems. A real-time schedulability analysis decide that a set of concurrent processes will always meet its deadline. In this paper, we present TCPN model for real-time system include timing constraints and propose a schedulability analysis algorithm using them. Also, When the TCPN model is unscheduleable under the imposed timing constraints, we propose a recovery technique that will be scheduleable by the system specification modifying or timing constraints relaxing in the optimized time

1. 서 론

실시간 시스템은 주어진 프로세스가 제한된 마감시간(deadline)을 지켜야 하는 시스템으로, 시스템의 명세 및 설계 과정에서 시간 제약 조건을 반드시 포함하여야 하며, 주어진 마감시간 이내에 정확한 결과를 산출하여야 한다[3, 7]. 또한, 실시간 스케줄가능성 분석은 프로세스 집합이 주어진 스케줄링 정책하에서 마감시간을 만족하는지를 판단하는 것이다[8, 12].

최근 명령 제어 시스템이나 우주선 제어, 비행 항로 제어, 공장 자동화, 프로세스 제어 시스템, 원자력 발전소, 그리고 정밀 로봇 제어 등, 다양하고 복잡한 실시간 시스템들이 생성되고, 보다 동적인 환경을 요구하기 때문에 실시간 시스템을 분석하기 위해서는 더욱 정교한 표현들이 사용되어야 한다[1, 11]. 실시간 시스템에서 응답이 생성되는 시점은 실시간 시스템의 정확

† 준 회 원 : 부경대학교 대학원 전자계산학과
†† 정 회 원 : 부경대학교 전자계산학과 교수
논문접수 : 1998년 12월 29일, 심사완료 : 1999년 9월 17일

성에 대한 중요한 요소 중 하나이다. 따라서, 실시간 시스템에서 실행 시간의 관리는 실시간 시스템에 명세된 정보를 기반으로 정확히 예측되어야 하는 부분 프로세스(partial processes)들의 실행 동작에 의존한다[3, 7].

petri net은 실시간 시스템을 기술하고 실행을 평가하기 위한 많은 표현 형식들을 제공한다[2, 4, 6, 10].그러나 petri net 만으로는 시간적인 특성을 표현할 수 없기 때문에 시간적인 특성을 추가한 petri net의 확장이 필요하다.

TCPN(Timing Constraint Petri Net)은 시간 요소가 첨가된 petri net의 변형된 형태의 하나로 실시간 시스템의 모델링과 분석을 위한 형식적인 방법을 제공한다[1, 6, 9].

본 논문에서는 실시간 시스템에 주어진 명세를 TCPN으로 모델링하고, 이러한 TCPN 모델을 기반으로 스케줄가능성을 분석하는 알고리즘을 제안하고자 한다. 또한, 제안된 스케줄가능성 분석 알고리즘을 통해 마감시간 이내에 스케줄이 가능하지 않으면, 스케줄을 불가능하게 하는 원인이 되는 부분 프로세스를 찾아 스케줄이 가능한 최적의 시간으로 시간명세를 수정함으로써 마감시간 이내에 스케줄이 가능하도록 하는 회복 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 Petri Net과 TCPN의 구성 요소와 특성을 비교·설명하고, 실시간 시스템을 위한 TCPN 명세 모델을 제시한다. 3장에서는 2장에서 제안된 TCPN 명세 모델을 이용하여 스케줄가능성을 분석하고, 스케줄이 불가능한 경우 스케줄이 가능하도록 하는 회복(Recovery) 기법을 제안한다. 4장에서는 제시된 알고리즘을 이용한 실시간 시스템의 적용예에 대한 구현 및 분석을 제시하고, 5장에서는 본 연구의 결론과 향후 연구 과제에 대해 기술한다.

2. Petri Net과 TCPN

복잡한 실세계 시스템을 모델링하고 이를 분석하기 위해 다양한 형태의 petri net은 병행성이 내재된 각종 실시간 시스템의 모델 및 분석에 사용됨으로써 우수한 병행성 모델이 입증되었다[5, 6, 11].

petri net은 플래이스와 트랜지션으로 구성된 이분화(bipartite) 방향 그래프이며, 일반적인 petri net(N)의 정의는 다음과 같다[11].

<정의 1> Petri Net(N)

N = (P, T, F, M_0)

- P : {P_1, P_2, ..., P_m}인 플래이스의 집합
- T : {t_1, t_2, ..., t_n}인 트랜지션의 집합
- F : (P, T) 또는 (T, P) 순서쌍에서 가능한 아크(arc)의 집합
- M_0 : 초기 마킹(marking)

이러한 petri net의 기본 구조에 다양한 구조적 제약조건과 수행규칙을 첨가함으로써 TPN(Timed Petri Net)[6], OCPN(Object Composition Petri Net)[10], SPN(Stochastic Petri Net)[4] 등에 응용되고 있다.

TCPN(Timing Constraint Petri Net)은 플래이스와 트랜지션에 최대·최소 시간제한쌍과 시간간격(duration)을 첨가한 petri net의 확장으로 6개의 튜플(tuple)로 정의된다[11, 12].

<정의 2> TCPN(Timing Constraint Petri Net)

N = (P, T, F, C, D, M)

- P : {P_1, P_2, ..., P_m}인 플래이스의 집합
- T : {t_1, t_2, ..., t_n}인 트랜지션의 집합
- F : (P, T) 또는 (T, P) 순서쌍에서 가능한 아크의 집합
- C : ($TCmin(pt_j)$, $TCmax(pt_j)$)의 정수쌍의 집합
  여기서, pt_j는 TCPN 내 j번째 플래이스 또는 트랜지션
- D : [$FIREdur(pt_j)$]인 점화 시간간격의 집합
- M : m-vector를 가진 마킹(marking)
  {$M(p_1)$, ..., $M(p_j)$, ..., $M(p_m)$)}의 집합
  여기서, $M(p_j)$는 플래이스 내 토큰의 수이고, M_0는 초기 마킹

이러한 TCPN의 정의를 기반으로 토큰에 대한 각 트랜지션의 활성화(enabling)와 점화 규칙(fire-rule)은 다음과 같다.

$TCmin(pt_j)$과 $TCmax(pt_j)$는 TCPN 내의 임의의 플래이스 또는 트랜지션에 토큰이 전달된 후 수행을 시작하기 위한 최소 시간제한(Timing Constraint Min)과 수행을 종료해야만 하는 최대 시간제한(Timing Constraint Max)을 의미한다. 이는 ($TCmin(t_j)$, $TCmax(t_j)$) 시간쌍을 가진 임의의 트랜지션은 $TCmax(t_j)$ - $TCmin(t_j)$ 시간 동안 점화가능하게 된다.

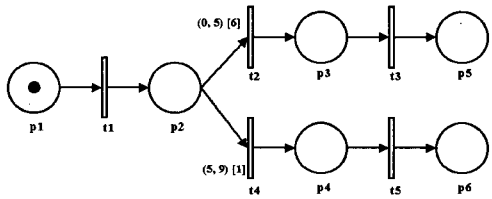
트랜지션 t_j를 활성화하기 위해 사용된 모든 토큰들은 t_j의 점화가능 시간 동안 플래이스에 유지되고, 토큰들은 t_j가 점화를 완료하는데 실패하면, 다른 트랜지션을 활성화하기 위해 사용된다. 또한, 시간 T_0에 활성화된 트랜지션 t_j의 점화는 $T_0 + TCmin(t_j)$ 시간에서부터 $T_0 + TCmax(t_j)$ 시간까지 가능하다. 이때, $TCmax(t_j) \geq TCmin(t_j)$이어야 한다. 그렇지만, 트랜지션 t_j는 주어진 시간간격 $FIREdur(t_j)$를 가지기 때문에 t_j의 점화가 $T_0 + TCmin(t_j)$ 시간에서 $T_0 + TCmax(t_j)$ 시간까지 성공적으로 완료된다는 것을 보장할 수 없다. 따라서, 트랜지션 t_j가 성공적으로 점화를 보장받기 위한 시간간격은 $TCmax(t_j)-TCmin(t_j) \geq FIREdur(t_j)$의 조건을 만족하여야 한다.

TCPN 내의 모든 플래이스와 트랜지션 각각은 시간제한쌍 ($TCmin(pt_j)$, $TCmax(pt_j)$)과 시간간격 [$FIREdur(pt_j)$]을 가질 수 있다. 플래이스와 트랜지션에 대해 명확한 시간제한쌍이 주어지지 않을 경우 트랜지션은 특정한 점화 시간을 가지지 않고 토큰이 전달되는 임의의 시간에 점화가능한 $(0, \infty)$의 시간쌍을 의미하고, 마찬가지로 명확하게 주어지지 않은 시간간격의 디폴트 값은 시간간격 [0]을 의미한다. 이러한 디폴트 값들은 TCPN의 표현에서 생략할 수 있다.

2.1 도달가능성과 스케줄가능성

TCPN에서 임의의 트랜지션 $TCmax(t_j) - TCmin(t_j) \geq FIREdur(t_j)$의 조건을 만족할 때, 트랜지션은 스케줄가능하고, TCPN 내의 임의의 마킹 M_n이 초기 마킹 M_0에서 M_n으로 변형되는 점화 연속 $\sigma = (M_0, t_1, M_1, \ldots, t_j, M_j, \ldots, t_n, M_n)$이 존재하면 마킹 M_n은 petri net 내에서 도달가능하다고 한다.

TCPN 모델에서 시간제한을 고려할 때, 마킹 M_n은 σ 내의 모든 트랜지션에 대해, 각 트랜지션에 주어진 시간제한이 $TCmax(t_j) - TCmin(t_j) \geq FIREdur(t_j)$의 조건을 만족할 때 스케줄가능하다고 한다. 그러나 σ 내의 트랜지션들은 TCPN의 주어진 명세에 따라 점화가 가능하거나 스케줄이 가능하지 않을 수 있다. 이는 TCPN에 부과된 시간제한 $TCmax(t_j) \geq TCmin(t_j)$과 $TCmax(t_j) - TCmin(t_j) \geq FIREdur(t_j)$의 특성에 의존하기 때문이다. 만약 σ 내의 적어도 하나의 트랜지션이 도달가능하지 않거나 점화가능하지 않다면 M_n은 스케줄가능하지 않다.



(그림 1) 스케줄가능과 불가능한 트랜지션

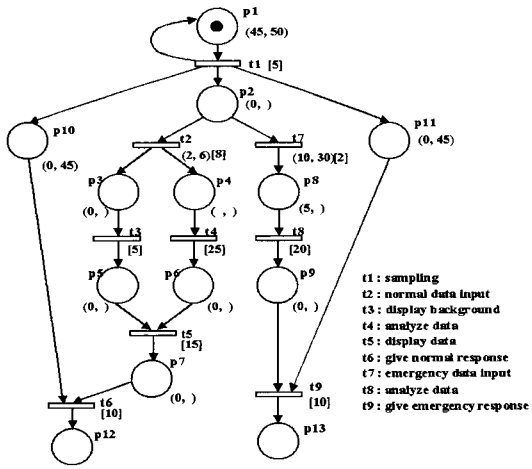
(그림 1)은 TCPN 내에서 스케줄가능한 트랜지션과 스케줄 불가능한 트랜지션의 예를 보여준다. (그림 1)에서 초기 마킹을 $M_0 = p_1$으로 가정하면, 마킹 $M_3 = p_5$와 $M_5 = p_6$을 추론할 수 있다. M_3와 M_5는 각각 $\sigma_1 = (t_1\ t_2\ t_3)$와 $\sigma_2 = (t_1\ t_4\ t_5)$인 점화 연속 후에 도달가능하다. σ_1 내의 t_2는 지연시간 [6]이 최대·최소 시간제한쌍 사이의 차 (5 - 0 = 5) 보다 크기 때문에 스케줄가능하지 않고, σ_2 내의 t_4는 지연시간 1이 최대·최소 시간제한쌍 사이의 차 (9 - 5 = 4) 보다 작기 때문에 스케줄가능하다. 결과적으로 M_3은 스케줄가능하지 않고, M_5는 시간제한의 고려 하에서도 스케줄가능하다고 판단할 수 있다.

2.2 실시간 시스템 명세와 TCPN 모델

TCPN을 이용한 스케줄가능성 분석은 세 단계로 구분할 수 있다[9, 12]. 먼저, 실시간 시스템을 구성하는 부분 프로세스에 부과된 시간 특성을 TCPN에 적용하는 모델링 단계로, 이때 생성된 TCPN을 N_s라 한다. 실시간 시스템을 TCPN으로 모델링하기 위해 초기 마킹 M_0는 시스템이 외부의 장치로부터 데이터를 샘플하는 상태로, 최종 마킹 M_n은 시스템이 장치를 통해 샘플된 데이터의 처리결과에 대한 응답을 사용자에게 제공하는 상태이다. 그리고 TCPN 모델에서 시스템 명세가 유효해야 하기 때문에 M_n은 M_0로부터 도달가능하다고 가정한다. 도달가능한 M_n에 대해 점화 연속 내의 모든 트랜지션이 스케줄가능하면 초기 마킹 M_0에 대해 M_n은 스케줄가능하게 된다. 다음으로 N_s의 초기 마킹 M_0에서 응답을 제공하는 최종 마킹 M_n까지 도달가능한지를 검증하는 도달가능성 시뮬레이션 단계이다. 마지막으로 도달가능성 시뮬레이션에서 발견되는 도달가능한 마킹 M_n이 N_s에 부과된 시간제한의 고려하에서 스케줄가능한지를 검증하는 시간 분석 단계이다. 이는 petri net에서 도달가능한 마킹이 TCPN에서 부과된 시간제한으로 인해 반드시 도달가능하지는 않

기 때문이다. 또한, 마킹 M_n이 도달가능하기 위해서는 초기 마킹 M_0에서부터 M_n으로 변형되는 $\sigma = (M_0\ t_1\ M_1 \ldots t_i\ M_i \ldots t_n\ M_n)$이나 $(t_1 \ldots t_i \ldots t_n)$인 점화 연속이 존재하여야 하고, 이는 σ 내의 모든 트랜지션이 M_0에 대해 스케줄가능하다는 것을 의미한다. 점화 연속 내에 M_n이 발견되면 주어진 시스템 명세는 부과된 시간제한하에서 스케줄가능하다고 주장할 수 있다. 만약, 스케줄가능하지 않으면 시스템의 명세를 변형하거나, 부가된 시간제한을 완화함으로써 스케줄가능하도록 변형하여야만 한다.

(그림 2)에서 TCPN으로 모델링된 실시간 시스템을 구성하는 각 프로세스에 대한 명세는 다음과 같다. 시스템은 매 50 단위시간마다 외부 환경으로부터 데이터를 샘플하고, 샘플링을 완료한 후 45 단위시간 내에 정상 또는 긴급 응답을 제공하여야 한다. 응답이 제공된 후 데이터를 샘플링하는데 소요되는 프로세스의 처리시간은 5 단위시간을 가진다.



(그림 2) 실시간 시스템을 위한 TCPN 모델

샘플된 데이터를 얻으면, 시스템은 먼저 정상 데이터 입력(Normal Data Input : NDI) 프로시저로 진행하고, NDI가 실패하면 긴급 데이터 입력(Emergency Data Input : EDI) 프로시저로 진행한다. 샘플된 데이터를 처리하기 위한 준비 시간으로 NDI는 2 단위시간 동안 대기하고, 샘플링 완료 후 6 단위시간 내에 샘플된 데이터의 처리를 완료해야만 한다. NDI 프로세스는 8 단위시간의 시간간격을 필요로 한다. EDI는 정상 데이터 입력 프로시저의 수행이 가능한 시간인 10 단위시간 동안 대기하고, 샘플링의 완료 후 30 단위시간 내에 긴급 응답의 시작을 완료하여야만 한다.

긴급 데이터 입력 프로시저인 EDI 프로세스에 필요한 시간간격은 2 단위시간이다. 시스템이 정상 데이터 입력 프로시저를 실행하면, 시스템은 정상 응답을 제공하기 전에 스크린에 샘플된 데이터를 보여주는 과정을 수행하여야 한다.

데이터 디스플레이는 분석된 데이터를 화면에 보여주는 동작을 실행하고, 배경 디스플레이와 데이터 분석이 완료될 때까지 대기하여야만 한다. 데이터 디스플레이 프로세스는 15 단위시간의 시간간격을 필요로 한다. 배경 디스플레이는 NDI가 완료되는 즉시 시작할 수 있고, 배경 디스플레이 프로세스에 필요한 시간간격은 5 단위시간이다. 데이터 전송 지연으로 인해, 데이터 분석 동작은 NDI의 완료 후 5 단위시간을 대기하여야 한다. 데이터 분석 프로세스는 25 단위시간의 시간간격을 필요로 한다. 긴급 데이터 입력 프로시저를 실행하면, 시스템은 단지 데이터 분석과 긴급 응답 제공을 필요로 한다(데이터의 디스플레이는 필요하지 않다). 긴급 응답 프로세스는 10 단위시간의 시간간격을 필요로 한다. 데이터 전송 지연으로 인해 데이터 분석 동작은 EDI의 완료 후 5 단위시간을 대기하여야 한다. 데이터를 분석하는 프로세스는 20 단위시간의 시간간격을 가진다.

3. 실시간 스케줄가능성 분석

본 논문에서는 실시간 시스템의 표현에 TCPN 모델을 이용하고, 이 모델을 기반으로 스케줄가능성을 분석함으로써 petri net을 이용한 실시간 시스템의 표현과 분석 모두를 제공하는 체계적인 분석 알고리즘을 제안하고, 만약 스케줄이 불가능한 부분 프로세스가 존재할 경우 스케줄이 가능한 최적의 시간으로 시간명세를 완화함으로써 스케줄을 가능하게 하는 회복 기법을 제시하고자 한다.

본 논문에서 TCPN 모델을 이용한 스케줄가능성 분석은 크게 두 단계로 구분할 수 있다. 첫 번째 단계는 각 트랜지션의 입력 플레이스들에 토큰이 도착하는 시간을 고려하지 않은 상태에서 도달가능한지를 검증하는 단계와 각 트랜지션에 주어진 시간간격 동안 점화 가능한지를 검증하는 단계로 나뉘어진다. 이는 TCPN

에서는 각 트랜지션에 주어진 시간지연 조건으로 인해 트랜지션이 도달가능하다고 해서 반드시 점화가능하다는 것을 보장받을 수 없기 때문이다.

두 번째 단계는 각 플레이스에 대해 토큰의 도착 시간을 고려한 상태에서 트랜지션이 점화가능한지를 검증하는 단계로, TCPN 모델 내의 가능한 작업 경로 상에 존재하는 모든 플레이스에 토큰이 도착하는 상한과 하한을 찾는 과정이다. 이러한 세부적인 단계를 거쳐 최종적으로 최초 작업이 수행되는 시작 시간으로부터 모든 가능한 작업 경로의 트랜지션들이 주어진 시간제한쌍과 시간간격 동안 점화가능을 나타냄으로써 전체 프로세스가 마감시간 내에 스케줄가능함을 검증하게 된다.

3.1 점화가능성 분석 알고리즘

도달가능성은 TCPN의 가능한 작업 경로 상의 모든 트랜지션들이 토큰의 도착시간을 고려하지 않은 상태에서 모든 부분 프로세스에 도달가능함을 보여 주는 단계로, 각 트랜지션의 최소·최대 시간제한쌍만을 가지고 분석이 가능하다. TCPN의 가능한 작업 경로 상에 임의의 트랜지션 t_j가 각 입력 플레이스에 토큰의 도착시간을 고려하지 않은 상태에서 적어도 하나의 토큰을 가질 때, 이 트랜지션은 도달가능하다.

점화가능성 역시 TCPN의 가능한 작업 경로 상의 모든 트랜지션들이 토큰의 도착시간을 고려하지 않은 상태에서 점화가능함을 보여 주는 단계로, 이를 분석하기 위해서는 각 트랜지션의 최소·최대 시간제한쌍과 각 트랜지션의 시간간격을 필요로 한다. TCPN의 가능한 작업 경로 상의 모든 트랜지션 t_j가 점화가능하기 위해서는 모든 트랜지션 t_j가 도달가능해야만 하고, 각 입력 플레이스의 토큰 도착시간을 고려하지 않은 상태에서 성공적으로 점화되어야 한다. 이는 트랜지션 t_j에 부과된 시간간격의 조건으로 인해 t_j에 토큰이 도착한다고 해서 성공적으로 점화된다는 것을 보장받을 수 없기 때문이다.

결국, 점화가능성은 도달가능의 조건에 각 트랜지션에 부과된 시간간격을 고려하기 때문에 도달가능성 분석은 점화가능성 분석에 대한 하나의 단계로 포함시킬 수 있고, 점화가능성 분석 단계에서 각 트랜지션에 부과된 시간간격의 조건을 달리 함으로써 두 가지 분석이 동시에 가능하게 된다. 가능한 작업 경로 상의 모든 트랜지션이 토큰의 도착시간을 고려하지 않은 상태에서 점화가능함을 증명하기 위해 각 경로 상에 존재하는 임의의 트랜지션 t_j가 활성화된 후 t_j가 점화를 시작하기 위한 최소시간(Earlist Time to Begin Firing : ETBF)과 t_j가 점화를 종료하기 위한 최대시간(Latest Time to End Firing : LTEF)을 계산해야 하고, 여기서 계산된 최대·최소 시간쌍은 트랜지션 t_j에 부과된 시간간격에 대한 조건을 만족하여야 한다.

각 트랜지션이 점화되는 최소시간과 최대시간은 다음과 같이 계산된다[12].

$$ETBF(t_j) = \max\{TCmin(p_j)\} + TCmin(t_j)$$
$$LTEF(t_j) = \min\{TCmax(p_j), TCmax(t_j)\}$$

모든 트랜지션이 주어진 시간제한쌍에 대해 도달가능한 조건, $TCmax(t_j) > TCmin(t_j)$,과 점화가능한 조건, $LTEF(t_j) - ETBF(t_j) \geq FIREdur(t_j)$,을 분석하는 방법은 알고리즘 1과 같다.

```
입력 : 경로 리스트의 포인터
출력 : 점화가능하지 않은 트랜지션 번호와 시간조건
pl : 플레이스 리스트의 시작 포인터
tl : 트랜지션 리스트의 시작 포인터
headpath[n] : 경로 리스트의 헤더 포인터
1 t_no : TList의 첫번째 node
2 while t_no ≠ null do
3    LTEF = t_no의 Max 값;
4    temp = t_no의 Min 값;
5    dur = t_no의 Dur 값;
6    for (모든 path에 대해)
7       if Exist_Path(t_no) then
8          p_no = 경로에서 t_no 이전의 노드
9          min = p_no의 Min 값;
10         if (min >= ETBF) then ETBF = min;
11         max = p_no의 Max 값;
12         if(max <= LTEF) then LTEF = max;
13      else for loop의 다음 경로에 대해 수행
14      end if
15   end for
16   ETBF = ETBF + 현재 t_no의 Min 값;
17   if LTEF - ETBF < FIREdur then
18      print(t_no, time_constraint);
19      Recovery_Path(최적값);
20      Recovery_TList(최적값);
21      return 1;
22   end if;
23   ETBF와 LTEF를 초기화;
24   t_no = TList에서 다음 트랜지션을 설정;
25 end while
26 return 0;
```

(그림 3) 알고리즘 1 : 점화가능성 분석 알고리즘

초기 마킹 M_0에서부터 최종 마킹 M_n까지는 여러 경로가 존재 가능하고, 임의의 트랜지션은 하나 이상의 경로에 포함될 수 있기 때문에, 각 트랜지션이 존재하

는 모든 경로를 고려해야 한다. 예를 들어, (그림 2)의 TCPN 모델에서 트랜지션 t_5는 하나 이상의 경로에 포함되어 있으므로 플래이스 p_5와 p_6에 의해 활성화될 수 있다. 그러므로, 트랜지션 t_5의 ETBF는 서로 다른 경로 상에 존재하는 두 입력 플래이스 p_5와 p_6의 TCmin 값 중 최대값에 t_5의 최소값을 더함으로써 결정된다. 임의의 트랜지션이 점화가능하기 위한 조건은 그 트랜지션이 도달가능한 조건을 포함한다. 따라서, 점화가능한 트랜지션은 분명히 도달가능하다. 그러나 트랜지션이 도달가능하다고 해서 점화가능한 것은 아니다.

모든 트랜지션에 대해 점화가능해야 하므로 프로그램의 시작 단계에서 트랜지션 리스트에서 첫 번째 트랜지션을 알고리즘 1행에서 얻은 다음, 각 트랜지션이 포함된 모든 경로에 대해 트랜지션의 ETBF와 LTEF를 계산하여야 한다. 2~25행은 트랜지션 리스트에 존재하는 모든 트랜지션에 대해 ETBF와 LTEF를 결정하고, 모든 트랜지션들이 점화가능한 조건을 만족하면 26행에 의해 모든 트랜지션이 점화가능함을 알리는 상태값 "0"을 반환하게 된다. 6~15행은 t_5의 경우처럼 트랜지션이 하나 이상의 경로에 포함되었을 때 포함된 모든 경로에 대해 ETBF와 LTEF를 결정하는 과정을 보여준다. 16행에서는 트랜지션이 하나 이상의 경로에 포함되었을 때, 10행에서 최종적으로 결정된 트랜지션 t_5의 ETBF에 t_5의 최소 시간값을 더함으로써 트랜지션 t_5에 대한 ETBF의 값을 결정하게 된다. 또한 LTEF(t_5)의 계산은 12행에 의해 결정된다. LTEF도 ETBF와 동일한 방법으로 트랜지션이 포함된 모든 경로에 대해 고려되어야 한다. LTEF는 트랜지션 t_5와 그의 입력 플래이스 TCmax의 순서쌍들 중 가장 적은 값을 LTEF로 결정한다. 여기서 t_5와 입력 플래이스의 TCmax 순서쌍은 {(∞, ∞),(∞, ∞)}이므로 결국 가장 늦게 점화되는 시간은 토큰이 도착하는 임의의 시간임을 알 수 있다. 17행은 임의의 트랜지션에 t_j에 대한 LTEF와 ETBF의 결정된 값을 이용해 LTEF - ETBF < FIREdur의 조건을 비교함으로써 트랜지션 t_j가 점화가능한지를 판단한다.

만약, 트랜지션이 주어진 시간 특성에 대해 점화가능하지 않은 경우 18~20행에서 점화가능하도록 트랜지션의 초기 명세와 트랜지션이 포함된 모든 경로에 대한 시간명세를 수정하는 루틴을 수행하고, 21행에서 주어진 시간명세에 대해 점화가능하지 않은 트랜지션이 존재함을 알리는 상태값 "1"을 반환함으로써 수정된 시간명세를 기반으로 전체 트랜지션에 대한 점화가능성 분석을 재수행하게 된다. 트랜지션이 도달가능한 경우 트랜지션 리스트에서 다음 트랜지션을 구하고, 동일한 과정을 반복한 후 모든 트랜지션이 주어진 시간 특성에 대해 도달가능한 경우 26행에서 상태값 "0"을 반환함으로써 다음 분석 단계를 수행하도록 한다.

3.1.1 점화가능성 회복 기법

TCPN으로 모델된 실시간 시스템에서 임의의 트랜지션 t_j가 주어진 시간명세에 대해 점화가능하지 않을 경우, 트랜지션에 부가된 시간제한을 완화함으로써 점화가능하도록 할 수 있다. 트랜지션에 부과된 시간명세의 유형에 따라 점화가능하도록 시간을 완화하기 위한 세 가지 시간 완화 방법을 적용할 수 있다. 첫 번째 방법은 트랜지션에 부과된 최소시간을 줄임으로써 점화가능하도록 시간제한을 완화하는 방법이다. 이 방법은 최소시간이 '0'으로 부여된 시간은 완화 규칙에 의해 음의 시간값을 나타낼 수 있기 때문에 시간제한의 의미를 상실하게 된다. 또한, 각 부분 프로세스의 부과된 작업 최소시간은 부분 프로세스를 수행하기 위한 준비시간이나 데이터를 대기하는 대기시간으로 이루어지기 때문에 최소시간을 줄이는 것은 프로세스의 준비나 대기시간을 줄임으로써 올바른 입력 결과를 얻지 못하는 경우가 발생할 수 있다. 두 번째로 트랜지션에 부과된 시간을 완화하는 방법은 트랜지션의 최대값을 완화함으로써 점화가능하도록 하는 것이다.

마지막으로 트랜지션의 시간을 완화하는 방법은 트랜지션에 주어진 시간간격을 줄이는 것이다. 이는 첫 번째 경우와 마찬가지로 부분 프로세스에서 처리되는 작업 수행시간을 줄임으로써 점화가능하도록 시간을 완화하는 방법인데, 이 방법 역시 각 부분 프로세스에서 처리되어야 하는 작업 처리시간은 일정하기 때문에 작업 처리시간의 단축은 첫 번째 시간 완화 방법에서와 마찬가지로 잘못된 결과를 초래하는 원인이 된다.

본 논문에서는 부분 프로세스가 점화 불가능한 원인이 될 때, 플래이스와 트랜지션에 부과된 최대시간을 완화하는 방법을 적용한다. 이는 플래이스와 트랜지션에 부과된 최대시간을 점화가능한 최적의 값으로 시간제한을 완화하여 점화 불가능한 트랜지션이 점화가능하도록 한다. 토큰의 도착시간을 고려하지 않은 상태에서 점화 불가능한 트랜지션이 발견되었을 때 점화가능하도록 시간을 완화하는 알고리즘 17~21행에 나타낸다.

(그림 2)의 실시간 시스템의 명세 모델에서 트랜지션 t_2에 대해 계산된 LTEF(t_2)와 ETBF(t_2)의 시간값은

$$LTEF(t_2) - ETBF(t_2) = 6 - 2 = 4$$

로, 이 값은 t_2에 부과된 시간간격 FIREdur(t_2)의 값인 8보다 작기 때문에, 토큰의 도착시간을 고려하지 않은 상태에서 트랜지션 t_2는 점화가능하지 않다. 따라서, 트랜지션 t_2가 점화가능하도록 시간제한을 완화하여야 한다. 트랜지션 t_2가 주어진 시간명세에 대해 점화가능하기 위한 시간 조건

$$LTEF(t_2) - ETBF(t_2) \geq FIREdur(t_2)$$

을 만족하기 위해서는 최소한 LTEF(t_2) - ETBF(t_2)의 값이 8이 되어야 한다.

본 논문에서는 이러한 시간값을 임의의 트랜지션이 점화 불가능할 때, 점화가능하도록 하는 최적의 시간 조건으로 평가한다. 따라서 트랜지션 t_2가 점화가능하도록 하기 위해서는 TCmax(t_2)를 6에서 8로 완화시켜야 한다. 경로에 대해 점화 불가능한 트랜지션이 존재하므로 19~20행에서 점화가능한 최적의 시간으로 시간명세를 변형한 후 점화 불가능한 트랜지션이 존재하는 상태값 "1"을 반환하여 변형된 시간명세를 기준으로 모든 트랜지션에 대한 점화가능성 분석을 재수행하게 된다.

3.2 스케줄가능성 분석 알고리즘

TCPN의 모든 트랜지션이 도달가능하고 주어진 조건하에서 점화가능하다고 판단되면, 전체 프로세스가 마감시간 이내에 스케줄가능한지를 분석하는 단계를 수행할 수 있다. 만약 입력 플레이스 중 하나가 활성화를 중지하면 트랜지션 t_j는 점화를 중지하게 된다. 따라서, 트랜지션 t_j가 스케줄가능한지를 결정하기 위해 t_j의 모든 입력 플레이스들에 토큰이 도착하는 시간을 고려해야만 한다. TCmin(p_j)와 TCmax(p_j)가 고정되어 있기 때문에 ETBF(t_j)와 LTEF(t_j)를 찾기 위해 각 플레이스에 토큰이 도착하는 시간의 상한과 하한을 결정하여야 한다. 플레이스에서 트랜지션까지 토큰의 이동시간은 필요하지 않다는 가정에서 임의의 플레이스에 토큰이 도착하는 시간은 플레이스의 입력 트랜지션이 점화를 종료하는 시간과 동일한 시간에 토큰이 도착하게 된다. 따라서, 초기 트랜지션 t_1의 점화 종료시간에서부터 추론된 각 플레이스의 입력 트랜지션이 점화를 종료하는 시간의 한계를 찾음으로써 플레이스에 토큰이 도착하는 시간의 한계를 결정할 수 있다. 임의의 트랜지션 t_j에 대한 ETBF와 LTEF는 다음과 같이 계산된다[12].

$$ETBF(t_j) = \max\{FIREend(t_1) + \sum TCmin(pt_{mk}) + \sum FIREdur(t_{nk})\}$$

$$LTEF(t_j) = \min\{(FIREend(t_1) + \sum TCmax(pt_{mk}))\}$$

TCPN 모델에서 토큰의 도착시간 고려하에서 작업가능한 경로의 모든 트랜지션이 점화가능하고 트랜지션에 부과된 FIREdur(t_j)에 대해 성공적으로 점화가능하면, 트랜지션 t_j는 초기 마킹 M_0에 대해 스케줄가능한 트랜지션이다. 트랜지션 t_j가 주어진 시간간격에 대해 성공적으로 점화가능하기 위해서는 계산된 (LTEF(t_j), ETBF(t_j))의 쌍과 부과된 시간간격 FIREdur(t_j)에 대해 다음의 조건을 만족하여야 한다.

$$LTEF(t_j) - ETBF(t_j) \geq FIREdur(t_j)$$

TCPN 모델을 기반으로 여러 경로 상에 존재하는 트랜지션이 토큰의 도착시간을 고려한 상태에서 스케줄가능한지를 분석하는 방법은 알고리즘 2와 같다.

```
입력 : 경로리스트 포인터 및 첫 번째 트랜지션의 점화 종료시간
출력 : 스케줄 불가능한 트랜지션 번호와 시간조건
1 first_tno : TList의 첫번째 노드;
2 t_no : TList의 두 번째 노드;
3 while t_no ≠ null do
4    dur = 현재 t_no의 Dur 값;
5    for (모든 path에 대해)
6       end_node = 현재의 t_no;
7       if Exist_Path(t_no) then
8          start_node = t_no가 포함된 path의 시작 노드;
9          last1_node = t_no의 이전 노드;
10         last2_node = last1_node의 이전 노드;
11         sum_min = start_node에서 last1_node까지
                     각 node Min값의 합을 계산;
12         sum_dur = start_node에서 last1_node까지
                     각 t_no Dur값의 합을 계산
13         if start_node와 last2_node가 같으면
14            then sum_max = 0;
15            else sum_max = start_node에서 last2_node까지
                          Max값의 합을 계산;
16         end if
17         min = fire_end + sum_max +
                Min(현재 t_no의 Max값, last1_node의 Max값);
18         if(min < LTEF) LTEF = min;
```

```
19          max = fire_end + sum_min + sum_dur;
20          if(max > ETBF) ETBF = max;
21        end if
22    end for
23    if LTEF - ETBF < FIREdur then
24      print(t_no, time_constraint);
25      Recovery_Path(최적값);
26      Recovery_TList(최적값);
27      return 1;
28    end if;
29    t_no = TList의 다음 트랜지션을 설정;
30 end while
31 return 0;
```

(그림 4) 알고리즘 2 : 실시간 스케줄가능성 분석 알고리즘

알고리즘 2는 토큰의 도착시간을 고려한 상태에서 모든 트랜지션들이 부과된 시간간격 내에 스케줄가능한지를 분석하고, 분석 과정에서 스케줄을 불가능하게 하는 트랜지션이 존재하는 경우 그 트랜지션이 시간 고려 하에서 스케줄이 가능하도록 주어진 시간 특성을 변형하는 알고리즘이다. 초기 마킹 M_0에서부터 최종 마킹 M_n까지의 여러 경로가 존재 가능하고, 임의의 트랜지션은 하나 이상의 경로에 포함될 수 있기 때문에, 각 트랜지션이 존재하는 모든 경로를 검색하여야 한다. 3~30행은 모든 트랜지션이 존재하는 경로에 대해 LTEF와 ETBF를 계산하고 스케줄가능한지를 분석한다.

5~22행은 트랜지션이 존재하는 모든 경로에 대해 LTEF와 ETBF를 결정하는 단계를 보여준다. 초기 트랜지션 t_1의 점화 종료 시간을 기준으로 경로에 존재하는 모든 트랜지션에 대한 ETBF와 LTEF를 결정하기 위해 경로 상에 존재하는 트랜지션의 이전 두 노드를 필요로 하고 있다. 7행에서는 트랜지션이 경로 상에 존재하는 경우 트랜지션의 두 이전 노드 last1과 last2를 9행과 10행에서 설정하고 있다. 11행에서는 start_node와 last1_node 사이에 존재하는 모든 노드의 TCmin(pk_j) 값을 계산하고, 12행에서는 start_node와 last1_node 사이에 존재하는 트랜지션들의 시간간격에 대한 합을 계산하고 있다. 17~18행은 하나의 트랜지션이 둘 이상의 경로에 포함될 때 LTEF를 결정하기 위한 과정을 수행한다. 13~16행은 트랜지션이 존재하는 경로가 짧을 경우, 두 이전 노드가 존재하지 않을 때 sum_max 값을 처리하기 위한 계산 과정이다. 또한, 임의의 트랜지션에 대한 ETBF의 최종 결정은 19~20행에서 결정된다. 임의의 트랜지션이 토큰의 도착시간 고려하에서 스케줄가능하지 않을 경우 스케줄가능하도록 복구하기 위한 회복루틴 23~27행을 수행하고 스케줄이 불가능한 트랜지션이 존재함을 알리는 상태값 "1"을 반환함으로써 회복된 트랜지션의 시간 속성 값을 기준으로 모든 트랜지션에 대해 스케줄가능성 분석을 재수행하게 된다. 모든 트랜지션이 스케줄가능하게 되면 모든 트랜지션이 스케줄가능함을 알리는 상태값 "0"을 반환하여 다음 과정을 수행하게 한다.

3.3 실시간 스케줄가능성 회복 기법

본 논문에서는 부분 프로세스가 스케줄을 불가능하게 하는 원인이 될 때 플래이스와 트랜지션에 부과된 최대 시간을 완화하는 방법을 선택한다. 플래이스와 트랜지션에 부과된 최대시간을 완화함으로써 스케줄을 가능하게 하는 방법을 적용하여 스케줄가능한 최적의 상한값으로 시간제한을 완화하여 스케줄이 불가능한 트랜지션을 스케줄가능하도록 한다. 토큰의 도착 시간을 고려한 상태에서 스케줄 불가능한 트랜지션이 발견되었을 때 스케줄가능하도록 시간을 완화하는 알고리즘의 부분은 23~27행에 나타낸다. 스케줄가능성 분석은 점화가능성 분석과는 달리 첫 번째 플래이스에 토큰이 도달하여 점화를 종료하는 시간을 필요로 한다. 따라서, 트랜지션 t_1이 시간 T_0에 점화를 종료한다고 가정할 때, 트랜지션 t_6과 t_9는 각각 다음과 같기 때문에 스케줄을 불가능하게 하는 원인이 된다.

$$\begin{aligned} LTEF(t_6) - ETBF(t_6) &= (T_0 + 45) - (T_0 + 53) \\ &= 45 - 55 < 0 \\ LTEF(t_9) - ETBF(t_9) &= (T_0 + 45) - (T_0 + 37) \\ &= 8 < 10 \end{aligned}$$

트랜지션 t_6은 $\delta_1(M_6) = (p_2\ t_2\ p_3\ t_3\ p_5\ t_5\ p_7\ t_6)$, $\delta_2(M_6) = (p_2\ t_2\ p_4\ t_4\ p_6\ t_5\ p_7\ t_6)$ 그리고 $\delta_3(M_6) = (p_{10}\ t_6)$의 세 경로에 포함되어 있고, t_6이 토큰의 도착시간의 고려하에서 스케줄가능하기 위해서는 두 입력 플래이스 p_{10}과 p_7에 의해 동시에 활성화되어야 한다. 트랜지션 t_5가 스케줄가능하기 때문에 t_6이 스케줄가능하지 않은 것은 t_6 이전의 입력 플래이스 p_7과 p_{11}에 그 원인이 존재한다. 플래이스 p_7에 주어진 시간명세 $(0, \infty)$에 의하면 토큰이 도착하는 임의의 시간에 활성화되기 때문에 트랜지션 t_6이 스케줄가능하지 않은 원인은 t_6의 또 다른 이전 플래이스 p_{10}의 최대값에 원인이 있음을 알 수

있다. 따라서 $LTEF(t_6) - ETBF(t_6) \geq FIREdur(t_6)$을 만족하는 최적값인 $(LTEF(t_6) - ETBF(t_6) = -10 \geq FIREdur(t_6) = 10)$을 만족하도록 p_{10}의 최대값을 65로 시간명세를 완화하고 스케줄 불가능한 트랜지션이 존재하는 상태값 "1"을 반환함으로써 회복된 시간명세를 기반으로 스케줄가능성 분석 알고리즘을 재수행하게 된다.

다음으로 스케줄이 불가능한 트랜지션으로 t_9를 찾게 된다. t9는 두 경로의 노드로 포함되고 t_8이 스케줄 가능하기 때문에 t_9가 스케줄 불가능한 원인은 두 입력 플래이스 p_9와 p_{11}에 있을 수 있다. t_9 이전의 입력 플래이스 p_9와 p_{10}의 최소값과 최대값의 순서쌍은 각각 (0, ∞), (0, 45)이다. p_9에 부여된 시간제한쌍은 토큰이 도착하는 임의의 시간에 활성화됨을 의미하기 때문에 결국 p_{11}에 부과된 시간제한 상한값 45에 그 원인이 있다. 따라서 t_{11}에 부과된 최대시간 값 45를 스케줄가능성 조건을 만족하기 위한 $LTEF(t_9) - ETBF(t_9) = 8 \leq FIREdur(t_9) = 10$에 의해 최적값 47로 완화함으로써 t_9를 스케줄가능하도록 한다.

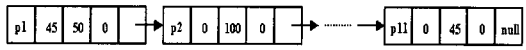
4. 구현 및 분석

본 장에서는 (그림 2)의 TCPN 명세 모델을 기반으로 스케줄가능성을 분석하고, 스케줄이 불가능한 트랜지션이 존재할 경우 스케줄가능하도록 시간명세를 완화하는 회복 기법에 대한 알고리즘의 정확성을 분석하기 위해, 객체 지향 언어인 C++를 이용하여 알고리즘을 구현하고 분석하였다.

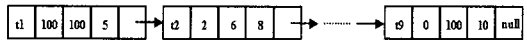
4.1 구현

(그림 2)에 제시된 TCPN 명세 모델을 표현하기 위해 플래이스와 트랜지션 그리고 경로에 대한 데이터는 동적 관리를 위해 링크드-리스트(linked-list) 구조를 이용하였고, 이들 데이터를 처리하기 위해 각 리스트의 포인터를 관리하는 기본 클래스를 구성하였다. 플래이스와 트랜지션 그리고 경로에 대한 링크드-리스트의 구조는 (그림 5)와 같다.

(그림 5)는 플래이스와 트랜지션의 링크드-리스트 구조로서 플래이스와 트랜지션의 번호, 각각의 Min값과 Max값 그리고 Dur값 순으로 데이터를 저장하고 있다. 또한, (그림 6)은 TCPN 모델 내에 가능한 작업경로(path)를 나타내고 있다. TCPN에서 가능한 작업경로의 수(n)는 각 경로가 시작되는 headnode[n]로 시



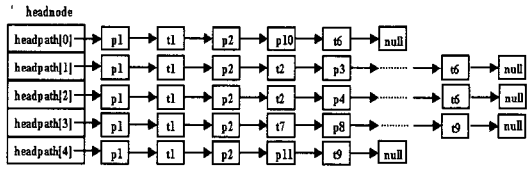
(a) 플래이스의 구조 (PList)



(b) 트랜지션의 구조 (TList)

(그림 5) 링크드 리스트 구조

작되어 각 경로에 포함되는 플래이스와 트랜지션의 리스트로 구성된다. 이때 경로에 포함된 노드들은 (그림 5)의 플래이스와 트랜지션에 기록된 Min값, Max값, 그리고 Dur값을 읽어 경로에 포함되나, (그림 6)에서는 경로에 포함되는 노드만을 축약해서 표현하였다. 그리고 (그림 2)의 TCPN 모델에서 플래이스와 트랜지션에 표현된 최대 시간값 "∞"는 #define DEF 100으로 사전 정의(pre-define)하여 "∞" 시간값을 100으로 한정하였다.



(그림 6) 경로 링크드 리스트의 구조

(그림 6)에 표현된 경로 리스트의 데이터를 처리하는 클래스의 구성에는 (그림 7)과 같다.

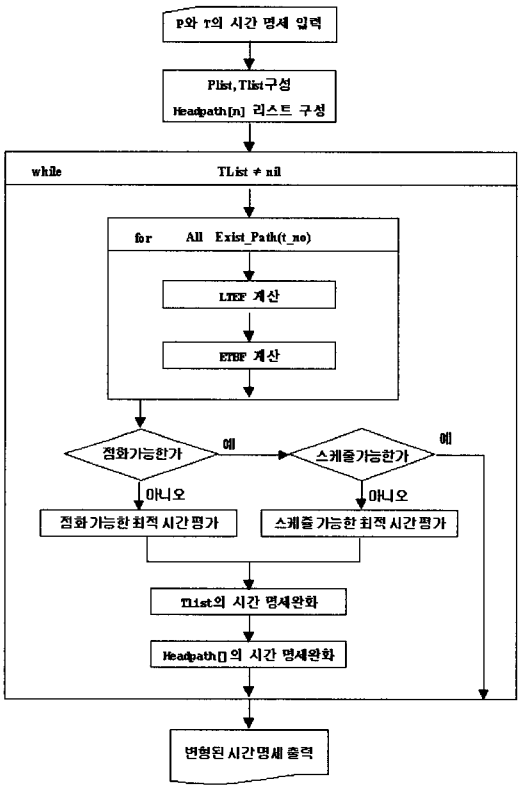
```
class Path {
public :
   char *Node;          // node 번호
   int  Min;            // node의 최소 시간
   int  Max;            // node의 최대 시간
   int  Dur;            // node의 주기
   Path *Next;          // 다음 node의 link

   Path(){} // constructor
   Path(char *node, int min, int max, int dur);
   void  Show_Path_Element(); // path의 요소를 보여준다.
   char* Return_Node();   // path의 Node를 반환
   int   Return_Min();    // node의 Min 값을 반환
   int   Return_Max();    // node의 Max 값을 반환
   int   Return_Dur();    // node의 Dur 값을 반환
};
```

(그림 7) 경로 링크드 리스트의 클래스 구조

4.2 분석

4.1절에서 제시된 리스트 구조를 기반으로 점화가능성을 분석하고 최종적으로 스케줄가능성을 분석하는 과정을 요약하면 (그림 8)과 같다.

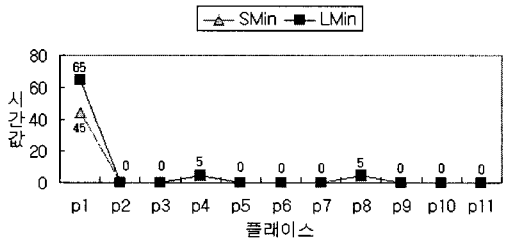


(그림 8) 스케줄가능성 분석 단계

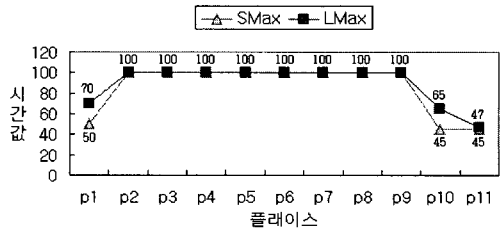
스케줄가능성 분석 단계에서 임의의 트랜지션이 포함된 모든 경로에 대해 LTEF와 ETBF를 결정하여 점화가능성과 스케줄가능성을 판단하고, 만약 점화가능하지 않거나 스케줄가능하지 않는 트랜지션이 존재할 경우 점화가능성과 스케줄가능성을 만족하는 최적의 시간값을 찾아 TList와 Headpath[]의 시간명세를 완화한 후 변형된 시간명세를 기반으로 스케줄가능성 분석 단계를 재수행하게 된다.

플래이스와 트랜지션의 초기 시간명세값을 기준으로 스케줄가능성 분석 알고리즘이 수행된 후의 시간명세값의 변화는 (그림 9)와 (그림 10)에 나타낸다. (그림 9)와 (그림 10)의 SMin, SMax는 알고리즘이 적용되기 전의 초기(Source) Min값과 Max값을 의미하고, LMin과 LMax는 스케줄가능성 알고리즘이 적용된 후(Last) 변형된 Min값과 Max값을 의미한다.

초기 플래이스와 트랜지션 시간명세를 기반으로 스케줄가능성 분석을 수행할 경우 점화가능성 분석 단계에서 트랜지션 t_2는 점화 불가능한 트랜지션으로 판정된다. 그러므로 점화가능하도록 시간명세를 수정하는 회복 알고리즘이 적용되어 (그림 10)의 (b)와 같이 t_2의 Max값이 6에서 10으로 완화되었다. 다음으로 스케줄가능성 분석 단계에서 스케줄을 불가능하게 하는 트랜지션 t_6이 발견되므로, 스케줄을 가능하기 위한 회복 기법에서 (그림 9)의 (b)와 같이 플래이스 p_{10}의 Max값을 45에서 65로 완화하여 스케줄이 가능하게 시간명세를 수정하였다.



(a) Min값의 변화



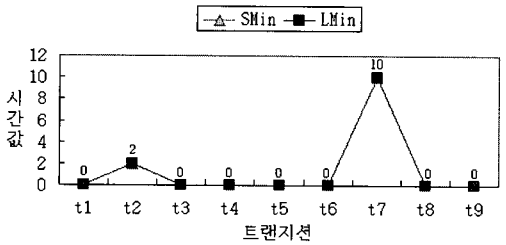
(b) Max값의 변화

(그림 9) 플래이스 Min값과 Max값의 변화

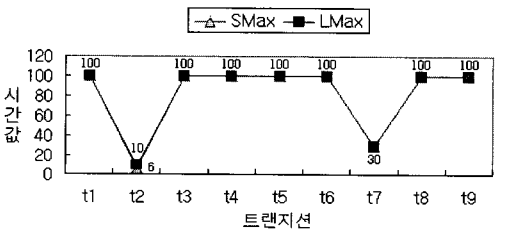
또한, 스케줄이 불가능한 트랜지션 t_9를 스케줄가능하게 시간을 완화하는 과정에서 (그림 9)의 (b)와 같이 플래이스 p_{11}의 Max값을 45에서 47로 완화하여 전체 프로세스를 스케줄가능하게 시간명세를 변형하였다.

본 논문에서는 트랜지션이 점화 불가능하거나 스케줄이 불가능할 경우 트랜지션의 Max 값을 평가된 최적의 시간값으로 완화하기 때문에 Min값의 변화보다는 Max값에 변화가 발생함을 알 수 있다. 그러므로, (그림 9)와 (그림 10)의 시간명세 변화 결과에 의해 초기 명세에 주어진 전체 프로세스의 작업은 65 단위시

간 이내에 정상 또는 긴급 응답을 받아 데이터를 샘플링하는데 5 단위시간을 소요하는 실시간 시스템임을 판단할 수 있다.



(a) Min값의 변화



(b) Max값의 변화

(그림 10) 트랜지션 Min값과 Max 값의 변화

5. 결론 및 향후 연구

본 논문에서는 실시간 시스템을 TCPN으로 모델링하고 이를 기반으로한 스케줄가능성을 분석한다. 만약 분석 단계에서 실시간에 스케줄이 불가능한 부분 프로세스가 존재하면 스케줄가능한 최적의 시간값으로 시간제한을 완화함으로써 전체 프로세스가 스케줄가능하도록 시스템의 초기 명세를 변형 회복 알고리즘을 제안하였다. 이는 기존의 petri net을 이용한 표현과 분석 방법을 결합함으로써 보다 복잡한 실시간 시스템의 체계적인 분석과 회복기법을 제시하고 있다.

또한, 스케줄이 불가능한 부분 프로세스를 제시함으로써 전체 프로세스 실행에 부하(overload)를 초래하는 부분 프로세스를 판단할 수 있다. 이러한 부분 프로세스의 수행시간을 분석함으로써 전체 프로세스에 대한 실행도를 높일 수 있다.

향후 연구 과제로는 TCPN으로 모델된 실시간 시스템의 각 부분 프로세스에 대한 최적의 시간을 분석함으로써 전체 프로세스의 최적 작업시간에 대한 분석과 분산 환경을 기반으로 한 실시간 시스템의 스케줄가능성과 최적 수행시간을 평가하고, 프로세스간의 상호동작에 대한 시각화(visualization)를 제공하는 추가의 연구가 필요하다.

참 고 문 헌

[1] J. L. Peterson, "Petri Net Theory and the Modeling of System," Prentice-Hall, 1981.

[2] J. E. Coolahan, Jr. and N. Roussopoulos, "Timing requirement for time-driven systems using augmented Petri nets," IEEE Trans. Software Eng., Vol.SE-9, pp.603-616, Sept. 1983.

[3] B. Dasarathy, "Timing constraints of real-time system : constructs for expressing them, methode of validating them," IEEE Trans. Software Eng., Vol.SE-11, pp.80-86, Jan. 1985.

[4] M. K. Molloy, "Discrete Time Stochastic Petri Nets," IEEE Transaction on Software Engineering, Vol.SE-11, No.4, Apr. 1985.

[5] N. G. Leveson and J. L. Stolzy, "Safety analysis using Petri nets" IEEE Trans. Software Engineering, SE-13(3) : 386-397, March 1987.

[6] B. Berthomieu and M. Diaz, "Modeling and verification of time dependent systems using time Petri nets," IEEE Trans. Software Eng., Vol.SE-17, pp.259-273, Mar. 1991.

[7] D. Niehaus, "Program Representation and Translation for Predictable Real-Time Systems," Proc. Real-Time Systems Symposium, pp.53-63, 1991.

[8] A. N. Fredette and R. Cleaveland, "A Generalized Approach to Real-Time Schedulability Analysis," 10th IEEE Workshop on Real-Time Operating Systems and Software, 1993.

[9] G. Bruno, A. Castella, I. Pavesio and M. P. Pescarmona, "A New Petri Net Based Formalism for Specification, Design and Analysis of Real-time Systems," Proceeding Real-Time Systems Symposium, pp.294-301, 1993.

[10] Y. Y. Al-Salqan, C. K. Chang and Y. V. Reddy, "MediaWare : On Multimedia Synchronization," IEEE Multimedia computer and Systems,

pp.150-157, May 1995.

[11] M. B. Dwyer and L. A. Clarke, "A Compact Petri Net Representation and Its Implications for Analysis," IEEE Transaction on Software Engineering, Vol.22, No.11, pp.794-811, Nov. 1996.

[12] J. P. Tsai, Yoadong Bi, J. H. Yang and A. W. Smith, "Distributed Real-Time Systems," A Welly-Interscience publication, 1996.

김 춘 배

e-mail : kcbkcb@unicorn.pknu.ac.kr

1993년 부산외국어대학교 컴퓨터공학(공학사)

1996년 부산외국어대학교 대학원 컴퓨터공학(공학석사)

1998년~현재 부경대학교 대학원 전자계산학과 박사과정

관심분야 : 실시간 시스템, petri nets 응용, 멀티미디어 응용, 프로그래밍 언어

박 흥 복

e-mail : multiphb@unicorn.pknu.ac.kr

1982년 경북대학교 컴퓨터공학(공학사)

1984년 경북대학교 대학원 컴퓨터공학(공학석사)

1995년 인하대학교 대학원 전자계산학전공(이학 박사)

1984년~1995년 동명대학 전자계산학과 부교수

1996년~현재 부경대학교 전자계산학과 부교수

관심분야 : 실시간 시스템, 프로그래밍 언어 및 컴파일러, 멀티미디어 시스템, 객체지향 프로그래밍