분산 실시간 시스템에서 개선된 EDF 정책을 사용한 메시지 스케줄가능성 분석

(A Message Schedulability Analysis using an Improved EDF Scheduling for Distributed Real-Time Systems)

이 은 미 † 허 신 ††
(Eun-Mi Lee) (Shin Heu)

요 약 본 논문은 분산 실시간 시스템에서 실시간 동적 알고리즘으로 메시지를 스케줄링하기 위한 개선된 접근법을 제안한다. EDF(Earliest Deadline First) 스케줄링 정책에서 메시지의 중간 종료시한을 계산하기 위한 방법을 제시하고, 이는 우선순위 할당을 위해 사용하는 슬랙값을 조정하는데 사용된다. 수정된 슬랙값으로 메시지의 우선순위값은 보다 정확하게 결정될 수 있으며, 결과적으로 메시지의 스케줄가능성 효율을 증가시킬 수 있다. 개선된 EDF 스케줄링 정책을 적용함으로써, 전송 메시지의 최악의 응답시간을 줄일 수 있었고, 메시지 전송의 실시간적 보장성 문제를 개선하였다. 또한, 개선된 EDF 스케줄링 정책을 기반으로 전송되는 메시지 집합에 대한 스케줄 가능성을 검사하기 위해 간단한 분석 방법을 제공하고 시뮬레이션을 통해 이전의 DM(Deadline Monotonic)과 기존의 EDF을 적용한 예와 비교함으로써, 본 연구 결과의 효율성을 보였다.

키워드 : 메시지 스케줄링, 동적 스케줄링, 스케줄가능성, EDF 스케줄링

Abstract This paper presents an approach for scheduling network messages with real-time dynamic algorithms. We present the method that calculates an intermediate relative deadline of the message based on the EDF(Earliest Deadline First) scheduling policy. We adjust the slack of message by using this intermediate relative deadline to allocate a priority of message. The priority of the message can be determined accurately by using the slack that calculates in our approach, which increases the schedulability efficiency of the message. As a result, we reduce the worst-case response time and improve the guarantee ratio of real-time messages. Also, we describe the analysis method to check the schedulability on message sets, and show the efficiency of our approach by comparing the results of the DM(Deadline Monotonic) approach and the existing EDF approach with that of the improved EDF in our approach through the simulation.

Key words : message scheduling, dynamic scheduling, schedulability, EDF scheduling

1. 서 론

분산 실시간 시스템은 강력한 성능과 높은 신뢰성을 위한 잠재력 때문에 실시간 응용을 위한 차세대 대안으로 부각되고 있다. 분산 실시간 시스템은 네트워크를 통해 연결된 단일 프로세서나 멀티 프로세서들로 구성되어 있으며, 이러한 네트워크에서 노드들 사이에 서로 중복되지 않는 다중 경로들이 존재하여 시스템을 링크와 노드 결함으로부터 더 안정하게 해 준다. 또한, 서로 다른 링크들에서 동시적인 접근을 지원하기 때문에 이러한 네트워크는 보다 높은 처리율을 제공한다. 분산 실시간 시스템에서 가장 중요한 문제는 예측 가능한 프로세스간 통신인데, 이는 메시지의 전송 중에 예측 불가능한 지연이 메시지 통신에 참여하는 태스크들의 응답시간에 영향을 미치기 때문이다. 만일 이러한 태스크들이 종료시한을 갖고 있다면, 예측 불가능한 통신 지연으로 인해 태스크들은 자신의 종료시한을 놓칠 수도 있다. 분산 실시간 시스템에서의 태스크는 각 노드에 동적으로 도착하고,

† 비 회 원 : 한양대학교 전자계산학과
em'ee@cse.hanyang.ac.kr
†† 종신회원 : 한양대학교 전자컴퓨터공학부 교수
shinheu@cse.hanyang.ac.kr
논문접수 : 2002년 5월 9일
심사완료 : 2002년 7월 31일

메시지 전달을 통해 다른 노드들에 있는 태스크들과 통신할 수 있다. 각 태스크는 실행 전에 자신의 실행을 완료해야 하는 종료시한을 가지며, 각 메시지는 전송 전에 자신의 목적노드에 도달해야 하는 종단간 종료시한(end-to-end deadline)을 갖는다. 분산 실시간 시스템에서의 스케줄링은 해당 노드에서의 태스크를 위한 지역 스케줄링(local scheduling)과 함께 통신 링크 상에서의 메시지 스케줄링(message scheduling)을 포함해야 한다. 분산된 경성 실시간 시스템에서 노드간 메시지 통신은 태스크 스케줄가능성에 영향을 끼치며, 따라서 이는 충분히 고려되어야 한다.

최근 들어 실시간 메시지 통신을 고려한 다수의 연구결과들이 발표되고 있다[1-3]. 이들 대부분의 연구는 메시지 통신의 실시간적 특성과 스케줄링을 위해 기존의 단일 시스템에서 사용되어 온 RM(Rate Monotonic)과 EDF(Earliest Deadline First) 정책을 크게 변형하지 않은 채 확장한 것으로, 보다 복잡하고, 동적인 분산환경을 충분히 다루고 있지 못하다. 기존의 여러 연구들로부터 동적 스케줄링은 정적 스케줄링 기법보다 더 효율적임을 보여 왔다. EDF는 프로세서 이용율을 100%까지 높일 수 있는 최적의 스케줄을 찾는 것을 보장하지만, 최적의 정적 스케줄링 기법으로 알려진 RM은 69% 이내의 이용율을 갖는 태스크 집합만이 허용가능하도록 보장한다[4]. 네트워크 상의 메시지 스케줄링에서도 동적 기법이 정적기법보다 더 효율적임이 여러 연구를 통해 밝혀져 왔다[5]. 특히, 메시지의 수가 변동될 수 있고, 긴급한 메시지가 즉시 처리될 수 있도록 보장되어야 할 땐 더욱 그렇다. [5]에서는 정적 스케줄링의 경우 네트워크에서 얻어질 수 있는 스케줄 가능성 한계가 단일 프로세서의 스케줄링에 대해 보장된 69%보다 낮아짐을 보였는데, 이는 일치성의 문제, 네트워크 프로토콜에서 허용되는 제한된 우선순위 비트수, 그리고, 메시지 패킷화로 인한 제한된 선점성의 문제들이 그 이유임을 자세히 설명하고 있다. 이러한 정적 스케줄링 기법에 비해, EDF와 같은 동적 스케줄링 기법은 스케줄링의 높은 이용율 때문에 많은 실시간 프로세서 스케줄링 알고리즘의 기반이 된다. 하지만, 실시간 통신에서 EDF 정책은 메시지를 위한 스케줄링 방법으로는 사용되어 오지 않았는데, 이는 메시지들이 채널에 대해 경쟁하는 매 순간마다 동적으로 할당되는 우선순위값과 스케줄링의 수행으로 인한 오버헤드 때문이다. [3]에서는 이러한 실시간 통신에서 적용되는 EDF 스케줄링의 문제점과 그 해결책이 잘 제시되어 있다. 그럼에도 불구하고 복잡하고 동적인 분산 환경에서의 메시지 스케줄링 문제를 해결하기 위해서는 정적 스케줄링 기법보다는 EDF정책과 같은 동적 스케줄링 알고리즘이 기반이 되어야 함은 분명하며, 현재 많은 문헌들에서 이에 대한 노력들이 진행되고 있다[2,6,7].

본 논문에서는 EDF 스케줄링 정책을 기반으로 하여 메시지의 시간 특성에 효율적으로 적용할 수 있는 스케줄링 정책과 그 스케줄가능성 분석 방법을 제안하고자 한다. EDF 스케줄링 정책에서 동적으로 우선순위를 결정하는 주요한 요인은 슬랙값이다. 슬랙값은 현재 시각에서 실행 종료까지 남은 시간으로 경쟁하는 메시지들 중 가장 작은 슬랙값을 갖는 메시지가 가장 높은 우선순위값을 갖고 전송된다. 이러한 슬랙값은 메시지 전송 중에 송신 태스크의 응답시간 지연이나 경쟁하는 다른 메시지의 전송 지연과 같은 요인들을 좀 더 면밀히 분석함으로써, 수정될 수 있는데, 이는 분산환경에서 실시간 메시지의 전송 중에 할당되는 중간 종료시한을 개선함으로써 가능하다. 본 논문에서는 이러한 수정된 슬랙값을 사용하여 우선순위를 결정함으로써 메시지의 응답시간을 개선하고, 결과적으로 메시지의 통신 이용율을 높일 수 있다. 또한, 개선된 EDF 스케줄링 정책을 기반으로 한 메시지 스케줄링 가능성 분석을 위한 방법도 제안하며, 시뮬레이션을 통해 기존의 정적 스케줄링 접근법과 EDF 스케줄링 접근법에 비해 본 논문에서 제안한 개선된 EDF를 사용한 접근법의 보장율이 더욱 개선되었음을 보인다.

본 논문은 다음과 같이 구성된다. 다음 장에서는 기존의 분산 실시간 시스템에 적용된 스케줄링 정책과 스케줄가능성 분석방법을 소개하고, 3장에서 본 연구의 개선된 EDF를 위한 우선순위 할당방법과 중간 종료시한 계산 방법을 제안하고, 이를 적용한 메시지 스케줄링 가능성 분석 방법을 논한다. 4장에서는 시뮬레이션을 통해 제안한 메시지 스케줄링 방법의 효율성을 검사하고, 5장에서 결론을 맺는다.

2. 기존의 스케줄가능성 분석

실시간 시스템을 위한 스케줄링 접근법으로 잘 알려진 DM(Deadline Monotonic) 정책과 EDF(Earliest Deadline First) 정책은 오래전부터 기본 정리와 함께 특성 등에 대해 많은 연구가 수행되어 왔으며, 특히 단일 프로세서 환경의 경우 많은 산업현장에서 우수한 효과들이 검증되어 왔다. 이 접근법들은 몇몇 연구에 의해 분산 실시간 환경 하에서도 적용되어 왔는데, 본 논문에서는 이들 중 이미 그 효과와 파생 연구들이 진행되고 있는 Spuri의 연구[7]와 Tindell의 연구[8,9]를 기반으

로 한다.

2.1 DM 정책을 기반한 스케줄가능성

Deadline Monotonic(이하 DM) 알고리즘은 잘 알려진 고정 우선순위 알고리즘으로, Rate Monotonic(이하 RM) 알고리즘의 개념과 유사하다. 이 알고리즘에서 우선순위는 태스크의 상대 종료시한 길이의 역에 비례한다. 즉, 상대 종료시한이 짧을 수록 우선순위가 높다. 이 알고리즘의 특징으로는 모든 태스크의 상대 종료시한이 자신의 주기에 비례적일 때, DM은 RM과 동일하게 스케줄 된다. 상대 종료시한이 임의로 주어질 때, DM 알고리즘은 때때로 RM 알고리즘이 실패하는 스케줄을 허용가능하도록 스케줄 한다는 점에서 더 효율적이지만, 반면에 DM 알고리즘이 실패하는 스케줄은 RM 알고리즘도 항상 실패한다. DM 알고리즘에 대한 스케줄 가능성 분석은 일반적으로 각 태스크의 최악의 응답시간을 계산하고, 이 최악의 응답시간이 태스크의 종료시한 보다 작음을 증명함으로써 수행된다. Tindell 등[8,9]은 DM 알고리즘을 기반으로 해서 분산 실시간 시스템에서 잘 적용시킬 수 있는 분석 방법을 제시하고 있는데, 일반적으로 분산 실시간 시스템에서 네트워크상의 메시지를 하나의 태스크로 모델링한다고 가정하면, 각 태스크의 최악의 응답시간은 다음의 식을 사용하여 구할 수 있다.

$$R_i^{(n+1)} = C_i + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^{(n)} + J_j}{T_j} \right\rceil C_j \quad (1)$$

이때, n : 계산단계

C_i : 태스크 τ_i의 계산시간

B_i : 태스크 τ_i가 낮은 우선순위 태스크에 의해 봉쇄되는 시간

$hp(i)$: 태스크 τ_i보다 높은 우선순위를 갖고 τ_i를 선점하는 태스크들의 집합

J_j : 태스크 τ_j의 지터

T_j : 태스크 τ_j의 주기

식 (1) 우변의 세번째 항은 태스크 τ_i가 수행 중에 자신보다 높은 우선순위를 갖는 태스크들에 의해 선점되어 지연되는 시간을 나타낸다. 최악의 응답시간을 구할 때, 이전 단계의 실행시간의 영향을 고려해야 하므로, 위 식은 $R_i^{(n+1)} = R_i^{(n)}$ 일 때까지 반복적으로 수행되고, 그 결과로 태스크 τ_i의 최악의 응답시간 $R_i = R_i^{(n)}$을 구할 수 있다. 이때, $R_i^{(0)} = C_i$ 이다. 이렇게 구해진 R_i은 주어진 종단간 종료시한 D_i과 비교되어, $R_i \le D_i$을 만족하면, 스케줄가능하다고 할 수 있다

2.2 EDF 정책을 기반한 스케줄가능성

EDF 알고리즘은 잘 알려진 동적 우선순위 알고리즘으로, 도착한 태스크들 중 현재 시점에서 가장 이른 종료시한을 갖는 태스크에게 가장 높은 우선순위를 할당한다. [7]에서는 일반적으로 EDF 스케줄링에 대한 허용 가능성을 결정하고, 최악의 실행시간을 계산하기 위해서 busy period 개념을 이용하고 있다. busy period는 $t = 0$부터 첫 번째 프로세서 유휴시간까지의 구간을 의미한다. Liu와 Layland는 모든 태스크들이 동시에 초기화되는 시점을 critical instant라고 정의하고, 이 시점부터 busy period 구간 내에 모든 태스크들이 종료시한을 만족시킬 수 있다면, 이 태스크 집합은 스케줄 가능함을 증명하였다[4]. 태스크 집합의 허용 가능성을 검사하기 위해서는 먼저, 이러한 busy period의 길이를 구하고, 그 다음에 이 구간 내에서 종료시한을 놓치는지 여부를 검사함으로써 결정한다. busy period를 구하는 절차는 Spuri의 holistic접근법[7]에서 자세히 설명하고 있는데, 임의 태스크 τ_i 에 대한 busy period를 구하기 위한 식은 다음과 같다.

$$\begin{cases} L_i^{(0)}(t) = \sum_{A_i} C_j \\ L_i^{(n+1)}(t) = W_i(t, L_i^{(n)}(t)) + (1 + \eta p^*) C_i + B_{M(d_i)} \end{cases} \quad (2)$$

이때, $A_i = \{\tau_j \mid \tau_j \neq \tau_i \wedge D_j \le t + D_i + J_j\}$

ηp^* : t 이전에 τ_i의 주기가 진행된 횟수

식 (2)의 첫 번째 식은 초기값이고, 두 번째 식은 busy period를 구성하고 있는 세 가지 요소를 포함한다. 식 (2)에서 사용한 인수값들은 식 (1)에서와 같고, 각 항의 계산값에 대해서는 [7]에 상세히 나와 있다. 식 (2)는 두 개 연속한 단계의 값 $L_i^{(n+1)} = L_i^{(n)}$일 때까지 반복되고, busy period의 길이 L_i은 $L_i^{(n)}$으로 설정된다. 계산된 L_i은 태스크 τ_i의 최악의 응답시간 R_i를 구하는데 사용된다. 직관적으로 종료시한 d를 갖는 태스크의 인스턴스의 완료시간을 busy period의 끝과 같도록 해서 실행되는 모든 태스크들이 d 보다 작거나 같은 종료시한을 갖도록 한다. 이러한 구간들을 모두 평가하고 최대 길이를 취함으로써, 태스크의 최악의 응답시간을 찾을 수 있다. 다음의 그림 1에서 임의의 시간 t (≥ 0)에서 태스크 τ_i가 도착한다고 가정하자.

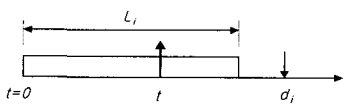

그림 1 busy period 구간

이때, (실행이 예정되어 있는) 다른 모든 태스크들은 t를 포함하는 busy period 구간의 시작에서 최대 비율

로 동기적으로 릴리즈되어야 한다. 그림 1의 $t = 0$에서 τ_i를 제외한 모든 태스크들이 동기적으로 릴리즈된다고 가정했을 때, 태스크 τ_i의 최악의 응답시간 R_i를 구하기 위해서는 busy period 내에서 도착할 가능성이 있는 모든 시간 t 일 때의 R_i중에서 가장 큰 값을 찾는다. 이때 t에는 각 태스크 τ_i의 릴리즈 지터를 함께 고려해야 하므로 태스크 τ_i의 최악의 응답시간 R_i는 다음과 같다.

$$R_i = \max_{t \geq -J_i}\{R_i(t)\} \quad (3)$$

$$R_i(t) = \max\{R_{\max}, L_i(t) - t\} \quad (4)$$

식 (4)에서 $L_i(t)$는 t에 대한 busy period이고, 실제 $R_i(t)$의 값은 busy period에서 t를 뺀 시간, 즉, 앞으로 busy period까지 남은 시간과 태스크 τ_i의 예정된 최대 계산시간 $R_{max}(= J_i + C_i + B_i)$ 중에서 최대값이다. 이와 같이 구해진 τ_i의 최악의 응답시간 R_i는 τ_i의 종료시한이 D_i 보다 작거나 같으면 τ_i는 스케줄가능하다.

3. 개선된 EDF 스케줄링

본 연구에서는 기존의 EDF 스케줄링을 적용한 메시지 스케줄 가능성 분석 방법에서의 문제점을 개선함으로써, 보다 개선된 응답시간을 찾아내어 스케줄 가능성 비율을 높일 수 있도록 한다. 이는 EDF의 특성상 스케줄링시에 동적으로 우선순위를 할당한다는 점을 고려하고, 이러한 특성은 기존의 지정된 종료시한을 수정함으로써, 보다 개선된 스케줄 가능한 비율을 높일 수 있다는 점에 기인한다.

3.1 메시지 모델과 표기 정의

네트워크 상에서 메시지 τ_m은 목적지까지 경로 상의 여러 중간 노드들을 거치게 된다. 통상적으로 분산 실시간 환경에서는 네트워크 상의 링크를 하나의 프로세서로, 링크를 통해 전송되는 메시지를 태스크로 모델링함으로써, 단일 시스템 환경에서의 태스크 스케줄링 문제로 일반화시켜 다루고 있다. 메시지 τ_m을 처음 전송하기 시작하는 송신 태스크를 $\tau_{s(m)}$으로, 메시지 τ_m이 목적지에 도착했을 때 이를 수신하고 처리하는 수신 태스크를 $\tau_{d(m)}$으로 정의한다. 전송되는 메시지 τ_m과 연관된 인수들로는 T_m, C_m, d_m, P_m^* 이고, 각각 메시지의 주기, 전송시간, 종료시한, 그리고, 메시지 τ_m을 구성하는 패킷들의 수이다. 이하 본 연구의 전반에 사용되는 표기들에 대해서는 표 1에 요약되어 있다. 이때, 메시지 τ_m은 태스크 τ_i의 매 η_m 호출 당 한번씩 보낼 수 있으므로, τ_m은 송신 태스크 $\tau_{s(m)}$로부터 $\eta_m T_{s(m)}$의 주기를 상속받게 된다. (이때, 태스크 $\tau_{s(m)}$이 실행중 한번의 메시지만을 호출한다면, $\eta_m = 1$ 이므로 태스크의 주기와 메시지의 주기는 같다.) 각 메시지는 송신 태스크에 의해 여러 개의 패킷들로 나뉘어 진다. 각 메시지에는 스케줄링 정책에 따른 우선순위가 할당되고, 이 메시지를 구성하는 모든 패킷들은 같은 우선순위를 갖는다. 이 패킷들은 호스트 프로세서와 네트워크 어댑터 사이에 공유하는 우선순위 순서 큐에 들어간다. 이때, 메시지를 패킷화하고 패킷을 큐잉하는 오버헤드를 고려해야 하며, 이는 송신 태스크의 최악의 실행시간에 포함된다. 수신측에서는 패킷들이 도착하면 패킷 인터럽트를 일으키거나 하드웨어 상태 레지스터에 도착함을 알리고, 이때 패킷 전달 태스크는 릴리즈되어 패킷을 처리한다. 이때, 역시 패킷 처리로 인한 오버헤드의 문제를 고려해야 하며, 이는 수신 태스크의 최악의 응답시간에 포함되어 한계지어져야 한다. 본 논문은 메시지의 스케줄가능성만을 다루므로, 이에 대한 자세한 분석은 생략하기로 한다.

표 1 메시지 스케줄 가능성 분석시 사용되는 표기들

표기	설명
T_m	메시지 τ_m의 주기
C_m	메시지 τ_m의 전송시간
d_m	메시지 τ_m의 도착시간에 상대적인 지역 종료시한
D_{ete}	메시지 τ_m의 종단간 종료시한
P_m^*	메시지 τ_m을 구성하는 패킷 수
B_m	메시지 τ_m의 봉쇄시간
J_m	메시지 τ_m의 지터
s_m	메시지 τ_m의 슬랙값
s_m^{new}	메시지 τ_m의 새로 수정되는 슬랙값
ϕ_g	태스크의 최소 생성지연
ϕ_c	메시지의 최소 전송지연
C_i^b	태스크 τ_i의 최선의 실행시간
R_i^b	태스크 τ_i의 최선의 응답시간
$\overline{R}_i^b$	태스크 τ_i의 최선의 응답시간의 하한
d_m^{new}	메시지 τ_m의 최종적으로 할당되는 중간 종료시한
$\widehat{d_m}$	메시지 τ_m의 중간 종료시한
ρ	단일 패킷의 최대 전송시간
R_m	메시지 τ_m의 최악의 응답시간
L_m	메시지 τ_m의 busy period

3.2 우선순위 할당

그림 2의 임의 시각 t 에서 메시지 τ_m이 채널에 대해 다른 메시지와 경쟁한다고 할 때, τ_m과 경쟁하는 메시지들은 EDF에 따라 스케줄링되어 채널 사용을 허가 받을 메시지를 결정한다. 기존의 EDF를 사용한다면, 각 메시지는 현재 시각에서 자신의 종료시한까지 남은 시간, 즉, 슬랙값 $s_m(= d_m - t)$의 역에 비례한 값으로 우선순위를 할당받게 되고, 이 우선순위에 따라 가장 높은 우선순위를 갖는 메시지가 채널을 사용할 수 있게 된다. 따라서, s_m는 태스크 실행의 나머지 부분을 완료하는데 필요한

시간으로, 메시지 τ_m의 종료시한과 경쟁이 시작되는 시각 t에 따라 달라지므로, 이 두 값들로 인해 결정되는 각 메시지들의 우선순위는 실제 메시지 집합의 스케줄 가능성 여부에 많은 영향을 끼친다. 지금까지 EDF를 적용한 분산 실시간 시스템에서의 스케줄 가능성 분석은 이러한 슬랙값을 기반으로 하여 수행되어 왔다.

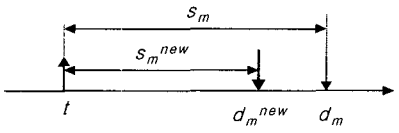

그림 2 슬랙값의 계산

하지만, 현재 분산 실시간 시스템이나 실시간 통신 문제에서 주요한 이슈 중 하나는 프로세서에서 EDF 스케줄링 정책이 적용되었을 때, 최대의 스케줄링 가능성을 실현하기 위해 어떻게 종료시한을 각 태스크의 종단간 계산의 중간 단계에 재 할당할 것인가 하는 문제이다. 다시 말하면, 분산 실시간 시스템의 태스크와 메시지의 종단간 계산시간의 중간단계에 자신의 종료시한을 재조정함으로써, 이들 태스크와 메시지들의 응답시간을 개선할 수 있고, 결과적으로 태스크와 메시지 집합의 스케줄 가능성 효율을 더 높일 수 있게 된다. 이러한 종료시한의 재 할당 문제는 다음 절의 후반에서 상세히 논의될 것이며, 본 논문에서는 EDF 스케줄링에서 우선순위 결정의 기반이 되는 슬랙값 s_m을 계산할 때, 주어진 종료시한 d_m이 아닌 새로 할당되는 조정된 종료시한 d_m^{new}을 기반으로 새로 계산되도록 한다. 본 논문에서 적용하는 우선순위 할당을 위한 슬랙값은 다음과 같다.

$$s_m^{new} = \min(d_m, d_m^{new}) - t \qquad (5)$$

그림 2에서 보면, 기존의 종료시한 d_m 보다 이른 d_m^{new} 가 존재하고, 이 값을 계산할 수 있다면, 슬랙값은 더 작아질 수 있으며, 이 태스크의 우선순위는 높아질 가능성이 있다. 식 (5)에서 새로 계산되는 슬랙값은 $d_m^{new} \le d_m$인 경우에만 새로운 종료시한의 설정이 유효하다. $d_m^{new} > d_m$ 인 경우는 새로운 중간 종료시한이 태스크의 실시간적 특성을 보장하지 못할 가능성이 큰 것으로 보여지며, 이 경우는 식 (5)에 의해 기존의 종료시한은 그대로 사용할 수 있도록 한다. 중간 종료시한 d_m^{new}을 구하는 문제는 다음 절에서 논의한다.

3.3 중간 종료시한의 계산

분산 실시간 시스템에서 메시지의 최악의 응답시간을 계산하는데 있어서 지터의 영향은 중요하다. 태스크의 지터는 태스크가 도착한 후 릴리즈되기까지의 최악의 지연으로 정의되며, 일반적으로 단일 시스템에서는 분석의 용이성을 이유로 무시하기도 한다. 하지만, 메시지의 지터는 메시지를 포함한 송신 태스크가 도착한 이후부터, 실제 메시지가 전송을 시작하는 시각까지로, 통신상에서는 가장 늦은 큐잉시간과 가장 이른 큐잉시간의 차이로 볼 수 있다. 이때, 메시지 지터는 노드에서 수행중인 태스크들과의 통신 요인들로 인해, 무시할 수 없을 정도로 커질 수 있으므로 중요하게 다루어져야 한다. Spuri와 Tindell의 holistic 접근법[7,9]에서는 메시지의 지터값을 선행하는 태스크의 최악의 응답시간으로 설정하고 있다.

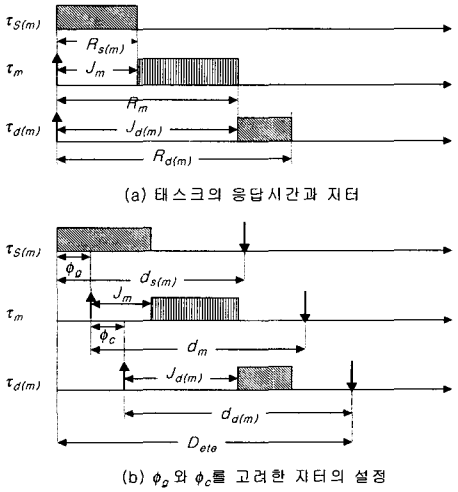

그림 3 태스크와 메시지의 종료시한 요소들

그림 3.(a)와 3.(b)에서 빗금친 부분은 $\tau_{s(m)}$과 $\tau_{d(m)}$의 경우 태스크의 계산시간($C_{s(m)}$과 $C_{d(m)}$)을, τ_m의 경우 메시지의 전송시간(C_m)을 나타낸다. 이러한 태스크들과 메시지의 계산시간($C_{s(m)}$, $C_{d(m)}$, C_m)은 자신의 지터($J_{s(m)}$, $J_{d(m)}$, J_m)와 더해져서 이들의 최악의 응답시간($R_{s(m)}$, $R_{d(m)}$, R_m)으로 계산되며, 이는 그림 3.(a)에 나타나 있다. 그림 3.(a)에서 메시지(τ_m), 메시지의 송신 태스크($\tau_{s(m)}$), 그리고 수신 태스크($\tau_{d(m)}$)의 관계를 서로 선행관계 ($\tau_{s(m)} \Rightarrow \tau_m \Rightarrow \tau_{d(m)}$)에 있다고 한다면, 송신 태스크의 최악의 응답시간은 메시지의 지터로, 메시지의 최악의 응답시간은 수신 태스크의 지터로 설정된다. 하지만, 그림 3.(a)와 같이 설정되는 지터값들은 실제값보다 크게 설정될 수 있다. 각 태스크(또는 메시지)의 계산시간(또는 전송시간)이 미리 지정되어 있다고 할 때, 메시

지의 최악의 응답시간에 중대한 영향을 끼치는 요소는 지터값으로, 이는 최악의 응답시간에 영향을 끼치고, 결과적으로 태스크의 스케줄 가능성 여부에도 영향을 준다.

그림 3.(b)에서 보면, 실제로 메시지 τ_m은 송신 태스크 $\tau_{s(m)}$이 도착한 이후 ϕ_g만큼 지연된 후 메시지 τ_m이 생성되며, 수신 태스크 $\tau_{d(m)}$도 메시지가 도착한 후 ϕ_c만큼의 지연이 있은 후에 도착한다. 이러한 ϕ_g와 ϕ_c는 Spuri의 접근법에서 최소 생성지연과 최소 전송지연으로 정의하고는 있으나, 실제 $\phi_g = 0$으로 가정하고, ϕ_c는 패킷전체의 전송시간에 전달지연을 더한 값으로 정의하였다. $\phi_g = 0$의 의미는 메시지의 최악의 응답시간의 한 요소인 지터는 송신 태스크의 최악의 응답시간과 같은 값임을 뜻한다. 하지만, 일반적인 경우 최소의 생성지연을 0으로 볼 수 없기 때문에, Spuri의 모델에서의 메시지의 지터는 실제값보다 훨씬 커질 수 있다. 실제값보다 크게 산정된 지터는 해당 태스크보다 낮은 우선순위를 가지는 태스크의 최악의 응답시간을 증가시키기 때문에 태스크의 최악의 응답시간 분석의 정확도를 떨어뜨리게 된다. 또한, 실제보다 크게 계산된 메시지의 최악의 응답시간은 메시지의 수신 태스크의 지터도 증가하게 되어 연쇄적으로 부정확한 최악의 응답시간을 산출하게 된다. 따라서, 결과적으로 그림 3.(a)의 J_m과 $J_{d(m)}$은 그림 3.(b)에서처럼 ϕ_g와 ϕ_c를 포함할 수 있으며, 이 값들을 제거함으로써, 메시지와 수신 태스크의 응답시간을 개선할 수 있다.

(1) 최소의 생성지연(ϕ_g)의 계산

그림 3.(b)에서 $\tau_{s(m)}$의 응답시간에는 $\tau_{s(m)}$이 수행 중에 어떠한 외부 간섭이나 선점으로 인해 지연되는 시간이 포함된다고 보면, ϕ_g는 $\tau_{s(m)}$이 이러한 지연없이 최선의 상황에서 수행되는 최소 실행시간으로 볼 수 있다. 다시 말하면, ϕ_g는 태스크 $\tau_{s(m)}$의 최선의 응답시간으로 볼 수 있으며, 이는 $\tau_{s(m)}$의 최악의 응답시간에서 ϕ_g 값만큼 제거함으로써 메시지의 지터(J_m)를 개선할 수 있다. (이는 수신태스크($\tau_{d(m)}$)의 지터($J_{d(m)}$)와 최소 전송지연(ϕ_c)도 마찬가지이다.) 이처럼 태스크가 최선의 응답시간만으로 수행될 수 있는 상황은 자신보다 높은 우선순위 태스크에 의해 선점되는 시간이 가장 작고, 자신의 수행시간이 가장 작을 때이다. 먼저, 후자의 경우는 자신의 최선의 실행시간 C_i^b로 정의하여, 실행시간 계산 알고리즘에 의해 계산될 수 있다고 가정한다. 전자의 경우는 자신보다 높은 우선순위 태스크들 모두 최선의 응답시간으로 완료됨과 동시에 자신이 수행을 시작하고 높은 우선순위 태스크들의 다음번 실행은 가능한 늦게 발생할 때로 설명될 수 있다. 다시 말하면, 높은 우선순위 태스크들로 인한 선점시간이 0일 때를 의미하며, 이는 그림 4에서 설명될 수 있다.

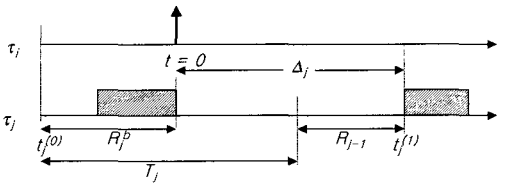

그림 4 태스크와 최선의 응답시간

그림 4에서 태스크 τ_i 의 도착시간이 $t = 0$라고 가정할 때, τ_i보다 높은 우선순위를 갖는 태스크 τ_j가 최선의 상황을 유지하면서 실행되기 위해서는 시각 $t = 0$이 τ_i가 도착함과 동시에 이전에 도착해서 실행된 높은 우선순위 태스크 τ_j가 완료되는 시간이라는 가정이 필요하다. $t = t_j^{(0)}$ 시각은 $t = 0$에서 종료된 태스크 τ_j가 도착한 시간이고, 이는 최선의 시나리오를 고려한다면, 가능한 가장 늦은 시간이 되어야 하므로, $t_j^{(0)} = -R_j^b$이다. τ_j의 다음 주기에서 $t = 0$ 이후에 τ_j의 첫 번째로 도착하는 태스크의 도착 시간 $t_j^{(1)}$ 역시 가능한 가장 늦은 시간으로 선택되어야 하므로, τ_j보다 선행하는 태스크 τ_{j-1}이 있다면, 이 태스크의 최악의 응답시간 이후에 발생하게 된다. 따라서, $t_j^{(1)}$ 시간은 $T_j + R_{j-1} - R_j^b$와 같게 되고, 이 시간을 Δ_j ($= T_j + R_{j-1} - R_j^b$)로 정의한다. Δ_j는 τ_i가 도착하고 나서 높은 우선순위 태스크 τ_j가 실행되기 까지의 시간이다. 따라서, 임의 시각 t에서, $\left\lceil \frac{t-\Delta_j}{T_j} \right\rceil$는 τ_j가 τ_i를 선점하는 횟수이고, 낮은 우선순위 태스크 τ_i의 응답시간에 대한 높은 우선순위 태스크 τ_j의 선점시간에 대한 하한은 $\max\left(0, \left\lceil \frac{t-\Delta_j}{T_j} \right\rceil\right)C_j^b$이다. 이 값은 낮은 우선순위 태스크 τ_i의 최선의 응답시간을 계산하는데 사용되는데, τ_i 자신의 최선의 실행시간에 모든 높은 우선순위 태스크들의 선점효과들의 합을 더해서 다음의 식을 만족하는 첫 번째 값을 찾을 때까지 계산을 반복한다.

$$\overline{R}_i^b = C_i^b + \sum_{\forall j \in hp(i)} \max\left(0, \left\lceil \frac{\overline{R}_i^b - \Delta_j}{T_j} \right\rceil\right) C_j^b \quad (6)$$

이때, $hp(i)$는 τ_i보다 높은 우선순위를 갖는 태스크들의 집합

식 (6)의 초기값은 $\overline{R}_i^b = C_i^b$이고, $\overline{R}_i^b$로 수렴할 때까지 반복한다. 이 값은 τ_i의 동작에 관련된 최선의 응답시간의 하한이고, 이 시간에 선행하는 태스크가 존재

하는 경우 이 태스크의 최선의 응답시간을 더함으로써, τ_i의 전체 최선의 응답시간 $R_i^b = \overline{R}_i^b + R_{i-1}^b$ 이다. 앞서 언급한 바와 같이 이와 같이 구해진 R_i^b은 그대로 최선의 응답시간 ϕ_g로 적용될 수 있으므로, 다음과 같은 ϕ_g를 얻을 수 있다.

$$\phi_g = \overline{R}_{s(m)}^b + R_{s(m)-1}^b \quad (7)$$

(2) 최소의 전송 지연 (ϕ_c)

ϕ_c 값은 메시지가 전송되는 최소한의 응답시간이며, 하나의 패킷이 전송되는 단위시간을 ρ, 메시지를 구성하는 패킷의 수를 P_m^*으로 정의할 때, 모든 패킷이 전송되는 최소한의 시간 ρP_m^*에 전파지연 P를 더한 값으로 지정할 수 있다. 즉, $\phi_c = \rho P_m^* + P$이다. 전파지연 P는 통신 프로토콜에 따라서 결정되는 것으로 시스템 전체에 일정한 값으로 적용될 수 있다. 이러한 ϕ_c값 역시 실제 메시지의 수신 태스크의 지터값에 영향을 미친다.

(1)과 (2)에서 계산된 ϕ_g와 ϕ_c값으로 메시지와 수신 태스크의 지터값은 감소되고, 이는 이들의 응답시간을 개선시킨다. 결과적으로 이 값들은 메시지의 새로운 중간 종료시한(d_m^{new})을 결정하는데 사용된다. 메시지를 포함한 송신 태스크가 릴리즈된 이후부터 메시지가 전송되고 수신 태스크에 도착하여 처리되는 과정은 종단간 종료시한(D_{ete}) 내에 처리되어야 하고, D_{ete}는 메시지의 전송 전에 지정된다. 통신 특성 상, 송신 태스크, 메시지, 수신 태스크는 서로 다른 프로세서에서 처리되므로, (이때, 메시지가 전송되는 링크는 하나의 프로세서로 모델링 된다.) 이들 세 형태의 태스크들은 각각 자신의 프로세서에서 처리될 때, 또 다른 지역 태스크들과 경쟁하기 위한 지역 종료시한이 필요하다. 즉, 분산 실시간 환경에서 수행되는 태스크와 메시지들은 초기에 주어지는 종단간 종료시한 외에 수행 단계에서 할당되어야 하는 종료시한이 필요하다. 본 논문에서는 송신 태스크, 메시지, 수신 태스크의 중간 종료시한을 $d_{s(m)}$, d_m, $d_{d(m)}$으로 표기하기로 하고, 사전에 지정되어 있다고 가정한다. 앞서 언급한 바와 같이 위의 세 형태의 태스크 모델이 선행관계에 있다고 볼 수 있다면, 각 태스크(메시지)의 종료시한은 선행하는 태스크(메시지)의 종료시한에 영향을 끼치므로, 태스크(메시지)의 중간 종료시한을 계산하는데 사용될 수 있다. 메시지의 중간 종료시한은 미리 알고 있는 종단간 종료시한(D_{ete})를 적용하여 그림 3.(b)에서 다음과 같이 구할 수 있다.

$$\widehat{d_m} = D_{ete} - C_{d(m)} - \phi_g \quad (8)$$

$\widehat{d_m}$은 또한, 수신 태스크의 종료시한($d_{d(m)}$)을 사용하면, 좀 더 근사값에 접근할 수 있는데, 그림 3.(b)에서 직접 다음의 식을 얻을 수 있다.

$$\widehat{d_m} = d_{d(m)} - C_{d(m)} + \phi_c \quad (9)$$

식 (8)와 (9)는 기준이 되는 종료시한을 각각 D_{ete}와 $d_{d(m)}$를 적용하여 근사한 값으로 유도되었다. 그림 3.(b)에서 보면, 직관적으로 D_{ete}가 $d_{d(m)}$보다 크기 때문에, 식 (9)에서 구해지는 $\widehat{d_m}$이 식 (8)에서 얻어지는 값보다 작다고 생각될 수 있다. 하지만, 사전에 알 수 없는 ϕ_g와 ϕ_c값에 따라 결과는 바뀔 수도 있다. 즉, ϕ_g가 ϕ_c값보다 상대적으로 큰 값으로 계산되는 경우는 식 (8)에서 더 작은 종료시한을 구할 수도 있다. 따라서 위의 두 식에서 최소값을 구하여 사용하는 것이 바람직하다. 따라서 $\widehat{d_m}$의 값은 다음의 식에서 찾을 수 있다.

$$\widehat{d_m} = \min(D_{ete} - \phi_g, d_{d(m)} + \phi_c) - C_{d(m)} \quad (10)$$

식 (10)에서 $\widehat{d_m}$은 D_{ete}와 $d_{d(m)}$를 기준으로 하여 새로 계산되는 ϕ_g와 ϕ_c값에 따라 새로 조정되는 메시지의 종료시한이다. 이는 3.2절의 s_m^{new}를 구하기 위한 메시지의 새로운 중간 종료시한값으로 사용될 수 있다. 또한, 메시지는 전송 전에 지정된 d_m 값을 인수로 갖고 있다고 가정하면, 다음과 같은 새로운 중간 종료시한을 얻을 수 있다.

$$d_m^{new} = \min(d_m, \widehat{d_m}) \quad (11)$$

식 (11)에서 계산된 새로운 중간 종료시한 d_m^{new}는 d_m과 마찬가지로 메시지 집합에 대해 스케줄 가능성을 보장받을 수 있다. 다시 말하면, 종료시한 d_m을 만족하면서 스케줄되는 메시지는 d_m^{new}에 대해서도 스케줄 가능하다. 이는 그림3.(b)와 식(8)와 (9)을 통해 쉽게 유도할 수 있다. 메시지 τ_m이 스케줄가능함을 보이기 위해서는 메시지 τ_m의 응답시간이 종료시한보다 작음을 보이면 된다. 그림 3.(b)에서 메시지 τ_m의 응답시간 $R_m = J_m + C_m$, 수신 태스크 $\tau_{d(m)}$의 응답시간 $R_{d(m)} = J_{d(m)} + C_{d(m)}$ 이라고 하자. 먼저 기존의 d_m이 아닌 새로 계산되는 d_m^{new}는 식 (10)에서 $D_{ete} - C_{d(m)} - \phi_g$이거나 $d_{d(m)} - C_{d(m)} + \phi_c$인 경우이고, 둘 모두 기존의 d_m보다는 작아야 한다. 이러한 사실을 기반으로 하여, 그림 3.(b)에서 직접 얻어지는 $D_{ete} \geq R_{d(m)} + \phi_g + \phi_c$와 $d_{d(m)} \geq R_{d(m)}$의 두 식을 이용하여 유도하면, 두 경우 모두 $R_m \leq d_m^{new} < d_m$임을 쉽게 구할 수 있다.

3.4 메시지 스케줄 가능성 분석

메시지 τ_m의 최악의 응답시간을 구하는 방법은 2.2절의 Spuri의 분석방법[7]을 기반으로 한다. 메시지 τ_m이 $t = t_a$시각에 생성된다고 하면, 이때, 지역적이거나 원격의 다른 메시지들 모두 동시에 최대 비율로 릴리즈된다.

패킷의 전송은 비선점이므로, 메시지 τ_m의 통신 지연 $R_m(t_a)$을 효율적으로 평가하기 위해서 다음의 접근법을 사용한다.

$$R_m = \max_{t_a \geq -J_m}\{R_m(t_a)\} \tag{12}$$

$$R_m(t_a) = \max\{J_m + B_m + P_m^* \rho + P, \; L_m(t_a) + \rho + P - t_a\} \tag{13}$$

식 (13)에서 ρ 는 하나의 패킷의 최대 전송시간, P_m^*을 메시지 τ_m을 구성하는 패킷들의 수라고 할 때, $P_m^* \rho$ 은 메시지의 모든 패킷이 순수하게 전송되는데 걸리는 시간이라고 할 수 있다. 또한, $\rho + P$는 목적지까지 자신의 전송지연(transmission delay)과 전달지연(propagation delay)을 위한 시간이다. 전달지연 P는 매체에 의존적이므로, 같은 값으로 처리될 수 있다. B_m은 τ_m의 봉쇄시간으로, 패킷 전송의 비선점적인 특성으로 인해, τ_m이 낮은 우선순위 패킷의 전송으로 인해 지연되는 시간이다. 선점은 패킷 크기 단위로 제한되므로, 이러한 봉쇄시간 $B_m = \rho$로 정의할 수 있다. 식 (12)와 식 (13)에서의 관건은 busy period $L_m(t_a)$를 구하는 일이다. 일반 태스크와는 달리, 네트워크 상에서의 $L_m(t_a)$는 일정시간 간격동안 다른 프로세서로 부터의 간섭이나, 다른 선행하는 메시지들로 인한 오버헤드 등을 포함해야 한다. busy period는 다음의 세 가지 구성요소들을 포함한다.

(i) 메시지 τ_m 자신의 전송시간

(ii) $t_a + d_m$ 이전에 종료시한을 갖는 메시지 중 지역적으로 높은 우선순위 메시지의 작업부하

(iii) 동기적 네트워크 접근에서 다른 호스트들의 간섭과 지역 비동기 트래픽의 간섭

첫 번째 요소는 네트워크 상에서 메시지의 송신 태스크($\tau_{s(m)}$)에서부터 수신 태스크($\tau_{d(m)}$)까지 메시지가 전송되는데 걸리는 시간으로, C_m으로 정의한다. 두 번째 요소는 $d_{m'} < t_a + d_m$인 종료시한 $d_{m'}$을 갖고 busy period 내에 릴리즈되는 모든 메시지들 $\tau_{m'}$의 전송시간 합으로, 다음의 식에서 구해진다.

$$H_m(t_a, t) = \rho \sum_{A_m} \min(\eta_{m'}^*, \; \eta d_m^*) C_{m'} \tag{14}$$

이때, $A_m = \{\tau_{m'} | \exists \tau_{m'} \in out(p), \tau_{m'} \neq \tau_m \wedge d_{m'} \leq t_a + d_m + J_{m'}\}$

$out(p)$: 프로세서 p에서 나가는 메시지들의 집합

$\eta_{m'}^*$: 시간 구간 $[0, t]$에서 태스크 τ_m이 아닌 다른 태스크들이 릴리즈되는 횟수

ηd_m^* : $a + d_m$보다 크거나 같은 종료시한을 갖는 다른 태스크들이 도착할 횟수

식 (14)에서는 현재 계산을 위한 시점인 t와 이전에 메시지 τ_m이 도착한 시각 t_a를 구별할 필요가 있으며, 식에서 $\eta_{m'}^*$와 ηd_m^*은 각각 $1 + \left\lfloor \frac{t + J_{m'}}{T_{m'}} \right\rfloor$, $1 + \left\lfloor \frac{t_a + d_m + J_{m'} - d_{m'}}{T_{m'}} \right\rfloor$ 이다. 세 번째 요소는 토큰이 다른 호스트의 네트워크 접근에 의해 많이 지연될 수 있으므로, 중요하게 다루어져야 하며, 각 노드에 방문시 최대 지연은 프로토콜을 주의깊게 분석함으로써 정확하게 계산될 수 있다. 이는 선택된 네트워크 프로토콜의 채널 접근방식에 상당히 의존한다. 하지만, 이 계산값은 제법 복잡한 과정을 거쳐 얻어질 수 있으므로, 본 연구의 주제에서 벗어난다. 따라서 본 연구에서는 더 이상 다루지 않으며, 다음의 busy period의 계산식에서는 오버헤드 값($O(t_a)$)을 0으로 처리하기로 한다. 위의 세 요소를 모두 고려한 busy period의 길이는 식 (15)와 같이 구할 수 있다.

$$\begin{cases} L_m^{(0)}(t_a) = \sum_{A_m} C_{m'} \\ L_m^{(n+1)}(t_a) = C_m + B_m + H_m(t_a, L_m^{(n)}(t_a)) + O(t_a) \end{cases} \tag{15}$$

이 식은 $L_m^{(n+1)}(t_a) = L_m^{(n)}(t_a)$ 일 때까지 반복 계산되고, $L_m(t_a)$ $(= L_m^{(n+1)}(t_a))$을 구한다. 이때, t_a의 범위는 $[0, \; L_{max} - B_m - \rho C_m]$ 이다. 이때, L_{max}는 busy period의 최대 길이이고, 2.2절에서 태스크의 실행시간 대신에 메시지의 전송시간을 대치한다. 3.2절에서 3.4 절까지 설명된 메시지 스케줄가능성 검사를 위한 절차를 그림 5의 알고리즘 SCM에 나타내었다.

```
Algorithm SCM
    초기 지역 종료시한을 할당
        새로운 중간 종료시한 d_m^new의 계산
        슬랙 s_m^new에 따른 우선순위 할당
        for ( 종료조건을 만족하지 않는 동안 )  {
                busy period의 계산
        }
        최악의 응답시간 계산
        메시지 스케줄가능성 여부검사
```

그림 5 알고리즘 SCM

4. 시뮬레이션

시뮬레이션을 위해, 세 개의 호스트 프로세서들에 36개의 태스크들이 임의로 할당하였다. 각 프로세서에 임의로 할당된 태스크들에는 지역적으로 수행되는 태스크와 메시지를 전송하는 태스크들이 임의의 비율로 포함되어 있으며, 메시지를 전송하는 태스크의 비율은 할당된 태스크 수에 대해 대략 0.3 ~ 0.7정도이고, 모두 48

개의 메시지 집합을 가정하였다. 전송되는 메시지들은 모두 주기적 스트림으로 가정한다. 패킷 크기는 1024 byte이고, 패킷 전송 단위시간 ρ = *0.8 ms*로 한다. 각 메시지들은 주기와 전송시간, 그리고, 종료시한의 인수값을 가지며, 주기는 10 *ms* 와 2000 *ms*사이로 하였으며, 종료시한은 주기에 대해 0.4에서 1 까지의 비율로 지정하였다. 또한, 메시지의 지터 계산을 위해 송신 태스크의 최악의 응답시간을 메시지 전송시간의 0.1 정도로 할당하였다. 각 프로세서에 할당된 메시지 집합의 전송시간과 주기는 네트워크 이용율을 일정한 비율로 변화시키기 위해 의도적으로 지정하였으며, 이때, 메시지의 네트워크 이용율은 $\sum_m C_m/T_m$으로 계산된다. 네트워크 이용율은 실제 프로세서 이용율이라고 할 수 있으며, 실제로 프로토콜 오버헤드 때문에 더 높은데, 가장 높은 네트워크 이용율은 데이타 이용율의 대략 0.58 정도임을 [2]에서 밝힌 바 있다. 하지만, 본 연구에서는 별도의 네트워크 프로토콜을 가정하지 않았으며, 메시지 집합에 대한 스케줄링의 효율성을 보이는 것이 목적이므로, 프로토콜 오버헤드는 0으로 가정한다. 시뮬레이션에 사용된 메시지 집합과 인수값들을 표 2에 나타내었다. 표 2의 메시지들은 시뮬레이션에서 사용된 메시지들의 집합 중 일부만을 나타낸 것으로, 시뮬레이션에 사용되는 인수값들과 중간단계에서 계산되는 결과값을 보여준다. 표 2에서 표시된 값들의 단위는 *ms* 이다.

표 2 시뮬레이션에 사용된 메시지들의 인수값과 계산결과

mess-age	source	destina-tion	초기 인수들			계산된 결과들		
			C_m	T_m	d_m	J_m	R_m	d_m^{new}
1	P_3	P_1	1	20	19	5	12	17
2	P_3	P_1	1	160	19	5	8	18
3	P_3	P_2	3	100	90	3	12	87
4	P_3	P_1	2	100	96	18	47	94
5	P_3	P_1	16	800	196	18	47	196
6	P_1	P_2	1	1000	998	42	67	988
7	P_1	P_2	1	200	199	45	63	199
8	P_2	P_1	1	40	14	4	12	14

시뮬레이션은 두 가지 방법으로 수행하였다. 먼저, 네트워크 이용율을 점차적으로 증가시켜 각 점에서 기존의 DM과 EDF, 그리고, 본 연구의 개선된 EDF 스케줄링 정책을 사용한 스케줄가능성 비율을 구한 것으로, 그 결과를 그림 6에 나타내었다.

그림 6에서 보면, 기존의 EDF 정책을 사용한 결과는 DM 정책을 사용한 결과보다 스케줄 가능성 비율이 현저하게 높아짐을 볼 수 있는데, 네트워크 이용율이 40

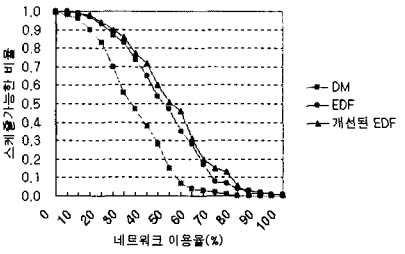

그림 6 DM, EDF, 개선된 EDF 정책에 대한 네트워크 이용율에 따른 스케줄가능성 비율

- 70% 범위에서 거의 두 배의 효율을 보여준다. 또한, 네트워크 이용율이 25 - 80% 범위에서 개선된 EDF가 기존의 EDF를 사용했을 때보다 약간의 차이지만, 스케줄되는 비율이 더 높음을 알 수 있다. 60%와 75 - 80% 부근에서는 0.1 비율의 차이를 보이고 있다. 프로토콜 오버헤드를 고려한다면, 전체적으로 네트워크 이용율이 감소하겠지만, 이는 그림 6에서 보여주는 스케줄링 정책들간의 수행 차이에 영향을 주지는 않는다. 두 번째는 네트워크 이용율 대신에 메시지 집합의 수에 따른 스케줄을 실패하는 비율을 구하기 위해 실험을 반복하였다. 이때는 프로세서에 할당되는 메시지 수의 비율을 점차적으로 증가하도록 조정하였으며, 메시지의 각 인수값들은 첫 번째 평가에서와 같은 인수들을 사용하였다. 이 경우는 네트워크 이용율을 고려하지 않고, 임의로 메시지를 할당 것으로, 그림 6의 스케줄가능성 비율보다는 낮게 산정될 수 있다. 이 결과는 그림 7에 나타내었다.

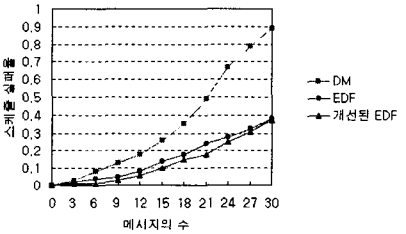

그림 7 DM, EDF, 개선된 EDF 정책에 대한 메시지 수에 따른 스케줄 실패율

그림 7에서도 DM은 EDF나 본 연구의 개선된 EDF를 사용했을 때 보다 실패율이 현저히 높아짐을 볼 수 있다. 메시지의 수가 많아지면, 그 만큼 다른 메시지로 부터의 간섭이나 선점 등으로 인한 지연이 발생하기 때문에 메시지가 종료시한을 놓치게 될 가능성이 커진다. 이런 경우, DM 정책보다는 EDF 정책을 기반한 접근법이

더 효율적임을 보여준다. 본 논문의 개선된 EDF 접근법이 기존의 접근법보다 많게는 0.08 비율 정도 개선되었음을 볼 수 있다.

5. 결 론

본 연구에서는 분산 실시간 시스템 하에서 EDF 정책을 사용하여 메시지의 스케줄가능성 방법을 알아보고, 메시지의 스케줄가능성의 효율을 높이기 위한 해결책을 제시하였다. EDF 정책을 사용하였을 때, 메시지 스케줄링의 시간특성과 요구조건을 분석하고, 기존 연구에서의 불필요한 가정을 제거하여, 스케줄링의 효율성을 높였다. 이는 채널에 대해 경쟁하는 메시지의 중간 종료시한을 면밀하게 분석하여, 그 하한을 메시지의 우선순위를 결정짓는데, 적용함으로써 실현된다. 이러한 절차는 알고리즘 SCM을 수행함으로써, 적용할 수 있다. 그 결과로, 최악의 응답시간에 지나치게 비관적으로 계산되어 포함된 메시지의 지터값은 상당히 개선될 수 있으며, 이는 전체 메시지 집합의 스케줄가능성 여부에도 많은 영향을 끼친다. 본 논문의 접근법의 효율성은 시뮬레이션을 통해 증명하였으며, 각 메시지의 네트워크 이용율과 메시지 수의 크기에 따른 스케줄가능성 비율과 실패율을 비교해 보았다. 그 결과로 제안한 알고리즘은 기존의 DM 보다는 현저한 개선을 보이고 있으며, 기존의 EDF 보다는 스케줄 가능성 비율 약간의 우수함을 보였다.

본 논문는 몇 가지 부분에서 개선이 요구된다. 본 연구는 주기적 메시지만을 고려하였으나, 실질적으로 네트워크상에서 비주기적 메시지로 부터의 영향을 무시할 수 없다, 또한, 본 논문에서는 제한된 프로토콜 오버헤드를 고려한 보다 면밀한 분석도 필요하다. 실시간 통신 분야에서 프로토콜을 고려한 통신지연문제는 여러 논문에서 다루고 있으나. 동적 스케줄링을 고려한 연구는 현재로선 그 성과가 미진하다. 앞서 언급한 비주기적 메시지 문제와 함께 이는 향후 연구되어야 할 주제이다.

참 고 문 헌

[1] L. Sha, S. S. Sathay, and J. K. Strosnider, "Scheduling Real-Time Communication on Dual-Link Networks," IEEE Proc. of Real-Time Systems Symposium, pp188-197, Dec., 1992.

[2] N. Malcolm and W. Zhao, "The Timed-Token Protocol for Real-Time Communications," IEEE Computer, Jan. 1994.

[3] M. D. Natale, "Scheduling the CAN Bus with Earliest Deadline Techniques," IEEE Proc. of Real-Time Symposium, 2000.

[4] C. L. Liu, and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," JACM 20(1), 1973, pp46 - 61

[5] J. P. Lehoczky and L. Sha, "Performance of Real-Time Bus Scheduling Algorithm," ACM Performance evaluation Review, 1986

[6] A. Meschi, M. D. Natale, and M. Spuri, "Earliest Deadline Message Scheduling with Limited Priority Inversion," IEEE Proc. of the Fourth Workshop on Parallel and Distributed Real-Time Systems, pp87-94, Apr., 1996 .

[7] M. Spuri, "Holistic Analysis for Deadline Scheduled Real-Time Distributed Systems," Rapport de recherche 2772, INRIA Rocquencourt, 1996.

[8] K. Tindell, "Analysis of Hard Real-Time Communication," Department of Computer Science, University of York, 1996.

[9] K. Tindell, and J. Clark, "Holistic Schedulability Analysis for Distributed Hard Real-Time Systems," Microprocessing & Microprogramming, Vol.50, Nos. 2-3, pp117-134, Apr. 1994.

[10] K. Tindell, H. Hanson, and A. Wellings, "Analysing Real-Time Communication: Controller Area Network(CAN)," procc. of the IEEE Real-Time Systems symposium, 1994.

[11] J. C. P. Gutierrez, J. J. G. Garcia, and M. G. Harbour, "Best-Case Analysis for Improving the Worst-Case Schedulability Test for Distributed Hard-Real-Time Syatems,"Proc. of Euromicro Workshop on Real-Time Systems, pp35-44, 1998.

이 은 미

1989년 한양대학교 수학과 학사. 1991년 한양대학교 전자계산학 석사. 현재 한양대학교 전자계산학 박사과정. 관심분야는 분산처리 시스템, 실시간 운영체제, 실시간 통신

허 신

1973년 서울대학교 전기공학과 졸업(공학사). 1979년 Univ. of Southern California 전자계산학 석사학위 취득. 1986년 Univ. of South Florida 전자계산학 박사학위 취득. 1986년 ~ 1988년 The Catholic University of America 조교수. 1988년 ~ 현재 한양대학교 전자컴퓨터공학부 교수. 관심분야는 분산처리 시스템, 결함허용 시스템, 실시간 운영체제