

멀티프로세서 시스템에서의 실시간 태스크 스케줄링 연구 동향과 도전 과제

성균관대학교 ■ 이진규*

1. 서 론

수행하는 작업을 논리적으로 정확(Logical correctness)하게 마감시간 내에 완료(Timeliness) 해야 하는 시스템을 실시간 시스템(Real-time system)이라고 한다. 이러한 시스템은 의료, 국방, 항공·우주 등 임무수행/안전에 필수적인 태스크(Mission/safety-critical task)를 다루는 산업에서 흔히 쓰인다. 예를 들면, 의료 수술 도중에 로봇 팔의 경로 계산이 제시간에 끝나지 못하면 환자의 생명이 위험할 수 있으며, 자율 주행 자동차의 전방 사물 인식이 마감시간 내 끝나지 못하면 보행자의 안전을 보장할 수 없다. 이러한 태스크들은 물리적인 기기의 제어를 다루는 계산 작업과 주로 관련되므로, 같은 작업을 (다른 입력에 대해서) 주기적으로 반복하고, 마감시간 내 작업 완료 보장을 위해 수행시간의 상한 값을 사전에 분석해 놓으며, 반복되어 실행되는 주기적인 작업은 정해진 시간 내에 반드시 완료되어야 한다.

마감시간 내 작업 완료 보장을 위한 스케줄링은 실시간 스케줄링(Real-time scheduling)이라고 불리며, 여러 실시간 태스크들이 하나의 플랫폼에서 수행될 때 (i) 어떤 태스크를 언제 수행할지를 결정하는 ‘알고리즘’을 설계하고 (ii) 그러한 알고리즘 하에서 모든 태스크가 마감시간 내에 완료됨을 보장하는 ‘스케줄 가능성 분석 기법’을 개발하는 연구 분야를 뜻한다. 이러한 연구는 단일 프로세서(Uniprocessor) 환경에서 먼저 시작되어[1], 이에 대한 충분한 연구 결과를 바-

탕으로 실제 시스템에 성공적으로 구현되었다. 또한 멀티 코어(Multi-core)를 탑재한 기기가 보편화 되면서, 멀티프로세서(Multiprocessor) 환경 하에서 실시간 스케줄링 연구가 활발히 이루어져 왔다.

본 논문에서는 실시간 시스템의 기본적인 태스크 모델을 대상으로 하여, 멀티프로세서 환경 하에서의 실시간 스케줄링 이론 연구 동향을 살펴보고 이에 대한 도전 과제를 논의하고자 한다. 첫 번째로, 마감시간이 주기와 동일한 함축적 마감시간 태스크(Implicit-deadline task)로 이루어진 집합에 대해서는, 많은 최적 알고리즘들이 개발되었으므로 선점 횟수를 줄이는 등 기 개발된 최적 알고리즘들을 실제 시스템에 활용 가능하게 하는 연구가 필요하다. 두 번째로, 마감시간이 주기보다 작거나 같은 제한된 마감시간 태스크(Constrained-deadline task)로 이루어진 집합에 대해서는, 아직 최적 알고리즘이 개발되지 않았고 최적 알고리즘의 스케줄링 성능이 어디까지 도달 가능한지에 대한 경계조차 밝혀진 바가 없으므로, 이에 대한 이론 연구가 도전 과제이다. 두 종류의 태스크 집합에 대한 연구 동향과 도전 과제는 본문에서 자세히 논의 하겠다.

본 논문의 구성은 다음과 같다. 2장에서는 실시간 태스크의 기본 모델과 실시간 스케줄링에 대한 배경 지식을 설명한다. 3장과 4장에서는 각각 함축적 마감시간 태스크 집합과 제한된 마감시간 태스크 집합에 대한 연구 동향과 도전 과제를 살펴본다. 마지막으로 5장에서는 본 논문을 마무리한다.

2. 실시간 태스크의 기본 모델 및 배경 지식

본 논문에서는 실시간 시스템에서 널리 사용되는 산발적 태스크 모델(Sporadic task model)[2]을 다룬다. 이 모델 하에서 하나의 실시간 태스크 τ_i 는 다음과 같

* 종신회원

† 본 연구는 2016년도 정부(교육부)와 2017년도 정부(미래창조과학부)의 지원으로 한국연구재단의 지원을 받아 수행된 연구임 (2016R1D1A1B03930580, 2017R1A2B2002458, 2017H1D8A2031 628). 본 연구는 또한 과학기술정보통신부 및 정보통신기술진흥센터의 SW중심대학지원사업의 연구결과로 수행되었음 (2015-0-00914).

이 표현된다. 그림 1에 나타나 있듯이, τ_i 가 만들어 내는 연속된 작업(Job) 사이의 최소 시간 간격(또는 태스크 주기)을 T_i 로 표현하고, τ_i 가 만들어 내는 작업의 최대 수행 시간(Worst-case execution time)을 C_i 로 표현하며, τ_i 가 만들어 내는 작업이 시작된 후 수행이 끝나야 하는 마감시간까지의 시간을 D_i 로 표현한다. T_i 와 D_i 의 관계에 따라서, 우리는 $D_i=T_i$ 를 만족하는 함축적 마감시간 태스크(Implicit-deadline task)의 집합과 $D_i \leq T_i$ 를 만족하는 제한된 마감시간 태스크(Constrained-deadline task)의 집합, 이렇게 두 가지의 태스크 집합을 고려한다. 예를 들면 그림 1은 T_1 , C_1 , D_1 가 각각 5, 2, 4인 제한된 마감시간 태스크 τ_1 을 나타내며, 이 태스크는 최소 5라는 시간 간격 마다 새 작업을 도착시키고, 도착된 작업의 최대 수행 시간은 2이며, 각 작업은 도착된 지 4라는 시간 안에 수행을 반드시 완료해야 한다.

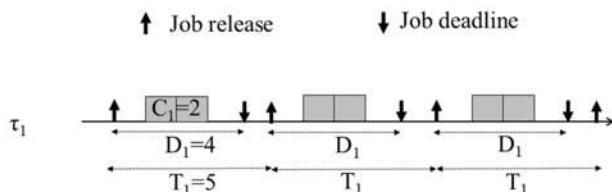


그림 1 T_i , C_i , D_i 로 표현되는 실시간 태스크 및 이 태스크가 만들어 내는 작업들의 예시

또한 본 논문에서는 선점형 태스크(Preemptive task)를 다루므로 각 태스크가 만들어내는 작업들은 언제든지 선점이 가능하며, 선점 오버헤드는 고려하지 않는다. 또한 본 논문은 m 개의 동종 프로세서(Homogeneous processors)로 이루어진 플랫폼을 대상으로 하고, 각 작업들은 어느 프로세서에서도 수행을 시작 할 수 있으며, 도중에 다른 프로세서로 옮겨서 작업을 수행하는 것도 가능하다. 이러한 스케줄링 방법을 전역 스케줄링(Global scheduling)이라고 한다.

주어진 태스크 집합이 주어진 플랫폼 하에서 주어진 알고리즘으로 스케줄을 하였을 때 ‘스케줄 가능하다’는 의미는 (T_i, C_i, D_i) 로 표현되는 각 태스크 집합이 만들어내는 어떠한 연속적인 작업들에 대해서도 항상 모든 작업들이 마감시간 내 수행 완료됨을 뜻한다. 실시간 스케줄링의 연구 분야에서 ‘알고리즘’은 여러 태스크들이 만들어 내는 많은 작업들 중에 어떤 작업을 언제 실행하는지를 결정하는 역할을 하며, ‘스케줄 가능성 분석 기법’은 이러한 알고리즘으로 작업들의 스케줄을 정했을 때 스케줄 가능함을 이론적으로 보장하는 역할을 수행 한다.

3. 함축적 마감시간 태스크 집합에 대한 실시간 스케줄링 연구 동향과 도전과제

함축적 마감시간 태스크 집합이란 주기와 마감시간이 일치하는 태스크($D_i=T_i$)로만 이루어진 집합을 의미하여 본 장에서는 이러한 태스크 집합을 위한 실시간 스케줄링 연구 동향과 이에 대한 도전 과제를 논의하겠다.

단일 프로세서 환경에서 널리 연구되고 좋은 성능을 보였던 EDF(Earliest Deadline First)[1]와 RM(Rate Monotonic)[1]이 멀티프로세서 환경에서 성능이 좋지 않음은 이론적으로 밝혀진 사실이다[3]. 따라서, EDF와 RM과 같은 기본적인 알고리즘 대신 멀티프로세서 환경에 특화된 스케줄링 알고리즘을 개발하는 연구가 많이 진행되어 왔다. 이러한 연구 중에서 멀티프로세서 환경에서 최적(Optimal)의 스케줄링을 달성하는 연구는 Pfair[4]로부터 시작되었다고 할 수 있다. Pfair의 아이디어는 간단하다. 최소 실행 시간 단위가 1인 환경에서 모든 작업의 수행시간이 1이면 EDF 알고리즘이 멀티프로세서에서 최적의 스케줄링이라는 이론적 결과를 활용하여[5], 모든 작업을 수행시간이 1인 부작업(Sub-job)로 나누고 이러한 부작업들을 EDF로 스케줄 하는 것이다. 예를 들면 주기와 마감시간이 8이고, 수행시간이 4인 작업은 주기와 마감시간이 2이고, 수행시간이 1인 부작업으로 분할되어 EDF에 의해 스케줄 된다. 그런데, 하나의 부작업의 수행시간을 1로 만들기 위해 주기와 마감시간이 정수로 쪼개지지 않으면 어떻게 될까? 예를 들어 주기와 마감시간이 11이고 수행시간이 8인 작업이 있다면, 주기와 마감시간을 어떻게 정수로 분할할 것인지에 대한 문제가 남아있다. Pfair에서는 그림 2와 같이 i 번째 부작업의 가상 도착(Virtual release) 시간을 $Floor(11/8*(i-1))$ 으로 하고, 마감시간을 $Ceil(11/8*i)$ 으로 설정한다. 예를 들면 그림 2의 첫 번째 부작업의 가상 도착시간은 $Floor(11/8*0)=0$ 이고 마감시간은 $Ceil(11/8*1)=2$ 이며, 여섯 번째 부작업의 가상 도착 시간은 $Floor(11/8*5)=6$ 이고 마감시간은 $Ceil(11/8*6)=9$ 이다. 여기서, 실제 i 번째 부작업의 도착 시간은 자신의 가상 도착 시간과 이전 부작업의 마감 시간 중 늦은 것으로 결정 된다. 이렇게 하면 태스크들의 C_i/T_i 의 합이 m 이하인 모든 태스크 집합들은 Pfair로 스케줄 가능하다.

Pfair는 이렇듯 (증명은 매우 복잡하지만) 간단한 방법으로 최적 스케줄링을 달성하였지만, 많은 선점을 일으킨다는 점이 단점이다. 이러한 단점을 극복하기 위해서 또는 좀 더 직관적인 스케줄링 개발을 위해서,

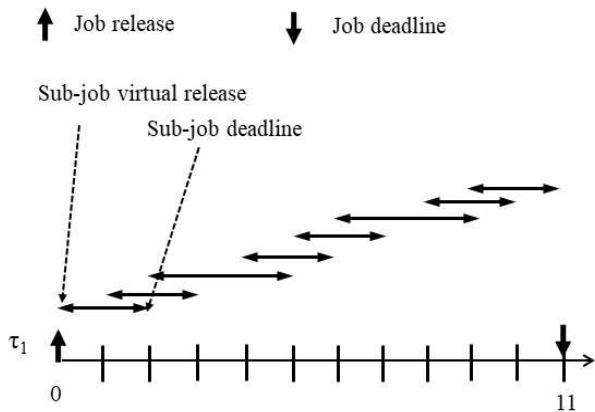


그림 2 Pair 알고리즘의 동작 예시: $T_1=11$, $C_1=8$, $D_1=11$

이후에 많은 알고리즘이 개발되었으며 대표적인 알고리즘에는 ER-Fair[6], LLREF[7], EKG[8], DP-Wrap[9], RUN[10], U-EDF[11], QPS[12] 등이 있다. 이러한 알고리즘들은 최적을 달성하면서 선점이 적고, 다양한 환경(주기적 태스크 모델 보다 까다로운 산발적 태스크 모델, 미래의 작업 도착 시간을 모르는 환경 등)에서도 동작할 수 있도록 진화하고 있다.

또한 함축적 마감시간 태스크 집합에 대한 실시간 스케줄링 연구의 또 다른 방향은 매우 실용적인 알고리즘의 개발이다. 이러한 알고리즘은 이론적으로 최적을 달성하지는 못하지만, 최적에 근접한(Near-optimal) 경험적인 성능을 보이면서도 앞서 언급한 최적 알고리즘에 비해 선점의 수가 획기적으로 적은 등 실제 시스템에 적용하기가 용이한 알고리즘이다. 현재 이러한 방향에서 가장 우수한 스케줄링은 2016년에 발표된 연구이며[13], 이전에도 [14-16] 등과 같은 많은 연구가 존재하였다.

이렇듯, 선점 오버헤드를 무시했을 때 이론적으로 최적을 달성하면서 좀 더 선점이 적고 다양한 환경에서도 동작할 수 있는 스케줄링의 개발과, 실용성을 극대화하며 최적에 근접한 스케줄링의 개발이라는 두 가지 방향에 대해 향후 도전과제는 다음과 같다. 첫 번째, 최적의 스케줄링에 대해서는 지금보다 선점이 더 적은 스케줄링 알고리즘의 설계가 가능할 것으로 예상된다. 두 번째, 실용성을 극대화 하는 스케줄링에 대해서는 경험적 근접 최적 달성을 뛰어 넘어 이론적으로 근접 최적을 보장하는 연구가 필요하다. 또한 이러한 두 가지 부류의 연구가 계속 진행되면, 결국에는 전자는 최적을 유지한 채로 실용성을 향상시키고, 후자는 실용성에 초점을 맞춘 채로 최적을 향해 스케줄링 성능이 향상해 가므로, 궁극적으로는 하나의 연구가 되어 실용성과 최적성을 동시에 갖추는 스케줄링을

개발하게 될 것이다. 그러나 함축적 마감시간 태스크 집합에 대한 실시간 스케줄링 연구는 제한된 마감시간 태스크 집합에 비해 그 동안 연구가 많이 진행되었음을 고려해 볼 때, 이러한 스케줄링 개발은 난이도 있는 연구가 될 것이라 예상된다.

4. 제한된 마감시간 태스크 집합에 대한 실시간 스케줄링 연구 동향과 도전과제

제한된 마감시간 태스크 집합이란 각 마감시간이 주기보다 작거나 같은 태스크($D_i \leq T_i$)로 이루어진 집합을 의미한다. 본 장에서는 이러한 제한된 마감시간 태스크 집합에 대한 실시간 스케줄링 연구 동향과 도전 과제를 논의하겠다.

제한된 마감시간 태스크 집합을 효과적으로 스케줄링 하기 위해 많은 스케줄링 알고리즘이 개발되어 왔다. 이러한 알고리즘에는 EDF와 RM의 단점을 보완하기 위해 개발된 EDZL[17]과 FPZL[18] 등을 비롯하여, LLF[19], EDF-CF[20], SPDF[21], EQDF[22], EQDZL[23] 등의 헤리스틱(Heuristic) 스케줄링 알고리즘이 있다. 함축적 마감시간 태스크 집합에 대해서는 최적성을 유지하는 하에서 알고리즘을 개발하였기 때문에 스케줄 가능성 분석 기법이 필요 없는 반면에(즉, 자동적으로 최적을 보장하는 스케줄 가능성 분석 기법이 개발되었던 반면에), 제한된 마감시간 태스크 집합에 대해서는 알고리즘 자체의 개발뿐만 아니라 이에 대한 우수한(Tight) 스케줄 가능성 분석 기법 개발도 어려운 문제이다.

한편, 함축적 마감시간 태스크 집합을 위해 개발되었던 최적 알고리즘들의 대부분은 제한된 마감시간 태스크 집합에도 적용이 가능하지만, 이 경우 최적성을 잃게 된다. 즉, 함축적 마감시간 태스크 집합에 대해서는 $\sum C_i / T_i \leq m$ 을 만족하면 스케줄 가능했으나(이를 어기면 어떤 알고리즘으로도 스케줄 가능하지 않으므로 최적성을 보장함), 이를 제한된 마감시간 태스크 집합에 적용하면 $\sum C_i / D_i \leq m$ 을 만족하면 스케줄 가능하다는 조건으로 바뀌게 된다. 하지만 $\sum C_i / D_i > m$ 을 만족하는 태스크 집합들 중에도 스케줄 가능한 집합이 존재하므로, $\sum C_i / D_i \leq m$ 을 만족하는 모든 태스크 집합에 대해 스케줄 가능하다는 것이 최적을 의미하지는 않는다.

따라서, 현재 제한된 마감시간 태스크 집합에 대한 멀티프로세서 실시간 스케줄링 성능은 다음과 같다. 첫 번째, 함축적 마감시간 태스크 집합을 위해 개발되었던 최적 알고리즘을 통해 $\sum C_i / D_i \leq m$ 를 만족하는 태스크 집합은 모두 스케줄 가능하다. 그러한 이러한 알고리즘은 $\sum C_i / D_i > m$ 을 만족하는 태스크 집합에 대

해서는 단 하나의 태스크 집합의 스케줄 가능성도 보장하지 못한다. 두 번째, 기존에 개발된 휴리스틱 알고리즘들은 $\sum C_i/D_i \leq m$ 을 만족하는 태스크 집합 모두에 대해 스케줄 가능한 것은 아니지만, $\sum C_i/D_i > m$ 을 만족하는 태스크 집합의 일부에 대해서도 스케줄 가능성을 보장한다. 따라서, 이 두 가지 부류의 스케줄링 알고리즘의 결과를 함께 사용하면, $\sum C_i/D_i \leq m$ 을 만족하는 태스크 집합을 모두 스케줄 가능하게 할 수 있고, $\sum C_i/D_i > m$ 을 만족하는 태스크 집합의 일부를 스케줄 가능하게 할 수 있으며, 이는 그림 3에 나타나 있다. 그림 3에서 FF-DBF[24]란 이것을 만족하지 못하는 태스크 집합은 절대로 스케줄 가능할 수 없음이 밝혀진 필수 실행 가능성 테스트(Necessary feasibility test)이며, 이를 만족하는 태스크 집합들이 모두 스케줄 가능한지는 밝혀진 바가 없다. 즉, FF-DBF는 달성할 수 있는 최대의 스케줄 성능이라고 할 수 있다. 이에 따라, 그림 3의 물음표 부분에 해당하는 태스크 집합들은 현재 어떤 알고리즘과 스케줄 분석 기법으로도 스케줄 가능성성이 보장되지 않았으며, 또한 스케줄이 불가능하다고 증명 된 바도 없다.

이러한 현재 상황을 고려해 볼 때 제한된 마감시간 태스크 집합을 위한 멀티프로세서 스케줄링의 당면한 도전 과제는 $\sum C_i/D_i > m$ 에 해당하는 태스크 집합을 되도록 많이 커버하는 알고리즘 설계와 이에 대한 스케줄 가능성 분석 기법 개발이다. 이러한 알고리즘 설계와 스케줄 가능성 분석 기법 개발과 관련하여, 본 연구진은 기존에 휴리스틱 알고리즘들 보다 많은 $\sum C_i/T_i > m$ 의 태스크 집합을 스케줄 가능하게 하는 스케줄링 구조를 제시하여 그림 3의 물음표 부분에 해당하는 태스크 집합들을 어느 정도 커버하는 등 스케줄 가능성 측면에서 괄목할만한 성능향상을 최근에 이루었다[25].

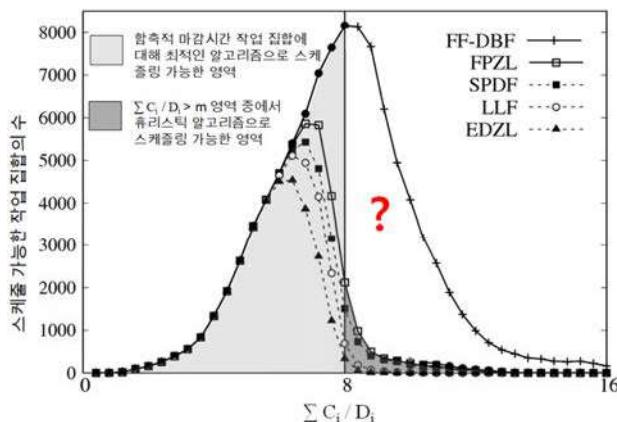


그림 3 제한된 마감시간 태스크 집합에 대한 멀티프로세서 실시간 스케줄 가능성 성능($m=8$)

그럼에도 불구하고, 향후 추가적인 연구들을 통해서 더 많은 태스크 집합에 대한 스케줄 가능성 보장이 가능할 것이라고 예상되며, 이는 향후 도전과제이다. 또한 이러한 연구를 통해 FF-DBF 보다 성능이 좋은(Tight) 필수 실행 가능성 테스트를 설계하는 것도 도전 과제이다.

또한 함축적 마감시간 태스크 집합의 경우와 마찬가지로 선점을 줄이고, 다양한 환경에서 동작하도록 하는 스케줄링 알고리즘 설계와 이에 대한 스케줄 가능성 분석 기법 개발 역시 향후 과제라고 할 수 있다.

5. 결 론

본 논문에서는 멀티프로세서 환경에서의 실시간 태스크 스케줄링에 관한 연구 동향을 살펴보고, 풀리지 않은 도전 과제들을 소개하였다. 함축적 마감시간 태스크 집합의 스케줄링에 대해서는, 많은 최적 알고리즘이 이미 개발되었으므로 선점이 적은 등 좀 더 실용적인 알고리즘 개발을 위한 연구가 필요하다. 반면에 제한된 마감시간 태스크 집합의 스케줄링에 대해서는, 최적 알고리즘이 존재하지 않고 최적 알고리즘의 스케줄링 성능이 어디까지 도달 가능한지에 대한 경계조사 밝혀진 바가 없으므로, 스케줄 가능성 측면에서 기존 연구 보다 성능이 좋은 스케줄링 알고리즘 설계와 이에 대한 스케줄 가능성 분석 기법 개발이 필요하다.

참고문헌

- [1] C. Liu and J. Layland, "Scheduling algorithms for multi-programming in a hard-real-time environment", Journal of the ACM, vol. 20, no. 1, pp. 46-61, 1973.
- [2] A. Mok, "Fundamental design problems of distributed systems for the hard-real-time environment", Ph.D. dissertation, Massachusetts Institute of Technology, 1983.
- [3] S. K. Dhall and C. L. Liu, "On a real-time scheduling problem", Operations Research 26(1), 1978, pp.127-140.
- [4] S. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, "Proportionate progress: a notion of fairness in resource allocation", Algorithmica, vol. 15, no. 6, pp. 600-625, 1996.
- [5] J. Lee, A. Easwaran, I. Shin and I. Lee, "Zero-laxity based real-time multiprocessor scheduling", Journal of Systems and Software, vol. 84, pp. 2324-2333, 2011.
- [6] J. H. Anderson and A. Srinivasan, "Early-release fair scheduling", in Proceedings of Euromicro Conference on Real-Time Systems (ECRTS), 2000, pp. 35-43.
- [7] H. Cho, B. Ravindran, and E. D. Jensen, "An optimal

- real-time scheduling algorithm for multiprocessors”, in Proceedings of IEEE Real-Time Systems Symposium (RTSS), 2006, pp. 101-110.
- [8] B. Andersson and E. Tovar, “Multiprocessor scheduling with few preemptions”, in Proceedings of IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2006, pp. 322-334.
 - [9] G. Levin, S. Funk, C. Sadowski, I. Pye, and S. Brandt, “DP-FAIR: A simple model for understanding optimal multiprocessor scheduling”, in Proceedings of Euromicro Conference on Real-Time Systems (ECRTS), 2010, pp. 3-13.
 - [10] P. Regnier, G. Lima, E. Massa, G. Levin, and S. Brandt, “RUN: Optimal multiprocessor real-time scheduling via reduction to uniprocessor”, in Proceedings of IEEE Real-Time Systems Symposium (RTSS), 2011, pp. 104-115.
 - [11] G. Nelissen, V. Berten, V. Nelis, J. Goossens, and D. Milojevic, “UEDF: An unfair but optimal multiprocessor scheduling algorithm for sporadic tasks”, in Proceedings of Euromicro Conference on Real-Time Systems (ECRTS), 2012, pp. 13-23.
 - [12] E. Massa, G. Lima, P. Regnier, G. Levin, and S. Brandt, “Quasipartitioned scheduling: optimality and adaptation in multiprocessor realtime systems”, Real-Time Systems, vol. 52, no. 5, pp. 566-597, 2016.
 - [13] B. B. Brandenburg and M. Gul, “Global Scheduling Not Required: Simple, Near-Optimal Multiprocessor Real-Time Scheduling with Semi-Partitioned Reservations”, in Proceedings of IEEE Real-Time Systems Symposium (RTSS), 2016, pp. 99-110.
 - [14] A. Bastoni, B. Brandenburg, and J. Anderson, “Is semi-partitioned scheduling practical?” in Proceedings of Euromicro Conference on Real-Time Systems (ECRTS), pp. 125-135, 2011.
 - [15] S. Kato, N. Yamasaki, and Y. Ishikawa, “Semi-partitioned scheduling of sporadic task systems on multiprocessors”, in Proceedings of Euromicro Conference on Real-Time Systems (ECRTS), pp. 249-258, 2009.
 - [16] A. Burns, R. Davis, P. Wang, and F. Zhang, “Partitioned EDF scheduling for multiprocessors using a C=D task splitting scheme”, Real-Time Systems, vol. 48, pp. 3-33, 2012.
 - [17] M. Cirinei and T. P. Baker, “EDZL scheduling analysis”, in Proceedings of Euromicro Conference on Real-Time Systems (ECRTS), 2007, pp. 9-18.
 - [18] R. I. Davis and A. Burns, “FPZL schedulability analysis”, in Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2011, pp. 245-256.
 - [19] J. Lee, A. Easwaran, and I. Shin, “LLF schedulability analysis on multiprocessor platforms”, in Proceedings of IEEE Real-Time Systems Symposium (RTSS), 2010, pp. 25-36.
 - [20] J. Lee, A. Easwaran, and I. Shin, “Maximizing contention-free executions in multiprocessor scheduling”, in Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2011, pp. 235-244.
 - [21] H. S. Chwa, H. Back, S. Chen, J. Lee, A. Easwaran, I. Shin, and I. Lee, “Extending task-level to job-level fixed priority assignment and schedulability analysis using pseudo-deadlines”, in Proceedings of IEEE Real-Time Systems Symposium (RTSS), 2012, pp. 51-62.
 - [22] H. Back, H. S. Chwa, and I. Shin, “Schedulability analysis and priority assignment for global job-level fixed-priority multiprocessor scheduling”, in Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012, pp. 297-306.
 - [23] H. S. Chwa, H. Back, J. Lee, K. M. Phan, and I. Shin, “Capturing urgency and parallelism using quasi-deadlines for real-time multiprocessor scheduling”, Journal of Systems and Software, vol. 101, pp. 15-29, 2015.
 - [24] T. P. Baker and M. Cirinei, “A necessary and sometimes sufficient condition for the feasibility of sets of sporadic hard-deadline tasks”, in Proceedings of IEEE Real-Time Systems Symposium (RTSS), 2006, pp. 178-190.
 - [25] H. Baek, H. S. Chwa and J. Lee, “Beyond Implicit-Deadline Optimality: A Multiprocessor Scheduling Framework for Constrained-Deadline Tasks”, To appear in Proceedings of IEEE Real-Time Systems Symposium (RTSS), 2017.

■ 약력



이진규

2004 한국과학기술원(KAIST) 전산학과 졸업(학사)
2006 한국과학기술원(KAIST) 전산학과 졸업(석사)
2011 한국과학기술원(KAIST) 전산학과 졸업(박사)
2011~2014 미국 University of Michigan 방문연구원/연구원

2014~현재 성균관대학교 컴퓨터 공학과 조교수
관심분야: 실시간 시스템, 배터리 관리 시스템 소프트웨어, 모바일 컴퓨팅
Email: jinkyu.lee@skku.edu