(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.
G06F 9/46 (2006.01)

(45) 공고일자 2006년12월27일
(11) 등록번호 10-0661851
(24) 등록일자 2006년12월20일

(21) 출원번호 10-2005-0008932
(22) 출원일자 2005년02월01일
심사청구일자 2005년02월01일

(65) 공개번호 10-2006-0064438
(43) 공개일자 2006년06월13일

(30) 우선권주장 1020040103023 2004년12월08일 대한민국(KR)

(73) 특허권자 한국전자통신연구원
대전 유성구 가정동 161번지

(72) 발명자 최승민
대전광역시 유성구 가정동 236-1번지

(74) 대리인 유미특허법인

(56) 선행기술조사문헌
KR1020040046863 A
* 심사관에 의하여 인용된 문헌

심사관 : 안철용

전체 청구항 수 : 총 25 항

(54) 플랫폼의 소비 전력 관리 방법 및 그 플랫폼

(57) 요약

본 발명은 플랫폼의 소비 전력 관리 방법 및 그 플랫폼에 관한 것이다. 본 발명에 따르면, 작업 완료를 위해 디바이스에 주어지는 명령어의 집합인 잡(job)의 스케줄링을 통해 플랫폼이 소비 전력을 관리할 때, 소정의 이벤트가 발생하면, 상기 이벤트가 작업부하에 따라 분석된다. 그리고 상기 잡(job)의 작업 부하에 따라 상기 잡(job)이 실행 완료되어야 하는 시간 제약, 상기 잡(job)의 버퍼링에 사용되는 메모리 크기 및 상기 버퍼링에 소비되는 에너지 중 하나 이상을 고려하여, 해당 디바이스가 실행해야 할 잡(job)의 스케줄링 주기 T가 계산되고, 그 결과에 따른 잡(job) 스케줄을 기초로 각 디바이스의 파워 상태가 조절된다. 따라서, 잡(job) 스케줄링을 통한 전력관리에 있어서 실질적인 에너지 절약 효과가 있다.

대표도

도 6

특허청구의 범위

청구항 1.

작업 완료를 위해 디바이스에 주어지는 명령어의 집합인 잡(job)의 스케줄링을 통해 플랫폼이 소비 전력을 관리하는 방법에 있어서,

a)소정의 이벤트가 발생하면,상기 이벤트를 작업부하에 따라 분석하는 단계;

b) 상기 잡(job)의 작업 부하에 따라 상기 잡(job)이 실행 완료되어야 하는 시간 제약, 상기 잡(job)의 버퍼링에 사용되는 메모리 크기 및 상기 버퍼링에 소비되는 에너지 중 하나 이상을 고려하여, 특정 시간대로 모아서 잡(job)을 실행하는 스케줄인 버스트(burst) 스케줄 혹은 잡(job)이 발생하는 즉시 처리하는 스케줄인 분할(split) 스케줄에 따라 해당 디바이스가 실행해야 할 잡(job)의 스케줄링 주기 T를 계산하는 단계;

c) 상기 b)단계의 결과를 기초로 한 잡(job) 스케줄에 따라 각 디바이스의 파워상태를 조절하는 단계를 포함하고,

상기 메모리 크기는,

상기 메모리가 버스트(burst) 스케줄에 따라 리퀘스트(request)를 버퍼링하는데 소비하는 에너지가 버스트(burst) 스케줄에 따라 절약되는 에너지보다 적을 때의 메모리 크기인

플랫폼의 소비 전력 관리 방법.

청구항 2.

제1항에 있어서,

상기 b) 단계는,

상기 메모리 크기로 인해 제약받는 스케줄 주기인 제1 주기 및 상기 시간 제약으로 인한 스케줄 주기인 제2 주기 중 하나 이상을 고려하여, 스케줄링 주기 T를 계산하는 단계인 플랫폼의 소비 전력 관리 방법.

청구항 3.

제2항에 있어서,

상기 b)단계는,

작업 부하에 따라 상기 제1 주기 혹은 상기 제2 주기를 최적 스케줄링 주기 T 결정에 제약조건으로 고려하는 단계인 플랫폼의 소비 전력 관리 방법.

청구항 4.

제3항에 있어서,

상기 b) 단계는,

상기 잡(job)을 분할(split)스케줄링 할 때의 소비 에너지 E_{split}과 버스트(burst) 스케줄링 할 때의 소비 에너지 E_{burst}가 같을 때의 스케줄 주기를 기초로, 상기 제1 주기 혹은 상기 제2 주기를 제약조건으로 고려하여 스케줄링 주기 T를 계산하는 단계인 플랫폼의 소비전력 관리 방법.

청구항 5.

제4항에 있어서,

상기 b) 단계는,

상기 잡(job)을 분할(split)스케줄링 할 때의 소비 에너지 E_{split}과 버스트(burst) 스케줄링 할 때의 소비 에너지 E_{burst}가 같을 때의 스케줄 주기값이, 상기 제1 주기 혹은 상기 제2 주기의 값보다 크거나 같을 때, 분할(split) 스케줄링을 선택하는 단계인 플랫폼의 소비전력 관리 방법.

청구항 6.

제4항에 있어서,

상기 b) 단계는,

상기 잡(job)을 분할(split)스케줄링 할 때의 소비 에너지 E_{split}과 버스트(burst) 스케줄링 할 때의 소비 에너지 E_{burst}가 같을 때의 스케줄 주기값이, 상기 제1 주기 혹은 상기 제2 주기의 값보다 작을 때,

상기 제1 주기 및 상기 제2 주기 중 더 작은 주기 값에 따른 버스트(burst) 스케줄링을 선택하는 단계인 플랫폼의 소비 전력 관리 방법.

청구항 7.

제4항에 있어서,

상기 E_{split}과 E_{burst}가 같을 때의 주기는 $E_{split} = E_{burst}$ 일 때의 스케줄 주기 T로부터 도출되며 상기 E_{split}과 상기 E_{burst}는 각각

$$E_{split} = \frac{T \times S \times N}{BW} \times p_{bw} + (T - \frac{T \times S \times N}{BW}) \times p_{iw}$$

$$E_{burst} = \frac{T \times S \times N}{BW} \times p_{bw} + (T - \frac{T \times S \times N}{BW} - t_0) \times p_s + e_0 + E_{Buf}$$

$$E_{Buf} = \{ Integer \quad (\frac{S \times N}{nKbyte}) + 1 \} \times T \times p_{nKbuf}$$

(S : 작업이 처리 요청하는 데이터 크기, N : 작업의 데이터 요청 횟수

e_o : 전력상태천이 소모 에너지 t_o : 전력상태 천이에 따른 지연시간

BW : 장치의 자료 처리능력(bps, fps 등)

P_{bw}: 비지 워킹(busy working) 상태의 소모전력

P_{iw} : 아이들 워킹(idle working) 상태의 소모전력

P_S: 슬리핑(sleeping) 상태의 소모전력

P_{nKbuf}: n Kbyte 버퍼 메모리의 소모 전력)

인 플랫폼의 소비 전력 관리 방법.

청구항 8.

제4항에 있어서,

상기 E_{split}과 E_{burst}가 같을 때의 주기는,

$$T_{eq} = \frac{e_{0} - t_{0} \times p_{s}}{p_{i} - p_{s}} \times \frac{1}{1 - \frac{N \times S}{BW}}$$

(S : 작업이 처리 요청하는 데이터 크기, N : 작업의 데이터 요청 횟수

e_o : 전력상태천이 소모 에너지 t_o : 전력상태 천이에 따른 지연시간

BW : 장치의 자료 처리능력(bps, fps 등)

P_i : 아이들 워킹(idle working) 상태의 소모전력

P_S : 슬리핑(sleeping) 상태의 소모전력)

인 플랫폼 소비 전력 관리 방법.

청구항 9.

제2항에 있어서,

상기 제1 주기는,

$$T_{buf} = \frac{Buffer \quad Limit}{S \times N}$$

(S : 작업이 처리 요청하는 데이터 크기, N : 작업의 데이터 요청 횟수)

로부터 구해지는 것을 특징으로 하는 플랫폼 소비 전력 관리 방법.

청구항 10.

삭제

청구항 11.

제1항에 있어서,

먼저 계산된 제1 디바이스의 최적 스케줄링 주기 T는 제2 디바이스의 상기 최적 스케줄링 주기 T를 구할 때의 상기 시간 제약으로 고려되는 플랫폼의 소비 전력 관리 방법.

청구항 12.

제1항에 있어서,

상기 이벤트는,

상기 작업의 생성, 상기 작업의 소멸, 상기 메모리 크기의 변화, 상기 시간 제약의 변화, 상기 작업의 데이터 요청 횟수(N)의 변화 및 상기 작업의 처리 요청 데이터 크기(S)의 변화 중 하나 이상을 포함하는 플랫폼의 소비 전력 관리 방법.

청구항 13.

제12항에 있어서,

상기 작업 부하는

해당 작업의 데이터 요청 횟수(N) 및 처리 요청 데이터 크기(S)로부터 결정되는 플랫폼의 소비 전력 관리 방법.

청구항 14.

작업 완료를 위해 디바이스에 주어지는 명령어의 집합인 잡(job)의 스케줄링을 통해 소비전력을 관리하는 플랫폼에 있어서,

소정의 이벤트를 발생시키는 응용부 ; 및

상기 응용부로부터 상기 이벤트 발생정보를 전달받은 때, 상기 잡(job)의 작업부하에 따라 시간 제약, 메모리 크기 및 상기 버퍼링에 소비되는 에너지 중 하나 이상을 고려하여, 특정 시간대로 모아서 잡(job)을 실행하는 스케줄인 버스트(burst) 스케줄 혹은 잡(job)이 발생하는 즉시 처리하는 스케줄인 분할(split) 스케줄에 따라 해당 디바이스가 실행해야 할 잡(job)의 스케줄링 주기 T를 도출한 후 해당 디바이스의 파워 상태를 조절하는 운영체제부를 포함하고,

상기 메모리 크기는 상기 메모리가 버스트(burst) 스케줄에 따라 리퀘스트(request)를 버퍼링하는데 소비하는 에너지가 버스트(burst) 스케줄에 따라 절약되는 에너지보다 적을 때의 메모리 크기인

잡(job)의 스케줄링을 통해 소비전력을 관리하는 플랫폼.

청구항 15.

제14항에 있어서,

상기 운영체제부는,

작업 생성 정보를 전달받고, 현재 실행중인 작업에 대한 모니터링, 관리 및 작업 상태 정보 전달을 수행하는 프로세스 매니저;

스케줄 주기에 따라 상기 잡(job)을 스케줄링하는 스케줄러;

상기 잡(job)이 실행되는 디바이스의 구동을 조정하는 디바이스 드라이버; 및

작업 부하에 따라, 시간 제약, 메모리 크기 및 상기 버퍼링에 소비되는 에너지 중 하나 이상을 고려하여 최적 스케줄링 주기 T를 도출하고, 상기 프로세스 매니저로부터 전달받은 작업(task) 상태 정보 및 상기 스케줄러부터 전달받은 잡(job) 스케줄을 기초로 상기 디바이스 드라이버가 해당 디바이스의 파워상태를 조절하도록 하는 파워매니저를 포함하는

잡(job)의 스케줄링을 통해 소비전력을 관리하는 플랫폼.

청구항 16.

제15항에 있어서,

상기 파워매니저는,

잡(job)을 분할(split)스케줄링 할 때의 소비 에너지 E_{split}과 버스트(burst) 스케줄링 할 때의 소비 에너지 E_{burst}가 같을 때의 스케줄 주기를 기초로,

상기 메모리 크기로 인해 제약받는 스케줄 주기인 제1 주기 및 상기 시간 제약으로 인한 스케줄 주기인 제2 주기 중 하나 이상을 고려하여 최적 스케줄링 주기 T를 도출하여 상기 스케줄러에 전달되도록 하는 T 계산부; 및

상기 프로세스 매니저로부터 전달받은 작업(task) 상태 정보 및 상기 스케줄러부터 전달받은 잡(job) 스케줄을 기초로 상기 디바이스 드라이버가 해당 디바이스의 파워상태를 조절하도록 하는 파워상태셋팅부를 포함하는

잡(job)의 스케줄링을 통해 소비전력을 관리하는 플랫폼.

청구항 17.

제16항에 있어서,

상기 T계산부는,

작업 부하에 따라 상기 제1 주기 혹은 상기 제2 주기를 최적 스케줄링 주기 T 결정에 제약조건으로 고려하는

잡(job)의 스케줄링을 통해 소비전력을 관리하는 플랫폼

청구항 18.

제17항에 있어서,

상기 T계산부는,

상기 잡(job)을 분할(split)스케줄링 할 때의 소비 에너지 E_{split}과 버스트(burst) 스케줄링 할 때의 소비 에너지 E_{burst}가 같을 때의 스케줄 주기값이, 상기 제1 주기 혹은 상기 제2 주기의 값보다 크거나 같을 때, 분할(split) 스케줄링을 선택하는

잡(job)의 스케줄링을 통해 소비전력을 관리하는 플랫폼.

청구항 19.

제17항에 있어서,

상기 T계산부는,

상기 잡(job)을 분할(split)스케줄링 할 때의 소비 에너지 E_{split}과 버스트(burst) 스케줄링 할 때의 소비 에너지 E_{burst}가 같을 때의 스케줄 주기값이, 상기 제1 주기 혹은 상기 제2 주기의 값보다 작을 때,

상기 제1 주기 및 상기 제2 주기 중 더 작은 주기 값에 따른 버스트(burst) 스케줄링을 선택하는

잡(job)의 스케줄링을 통해 소비전력을 관리하는 플랫폼.

청구항 20.

제16항에 있어서,

상기 E_{split}과 E_{burst}가 같을 때의 주기는 $E_{split} = E_{burst}$ 일 때의 스케줄 주기 T로부터 도출되며 상기 E_{split}과 상기 E_{burst}는 각각

$$E_{split} = \frac{T \times S \times N}{BW} \times p_{bw} + (T - \frac{T \times S \times N}{BW}) \times p_{iw}$$

$$E_{burst} = \frac{T \times S \times N}{BW} \times p_{bw} + (T - \frac{T \times S \times N}{BW} - t_0) \times p_s + e_0 + E_{Buf}$$

$$E_{Buf} = \{ Integer \quad (\frac{S \times N}{nKbyte}) + 1 \} \times T \times p_{nKbuf}$$

(S : 작업이 처리 요청하는 데이터 크기, N : 작업의 데이터 요청 횟수

e_o : 전력상태천이 소모 에너지 t_o : 전력상태 천이에 따른 지연시간

BW : 장치의 자료 처리능력(bps, fps 등)

P_{bw}: 비지 워킹(busy working) 상태의 소모전력

P_{iw} : 아이들 워킹(idle working) 상태의 소모전력

P_s: 슬리핑(sleeping) 상태의 소모전력

P_{nKbuf}: n Kbyte 버퍼 메모리의 소모 전력)

인 잡(job)의 스케줄링을 통해 소비전력을 관리하는 플랫폼

청구항 21.

제16항에 있어서,

상기 E_{split}과 E_{burst}가 같을 때의 주기는,

$$T_{eq} = \frac{e_0 - t_0 \times P_s}{P_i - P_s} \times \frac{1}{1 - \frac{N \times S}{BW}}$$

(S : 작업이 처리 요청하는 데이터 크기, N : 작업의 데이터 요청 횟수

e_o : 전력상태천이 소모 에너지 t_o : 전력상태 천이에 따른 지연시간

BW : 장치의 자료 처리능력(bps, fps 등)

P_i : 아이들 워킹(idle working) 상태의 소모전력

P_S : 슬리핑(sleeping) 상태의 소모전력)

인 잡(job)의 스케줄링을 통해 소비전력을 관리하는 플랫폼.

청구항 22.

제16항에 있어서,

상기 제1 주기는,

$$T_{buf} = \frac{Buffer \quad Limit}{S \times N}$$

(S : 작업이 처리 요청하는 데이터 크기, N : 작업의 데이터 요청 횟수)

인 잡(job)의 스케줄링을 통해 소비전력을 관리하는 플랫폼.

청구항 23.

제14항에 있어서,

상기 T계산부는,

먼저 계산한 제1 디바이스의 최적 스케줄링 주기 T를, 제2 디바이스의 상기 최적 스케줄링 주기 T 계산에 있어서 시간 제약으로 고려하는

잡(job)의 스케줄링을 통해 소비전력을 관리하는 플랫폼.

청구항 24.

제14항에 있어서,

상기 메모리 크기는,

상기 메모리가 버스트(burst) 스케줄에 따라 리퀘스트(request)를 버퍼링하는데 소비하는 에너지가 버스트(burst) 스케줄에 따라 절약되는 에너지보다 적을 때의 메모리 크기인

잡(job) 스케줄링을 통해 소비전력을 관리하는 플랫폼.

청구항 25.

제14항에 있어서,

상기 이벤트는,

상기 작업의 생성, 상기 작업의 소멸, 상기 메모리 크기의 변화, 상기 시간 제약의 변화, 상기 작업의 데이터 요청 횟수(N)의 변화 및 상기 작업의 처리 요청 데이터 크기(S)의 변화 중 하나 이상을 포함하는

잡(job) 스케줄링을 통해 소비전력을 관리하는 플랫폼.

청구항 26.

제25항에 있어서,

상기 작업 부하는

해당 작업의 데이터 요청 횟수(N) 및 처리 요청 데이터 크기(S)로부터 결정되는 잡(job) 스케줄링을 통해 소비전력을 관리하는 플랫폼.

명세서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 플랫폼의 소비 전력 관리 방법 및 그 플랫폼에 관한 것으로, 보다 상세하게는 특정 작업 완료를 위한 명령어의 집합인 잡(job) 스케줄링을 통해 상기 플랫폼이 소비 전력을 관리하는 방법 및 그 플랫폼에 관한 것이다.

플랫폼, 특히 모바일 플랫폼들은 이동을 하면서 다양한 기능을 수행하므로 그 동작을 위한 전원을 전지로부터 공급받는다. 그러나, 제한된 용량의 전지로부터 모바일 플랫폼을 작동시킬 충분한 에너지를 공급받는 것에는 한계가 있으므로, 플랫폼 설계과정에서 전지의 용량을 증가시키는 노력과 함께 효율적인 전력관리에 대한 연구 또한 중요한 일이다.

시스템에서의 전력관리는 여러 단계에서 이루어질 수 있는데, 시스템 수준에서 OS를 통해 이루어지는 전력 관리는 특히 모바일 플랫폼과 같은 대화형 시스템에서 효과적이어서, OS 수준에서의 다양한 전력관리 기법들이 적용되어 왔다.

예를 들어, 시스템을 구성하는 장치들의 아이들(idle) 주기를 예측하여 셧-다운(shutdown)을 결정하는 기법으로서, 장치의 아이들(idle) 주기를 예측하여 셧-다운(shutdown)을 결정하는 방법이 사용되었다. 이 때, 장치가 앞으로 얼마동안 사용될 것인가에 대한 정보는 과거의 장치 사용 패턴 분석을 통해 예측된다. 따라서, 예측의 정확도에 따라 시스템의 성능과 에너지 절약 효율이 크게 변하게 된다.

또, 플랫폼의 특정 작업 완료를 위한 명령어의 집합(이하, '잡(job)'이라 한다)스케줄링을 통해 불연속적 장치나 CPU등의 아이들(idle) 주기를 연속적으로 만들어 셧-다운(shutdown) 기회를 늘이는 기법이 있었다. 이 때 수행중인 잡(job)은 장치를 사용할 수 있고, 커널(kernel)의 스케줄러에 의해 그 수행순서가 정해지므로, 스케줄링 방식에 따라 장치의 사용시간대가 변하는 것을 이용한 기법이 있었다. 즉 장치를 사용하는 잡(job)을 스케줄링 하는 것이 장치의 아이들(idle) 주기 분포에 직접적인 영향을 주기 때문에, 스케줄을 통해 장치의 아이들(idle)주기를 가능한 연속적으로 만들어 셧-다운(shutdown) 기회를 늘리는 방법이다.

기타, 메모리 소비전력을 줄이기 위해 뱅크(bank)별로 다른 전력상태를 선택하는 기법등이 적용되어 왔다.

그러나, 실질적으로 장치들에 부과되는 잡(job)의 작업부하(workload)에 따라 버스트(burst) 주기 혹은 분할(split) 주기에 따른 전력 관리의 효율성이 달라진다. 또한 스케줄에 사용 가능한 버퍼의 크기가 시스템의 상황에 따라 시스템 실행중 변하므로, 시스템의 성능을 저하시키지 않으면서 에너지를 절약하기 위해서는 현재 사용 가능한 버퍼의 크기 또한 고려 해야 한다. 또, 잡(job) 스케줄링을 통해 잡(job)의 수행순서를 바꾸게 되면 사용자가 원하는 제한시간(deadline)을 만족하지 못할 가능성이 있으므로 시간 제약에 대한 고려도 필요하다.

그러나, 상술한 종래의 기법들은 장치를 사용하는 잡(job)의 작업부하(workload), 잡(job)의 버퍼링(buffering)에 사용가능한 메모리의 크기 , 버퍼링(buffering)으로 인해 발생하는 시간 지연등에 대한 고려가 없어 효율적인 전력 관리가 이루어지지 못했다. 또한, 잡(job)을 버스트(burst) 주기로 수행함에 있어서, 잡(job)이 대기하는 큐(queue)가 소모하는 에너지가 오히려 버스트(burst) 스케줄로 인해 얻어지는 전력 절감의 이익을 상쇄시킬 가능성도 있었다.

발명이 이루고자 하는 기술적 과제

따라서, 본 발명이 이루고자 하는 기술적 과제는, 잡(job)의 작업부하(workload), 잡(job)의 버퍼링(buffering)에 사용 가능한 메모리의 크기, 버퍼링(buffering)에 의해 발생하는 시간 지연 및 버퍼(buffer)의 소모 에너지등을 고려하여, 시스템의 성능을 저하시키지 않고 실질적으로 에너지를 절약할 수 있도록 하는데 있다.

발명의 구성

상기한 기술적 과제를 달성하기 위한 본 발명의 특징에 따른 플랫폼의 소비 전력을 관리하기 위한 방법은, 작업 완료를 위해 디바이스에 주어지는 명령어의 집합인 잡(job)의 스케줄링을 통해 플랫폼이 소비 전력을 관리하는 방법으로서, a) 소정의 이벤트가 발생하면, 상기 이벤트를 작업부하에 따라 분석하는 단계; b) 상기 잡(job)의 작업 부하에 따라 상기 잡(job)이 실행 완료되어야 하는 시간 제약, 상기 잡(job)의 버퍼링에 사용되는 메모리 크기 및 상기 버퍼링에 소비되는 에너지 중 하나 이상을 고려하여, 해당 디바이스가 실행해야 할 잡(job)의 스케줄링 주기 T를 계산하는 단계; c) 상기 b)단계의 결과를 기초로 한, 잡(job) 스케줄에 따라 각 디바이스의 파워상태를 조절하는 단계를 포함한다.

이 때, 상기 b) 단계는, 상기 메모리 크기로 인해 제약받는 스케줄 주기인 제1 주기 및 상기 시간 제약으로 인한 스케줄 주기인 제2 주기 중 하나 이상을 고려하여, 스케줄링 주기 T를 계산하는 단계일 수 있다.

한편 본 발명의 특징에 따른 플랫폼은, 작업 완료를 위해 디바이스에 주어지는 명령어의 집합인 잡(job)의 스케줄링을 통해 소비전력을 관리하는 플랫폼으로서, 소정의 이벤트를 발생시키는 응용부 ; 및 상기 응용부로부터 상기 이벤트 발생정보를 전달받은 때, 상기 잡(job)의 작업부하에 따라 시간 제약, 메모리 크기 및 상기 버퍼링에 소비되는 에너지 중 하나 이상을 고려하여 최적 스케줄링 주기 T를 도출한 후 해당 디바이스의 파워 상태를 조절하는 운영체제부를 포함할 수 있다.

이 때 상기 운영체제부는, 작업 생성 정보를 전달받고, 현재 실행중인 작업에 대한 모니터링, 관리 및 작업 상태 정보 전달을 수행하는 프로세스 매니저; 스케줄 주기에 따라 상기 잡(job)을 스케줄링하는 스케줄러; 상기 잡(job)이 실행되는 디바이스의 구동을 조정하는 디바이스 드라이버; 및 작업 부하에 따라, 시간 제약, 메모리 크기 및 상기 버퍼링에 소비되는 에너지 중 하나 이상을 고려하여 최적 스케줄링 주기 T를 도출하고, 상기 프로세스 매니저로부터 전달받은 작업(task) 상태 정보 및 상기 스케줄러부터 전달받은 잡(job) 스케줄을 기초로 상기 디바이스 드라이버가 해당 디바이스의 파워상태를 조절하도록 하는 파워매니저를 포함할 수 있다.

이하 첨부도면을 참조하여 본 발명을 상세히 설명한다.

우선, 본 문서에서 정의한 symbol을 나타내면 다음 표와 같다.

Symbol	의 미
t_{sd} (t_{wu})	Shutdown (wakeup) 지연
t_o	전력 상태 천이 지연 ($t_{sd} + t_{wu}$)
$t_{be,k}$	장치 d_k 의 break-even time
T_u	Task u 의 장치사용 job 의 burst 주기
p_{iw} (p_s)	Idle working(sleeping)상태의 소모전력
$p_{iw,k}(p_{s,k})$	장치 d_k 의 idle working(sleeping)소모 전력
$p_{bw,k}$	장치 d_k 의 busy working 소모 전력
p_{nKbuf}	n Kbyte buffer memory 소모 전력
e_{sd} (e_{wu})	Shutdown (wakeup) 소모 에너지
e_o	전력 상태 천이 소모 에너지($e_{sd} + e_{wu}$)
E_{split}	요청 발생 즉시 처리할 때 소모 에너지
E_{burst}	요청을 burst 로 처리 할 때 소모 에너지
E_{Buf}	Buffer memory 가 소모하는 에너지
BW_k	장치 d_k 의 자료 처리 능력 (bps, fps, etc)
$S_{k,u}$	Task u 가 d_k 에 한번에 처리를 요청 하는 자료의 크기 (Byte)
$N_{k,u}$	Task u 가 초당 $S_{k,u}$ 크기의 자료를 요청 하는 횟수

< 표 1 > 본 문서에서 정의한 symbol.

또한, 본 발명에서 주기를 계산하는 각각의 경우를 이벤트라 하며, 이것은 주기 T를 계산하는 식이 호출되는 조건을 의미한다. 이벤트의 종류는 다음과 같다.

(1) add tasks to run queue

(2) remove tasks from run queue

(3) time constraint change

(4) memory constraint change

(5) task's S*N change

도 1은 작업부하(workload)가 없을 때 장치를 휴면(sleep) 시켰다가 작업부하(workload)가 발생하면 다시 깨우는(wake-up) 과정에 대한 그림이다.

도 1에서 장치가 처리해야 할 요청(request)가 있는 경우 장치의 동작상태는 비지(busy)이며, 처리해야 할 요청(request)가 없는 경우 장치의 동작상태는 아이들(idle)이다. 그림에서 아이들(idle)상태는 t_1과 t_3 사이이며, 전력관리를 위해 이 구간을 휴면(sleep)상태로 만들 수 있다. 이 때, 전력 상태를 변화시킬때는 셧-다운 딜레이(shutdown delay, t_{sd})와 웨이크-업 딜레이(wakeup delay, t_{wd})라는 시간상의 지연뿐만 아니라 전력 상태의 천이과정에서 발생하는 추가적 에너지 소모가 발생한다. 따라서 일정시간 이상 장치가 아이들(idle) 상태로 있을 때 장치를 휴면(sleep) 시켜야 실질적인 에너지 이득을 볼 수 있다. 이렇게 셧-다운(shutdown)을 통한 전력관리로 에너지 이득을 볼 수 있는 최소의 아이들(idle) 길이를 브레이크-이븐 타임(break-even time, t_{be})이라고 하는데, 이 시간은 장치가 가지고 있는 고유한 특성 상수이다.

도 2a 및 도 2b는 장치가 아이들(idle)일 때 전력상태를 계속 아이들 워킹(idle working)으로 유지하는 경우와 셧-다운(shutdown)으로 유지하는 경우의 전력 소모량을 보여준다.

이 때, t_{be}는 (a)와 (b)에서, 각각 계산된 에너지의 소모량이 같을 때의 시간이다. 장치의 아이들 워킹(idle working)과 휴면(sleep)상태의 소모전력을 각각 P_{iw}, P_s(P_{iw}??P_s)라 하고, 장치의 셧-다운(shutdown)과 웨이크-업(wakeup)에 소요되는 시간과 소모되는 에너지를 각각 t_o, e_o라 하면, t_{be}는 다음과 같다.

$$p_{iw} \times t_{be} = e_o + p_s \times (t_{be} - t_o) \;\; ; \;\; t_{be} > t_o$$

$$t_{be} = \max\left(\frac{e_o - p_s \times t_o}{p_{iw} - p_s}, t_o \right) \quad \cdots\cdots (1)$$

즉, 식(1)에서 t_{be}는 각 장치에 의존적인 값으로, 수행되는 프로그램이나 전력관리 정책과는 무관한 값이다. 장치가 최소한 t_{be}이상의 시간을 아이들(idle) 상태로 있는 경우, 셧-다운(shutdown)을 통해 에너지 절약 효과를 얻을 수 있다.

도 3은 어느 모바일 플랫폼에서 fps가 10인 H.263 인코더의 출력값(output)을 호스트 PC로 전송하는 잡(job)이 분할(split)방식에 따라 스케줄링 될 때와 버스트(burst) 주기에 따라 스케줄링될 때, 무선랜이 호출되는 빈도와 무선랜의 전력상태를 보여준다. 이 때, 분할(split) 방식이란 잡(job)이 발생할 때 즉시 처리하는 방식을 가리키며, 버스트(burst)는 잡(job)이 특정 시간대로 모여서 수행되는 방식을 가리킨다.

도 3에 제시된 바와 같이, 분할(split) 스케줄의 경우 무선랜이 0.1초마다 호출되는데, 이것은 무선랜의 t_{be}의 값인 0.67 보다 작은 값이므로, 정확하게 아이들(idle) 주기를 예측했다고 하더라도 아이들(idle) 주기에 따라 장치를 셧-다운(shutdown)하는 것은 오히려 에너지 소모를 증가하게 만든다. 반면, 버스트(burst) 스케줄의 경우 무선랜이 0.75초마다 호출되므로 셧-다운(shutdown)을 통해 에너지를 절약할 수 있다.

한편, 식 (2)~(4)은 도 3에서 분할(split) 방식 스케줄과 버스트(burst) 주기 T에 따른 스케줄 중 burst 주기동안 무선랜이 소모하는 에너지를 계산하는 식이다.

$$E_{split} = \frac{T \times S \times N}{BW} \times p_{bw} + (T - \frac{T \times S \times N}{BW}) \times p_{iw} \quad \cdots\cdots(2)$$

$$E_{burst} = \frac{T \times S \times N}{BW} \times p_{bw} + (T - \frac{T \times S \times N}{BW} - t_0) \times p_s + e_0 + E_{Buf} \quad \cdots\cdots(3)$$

$$E_{Buf} = \{ Integer \quad (\frac{S \times N}{nKbyte}) + 1 \} \times T \times p_{nKbuf} \quad \cdots\cdots(4)$$

식 (2)~(4)의 첫항은 T시간 동안 리퀘스트(request)를 처리하는데 소모되는 에너지 값이다. 그리고, 식 (2)의 두번째 항은 무선랜이 아이들(idle)일 때 전력관리를 하지 않고 아이들(idle working) 상태로 유지할 때 소모하는 에너지이다. 그리고 식 (3)의 두번째 항은 휴면(sleep) 구간에서 소모되는 에너지이고, e_0는 셧-다운(shutdown)과 웨이크-업(wakeup) 과정에서 소모되는 상태 천이 에너지이다. 이 때, 식 (3)의 마지막 항인 E_{Buf}는 식 (4)으로 계산할 수 있다.

이 때, E_{Buf}는 버스트(burst) 스케줄을 위해 대기하는 리퀘스트(request)가 버퍼링(buffering)되는데 쓰이는 메모리의 소모 에너지이다. 따라서, 잡(job) 스케줄링을 통해 절약한 에너지와, 버퍼링(buffering)으로 소모되는 에너지의 트레이드-오프(trade-off) 관계를 식 (2),(3)을 통해 분석할 수 있다. 따라서, 버퍼 설계시 버퍼의 크기를 너무 작게 할 경우 전력 조절 회로를 구현하는데 많은 비용이 소모되며, 버퍼의 크기를 너무 크게 할 경우 버퍼의 저장공간과 소모되는 에너지가 비효율적으로 활용될 수 있으므로, 전력조절이 가능한 버퍼의 크기를 합리적으로 선택하는 작업 또한 필요할 것이다.

효율적 전력관리를 위한 최적의 잡(job) 스케줄을 찾기 위해서는 분할(split)스케줄과 버스트(burst) 스케줄이 각각 소모하는 에너지, 버스트(burst) 스케줄이 사용 가능한 버퍼 크기, 작업(task, process라고도 한다)의 제한시간(deadline)을 고려할 수 있으며, 이들은 각각 아래의 식 (5)~(7)의 관계식으로 나타낼 수 있다. 각각의 식에서 우변의 상수값(소문자)은 표 2로부터, 변수값(대문자)는 각 작업실행시, 작업 특성 및 OS로부터 얻을 수 있다.

$$T_{eq} = \frac{e_0 - t_0 \times p_s}{p_i - p_s} \times \frac{1}{1 - \frac{N \times S}{BW}}; E_{burst} = E_{split} \quad \cdots\cdots(5)$$

$$T_{buf} = \frac{Buffer \quad Limit}{S \times N}\ ;\ 메모리제약 \quad \cdots\cdots(6)$$

$$T_{dl} = Deadline\ ;\ 시간제약 \quad \cdots\cdots(7)$$

파라미터 \ 장치 명	TM1300 DSP	Orinoco 무선랜
p_{iw}	0.40	0.46
p_s	0.05	0.05
p_{bw}	0.44	0.65
t_0	0.59	0.5
e_0	0.17	0.3

< 표 2 > 무선랜과 DSP의 특성 파라미터 측정치

예를 들어, 상기 < 표 2 >에서 측정한 무선랜의 특성 파라미터와 대역폭(Bandwidth) =5.5Mbps, fps=10, 제한시간(deadline)=4, 메모리크기(buffer limit)=1MB를 식 (5)~(7)에 대입하면, 아래의 식 (5')~(7')과 같은 결과가 나타난다.

$$T = \frac{e_0 - t_0 \times p_s}{p_i - p_s} \times \frac{1}{1 - \frac{N \times S}{BW}} = \frac{0.3 - 0.5 \times 0.05}{0.46 - 0.05} \times \frac{1}{1 - \frac{10 \times S}{5.5Mbps}} \quad \cdots\cdots\cdots\cdots\cdots\cdots (5')$$

$$= 0.67 \times \frac{1}{1 - \frac{10 \times S}{720896}}$$

$$T = \frac{Buffer\ Limit}{S \times N} = \frac{1 \times 1024 \times 1024}{S \times 10} = \frac{104857.6}{S} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (6')$$

$$T = Deadline = 4 \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (7')$$

도 4는 이 식을 그래프로 나타낸 결과이다. 도 4에 도시된 바와 같이, T가 S값에 따라 I~III 의 세 영역으로 구별되어 나타난다.

즉, 도 4의 그래프에서 식 5' 영역은 분할(split) 스케줄에 따라 job을 실행할 때 소비되는 에너지와 동일한 에너지를 소비하는 버스트(burst) 스케줄 주기 그래프이다.

이 때, 식 5'의 왼쪽 상단영역에 속하는 I, II 영역에서는 버스트(burst) 스케줄 방식에 따를 경우 분할(split) 스케줄을 따를 때 보다 더 적은 에너지를 소비할 수 있다. 또, III의 경우는 분할(split) 스케줄 방식이 유리하다. 이것은 식(2) 및 식(3)을 상술한 무선랜 특성 파라미터 측정치 및 주어진 대역폭(Bandwidth), 시간제약(deadline) 및 메모리 크기(buffer limit) 값를 이용해 나타낸 그래프, 즉 스케줄별 에너지 소모량을 나타낸 삼차원 그래프인 도 5a에서, 두 평면의 교점의 그래프인 식(5)를 경계로 하여 결정된다.

다음, 버스트 스케줄 방식을 따르는 것이 유리한 영역에서 소비 에너지의 최적화가 실행되는 최적 버스트(burst) 주기 T는, 실행되는 작업에 의해 결정되는 S값에 따라 달라지게 된다.

I의 경우 응용 프로그램이 S값을 0<S≤26KB 범위로 생성하는데, 이 때 식(6')를 근거로 T를 무한으로 설정할 수도 있다 그러나 식(7')에 의한 시간제약이 존재하므로, 결국 에너지 최적의 버스트(burst) 주기 T는 시간 제약 4초가 됨을 알 수 있다.

같은 방식으로 II의 경우 응용 프로그램이 S값을 26KB<S≤45.5KB 범위로 생성한다. 이 때에도 식(7')에 의한 시간 제약에 근거할 경우, T는 여전히 4초가 될 수 있다. 그러나, 식(6')에 의한 메모리 크기에 따른 제약이 존재하므로, 결국 T는 식(6')가 된다.

한편 III의 경우, 즉 S가 45.5KB보다 클 경우는 앞서 설명한 바와 같이 분할(split) 스케줄 방식을 따를 때 더 적은 에너지를 소비하게 되므로, 무선랜 사용을 요청(request)하는 잡(job)이 생성되는 즉시 해당 잡(job)을 수행하는 분할(split) 스케줄링이 선택된다.

상술한 예제는 <표2>의 DSP의 특성 파라미터 측정치, 식(5)~식(7) 및 DSP의 스케줄별 에너지 소모량을 나타낸 그래프인 도 5b를 이용하여 DSP에도 동일하게 적용된다. 또한 무선랜과 DSP를 동시에 사용하는 H.263 인코더 등의 작업(task, process라고도 한다)의 경우 잡(job)들 간 수행 순서의 제약이 존재하므로, 한 장치를 위해 구한 버스트(burst) 주기 T값을 다른 장치의 시간 제약(deadline)으로 대입하여 나머지 장치의 버스트(burst) 주기 T값을 구하는 순차적 방법을 이용할 수 있다.

이하 도 6을 참조하여 본 발명의 특징에 따른 플랫폼의 실시예에 대해 설명한다.

상기 플랫폼은 특정 작업 완료를 위한 명령어의 집합인 잡(job)의 스케줄링을 통해 소비 전력을 관리하는 플랫폼으로서, 응용부(100) 및 운영체제부(200)를 포함한다.

응용부(100)는 특정 작업을 수행하도록 설계된 프로그램을 포함하는 모듈로서, 상기 프로그램의 예로는 MP3 인코더, H.263 인코더등이 있으며 기타 다양한 응용 프로그램이 응용부(100)에 포함될 수 있다.

운영체제부(200)는 프로세스 매니저(210), 파워 매니저(220), 스케줄러(230) 및 디바이스 드라이버(240)를 포함하여, 상기 응용부(100)로부터 전달받은 이벤트(즉 특정 작업 생소멸 이벤트, S, N 시간 및 메모리 제약 변경등)를 기초로 최적의 버스트 주기 T를 계산하고, T에 따라 각 장치를 사용하는 잡(job)들을 스케줄링 한다.

보다 구체적으로 디바이스 드라이버(240)는 플랫폼에 포함된 디바이스, 즉 장치들과 상호작용을 하며, 장치나 특수 소프트웨어 인터페이스에 대한 정보를 포함하여 해당 장치 구동을 조정한다. 디바이스 드라이버(240)는 플랫폼이 포함하고 있는 다양한 디바이스들의 드라이버를 포함하며, 예를 들어 무선랜 디바이스 드라이버(240), DSP 디바이스 드라이버(240)를 포함할 수 있다.

프로세스 매니저(210)는 각 작업이 생성되면, 운영체제부(200)(예를 들면, linux 커널)로부터 작업 생성 정보를 전달받은 후, 현재 실행중인 작업을 모니터링하고 관리한다. 그리고 현재 작업 진행 상태 정보를 파워 매니저(220)에게 전달한다.

스케줄러(230)는 계산된 스케줄 주기에 따라 특정 작업 완료를 위한 명령어들, 즉 잡(job)들을 스케줄링한다.

파워 매니저(220)는 T 계산부(221) 및 파워상태 셋팅부(223)를 포함하여, 잡(job) 스케줄링 주기를 계산하고 잡(job) 스케줄에 따라 각 장치들의 파워상태를 셋팅한다.

특히, T 계산부(221)는 소정의 이벤트가 발생하면 작업부하에 따라, 상기 잡(job)을 분할(split)스케줄링 할 때의 소비 에너지 E_{split} 과 버스트(burst) 스케줄링 할 때의 소비 에너지 E_{burst}가 같을 때의 주기를 기초로, 메모리 제약, 시간 제약 등을 고려하여 스케줄링 주기 T를 계산하여, 잡(job) 스케줄링 방식 및 그에 따른 최적 주기를 최종 결정하고 그 결과를 스케줄러에 전달한다.

보다 구체적으로 T 계산부(221)는 상기 잡(job)을 분할(split)스케줄링 할 때의 소비 에너지 E_{split}과 버스트(burst) 스케줄링 할 때의 소비 에너지 E_{burst}가 같을 때의 스케줄 주기값이, 상기 제1 주기 혹은 상기 제2 주기의 값보다 크거나 같을 때는 분할(split) 스케줄링을 선택한다. 그리고, 상기 잡(job)을 분할(split)스케줄링 할 때의 소비 에너지 E_{split}과 버스트(burst) 스케줄링 할 때의 소비 에너지 E_{burst}가 같을 때의 스케줄 주기값이, 상기 제1 주기 혹은 상기 제2 주기의 값보다 작을 때에는, 상기 제1 주기 및 상기 제2 주기 중 더 작은 주기 값에 따른 버스트(burst) 스케줄링을 선택한다.

그리고, 파워상태셋팅부(223)는 프로세스 매니저로부터 전달받은 작업 상태 정보 및 스케줄러로부터 전달받은 잡(job) 스케줄에 따라 디바이스 드라이버(240)를 통해 각 장치들의 파워상태를 셋팅한다.

이하 도 7 및 도 8을 참조하여 본 발명의 특징에 따른 플랫폼의 소비 전력을 관리하는 방법을 설명한다.

우선, 작업(task)의 생성 혹은 소멸, 메모리 제약 변경, 작업(task)의 S, N변경, 시간 제약 변경 등의 이벤트가 발생하면, 상기 이벤트 발생 사실이 응용부로부터 운영체제부로 전달된다(S100).

다음, T 계산부(221)는 소정의 이벤트가 발생하면 상기 잡(job)을 분할(split)스케줄링 할 때의 소비 에너지 E_{split} 과 버스트(burst) 스케줄링 할 때의 소비 에너지 E_{burst}가 같을 때의 주기를 기초로, 메모리 제약, 시간 제약 등을 고려하여 스케줄링 주기 T를 계산하여, 잡(job)의 최적 스케줄링 주기를 결정한다.

이 때, 도 8에 도시된 바와 같은 방법으로, T 계산부(221)는 상기 잡(job)을 분할(split)스케줄링 할 때의 소비 에너지 E_{split} 과 버스트(burst) 스케줄링 할 때의 소비 에너지 E_{burst}가 같을 때의 스케줄 주기값이, 상기 제1 주기 혹은 상기 제2 주기의 값보다 크거나 같을 때는 분할(split) 스케줄링을 선택한다. 그리고, 상기 잡(job)을 분할(split)스케줄링 할 때의 소비 에너

지 E_{split}과 버스트(burst) 스케줄링 할 때의 소비 에너지 E_{burst}가 같을 때의 스케줄 주기값이, 상기 제1 주기 혹은 상기 제2 주기의 값보다 작을 때에는, 상기 제1 주기 및 상기 제2 주기 중 더 작은 주기 값에 따른 버스트(burst) 스케줄링을 선택하여, 잡(job)의 최적 스케줄링 주기 T를 결정할 수 있다.

이 때, 예를 들어, 무선랜에 할당되는 잡(job)의 최적 스케줄링 주기를 우선 계산하고, 플랫폼에 포함된 또 다른 장치인 DSP에 할당되는 잡(job)의 최적 스케줄링 주기를 계산할 때, 마찬가지로 작업의 S*N, 시간 제약 및 메모리 제약도 함께 고려된다. 이 경우 우선 계산된 무선랜의 최적 주기 값이 DSP에 할당되는 잡(job)에 대한 시간제약으로 고려될 수 있다 (S101, S103).

스케줄러는 각 디바이스들의 최적 스케줄링 주기에 따라 작업 완료를 위한 잡(job)을 스케줄링하고, 파워상태셋팅부는 상기 스케줄에 따라 해당 장치의 파워 상태(power state)를 변화시켜, 작업 수행을 위해 상기 플랫폼에서 소비하는 전력을 효율적으로 관리한다(S105).

이상 설명한 바는 본 발명의 실시예에 불과한 것으로, 본 발명의 권리범위가 이에 한정되는 것은 아니며 기타 다양한 응용 실시가 가능하며, 본 발명의 권리범위가 후술할 특허청구범위 기재사항 및 그 균등물로 해석되는 모든 사항을 포함하는 것은 당연하다.

예를 들어, 플랫폼에 포함된 장치로서 상술한 DSP, 무선랜 이외 기타 다양한 장치에 대해서도 적용될 수 있다. 이 때 하나의 파워매니저가 여러 장치에 적용될 경우, 각 장치마다 버퍼가 필요할 수 있고, 이 경우 시스템 전체에서 사용 가능한 버퍼를 장치별로 할당할 때, 전체장치에서 절약되는 에너지의 합을 최대로 만드는 비율로 버퍼를 할당할 수도 있다.

본 발명은 특히 로봇과 같은 모바일 플랫폼에서 유용하게 적용될 수 있다. 제한된 용량의 전지로부터 모바일 장치의 동작 전원을 공급받는 것에는 한계가 있으므로, 장치 설계 과정에서 전지의 용량을 늘리는 것 못지 않게 효율적 전력 관리는 특히 중요하다. 따라서, 로봇의 운영체제(OS) 수준에서, 입출력장치를 사용하는 잡(job)들을 본 발명에 따라 스케줄링하는 경우, 입출력장치를 사용하는 잡(job)의 작업부하, 잡(job) 버퍼링에 사용하는 메모리 크기 및 시간 지연등이 고려되므로, 에너지를 절약하면서도 로봇 시스템의 성능 저하를 최소화 할 수 있다.

이 때, 모바일 플랫폼에서 현재 사용 가능한 전지의 용량에 따라 여러 단계의 전력 관리를 적용하는 방법도 가능하다. 즉 일반적으로 전력 관리기법을 적용하면 에너지가 절약되는 만큼 시스템 성능이 낮아지는 문제가 발생하는데, 따라서 남아있는 전지 용량에 따라 제공 가능한 QoS를 정의하고 전력상태에 따라 다른 전력 관리 기법을 적용할 수 있다.

발명의 효과

이상 설명한 바와 같이 본 발명에 따르면, 플랫폼, 특히 전력관리의 필요성이 절실한 모바일 플랫폼에서 OS 수준의 전력관리를 할 때, 잡(job)의 작업 부하(workload), 잡(job)의 버퍼링에 사용 가능한 메모리의 크기, 버퍼링(buffering)에 의해 발생하는 시간 지연 및 버퍼(buffer)의 소모 에너지등이 고려되어 최적의 잡(job) 스케줄링 주기가 계산된다. 따라서, 시스템의 성능은 저하되지 않고 실질적으로 에너지를 절약할 수 있다.

도면의 간단한 설명

도 1은 작업부하(workload)가 없을 때 장치를 휴면(sleep) 시켰다가 작업부하 (workload)가 발생하면 장치를 다시 깨우는(wake-up) 과정에 대한 그림이다.

도 2a, 도 2b는 장치가 아이들(idle)상태일 때 전력상태를 계속 아이들 워킹(idle working)으로 유지하는 경우와 셧-다운(shutdown)으로 유지하는 경우의 전력 소모량에 대한 그래프이다.

도 3은 스케줄 방식에 따른 무선랜의 호출 패턴 및 전력상태 그림이다.

도 4는 S값에 따라 에너지 최적의 T를 선택하기 위한 그래프이다.

도 5a, 도 5b는 분할 방식과 버스트(burst) 스케줄에 따른 각 장치의 에너지 소모량을 나타낸 3차원 그래프이다.

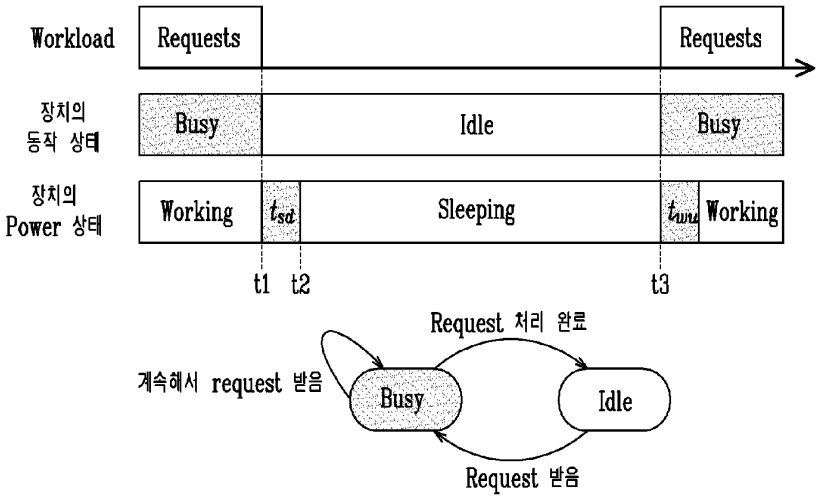
도 6은 본 발명의 특징에 따라 잡(job) 스케줄링을 통해 소비 전력을 관리하는 플랫폼 구성도의 예이다

도 7은 본 발명의 특징에 따른 실시예로서 잡(job) 스케줄링을 통한 소비전력 관리 방법 흐름도이다.

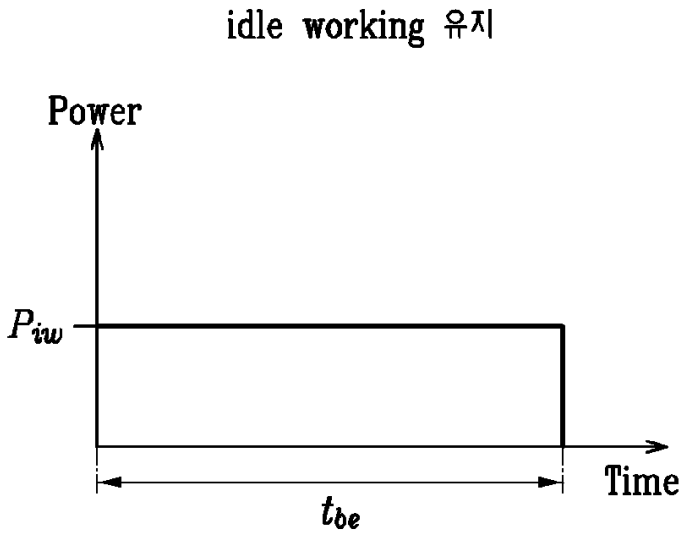
도 8은 잡(job) 스케줄링을 위한 버스트주기(burst T)를 구하여 전력 관리에 사용하기 위한 수도 코드(pseudo code)의 예이다

도면

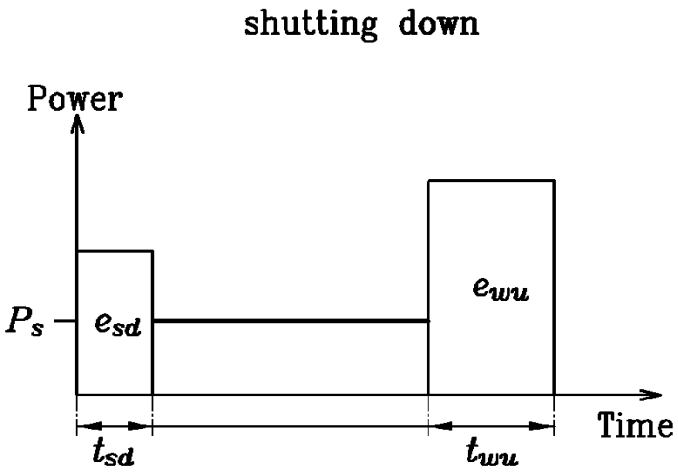
도면1



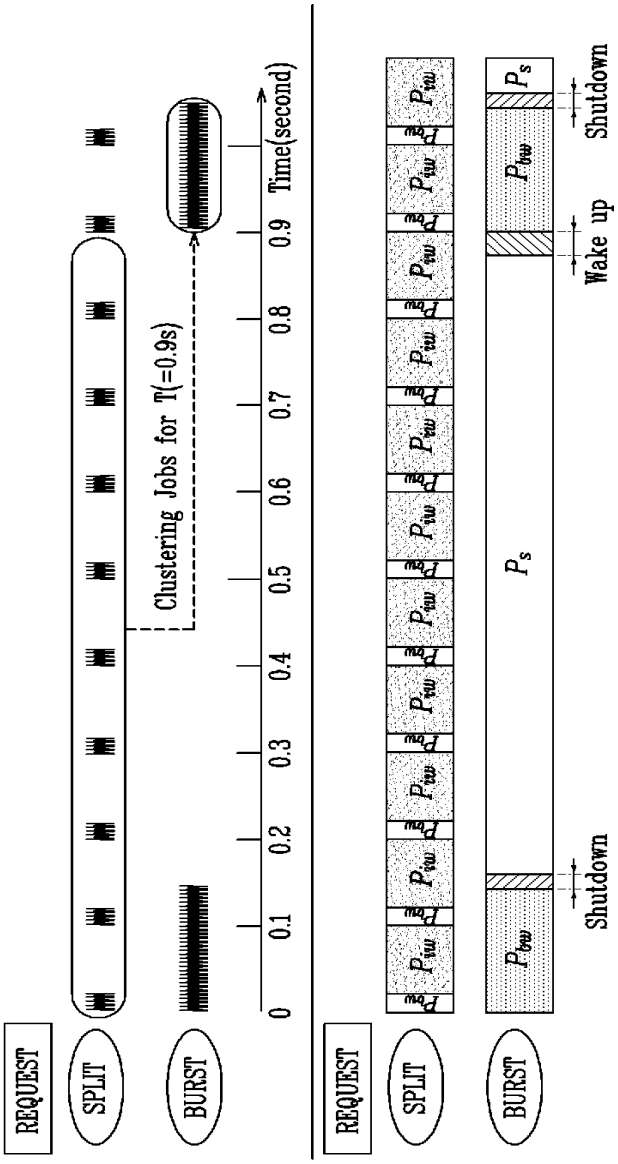
도면2a



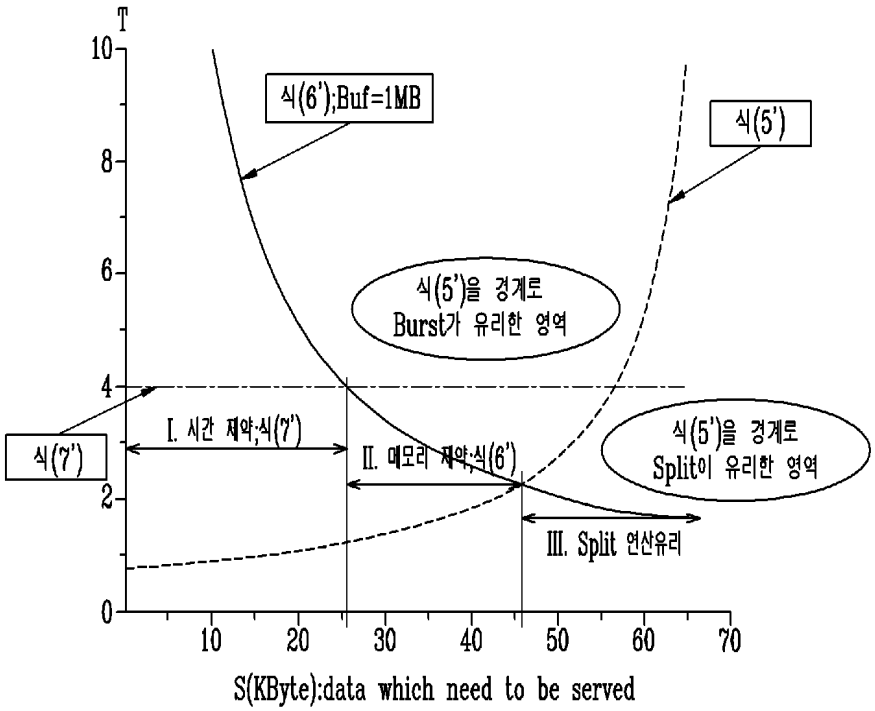
도면2b



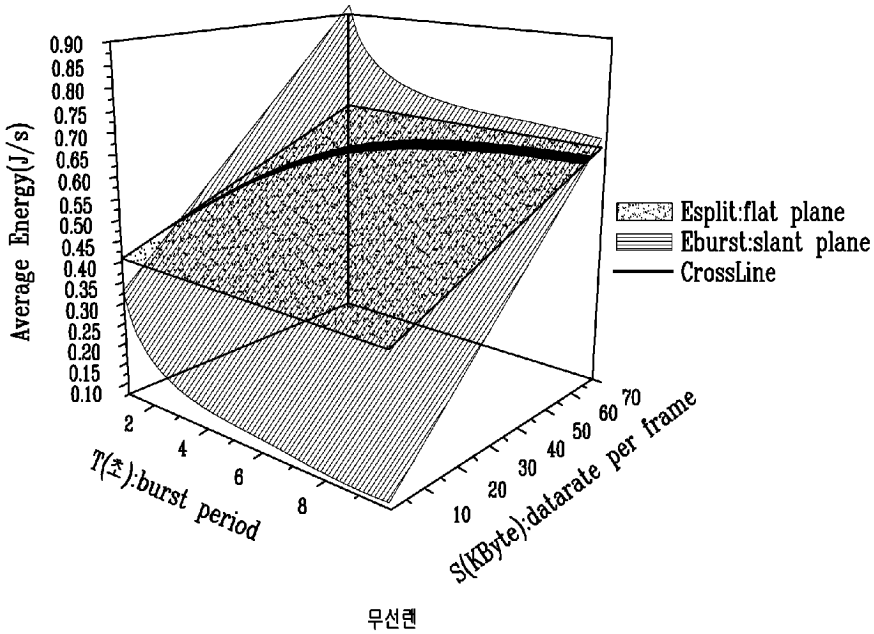
도면3


REQUEST
SPLIT
BURST
Clustering Jobs for T(=0.9s)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Time(second)
REQUEST
SPLIT
BURST
P_{tw}
P_s
P_{bw}
Shutdown
Wake up
Shutdown

도면4

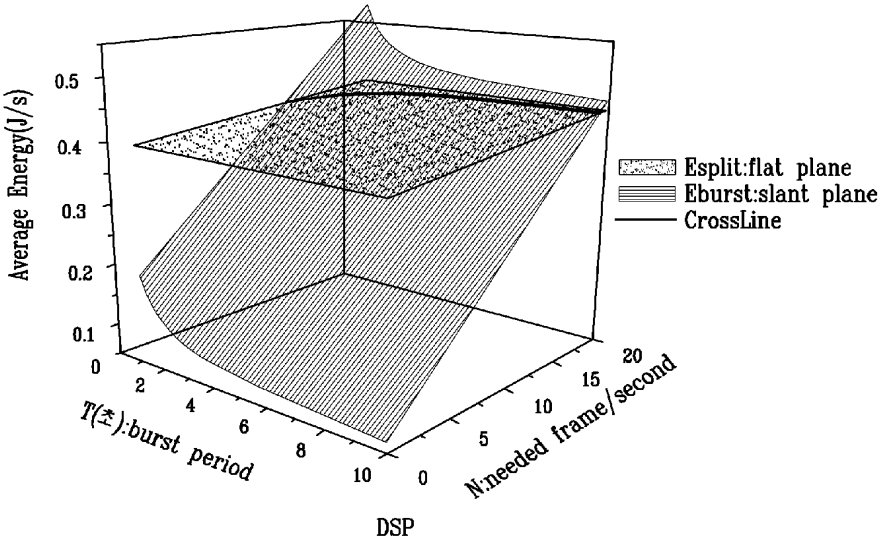

T
10
8
6
4
2
0
식(6');Buf=1MB
식(5')
식(5')을 경계로
Burst가 유리한 영역
식(7')
I. 시간 제약;식(7')
II. 메모리 제약;식(6')
식(5')을 경계로
Split이 유리한 영역
III. Split 연산유리
10
20
30
40
50
60
70
S(KByte):data which need to be served

도면5a

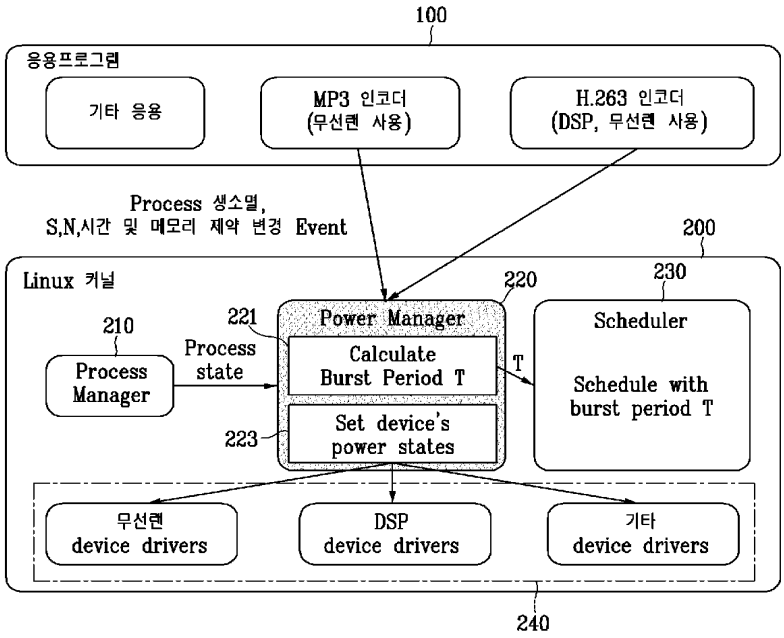

Average Energy(J/s)
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
2
4
6
8
T(초):burst period
10
20
30
40
50
60
70
S(KByte):datarate per frame
Esplit:flat plane
Eburst:slant plane
CrossLine

무선랜

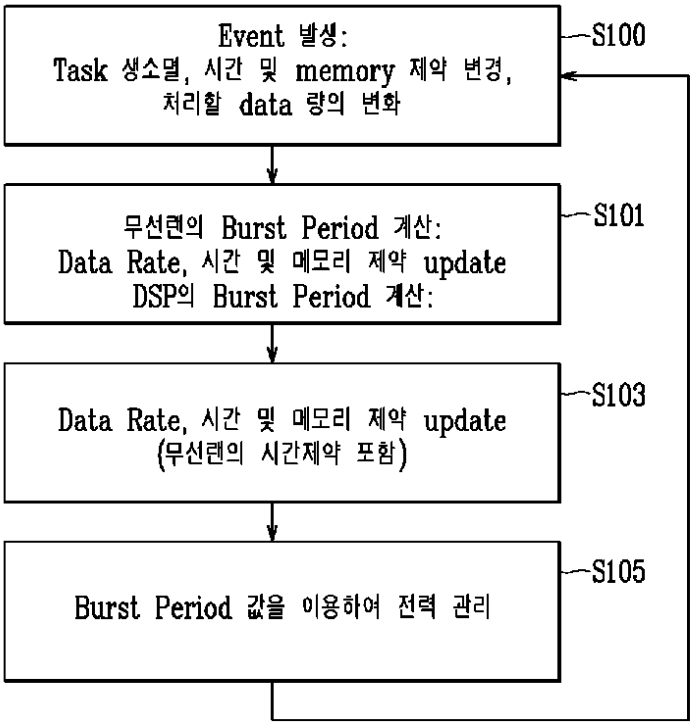
도면5b


Average Energy(J/s)
0.5
0.4
0.3
0.2
0.1
0
2
4
6
8
10
T(초):burst period
0
5
10
15
20
N:needed frame/second
DSP
Esplit:flat plane
Eburst:slant plane
CrossLine

도면6


100
응용프로그램
기타 응용
MP3 인코더
(무선랜 사용)
H.263 인코더
(DSP, 무선랜 사용)
Process 생소멸,
S,N,시간 및 메모리 제약 변경 Event
200
Linux 커널
220
230
210
221
Process
Manager
Process
state
Power Manager
Calculate
Burst Period T
T
Scheduler
Schedule with
burst period T
223
Set device's
power states
무선랜
device drivers
DSP
device drivers
기타
device drivers
240

도면7


Event 발생:
Task 생소멸, 시간 및 memory 제약 변경,
처리할 data 량의 변화
S100
무선랜의 Burst Period 계산:
Data Rate, 시간 및 메모리 제약 update
DSP의 Burst Period 계산:
S101
Data Rate, 시간 및 메모리 제약 update
(무선랜의 시간제약 포함)
S103
Burst Period 값을 이용하여 전력 관리
S105

도면8

/* Parameter Initialization : S,N,Bw,Tlatency */

/* Burst period 'T' 선택 */

If (add tasks to run queue) or (remove tasks from run queue) or (time constraint change) or (memory constraint change) or (task's S*N change) {
switch(EVENT){
case "add task Tn to run queue":
S*N = S*N + (S*N of Tn);
Tlatency = min(Tlatency, Tn's Tlatency);
case "remove task Tn from run queue" :
S*N = S*N - (S*N of Tn);
Tlatency=min(all running tasks's Tlatency except Tn's);
case "new time constraint"?:
Tlatency = (new Tlatency);
case "new buffer constraint" :
BufSize = (new BufSize);
case "Task Tn's S*N change" :
S*N = S*N - (S*N of Tn) + (new S*N of Tn);
}

$$T_{BufLimit} = \frac{BufSize}{S \times N};$$

$$T_{(Eburst=Esplit)} = \frac{E_0 - T_0 \times P_{sleep}}{P_{idle} - P_{sleep}} \times \frac{1}{1 - \frac{N}{B_W} \times S};$$

T =min($T_{BufLimit}$, $T_{Latency}$);
IF(($T_{(Eburst=Esplit)} >= T_{BufLimit}$) or ($T_{(Eburst=Esplit)} > T_{Latency}$)) split 으로 스케줄;
else T 값을 가지고 burst 로 스케줄;