

발송번호: 9-5-2008-005814338
발송일자: 2008.01.31

수신 서울 강남구 역삼1동 642-16 성지하이츠빌
딩2차 805호

이원희

135-717

YOUR INVENTION PARTNER

특 허 청
특허결정서



출원인명칭	명칭	한국과학기술원 (출원인코드: 319980988661)
주소	주소	대전 유성구 구성동 373-1
대리인명칭	명칭	이원희
주소	주소	서울 강남구 역삼1동 642-16 성지하이츠빌딩2차 805호
발명자성명	성명	임지수
주소	주소	대전 유성구 구성동 373-1 KAIST
발명자성명	성명	박대연
주소	주소	경기 성남시 분당구 수내동 양지마을아파트 207동 702호
발명자성명	성명	박기웅
주소	주소	서울 노원구 월계4동 500-11번지
발명자성명	성명	박규호
주소	주소	충청남도 공주시 장기면 금암리 314-98 번지
출원번호	호	10-2006-0026518
발명명의명칭	명칭	무선 센서 네트워크 환경에서 프로그램을 변경하기 위한확장 가능한 가상 머신 및 이를 이용한 리프로그래밍 방법
청구항	항	13

이 출원은 특허법 제66조의 규정에 의하여 특허결정합니다.
(특허법 제87조의 규정에 따라 특허권은 특허료를 납부하여 설정등록함으로써 발생하게 됩니다.) 끝.

[참고문헌]

1. EP0930793 A1
2. KR100378565 B1
3. US6418310 B1
4. KR1020040048246 A

2008.01.31

특허청

정보통신심사본부
컴퓨터심사팀

심사관

신성길



<<특허등록료, 실용신안등록료 납부, FD(플로피디스크제출 및 특허기술상신청안내)>>

※ 등록료 납부안내

1. 특허료 납부예정금액(1~3년차분) : 651,000 원(1~3년차분을 정상납부기간내에 납부하는 경우), 325,500 원(50% 감면대상자인 경우), 195,300 원(70% 감면대상자인 경우)

* 특허료의 정상납부기간을 경과하여 추가납부하거나, 특허료납부기간 중 특허료의 변경요인이 있을 때에는 위 납부금액이 달라질 수 있습니다.

2. 납부기간 : 이 특허결정서를 받은 날로부터 3월 이내에 최초 3년분의 특허료를 일시에 납부하여야 하며, 이 납부기간이 경과한 후에는 6월이내(추가납부기간)에 한하여 2배의 금액을 납부하여야 합니다. 이 추가납부기간도 경과하면 출원포기로 간주되어 등록받을 수 없습니다.

3. 납부방법 :

① 「납입고지서 및 영수증」을 이용하는 방법 : 특허결정서와 함께 송달받은 「납입고지서 및 영수증」상의 정상납부일까지 은행, 우체국 등 국고수납기관 또는 인터넷지로서이트(<http://giro.or.kr>)에서 특허료를 납부합니다.

② 특허료납부서를 이용하는 방법 : 특허료납부서를 특허청(서울사무소)에 먼저 제출한 후 부여받은 접수번호를 납부자번호로 하여 제출일의 다음날까지 은행, 우체국 등 국고수납기관 또는 인터넷지로서이트(<http://giro.or.kr>)에서 특허료를 납부합니다. 다만, 특허료납부서를 우편으로 제출하려는 경우에는 특허료에 해당하는 통상환증서를 함께 보내셔야 합니다.

* 보내실 곳 : 우)302-701 대전광역시 서구 둔산동 920 특허청 등록서비스팀

4. 국민기초생활보장법상 수급자, 학생, 소기업, 중기업 등이 특허료를 감면받고자 하는 경우에는 특허료납부서와 함께 감면 또는 면제 대상임을 입증하는 서류를 제출한 후 특허료를 납부하여야 하고, 일부청구항을 포기하려는 자는 특허료납부서와 함께 일부청구항포기서를 제출한 후 특허료를 납부하여야 합니다.

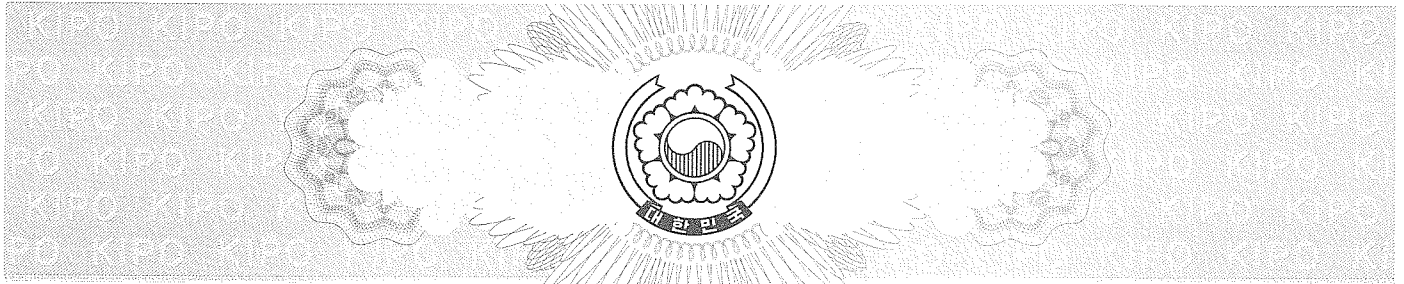
※ FD출원

- FD출원 후 특허결정의 등본을 받기 전에 보정을 한 자는 특허료납부서 제출시까지 보정내용의 전부가 반영된 최종본의 FD 1통을 첨부하여 민원실에 제출하여야 합니다.

※ 특허기술상은 내국인의 경우, 이의신청기간만료일로부터 설정등록 후 2년이내에 해당 심사팀에 추천의뢰신청을 할 수 있습니다.

※ 서식 또는 절차에 대하여는 특허고객 콜센터(☎1544-8080)로 문의하시기 바라며, 기타 문의사항이 있으시면 ☎ 042-481-5688(담당심사관 신성길)로 문의하시기 바랍니다.

※ 우 302-701 대전광역시 서구 선사로 139(둔산동 920) 정부대전청사 특허청



특 허 증

CERTIFICATE OF PATENT

특 허 제 10-0803290 호 (PATENT NUMBER) 출원번호 (APPLICATION NUMBER) 제 2006-0026518 호
출원일 (FILING DATE:YY/MM/DD) 2006년 03월 23일
등록일 (REGISTRATION DATE:YY/MM/DD) 2008년 02월 04일

발명의명칭 (TITLE OF THE INVENTION)
무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신 및 이를 이용한 리프로그래밍 방법

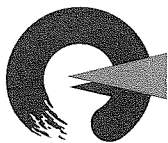
특허권자 (PATENTEE)
한국과학기술원(114471-0*****)
대전 유성구 구성동 373-1

발명자 (INVENTOR)
등록사항란에 기재

위의 발명은 「특허법」에 의하여 특허등록원부에 등록되었음을 증명합니다.

(THIS IS TO CERTIFY THAT THE PATENT IS REGISTERED ON THE REGISTER OF THE KOREAN INTELLECTUAL PROPERTY OFFICE.)

2008년 02월 04일



특 허 청
COMMISSIONER, THE KOREAN INTELLECTUAL PROPERTY OFFICE



등록사항

특허 등록 제 10-0803290 호
(PATENT NUMBER)

발명자 (INVENTOR)

임지수(790113-1*****)

대전 유성구 구성동 373-1 KAIST 전기및전자공학과시스템소프트웨어
연구실

박대연(560314-1*****)


경기 성남시 분당구 수내동 양지마을아파트 207동 702호

박기웅(791002-1*****)

서울 노원구 월계4동 500-11번지

박규호(501019-1*****)

충청남도 공주시 장기면 금암리 314-98 번지

	(19) 대한민국특허청(KR)	(45) 공고일자	2008년02월13일
	(12) 등록특허공보(B1)	(11) 등록번호	10-0803290
		(24) 등록일자	2008년02월04일
(51) Int. Cl.	(73) 특허권자		
G06F 9/44 (2006.01) G06F 9/06 (2006.01)	한국과학기술원		
(21) 출원번호 10-2006-0026518	대전 유성구 구성동 373-1		
(22) 출원일자 2006년03월23일	(72) 발명자		
심사청구일자 2006년03월23일	임지수		
(65) 공개번호 10-2007-0096316	대전 유성구 구성동 373-1 KAIST 전기및전자공학		
(43) 공개일자 2007년10월02일	과시스템소프트웨어 연구실		
(56) 선행기술조사문헌	박대연		
EP0930793 A1	경기 성남시 분당구 수내동 양지마을아파트 207동		
KR100378565 B1	702호		
US6418310 B1	(뒷면에 계속)		
KR1020040048246 A	(74) 대리인		
전체 청구항 수 : 총 13 항	이원희		
		심사관 :	신성길

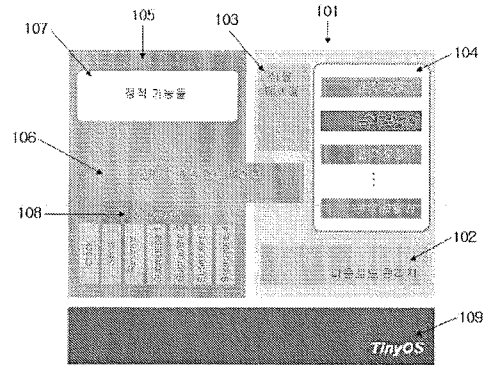
(54) 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신 및 이를 이용한 리프로그래밍 방법

(57) 요약

본 발명은 기존 가상 머신의 오버헤드를 줄여 프로그램의 변경과 실행을 저전력화할 수 있도록 한 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신 및 이를 이용한 리프로그래밍 방법을 제공한다.

본 발명에 따른 확장 가능한 가상 머신은, 네트워크를 통해 메타데이터 패키지가 수신되면 다운로드 관리자는 네트워크를 통해 새로운 동적 모듈을 다운받아 이를 프로그램 메모리로 로드하는 제1단계; 동적 모듈이 저장된 프로그램 메모리의 주소를 심볼 테이블에 업데이트하는 제2단계; 동적 모듈을 호출하는 바이트 코드를 바이트 코드 인터프리터가 해석하여 상기 심볼 테이블을 검색하여 동적 모듈이 저장된 프로그램 메모리의 주소를 얻는 제3단계; 상기 제3단계에서 얻어진 프로그램 메모리 주소에 따라 바이트 코드 인터프리터는 그 바이트 코드에 해당하는 동적 모듈을 호출하여 호출된 동적 모듈이 실행되도록 하는 제4단계;를 수행하여 무선 센서 네트워크 환경에서 리프로그래밍을 가능케 한다.

도1 - 도1



(72) 발명자
박기웅
서울 노원구 월계4동 500-11번지

박규호
충청남도 공주시 장기면 금암리 314-98 번지

특허청구의 범위

청구항 1

바이트 코드가 저장되는 캡슐 및 상기 캡슐에 저장된 바이트 코드를 수행하며 동적 모듈을 호출하는 바이트 코드 인터프리터를 구비하는 무선 센서 네트워크 환경에서 이미 배치된 센서 노드를 다시 프로그램하기 위한 가상 머신 장치에 있어서,

업데이트되는 동적 모듈의 프로그램 메모리 주소가 저장되는 심볼 테이블; 및

네트워크를 통해 새로운 동적 모듈을 다운받고 이를 상기 프로그램 메모리로 로드하며, 상기 동적 모듈이 저장된 프로그램 메모리의 주소를 상기 심볼 테이블에 업데이트하는 다운로드 관리자;

를 더 포함하여 구성되는 것을 특징으로 하는 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신 장치.

청구항 2

제 1 항에 있어서, 상기 동적 모듈은

상기 프로그램 메모리의 중간의 빈 공간 임의의 위치에 로드되는 것을 특징으로 하는 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신 장치.

청구항 3

제 1 항에 있어서, 상기 다운로드 관리자는

상기 네트워크를 통해 수신되는 메타데이터 패키지에 따라 상기 동적 모듈을 다운받는 것을 특징으로 하는 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신 장치.

청구항 4

제 3 항에 있어서, 상기 메타데이터 패키지는

상기 동적 모듈을 호출하는 바이트 코드의 이름, 동적 모듈의 버전, 사이즈, 전체 패키지 수에 대한 정보를 포함하는 것을 특징으로 하는 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신 장치.

청구항 5

제 3 항에 있어서, 상기 다운로드 관리자는

센서 노드가 배치된 후 추가된 동적 모듈의 정보를 포함하는 동적 모듈 테이블; 및

상기 프로그램 메모리의 빈 페이지 정보를 갖는 프로그램 메모리 페이지 리스트;

를 포함하는 것을 특징으로 하는 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신 장치.

청구항 6

제 5 항에 있어서, 상기 동적 모듈 테이블은

상기 메타데이터 패키지의 정보와 상기 동적 모듈이 로드된 프로그램 메모리 주소 정보를 갖는 것을 특징으로 하는 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신 장치.

청구항 7

바이트 코드가 저장되는 캡슐; 상기 캡슐에 저장된 바이트 코드를 수행하며 동적 모듈을 호출하는 바이트 코드 인터프리터; 업데이트되는 동적 모듈의 프로그램 메모리 주소가 저장되는 심볼 테이블; 및 네트워크를 통해 새로운 동적 모듈을 다운받고 이를 상기 프로그램 메모리로 로드하며, 상기 동적 모듈이 저장된 프로그램 메모리의 주소를 상기 심볼 테이블에 업데이트하는 다운로드 관리자;를 포함하는 무선 센서 네트워크 환경에서

의 확장 가능한 가상 머신을 이용한 리프로그래밍 방법에 있어서,

상기 네트워크를 통해 메타데이터 패키지가 수신되면 상기 다운로드 관리자는 네트워크를 통해 새로운 동적 모듈을 다운받아 이를 상기 프로그램 메모리로 로드하는 제1단계;

상기 동적 모듈이 저장된 프로그램 메모리의 주소를 상기 심볼 테이블에 업데이트하는 제2단계;

상기 동적 모듈을 호출하는 바이트 코드를 상기 바이트 코드 인터프리터가 해석하여 해석 결과에 따라 상기 심볼 테이블을 검색하여 동적 모듈이 저장된 프로그램 메모리의 주소를 얻는 제3단계; 및

상기 제3단계에서 얻어진 프로그램 메모리 주소에 따라 상기 바이트 코드 인터프리터는 그 바이트 코드에 해당하는 동적 모듈을 호출하여 호출된 동적 모듈이 실행되도록 하는 제4단계;

를 포함하여 이루어짐을 특징으로 하는 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신을 이용한 리프로그래밍 방법.

청구항 8

제 7 항에 있어서, 상기 메타데이터 패키지는

상기 동적 모듈을 호출하는 바이트 코드의 이름, 동적 모듈의 버전, 사이즈, 전체 패키지 수 정보를 포함하는 것을 특징으로 하는 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신을 이용한 리프로그래밍 방법.

청구항 9

제 7 항에 있어서, 상기 메타데이터 패키지가 수신되면 상기 다운로드 관리자는

센서 노드가 배치된 후 추가된 동적 모듈의 정보를 가지고 있는 동적 모듈 테이블을 이용하여 업데이트해야 할 동적 모듈의 버전을 판단하도록 된 것을 특징으로 하는 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신을 이용한 리프로그래밍 방법.

청구항 10

제 7 항에 있어서, 상기 동적 모듈의 프로그램 메모리로의 로드는

상기 다운로드 관리자로부터의 프로그램 메모리의 부트 로더에게의 쓰기 요청에 따라, 상기 프로그램 메모리의 빈 페이지 정보를 갖는 프로그램 메모리 페이지 리스트의 빈 페이지에, 상기 동적 모듈을 로드하도록 된 것을 특징으로 하는 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신을 이용한 리프로그래밍 방법.

청구항 11

제 7 항에 있어서, 상기 제4단계에서 상기 바이트 코드 인터프리터는 상기 바이트 코드의 해석 결과에 따라 얻어지는 심볼 테이블 주소를 바탕으로 함수 포인터를 이용하여 바이트 코드에 해당하는 동적 모듈을 호출하도록 된 것을 특징으로 하는 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신을 이용한 리프로그래밍 방법.

청구항 12

제 11 항에 있어서, 상기 동적 모듈 호출시 상기 바이트 코드 인터프리터는 함수의 인자로 오퍼랜드 스택의 포인터를 상기 동적 모듈로 전달하도록 된 것을 특징으로 하는 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신을 이용한 리프로그래밍 방법.

청구항 13

제 12 항에 있어서, 상기 동적 모듈은 상기 전달받은 오퍼랜드 스택의 포인터를 사용하여 계산에 필요한 피연산자를 상기 오퍼랜드 스택에서 가져오고, 피연산자를 연산한 결과를 다시 상기 오퍼랜드 스택의 포인터를 사용하여 오퍼랜드 스택에 넘겨주도록 된 것을 특징으로 하는 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신을 이용한 리프로그래밍 방법.

명세서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

- <17> 본 발명은 무선 센서 네트워크 시스템에 관한 것으로, 특히 무선 센서 네트워크 환경에서 각각의 센서 노드의 프로그램을 네트워크를 통해 무선으로 수정 보완할 수 있도록 하는 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신 및 이를 이용한 리프로그래밍 방법에 관한 것이다.
- <18> 무선 센서 네트워크는 유비쿼터스 시대와 더불어 현재 가장 많이 연구되는 분야이다. 무선 센서 네트워크는 무선으로 연결된 작은 센서 노드를 이용하여 분산 환경에 적용될 수 있는 네트워크를 말한다. 센서 노드의 자원은 제한되기 때문에 센서 노드의 프로그램은 자원이 풍부한 호스트 머신에서 컴파일 된 프로그램 이미지를 사용해 전송한다.
- <19> 무선 센서 네트워크는 환경을 모니터링 하는 어플리케이션으로 많이 활용되므로 각 센서노드의 어플리케이션은 환경 변화에 맞게 프로그램을 수정될 필요가 있다.
- <20> 또한, 하나의 센서 노드는 긴 수행 시간을 가지기 때문에 라이프 타임 동안 버그의 수정이나 시스템의 변경 등의 이유로 소프트웨어를 바꾸어야 할 필요가 있다.
- <21> 그러나 무선 센서 네트워크 환경은 하나의 어플리케이션을 위해 수백 수천 개의 센서 노드가 사용되고 관리자가 센서 노드의 프로그램의 변경을 위하여 각 센서 노드를 물리적으로 제거 및 재설치하는 것을 불가능 하다. 그러므로 효율적으로 무선 센서 네트워크의 프로그램을 변경하기 위해 네트워크를 통한 센서 노드의 프로그램 변경 기능은 필수적이다.
- <22> 일반적으로 센서 노드는 제한된 파워를 가지므로 다시 프로그램하는 과정이 에너지 관점에서 효율적이어야 한다. 이 효율성은 프로그램 코드의 전송과 수행을 말한다.
- <23> TinyOS는 센서 노드의 가장 대표적인 운영체제로 사용되고 있다(Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System Architecture Directions for Networked Sensors. In the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems, 2000.), (P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler. The Emergence of Networking Abstractions and Techniques in Tinyos. In Proceedings of the First Symposium on Networked Systems Design and Implementation, pages 1-14. USENIX Association, 2004.).
- <24> 최근, 대부분의 센서 네트워크 어플리케이션은 TinyOS을 기반으로 개발되었으며, 센서 노드의 한정된 자원을 위해 디자인 되었다.
- <25> TinyOS는 XNP(Crossbow Network Programming)(Jaein Jeong, Sukun Kim and Alan Broad. Network Reprogramming. TinyOS document, .), (Crossbow Technology Note In Network Programming User Reference. TinyOS document, . 24)와 Deluge(Jonathan W. Hui and David Culler. The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale. In proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems. 2004.)을 사용한다.
- <26> XNP는 TinyOS 1.1 release에 나와 있는 네트워크 프로그래밍 구조이며, TinyOS 프로그램 이미지는 컴파일할 때 정적으로 링크된 것이기 때문에 모든 프로그램 이미지 전체를 업데이트 해야 한다.
- <27> Deluge는 멀티-홉(다중도약) 지원을 위한 데이터 전송 프로토콜이다. 큰 프로그램 이미지를 멀티-홉 센서 네트워크를 통해 극소수 노드에서 다수 노드로 전송하는 것을 지원한다. 각 노드들은 주기적으로 프로그램 이미지의 버전을 알리게 된다. 만약, 노드가 새로운 버전을 받아들이게 되면, 노드는 새로운 프로그램 이미지를 요청하고 프로그램 전체의 이미지를 받아 새로운 프로그램 이미지의 버전을 알리게 된다.
- <28> 그러나 TinyOS 네트워크 리프로그래밍 구조의 문제점은 꼭 프로그램 이미지 전체를 사용해야 한다는 것이다. 이는 다음과 같은 문제점을 갖게 된다.

- <29> 첫째, 전체 프로그램 이미지가 전송되어야 하므로 코드 전송 손실이 너무 많다는 것이다.
- <30> 둘째, 전체 프로그램 이미지가 보급되어야 하므로 증가된 업데이트는 불가능하다.
- <31> 셋째, 만약 새로운 프로그램 이미지에 오류가 있으면, 깨진 센서 노드를 복원시킬 수 없어 업데이트 된 프로그램 이미지가 센서 노드를 망가뜨릴 수 있다는 것이다.
- <32> 따라서, 에너지 측면에서 효율적인 네트워크 리프로그래밍 구조가 아니라는 단점이 있다.
- <33> 이와 같이, TinyOS의 네트워크 리프로그래밍 구조는 반드시 프로그램 이미지 전체를 전송해야 하므로 에너지 관점에서 볼 때 효과적이지 못하다.
- <34> 이 밖에도 가상 머신을 이용해 가상 머신의 코드인 바이트 코드를 네트워크를 통해 전송하는 방식인 Mate가 있다.
- <35> Mate는 TinyOS 네트워크 리프로그래밍 구조의 문제점을 해결하였다. Mate는 네트워크 센서를 위한 작은 가상 머신이다. 이 가상 머신은 네트워크 리프로그래밍을 하는데 있어서 몇 가지 장점이 있다. 분포된 센서 노드를 리프로그래밍하기 위해서, 가상 머신의 명령어로 바이트 코드를 사용한다. 가상 머신의 복잡한 기능을 바이트 코드로 인용한 것이므로, 알려진 바와 같이 간단한 바이트 코드의 세트로 복잡한 프로그램을 표현할 수 있다. 가상 머신 위에서 실행되었기 때문에 바이트 코드는 탄력성(시스템이 부분적으로 고장 나도 처리를 실행할 수 있는 능력)이 있다.
- <36> Mate의 눈에 띄는 특징은 네트워크 리프로그래밍이 수백개의 바이트 코드를 보내는 것처럼 실행된다는 것이다. 이것은 Mate가 새로운 프로그램 코드를 빨리 전송하는 것을 가능하게 해준다. 그러므로, Mate는 에너지 측면에서 효율적인 프로그램 코드 전송 구조와 탄력성을 제공해준다.
- <37> 그러나, 가상 머신 위에서 바이트 코드가 실행되어야 하므로 Mate는 엄청난 오버헤드를 포함하게 된다.
- <38> 첫째, 실제 머신의 네이티브 코드가 아니기 때문에 인터프리테이션 오버헤드가 발생한다.
- <39> 둘째, 가상 머신의 오퍼랜드 스택 위에서 실행되므로 오퍼랜드 스택 오버헤드를 포함한다.
- <40> 만약, 복잡한 계산을 필요로 하는 센서 네트워크 어플리케이션의 경우, 바이트 코드를 이용하여 실행을 하게 되면 수행할 연산량과 소비 에너지가 매우 커지게 된다.
- <41> 이 방식의 경우 연산에 드는 에너지가 매우 크기 때문에 무선 센서 네트워크 어플리케이션을 변경하기 않고 계속 사용하게 된다면 프로그램 이미지 전체를 네트워크로 전송하는 방식보다 더 큰 에너지를 소모할 수 있다.

발명이 이루고자 하는 기술적 과제

- <42> 본 발명은 이러한 문제점을 해결하기 위한 것으로, 본 발명의 목적은 센서 노드의 프로그램을 변경하기 위해 다운로드 관리자 모듈과 심볼 테이블을 센서 노드 내부의 가상 머신에 추가시켜 네이티브 코드로 된 프로그램을 실행가능하게 하고, 기존 가상 머신의 오버헤드를 줄여 프로그램의 변경과 실행을 저전력화할 수 있도록 한 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신 및 이를 이용한 리프로그래밍 방법을 제공함에 있다.

발명의 구성 및 작용

- <43> 상기 목적을 달성하기 위한 본 발명에 따른 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신은, 바이트 코드가 저장되는 캡슐 및 상기 캡슐에 저장된 바이트 코드를 수행하며 동적 모듈을 호출하는 바이트 코드 인터프리터를 구비하는 무선 센서 네트워크 환경에서 이미 배치된 센서 노드를 다시 프로그램하기 위한 가상 머신에 있어서, 업데이트되는 동적 모듈의 프로그램 메모리 주소가 저장되는 심볼 테이블; 및 네트워크를 통해 새로운 동적 모듈을 다운받고 이를 상기 프로그램 메모리로 로드하며, 상기 동적 모듈이 저장된 프로그램 메모리의 주소를 상기 심볼 테이블에 업데이트하는 다운로드 관리자;를 더 포함하여 구성되는 것을 특징으로 한다.
- <44> 상기 동적 모듈은 상기 프로그램 메모리의 중간의 빈 공간 임의의 위치에 로드된다.
- <45> 상기 다운로드 관리자는 상기 네트워크를 통해 수신되며, 상기 동적 모듈을 호출하는 바이트 코드의 이름, 동적

모듈의 버전, 사이즈, 전체 패킷 수에 대한 정보를 포함하는 메타데이터 패킷에 따라 상기 동적 모듈을 다운받는다.

<46> 또한, 상기 다운로드 관리자는 센서 노드가 배치된 후 추가된 동적 모듈의 정보를 포함하는 동적 모듈 테이블; 및 상기 프로그램 메모리의 빈 페이지 정보를 갖는 프로그램 메모리 페이지 리스트;를 포함하며, 상기 동적 모듈 테이블은 상기 메타데이터 패킷의 정보와 상기 동적 모듈이 로드된 프로그램 메모리 주소 정보를 갖는다.

<47> 상기 목적을 달성하기 위한 본 발명에 따른 무선 센서 네트워크 환경에서 프로그램을 변경하기 위한 확장 가능한 가상 머신을 이용한 리프로그래밍 방법은, 바이트 코드가 저장되는 캡슐; 상기 캡슐에 저장된 바이트 코드를 수행하며 동적 모듈을 호출하는 바이트 코드 인터프리터; 업데이트되는 동적 모듈의 프로그램 메모리 주소가 저장되는 심볼 테이블; 및 네트워크를 통해 새로운 동적 모듈을 다운받고 이를 상기 프로그램 메모리로 로드하며, 상기 동적 모듈이 저장된 프로그램 메모리의 주소를 상기 심볼 테이블에 업데이트하는 다운로드 관리자;를 포함하는 무선 센서 네트워크 환경에서의 확장 가능한 가상 머신을 이용한 리프로그래밍 방법에 있어서, 상기 네트워크를 통해 메타데이터 패킷이 수신되면 상기 다운로드 관리자는 네트워크를 통해 새로운 동적 모듈을 다운받아 이를 상기 프로그램 메모리로 로드하는 제1단계; 상기 동적 모듈이 저장된 프로그램 메모리의 주소를 상기 심볼 테이블에 업데이트하는 제2단계; 상기 동적 모듈을 호출하는 바이트 코드를 상기 바이트 코드 인터프리터가 해석하여 해석 결과에 따라 상기 심볼 테이블을 검색하여 동적 모듈이 저장된 프로그램 메모리의 주소를 얻는 제3단계; 및 상기 제3단계에서 얻어진 프로그램 메모리 주소에 따라 상기 바이트 코드 인터프리터는 그 바이트 코드에 해당하는 동적 모듈을 호출하여 호출된 동적 모듈이 실행되도록 하는 제4단계;를 포함하여 이루어짐을 특징으로 한다.

<48> 상기 메타데이터 패킷이 수신되면 상기 다운로드 관리자는, 센서 노드가 배치된 후 추가된 동적 모듈의 정보를 가지고 있는 동적 모듈 테이블을 이용하여 업데이트해야 할 동적 모듈의 버전을 판단한다.

<49> 또한, 상기 동적 모듈의 프로그램 메모리로의 로드는, 상기 다운로드 관리자로부터의 프로그램 메모리의 부트 로더에게의 쓰기 요청에 따라, 상기 프로그램 메모리의 빈 페이지 정보를 갖는 프로그램 메모리 페이지 리스트의 빈 페이지에, 상기 동적 모듈이 로드된다.

<50> 그리고 상기 제4단계에서 상기 바이트 코드 인터프리터는, 상기 바이트 코드의 해석 결과에 따라 얻어지는 심볼 테이블 주소를 바탕으로 함수 포인터를 이용하여 바이트 코드에 해당하는 동적 모듈을 호출하도록 되며, 상기 동적 모듈 호출시 상기 바이트 코드 인터프리터는 함수의 인자로 오퍼랜드 스택의 포인터를 상기 동적 모듈로 전달한다.

<51> 상기 동적 모듈은 상기 전달받은 오퍼랜드 스택의 포인터를 사용하여 계산에 필요한 피연산자를 상기 오퍼랜드 스택에서 가져오고, 피연산자를 연산한 결과를 다시 상기 오퍼랜드 스택의 포인터를 사용하여 오퍼랜드 스택에 넘겨주고 리프로그래밍 과정을 종료한다.

<52> 이하, 본 발명의 바람직한 실시 예를 첨부된 도면을 참조하여 보다 상세하게 설명한다. 단, 하기 실시 예는 본 발명을 예시하는 것일 뿐 본 발명의 내용이 하기 실시 예에 한정되는 것은 아니다.

<53> 도 1은 본 발명에 따른 무선 센서 네트워크 노드 위에 올라가는 확장 가능한 가상 머신의 전체 구조를 보여주고 있다.

<54> 본 발명에 따른 확장 가능한 가상 머신(101)은 에너지 측면에서 효율적으로 센서 노드 내부의 어플리케이션을 리프로그래밍하기 위해서 가상 머신(105)에 추가적으로 동적 모듈(104)을 실행 중에 로드하고 링킹하는 기능을 가지는 모듈을 추가한다.

<55> 기존의 가상 머신(105)이 정적인 기능(107)만을 가지기 때문에 센서 노드를 다시 프로그램하기 위해서 바이트 코드만을 사용해야 하고, 바이트 코드 인터프리터(106)가 바이트 코드로 된 어플리케이션을 실행하기 위해 인터프리터, 오퍼랜드 스택, 바이트 코드의 스케줄 등의 많은 실행 오버헤드를 가지기 때문이다. 따라서 확장 가능한 가상 머신(101)은 동적 모듈(104)을 실행 중에 동적으로 로드하고 링킹하기 위해 다운로드 관리자(102)와 심볼 테이블(103)을 가진다.

<56> 우선, 확장 가능한 가상 머신(1)의 바탕이 되는 소프트웨어 플랫폼을 살펴보면, 무선 센서 네트워크 어플리케이션은 일반적으로 TinyOS(109)를 운영 체제로 사용하여 센서 노드를 프로그램한다. TinyOS(109)는 센서, 액츄에이터, 통신을 위한 디바이스 드라이버와 기본적인 스케줄러 기능을 제공한다.

- <57> TinyOS(109)환경에서 센서 노드의 어플리케이션의 리프로그래밍을 가능하게 하기 위해 만들어진 기존의 가상 머신(105)은 TinyOS(109) 상에서 동작한다. 확장 가능한 가상 머신(101)의 바탕이 되는 기존의 가상 머신(105)은 간단히 바이트코드 인터프리터(106) 역할을 한다.
- <58> 그리고 바이트코드는 캡슐(Capsules)(108) 단위로 저장된다. 각 캡슐(108)의 바이트코드는 특정 이벤트가 발생하면 수행된다. 여기서, 특정 이벤트는 Clock, Send, Receive 세 가지로 정의하고 있다. Clock은 센서 노드의 내부 타이머 인터럽트에 의해 발생한다. Send는 센서 노드가 패킷을 네트워크로 보낼 때 발생한다. Receive는 패킷을 받으면 발생한다.
- <59> 확장 가능한 가상 머신(101)의 핵심은 어플리케이션에서 필요한 연산을 하는 부분을 동적 모듈(104)로 작성하여 이 동적 모듈(104)의 프로그램 이미지를 네트워크를 통해 배치된 센서 노드에게 동적으로 로드하고 링킹할 수 있다는 것이다. 통상 상기 동적 모듈(104)은 어플리케이션의 계산 부분을 구현한다.
- <60> 도 2는 동적 모듈(104)이 실제 센서 노드의 프로그램 메모리(201)에 로드되고 링크되는 과정을 설명하기 위한 도이다.
- <61> 프로그램 메모리(201)는 무선 센서 네트워크 환경에서 사용되는 마이크로 컨트롤러의 프로그램 메모리이다. 이 프로그램 메모리(201)는 플래쉬 메모리이므로 한 번의 쓰기 단위가 한 페이지 256바이트 단위로 이루어진다. 동적 모듈(104)을 프로그램 메모리(201)에 쓰기 위해 다운로드 관리자(102)는 프로그램 메모리(201)의 페이지를 관리한다.
- <62> 프로그램 메모리(201)에 코드를 쓰기 위해서는 부트 로더 코드 영역(204)에 코드를 쓰는 코드가 포함되어야 한다. 프로그램 메모리(201)의 안전을 위해서 어플리케이션 코드 영역(203)의 코드가 프로그램 메모리(201)에 코드를 쓸 수 없기 때문이다.
- <63> 마이크로 컨트롤러를 사용하는 센서 노드의 프로그램 메모리(201)는 도 2에서처럼 초기 센서 노드를 환경에 배치할 때 TinyOS(109), 확장 가능한 가상 머신(101)이 프로그램 메모리(201)의 상위 부분에, 부트 로더(202)가 하위 부분에 로드된다.
- <64> 이에, 동적 모듈(104)은 중간의 빈 프로그램 메모리(201)의 공간의 임의의 위치에 저장되게 된다. 동적 모듈(104)은 확장 가능한 가상 머신(101)의 바이트 코드 인터프리터(106)로부터 호출 시 함수의 인자로 오퍼랜드 스택(509)의 포인터를 가지게 된다. 이를 이용해 계산에 필요한 피연산자를 가져오고 계산된 결과를 다시 확장 가능한 가상 머신(101)에게 전달한다.
- <65> 도 3 및 도 4는 확장 가능한 가상 머신(101)의 다운로드 관리자(102)의 구조 및 기능을 설명하기 위한 도이다.
- <66> 확장 가능한 가상 머신(101)은 기존의 가상 머신(105)을 동적으로 만들기 위해 다운로드 관리자(102)와 심플 테이블(103)이 필요하다. 다운로드 관리자(102)는 동적 모듈(104)을 네트워크로부터 다운로드하고, 확장 가능한 가상 머신(101)의 바이트 코드 인터프리터(106)가 동적 모듈(104)을 호출하는 바이트 코드를 수행할 경우, 그 동적 모듈(104)을 호출할 수 있도록 다운받은 동적 모듈(104)을 센서 노드의 프로그램 메모리(201)에 로드한다.
- <67> 다운로드 관리자(102)가 동적 모듈(104)을 네트워크로부터 다운로드하는 과정은 다음과 같다.
- <68> 우선, 어플리케이션 개발자는 센서 노드를 리프로그래밍할 경우 호스트 머신에서 새로운 프로그램을 작성한다. 개발자는 에너지 측면에서 효율적으로 무선 센서 네트워크를 다시 프로그래밍하기 위해 새로 작성하는 프로그램은 기존의 가상 머신과 같이 어플리케이션을 위한 바이트 코드와 추가적으로 많은 계산이 필요한 부분은 동적 모듈(104) 형태로 구현하여 작성해야 한다.
- <69> 개발자는 새롭게 작성된 어플리케이션을 센서 노드에게 전달하기 위해 도 3과 같이 새로운 동적 모듈(104)의 이미지가 있다는 것을 알리기 위한 메타데이터 패킷(301)을 네트워크를 통해 보낸다.
- <70> 이 패킷(301)은 동적 모듈(104)을 호출하는 바이트 코드의 이름과 동적 모듈(104)의 버전, 사이즈, 전체 패킷 수 정보를 포함한다.
- <71> 그리고 이 패킷(301)을 받은 센서 노드는 동적 모듈(104)을 다운로드 하기 위한 상태가 된다. 다운로드 관리자(102)는 이 정보를 가지고 업데이트해야 할 동적 모듈(104)이 최신 버전인지 동적 모듈 테이블(302)과 비교하여 판단한다.
- <72> 동적 모듈 테이블(302)은 센서 노드가 배치된 후, 새롭게 확장 가능한 머신(101)에 추가된 동적 모듈(104)의 정

보를 포함한다. 이 동적 모듈 테이블(302)에는 메타 데이터 패킷(301)의 정보와 동적 모듈(104)이 로드된 실제 프로그램 메모리(201) 주소가 저장된다.

- <73> 다운로드 관리자(102)는 추가적으로 프로그램 메모리 페이지 리스트(303)를 관리하고, 동적 모듈(104)의 프로그램 이미지를 완전히 다운받은 후, 상기 프로그램 메모리 페이지 리스트(303)에서 빈 페이지(304)에 동적 모듈(104)의 이미지를 쓰라고 부트 로더(202)에게 요청한다.
- <74> 센서 노드는 이 패킷(301)을 전달한 호스트 머신 혹은 센서 노드에게 새로운 동적 모듈(104)의 이미지를 요청한다.
- <75> 도 4는 다운로드가 완료된 후 프로그램 메모리(201)에 동적 모듈(104)의 코드 이미지를 쓰는 과정을 나타낸다. 다운로드 관리자(102)는 동적 모듈(104)을 프로그램 메모리(201)에 쓰기 위해서 부트 로더 코드 영역(204)에 들어있는 프로그램 메모리(201)에 코드를 쓰는 함수를 호출한다.
- <76> 부트 로더(202)에게 쓰기 요청시 다운로드 관리자(102)는 동적 모듈(104)의 코드가 저장되어 있는 프로그램 메모리(201)의 주소와 이 동적 모듈(104)이 프로그램 메모리(201)에 저장될 주소를 함수 호출시 인자로 넘겨주게 된다.
- <77> 부트 로더(202)에 포함된 프로그램 코드를 쓰는 함수는 이 인자를 사용하여 동적 모듈(104)을 실제 프로그램 메모리(201)에 로드한다. 성공적으로 프로그램 메모리(201)에 동적 모듈(104)을 쓰고 나면 이 동적 모듈(104)이 저장된 프로그램 메모리(201)의 시작 주소를 심볼 테이블(103)에 업데이트한다.
- <78> 심볼 테이블(103)은 동적으로 업데이트된 모듈(104)의 실제 프로그램 메모리 주소를 저장하는 테이블이다. 즉, 이 심볼 테이블(103)은 다운로드 관리자(102)에 의해서 업데이트된다.
- <79> 도 5는 본 발명에 따른 확장 가능한 가상 머신(101)의 동적 모듈(104)을 링킹하는 과정을 나타낸 도이다.
- <80> 확장 가능한 가상 머신(101)은 동적 모듈(102)을 호출하는 바이트 코드를 실행하면 이 바이트 코드는 바이트 코드 인터프리터(106)에 의해 해석되어 심볼 테이블(103)의 주소를 가지고 온다(①),(②).
- <81> 상기 바이트 코드 인터프리터(106)는 심볼 테이블(103)의 주소를 가지고 함수 포인터를 이용해 바이트 코드에 해당하는 동적 모듈(104)을 호출한다(③). 동적 모듈(104)을 호출 시 확장 함수의 인자로 오퍼랜드 스택(509)의 포인터를 전달한다.
- <82> 오퍼랜드 스택(509)은 확장 가능한 가상 머신(101)이 동적 모듈(104)과 계산에 필요한 혹은 계산된 데이터를 서로 주고받는 유일한 방법이다. 동적 모듈(104)은 함수 포인터에 의한 함수의 호출시 인자로 전달받은 오퍼랜드 스택의 포인터(509)를 사용하여 계산에 필요한 피연산자를 오퍼랜드 스택(509)에서 가지고 온다.
- <83> 그리고 상기 동적 모듈(104)은 이 피 연산자를 사용하여 어플리케이션의 계산 부분을 수행하고 그 결과를 다시 오퍼랜드 스택(509)의 포인터를 사용하여 오퍼랜드 스택(509)으로 넘겨준다.
- <84> 즉, 도 6의 본 발명의 전체적인 동작 흐름도에 나타난 바와 같이, 본 발명은 기본적인 종래의 가상 머신(105)의 기능에 동적으로 모듈을 추가하고 수정하기 위해 다운로드 관리자(102)와 심볼 테이블(103)을 가지며, 네트워크를 통해 메타데이터 패킷(301)이 수신되면 다운로드 관리자(102)는 수행을 시작한다.
- <85> 다운로드 관리자(102)는 네트워크를 통해 새로운 동적 모듈(104)을 다운받고 이를 프로그램 메모리(201)로 로드한다(S10). 그리고 동적 모듈(104)이 저장된 프로그램 메모리(201)의 주소를 심볼 테이블(103)에 업데이트한다(S20).
- <86> 동적 모듈(104)의 업데이트가 끝나고 동적 모듈(104)을 호출하는 바이트 코드가 수행되면 동적 모듈(104)이 실행된다. 즉, 바이트 코드 인터프리터(106)는 실행할 바이트 코드가 어떤 동적 모듈(104)을 호출할지 해석하고, 심볼 테이블(103)을 검색하여 동적 모듈(104)이 저장된 프로그램 메모리(201) 주소를 얻는다(S30).
- <87> 이 주소를 사용해 바이트 코드 인터프리터(106)는 그 바이트코드에 해당하는 동적 모듈(104)을 호출한다(S40). 호출된 동적 모듈(104)은 어플리케이션 실행시 필요한 부분을 수행한다(S50).
- <88> 또한, 바이트 코드 인터프리터(106)는 동적 모듈(104)을 호출할 때 전달 인자로 가상 머신(105)의 오퍼랜드 스택(509)의 포인터를 넘겨준다. 동적 모듈(104)은 계산에 필요한 피연산자를 오퍼랜드 스택(509)에서 가져와 피연산자를 연산한 결과를 다시 오퍼랜드 스택(509)에 넘겨주고 수행 과정을 종료한다.
- <89> 상술한 바와 같이, 본 발명의 바람직한 실시 예를 참조하여 설명하였지만, 해당 기술 분야의 숙련된 당업자는

하기의 특허청구범위에 기재된 본 발명의 사상 및 영역으로부터 벗어나지 않는 범위 내에서 본 발명을 다양하게 수정 또는 변형하여 실시할 수 있다.

발명의 효과

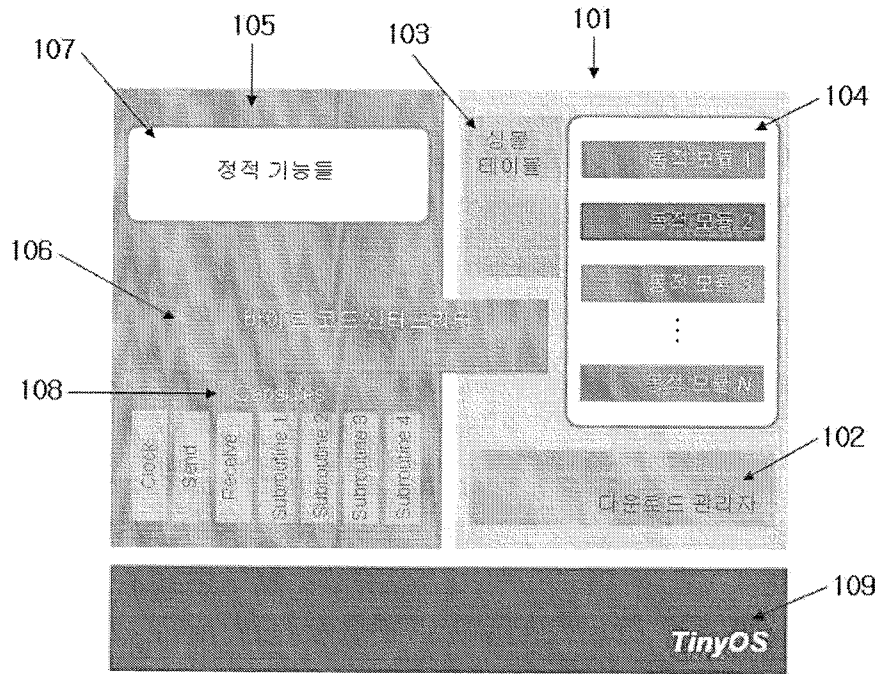
- <90> 이상에서 살펴본 바와 같이, 본 발명은 무선 센서 네트워크 환경에서 센서 노드의 프로그램을 변경하기 위해 다운로드 관리자 모듈과 심볼 테이블을 센서 노드 내부의 가상 머신에 추가시켜 네이티브 코드로 된 프로그램을 실행가능하게 함으로써 기존 가상 머신의 오버헤드를 줄여 프로그램의 변경과 실행을 저전력화 할 수 있는 효과가 있다.
- <91> 또한, 본 발명은 동적인 프로그램 변경이 필요한 모든 무선 센서 네트워크 어플리케이션에 적용 가능한 효과가 있다.

도면의 간단한 설명

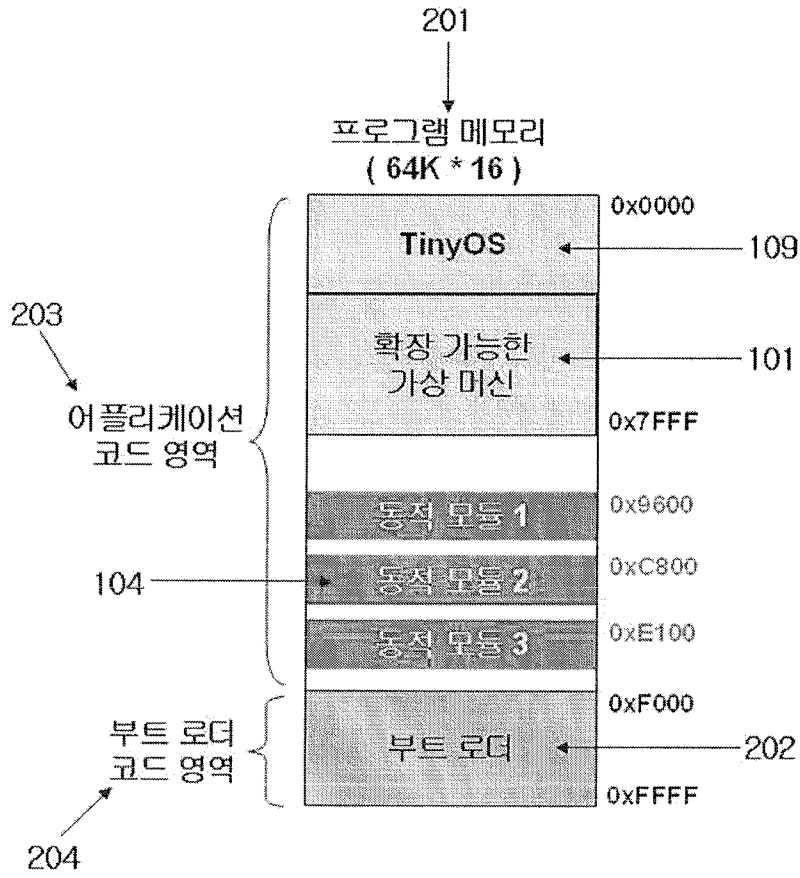
- <1> 도 1은 본 발명에 따른 확장 가능한 가상 머신의 전체적인 구조도.
- <2> 도 2는 본 발명에 따라 동적 모듈이 실제 센서 노드의 프로그램 메모리에 로드되고 링크되는 과정을 설명하기 위한 도.
- <3> 도 3은 본 발명에 따른 다운로드 관리자의 구조 및 확장 가능한 가상 머신이 새로운 모듈을 받았을 경우의 다운로드 관리자의 기능 설명도.
- <4> 도 4는 본 발명에 따른 확장 가능한 가상 머신의 다운로드 관리자가 동적 모듈을 센서 노드의 프로그램 메모리로 로드하는 과정의 설명도.
- <5> 도 5는 본 발명에 따른 확장 가능한 가상 머신의 바이트 코드 인터프리터가 동적 모듈을 링크하는 과정의 설명도.
- <6> 도 6은 본 발명의 전체적인 동작 흐름도.
- <7> <도면의 주요 부분에 대한 부호의 설명>
- <8> 101 : 확장 가능한 가상 머신 102 : 다운로드 관리자
- <9> 103 : 심볼 테이블 104 : 동적 모듈
- <10> 106 : 바이트 코드 인터프리터 108 : 캡슐
- <11> 109 : TinyOS 201 : 프로그램 메모리
- <12> 202 : 부트 로더 203 : 어플리케이션 코드 영역
- <13> 204 : 부트 로더 코드 영역 301 : 메타데이터 패킷
- <14> 302 : 동적 모듈 테이블 304 : 빈 페이지
- <15> 303 : 프로그램 메모리 페이지 리스트
- <16>

도면

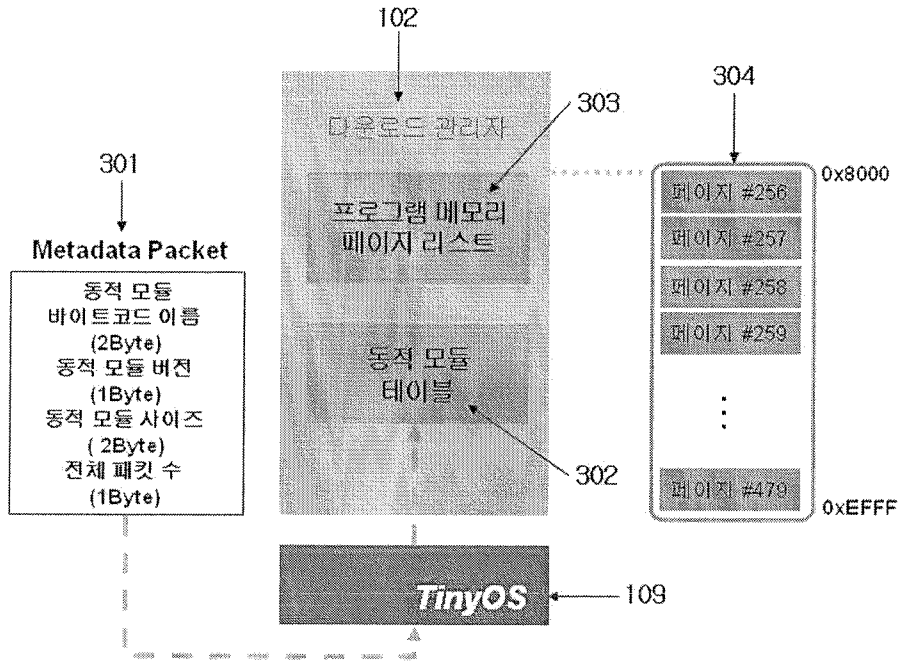
도면1



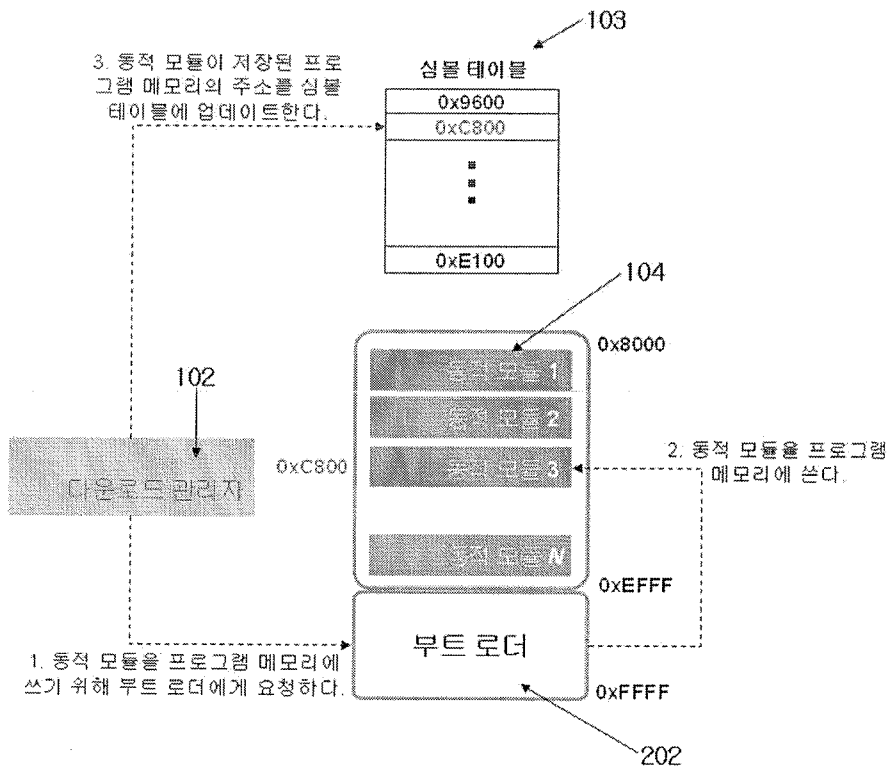
도면2



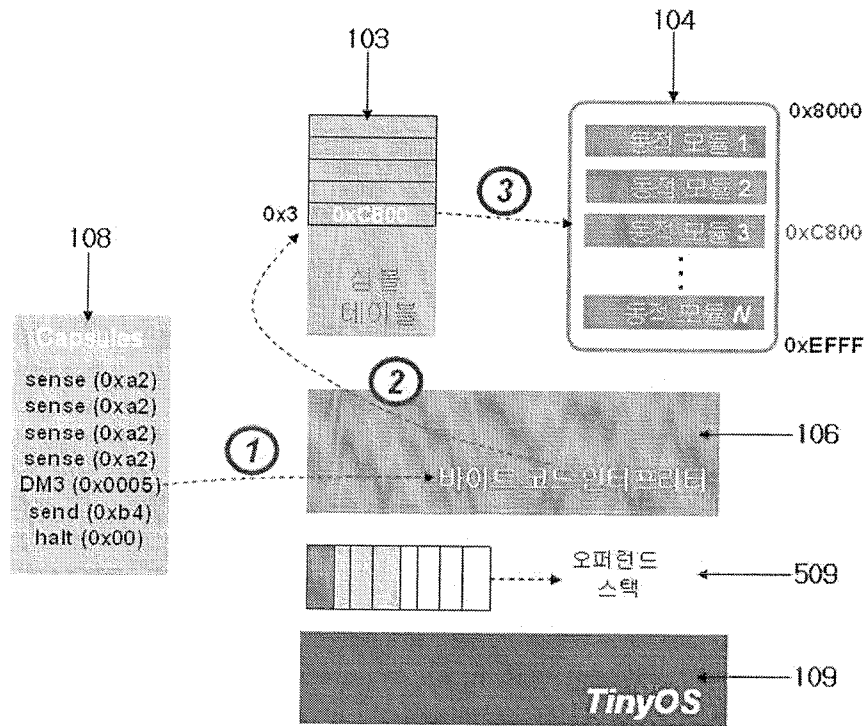
도면3



도면4



도면5



도면6

