(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2019년08월14일
(11) 등록번호 10-2011215
(24) 등록일자 2019년08월08일

(51) 국제특허분류(Int. Cl.)
H04L 12/24 (2006.01) *H04L 29/08* (2006.01)
(52) CPC특허분류
H04L 41/145 (2013.01)
H04L 67/08 (2013.01)
(21) 출원번호 10-2018-0050797
(22) 출원일자 2018년05월02일
심사청구일자 2018년05월02일
(56) 선행기술조사문헌
KR1020100020822 A*
KR1020100073127 A*
KR1020150058943 A*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
세종대학교산학협력단
서울특별시 광진구 능동로 209 (군자동, 세종대학교)
(72) 발명자
아볼가셈
서울특별시 광진구 능동로 209
최수미
서울특별시 광진구 능동로 209
(74) 대리인
양성보

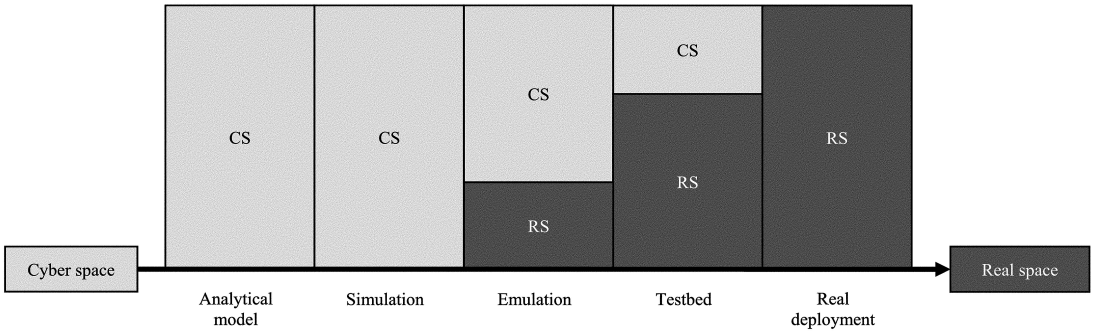
전체 청구항 수 : 총 4 항 심사관 : 이주민

(54) 발명의 명칭 **유비쿼터스 센서 네트워크의 시뮬레이션 및 에뮬레이션 환경**

(57) 요 약

유비쿼터스 센서 네트워크의 시뮬레이션 및 에뮬레이션 환경이 개시된다. 유비쿼터스 센서 네트워크의 성능을 평가하기 위한 장치는, 컴퓨터에서 판독 가능한 명령을 실행하도록 구현되는 적어도 하나의 프로세서를 포함하고, 상기 적어도 하나의 프로세서는, 상기 유비쿼터스 센서 네트워크의 도메인에서 무선 네트워크 프로토콜과 통신 프레임워크의 성능을 평가하는 시뮬레이션을 수행하는 시뮬레이터; 및 상기 유비쿼터스 센서 네트워크 내에서 노드와 관련된 성능을 평가하는 에뮬레이션을 수행하는 에뮬레이터를 포함할 수 있다.

대 표 도



이 발명을 지원한 국가연구개발사업
과제고유번호 1711070409(세부과제번호: 2016-0-00312-003)
부처명 과학기술정보통신부(P71)
연구관리전문기관 정보통신기술진흥센터(ACJ0788)
연구사업명 정보통신기술인력양성(R&D,정보화)(1350801301314500453830 2)
연구과제명 모바일 플랫폼 기반 엔터테인먼트 VR 기술 연구
기 여 율 1/1
주관기관 세종대학교 산학협력단
연구기간 2018.01.01 ~ 2018.12.31
공지예외적용 : 있음

명 세 서

청구범위

청구항 1

유비쿼터스 센서 네트워크(ubiquitous sensor network)의 성능을 평가하기 위한 장치에 있어서,

컴퓨터에서 판독 가능한 명령을 실행하도록 구현되는 적어도 하나의 프로세서

를 포함하고,

상기 적어도 하나의 프로세서는,

상기 유비쿼터스 센서 네트워크의 도메인에서 무선 네트워크 프로토콜과 통신 프레임워크의 성능을 평가하는 시뮬레이션(simulation)을 수행하는 시뮬레이터; 및

상기 유비쿼터스 센서 네트워크 내에서 노드의 성능을 평가하는 에뮬레이션(emulation)을 수행하는 에뮬레이터

를 포함하고,

상기 유비쿼터스 센서 네트워크에 대한 시뮬레이터와 에뮬레이터를 비교 조사하여 상기 유비쿼터스 센서 네트워크의 성능을 평가하기 위한 것으로,

상기 시뮬레이션의 결과와 상기 에뮬레이션의 결과를 그래프, 텍스트 파일, 및 추적 파일의 애니메이션으로 출력하되,

다수의 프로토콜 간 비교가 가능하도록 상기 그래프를 통해 네트워크 성능 평가 파라미터를 나타내고,

상기 텍스트 파일을 통해 다른 시뮬레이터 또는 프로그램에 대한 입력을 나타내고,

상기 추적 파일을 통해 시뮬레이션 과정에서 발생하는 이벤트를 나타내고,

상기 시뮬레이터는,

무선 네트워크 동작(operation) 및 프로토콜 스택을 중심으로 지향 네트워크(oriented network)의 시뮬레이션을 수행하고,

통신 모델을 포함하는 노드의 기능을 중심으로 지향 노드(oriented node)의 시뮬레이션을 수행하고,

상기 시뮬레이터는,

상기 유비쿼터스 센서 네트워크 내에서 하드웨어와 소프트웨어로 구성된 노드; 노드의 기능을 나타내는 이벤트; 노드 간에 신호를 전송하는 중간 모듈(medium module); 모델링 환경을 전달하는 환경 모듈(environment module); 각 노드의 상태를 결정하는 송수신 하드웨어; 및 물리적인 프로토콜, MAC 프로토콜, 라우팅 프로토콜, 어플리케이션 계층으로 이루어진 네트워크 스택;을 포함하는 다중 모듈로 구성되고,

추상화 수준(abstraction levels)에 기반하여 카테고리화 됨에 따라, 네트워크 시뮬레이션을 위한 네트워크 시뮬레이터; 코드 시뮬레이션을 위한 코드 레벨 시뮬레이터; 펌웨어 시뮬레이션을 위한 펌웨어 레벨 시뮬레이터; 알고리즘 시뮬레이션을 위한 알고리즘 레벨 시뮬레이터; 네트워크 패킷 시뮬레이션을 위한 패킷 레벨 시뮬레이터; 및 명령(instruction) 시뮬레이션을 위한 명령 레벨 시뮬레이터를 포함하고,

상기 에뮬레이터는,

물리적 계층과 MAC 계층 클래스로 카테고리화되고,

네트워크 시뮬레이터에 포함되고,

상기 유비쿼터스 센서 네트워크에 대한 평가 도구가 노드 모델을 가진 네트워크 시뮬레이터, 노드 에뮬레이터를 가진 네트워크 시뮬레이터, 네트워크 모델을 가진 노드 시스템 시뮬레이터, 네트워크 모델을 가진 노드 에뮬레이터로 분류되는 것

을 특징으로 하는, 장치.

청구항 2

삭제

청구항 3

삭제

청구항 4

삭제

청구항 5

제1항에 있어서,

상기 시뮬레이터는,

시스템 내 현상에 대한 시간 순서 히스토리로 정의된 자취(trace)를 입력으로 사용하는 트레이스-구동(trace-driven) 시뮬레이션, 또는 시스템의 이산 상태 모델(discrete-state model)을 사용하는 이산-이벤트(discrete-event) 시뮬레이션을 수행하는 것

을 특징으로 하는, 장치.

청구항 6

삭제

청구항 7

제1항에 있어서,

네트워크 모델을 가진 노드 에뮬레이터는 특정 마이크로컨트롤러를 위한 명령 세트 시뮬레이터와 운영체제(OS)의 어플리케이션 코드 실행을 에뮬레이트하기 위한 에뮬레이터로 분류되는 것

을 특징으로 하는, 장치.

청구항 8

삭제

청구항 9

유비쿼터스 센서 네트워크의 성능을 평가하기 위한 방법에 있어서,

상기 유비쿼터스 센서 네트워크의 도메인에서 무선 네트워크 프로토콜과 통신 프레임워크의 성능을 평가하는 시뮬레이션을 수행하는 단계; 및

상기 유비쿼터스 센서 네트워크 내에서 노드와 관련된 성능을 평가하는 에뮬레이션을 수행하는 단계

를 포함하고,

상기 유비쿼터스 센서 네트워크에 대한 시뮬레이터와 에뮬레이터를 비교 조사하여 상기 유비쿼터스 센서 네트워크의 성능을 평가하기 위한 것으로,

상기 시뮬레이션의 결과와 상기 에뮬레이션의 결과를 그래프, 텍스트 파일, 및 추적 파일의 애니메이션으로 출력하는 단계

를 더 포함하고,

다수의 프로토콜 간 비교가 가능하도록 상기 그래프를 통해 네트워크 성능 평가 파라미터를 나타내고,

상기 텍스트 파일을 통해 다른 시뮬레이터 또는 프로그램에 대한 입력을 나타내고,

상기 추적 파일을 통해 시뮬레이션 과정에서 발생하는 이벤트를 나타내고,

상기 시뮬레이션을 수행하는 단계는,

무선 네트워크 동작(operation) 및 프로토콜 스택을 중심으로 지향 네트워크(oriented network)의 시뮬레이션을 수행하는 단계; 및

통신 모델을 포함하는 노드의 기능을 중심으로 지향 노드(oriented node)의 시뮬레이션을 수행하는 단계

를 포함하고,

상기 시뮬레이션을 위한 시뮬레이터는,

상기 유비쿼터스 센서 네트워크 내에서 하드웨어와 소프트웨어로 구성된 노드; 노드의 기능을 나타내는 이벤트; 노드 간에 신호를 전송하는 중간 모듈(medium module); 모델링 환경을 전달하는 환경 모듈(environment module); 각 노드의 상태를 결정하는 송수신 하드웨어; 및 물리적인 프로토콜, MAC 프로토콜, 라우팅 프로토콜, 어플리케이션 계층으로 이루어진 네트워크 스택;을 포함하는 다중 모듈로 구성되고,

추상화 수준(abstraction levels)에 기반하여 카테고리화 됨에 따라, 네트워크 시뮬레이션을 위한 네트워크 시뮬레이터; 코드 시뮬레이션을 위한 코드 레벨 시뮬레이터; 펌웨어 시뮬레이션을 위한 펌웨어 레벨 시뮬레이터; 알고리즘 시뮬레이션을 위한 알고리즘 레벨 시뮬레이터; 네트워크 패킷 시뮬레이션을 위한 패킷 레벨 시뮬레이터; 및 명령(instruction) 시뮬레이션을 위한 명령 레벨 시뮬레이터를 포함하고,

상기 에뮬레이션을 위한 에뮬레이터는,

물리적 계층과 MAC 계층 클래스로 카테고리화되고,

네트워크 시뮬레이터에 포함되고,

상기 유비쿼터스 센서 네트워크에 대한 평가 도구가 노드 모델을 가진 네트워크 시뮬레이터, 노드 에뮬레이터를 가진 네트워크 시뮬레이터, 네트워크 모델을 가진 노드 시스템 시뮬레이터, 네트워크 모델을 가진 노드 에뮬레이터로 분류되는 것

을 특징으로 하는, 방법.

발명의 설명

기 술 분 야

[0001] 아래의 설명은 유비쿼터스 센서 네트워크(ubiquitous sensor network, USN)의 성능을 평가하는 기술에 관한 것이다.

배 경 기 술

[0002] 최근 인간의 노력은 만연하는 스마트 환경을 확장하는 데에 집중되고 있다. 이와 관련하여, 유비쿼터스 컴퓨팅 기술은 임의의 서비스나 임의의 장치를 임의의 네트워크 하에서 언제든지 어디에서나 모든 사용자들에게 사용하기 위하여 도입된다.

[0003] 유비쿼터스 환경은 일상 생활 곳곳에 존재하는 사물 간의 네트워킹을 통한 보이지 않는 서비스를 제공하는 것으로 목표로 하고 있으며, 이러한 서비스를 위해 일상에 존재하는 "사용자, 장소 혹은 대상물 등의 개체 상태를 나타내는 정보"들을 컨텍스트라고 정의하고, 이러한 컨텍스트의 상태를 파악하고 추론하여 효율적으로 응용프로그램에 제공하기 위한 방법들이 연구되고 있다.

[0004] 예컨대, 한국등록특허공보 제10-0994151호(등록일 2010년 11월 08일)에는 주변 무선네트워크와의 통신시에 얻어지는 RSSI 값을 이용하여 사용자의 다양한 위치 정보를 수집하고 센서 정보와 시간 정보를 수집된 위치 정보와 결합시켜 사용자의 선호도 및 생활패턴을 추론할 수 있는 유비쿼터스 컴퓨팅 환경에서 사용자의 생활패턴에 최적화된 서비스 제공 시스템 및 방법이 개시되어 있다.

발명의 내용

해결하려는 과제

[0005] 유비쿼터스 센서 네트워크에 대한 시뮬레이터와 에뮬레이터를 비교 조사하여 그 성능을 평가할 수 있는 기술을

제공한다.

[0006] 유비쿼터스 센서 네트워크의 동작 환경을 고려하여 적절한 시뮬레이터와 에뮬레이터를 선택할 수 있는 기술을 제공한다.

과제의 해결 수단

[0007] 유비쿼터스 센서 네트워크(ubiquitous sensor network)의 성능을 평가하기 위한 장치에 있어서, 컴퓨터에서 판독 가능한 명령을 실행하도록 구현되는 적어도 하나의 프로세서를 포함하고, 상기 적어도 하나의 프로세서는, 상기 유비쿼터스 센서 네트워크의 도메인에서 무선 네트워크 프로토콜과 통신 프레임워크의 성능을 평가하는 시뮬레이션(simulation)을 수행하는 시뮬레이터; 및 상기 유비쿼터스 센서 네트워크 내에서 노드와 관련된 성능을 평가하는 에뮬레이션(emulation)을 수행하는 에뮬레이터를 포함하는, 장치를 제공한다.

[0008] 일 측면에 따르면, 상기 시뮬레이터는, 네트워크 시뮬레이션을 위한 네트워크 시뮬레이터; 코드 시뮬레이션을 위한 코드 레벨 시뮬레이터; 펌웨어 시뮬레이션을 위한 펌웨어 레벨 시뮬레이터; 알고리즘 시뮬레이션을 위한 알고리즘 레벨 시뮬레이터; 네트워크 패킷 시뮬레이션을 위한 패킷 레벨 시뮬레이터; 및 명령(instruction) 시뮬레이션을 위한 명령 레벨 시뮬레이터를 포함할 수 있다.

[0009] 다른 측면에 따르면, 상기 시뮬레이터는, 무선 네트워크 동작(operation) 및 프로토콜 스택을 중심으로 지향 네트워크(oriented network)의 시뮬레이션을 수행하고, 통신 모델을 포함하는 노드의 기능을 중심으로 지향 노드(oriented node)의 시뮬레이션을 수행할 수 있다.

[0010] 또 다른 측면에 따르면, 상기 시뮬레이터는, 상기 유비쿼터스 센서 네트워크 내에서 하드웨어와 소프트웨어로 구성된 노드; 노드의 기능을 나타내는 이벤트; 노드 간에 신호를 전송하는 중간 모듈(medium module); 모델링 환경을 전달하는 환경 모듈(environment module); 및 물리적인 프로토콜, MAC 프로토콜, 라우팅 프로토콜, 어플리케이션 계층으로 이루어진 네트워크 스택을 포함하는 다중 모듈로 구성될 수 있다.

[0011] 또 다른 측면에 따르면, 상기 시뮬레이터는, 시스템 내 현상에 대한 시간 순서 히스토리로 정의된 자취(trace)를 입력으로 사용하는 트레이스-구동(trace-driven) 시뮬레이션, 또는 시스템의 이산 상태 모델(discrete-state model)을 사용하는 이산-이벤트(discrete-event) 시뮬레이션을 수행할 수 있다.

[0012] 또 다른 측면에 따르면, 상기 에뮬레이터는 네트워크 시뮬레이터에 포함되고, 상기 유비쿼터스 센서 네트워크에 대한 평가 도구가 노드 모델을 가진 네트워크 시뮬레이터, 노드 에뮬레이터를 가진 네트워크 시뮬레이터, 네트워크 모델을 가진 노드 시스템 시뮬레이터, 네트워크 모델을 가진 노드 에뮬레이터로 분류될 수 있다.

[0013] 또 다른 측면에 따르면, 네트워크 모델을 가진 노드 에뮬레이터는 특정 마이크로컨트롤러를 위한 명령 세트 시뮬레이터와 운영체제(OS)의 어플리케이션 코드 실행을 에뮬레이트하기 위한 에뮬레이터로 분류될 수 있다.

[0014] 또 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 상기 유비쿼터스 센서 네트워크에 대한 시뮬레이션 및 에뮬레이션의 결과를 그래프, 텍스트 파일, 및 추적 파일의 애니메이션 중 적어도 하나로 출력할 수 있다.

[0015] 유비쿼터스 센서 네트워크의 성능을 평가하기 위한 방법에 있어서, 상기 유비쿼터스 센서 네트워크의 도메인에서 무선 네트워크 프로토콜과 통신 프레임워크의 성능을 평가하는 시뮬레이션을 수행하는 단계; 및 상기 유비쿼터스 센서 네트워크 내에서 노드와 관련된 성능을 평가하는 에뮬레이션을 수행하는 단계를 포함하고, 상기 시뮬레이션을 위한 시뮬레이터는, 네트워크 시뮬레이션을 위한 네트워크 시뮬레이터; 코드 시뮬레이션을 위한 코드 레벨 시뮬레이터; 펌웨어 시뮬레이션을 위한 펌웨어 레벨 시뮬레이터; 알고리즘 시뮬레이션을 위한 알고리즘 레벨 시뮬레이터; 네트워크 패킷 시뮬레이션을 위한 패킷 레벨 시뮬레이터; 및 명령 시뮬레이션을 위한 명령 레벨 시뮬레이터를 포함하는 것을 특징으로 하는, 방법을 제공한다.

발명의 효과

[0016] 본 발명의 실시예들에 따르면, 유비쿼터스 센서 네트워크에 대한 시뮬레이터와 에뮬레이터를 비교 조사하여 그 성능을 평가할 수 있다.

[0017] 본 발명의 실시예들에 따르면, 유비쿼터스 센서 네트워크의 동작 환경을 고려하여 적절한 시뮬레이터와 에뮬레이터를 선택할 수 있다.

도면의 간단한 설명

[0018] 도 1은 본 발명의 일 실시예에 있어서 정보 기술(information Technology, IT) 공간을 설명하기 위한 예시 도면이다.

도 2는 본 발명의 일 실시예에 있어서 평가 기술들이 사용되는 가상 공간(CS) 및 실제 공간(RS)의 비율 관점에서 각자의 기여분을 도시한 것이다.

도 3은 본 발명의 일 실시예에 있어서 평가 기술들이 사용되는 에이전트 기반 모델 및 유비쿼터스 컴퓨팅의 단계를 도시한 것이다.

도 4는 본 발명의 일 실시예에 있어서 시뮬레이션 과정의 일례를 개략적으로 도시한 것이다.

도 5는 본 발명의 일 실시예에 있어서 센서 네트워크 운영체제(OS)의 목록을 도시한 것이다.

도 6은 본 발명의 일 실시예에 있어서 유비쿼터스 센서 네트워크의 성능을 평가하기 위한 장치의 내부 구성의 일례를 설명하기 위한 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0019] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0020] 정보 기술(information technology, IT)은 우리의 삶에 침투하여 우리 일상 생활에 매우 관여되고 긴밀해지고 있다. 사용자 인터페이스로서의 컴퓨터는 개인들이 가상 공간과 연결되는 것을 가능하게 하며, 사람 대 사람(person-to-person) 및 사람 대 기계(person-to-machine)의 상호작용을 이용하는 것을 가능하게 한다.

[0021] 도 1은 IT 공간을 나타낸 것으로, (a)는 가상 공간(cyber space, CS) 및 실제 공간(real space, RS)의 융합을, (b)는 가상 공간과 실제 공간의 융합에 의해 생성된 유비쿼터스 공간을 도시한 것이다. IT의 빠른 진보 및 발전에 의하여, 가상 공간은 우리의 실제적 공간의 일부가 되고 있기 때문에(예를 들어, 증강 현실 적용), 가상 공간은 실제적(물리적) 공간과 점점 더 닮아가고 있다. 가상 공간 및 실제적 공간의 융합 지점은 유비쿼터스 공간이라 할 수 있다. 이러한 IT의 스마트 공간에서 컴퓨터들은 단편화되고(fragmented) 환경에 배치되며, 계산은 유비쿼터스 컴퓨팅을 통해 모든 장소 및 임의의 장소에서 이용 가능하게 된다.

[0022] 유비쿼터스 컴퓨팅 기술은 임의의 서비스나 임의의 장치를 임의의 네트워크 하에서 언제든지 어디에서나 모든 사용자들에게 사용하기 위하여 도입된다. 그러나, 고비용, 장시간 소모, 광범위한 노력 및 일부 경우에서의 불가역성(irrevocability)들은 유비쿼터스 시스템을 개발하는 것에 대한 주된 당면 과제 및 어려움이다.

[0023] 하나의 해결책은 실제 세계에서 감지 장치 및 컴퓨팅 장치들을 개발하기 전에 연습 시행(practice)들을 초기에 시뮬레이트하고 분석하고 증명하는 것이다. 본 발명은 이용 가능한 시뮬레이터 및 에뮬레이터 환경과 유비쿼터스 센서 네트워크(이하, 'USN'라 칭함) 적용을 위한 프레임워크를 소개하고 비교하는 것에 목표를 둔다.

[0024] 먼저, USN과 관련된 성능 평가 기술들을 설명한다.

[0025] 분석적 모델(analytical model), 시뮬레이션(simulation), 에뮬레이션(emulation), 테스트 베드(test bed), 및 실제 배치(real deployment)를 포함하는 일부 기술들은 USN 내 프로토콜 및 알고리즘들의 성능 평가를 위해 활용되고 있다.

[0026] 분석적 모델이란 시스템의 성능을 나타내는 방정식들의 집합이다. 분석적 모델들이 모델링 과정을 단순화함에도, 그들은 센서 네트워크들의 내재적인 복잡도를 정확하게 표현할 수 없다.

[0027] 시뮬레이션은 네트워크 프로토콜 및 알고리즘을 설계하고 개발하는 데 가장 흔하고 효과적인 방법으로 인용되어 왔다. 시뮬레이션을 사용함으로써 실제 환경의 다양한 시나리오들이 모델링될 수 있다. 또한, 그들은 설계의 임의 단계에서의 프로토콜을 시험하고 디버깅하는 가능성을 제공한다.

[0028] 하이브리드 방식으로서의 에뮬레이션은 네트워크 모델링에 대한 시뮬레이션 가능성이 동반되는 하드웨어 및 소프트웨어 구성요소들의 조합이다. 에뮬레이터들은 실험 조건에서 시뮬레이션을 실행하기 위하여 하드웨어뿐만 아니라 펌웨어를 사용한다. 에뮬레이터들이 실제 환경에서 활용될 수 있기 때문에, 그들은 시뮬레이터들과 비교할 때 잠재적으로 정확하게 기능한다.

[0029] 물리적인 테스트 베드들은 프로토콜 및 알고리즘의 실제 구현을 위한 프레임워크이다. 테스트 베드들은 실험을 모니터링하고 시행하는 원격 설정(remote configuration)을 허용할 뿐만 아니라 모델, 프로토콜 및 알고리즘 평가를 지원한다. 그것들은 시뮬레이션과 실제 장치의 배치 간의 차이를 좁힌다.

[0030] 실제 배치는 실현 가능하며 실제적인 센서 배치 연습 시행을 가능하게 한다. 모든 기능들은 실세계에서 설정되며 부정확하거나 불확실한 가정은 만들어지지 않는다. 이는 프로토콜 및 알고리즘 유효성 증명의 궁극적인 단계이다.

[0031] USN 성능 평가 기술들은 순수 소프트웨어 기반 기술에서 오직 하드웨어 기반 기술까지에 이른다. 도 2는 평가 기술들이 사용하는 가상 공간 및 실제 공간의 비율 관점에서 각자의 기여분을 도시한 것이다. 이 관점에서, 분석적 모델 및 시뮬레이터는 오직 가상 공간에서만 기능하며, 어떠한 물리적인 배치도 실제 공간에 구현되지 않는다. 에뮬레이터와 테스트 베드의 경우는 또 다르다. 이 기술들은 그들의 스루풋을 가상 및 실제 공간에 배분한다. 전자에서, 가상 공간은 더 큰 부분을 갖는 반면, 후자에서 구현의 대다수가 실제 공간에 할당된다. 최신 평가 기술로서의 실제 배치는 완전히 실제 공간에만 집중한다. 이 기술에 대한 모든 구현 및 조작(manipulation)은 물리적인 배치를 고려한다.

[0032] 본 발명에서는 USN 성능 평가 기술들의 모든 긍정적 및 부정적 양상을 고려하면서 시뮬레이션 및 에뮬레이션 컨셉 및 환경을 살피는 것에 목표를 둔다. 이는 USN 시뮬레이터 및 에뮬레이터의 생산성, 능력, 성질에 접근하는 것을 가능하게 할 뿐만 아니라, 순수하게 가상 공간에서 수행되는 평가 기술들 및 가상 및 실제 공간의 혼합에서 수행되는 평가 기술들을 가능하게 한다.

[0033] 에이전트 기반 모델링과 유비쿼터스 컴퓨팅 간의 차이점은 논쟁을 초래할 수 있다. 한편, 에이전트 기반 모델들(agent-based models, ABM)은 실세계 객체들의 복잡한 상호 작용을 연구하기 위하여 다양하게 사용되어 왔다. 분석적 모델들 및 시뮬레이터들은 에이전트 기반 모델에 의해 사용되는 두드러진 성능 평가 기술들이다. 달리 말해, 에이전트 기반 모델은 가상 공간 전반 시뮬레이터들 내의 알고리즘으로 요약된다. 에이전트 기반 모델링으로부터 어느 정도의 확실성을 달성함으로써 실제적인 연습 시행들은 실제 공간에서 구현될 수 있다.

[0034] 그러나, 에이전트 기반 모델은 몇 가지 예를 들어, 지나친 단순화 및 높은 수준의 추상화와 같은 분석적 모델들 및 시뮬레이터들의 결점을 겪는다. 반면, 유비쿼터스 컴퓨팅은 모든 곳에서 발생하며, 경계로 한정되지 않는다. 에이전트 기반 모델과 달리, 유비쿼터스 컴퓨팅은 실제 공간에 널리 보급된 컴퓨팅 장치들의 실제적인 배치에 의해 달성된다. 그러나, 에이전트 기반 모델은 유비쿼터스 컴퓨팅의 전제 단계가 될 수 있다. 도 3은 성능 평가 기술들을 사용하여 에이전트 기반 모델 및 유비쿼터스 컴퓨팅의 단계를 도시한 것이다.

[0035] 에이전트 기반 모델과 시뮬레이터 간에는 교환 관계(tradeoff)가 있다. 에이전트 기반 모델에 의해 에이전트 수준(agent level)에서 일련의 규칙들이 정의되고 그들의 상호 작용은 명시적으로 모델링된다. 실험적인 도구로서의 시뮬레이터들은 보통 일반적인 의미를 전달하고 더 넓은 영역을 커버한다. 그들은 소프트웨어 내의 사전 정의된 규칙들에 더 또는 덜 의존적이다. 그러나, 에이전트 기반 모델에서의 규칙들은 전체적인 수준(global level)에서 전체 시스템의 행동을 결정하기 위하여 시뮬레이터에 내장될 수 있다. 이 과정은 다중 에이전트 시뮬레이션(multi-agent simulation)으로 알려진다. 그럼에도 불구하고, 시뮬레이터들은 에이전트 및 이에 대응하는 규칙들을 취급할 수 없다.

[0036] 시뮬레이션

[0037] 시간에 따른 실제 세계 상태들 및 진행들의 모방을 시뮬레이션이라고 한다. 시뮬레이션에 의해 시스템 동작은 특징 지어지고 분석되며, 가정 의문점들(what-if questions)이 발생되고, 실제 상태와 가까운 유사성을 갖는 시스템들이 설계될 수 있다. 시스템의 효율성, 생산성 및 실현 가능성을 다루는 상당한 정보는 실제적인 구현의 실제 배치 이전에 시뮬레이션에 의해 평가될 수 있다. 보통, 시뮬레이션을 수행하기 위하여, 모델이 개발되어야 한다. 그러한 모델은 바라는 시스템/과정의 취급, 특성 및 주된 성질을 설명한다. 시간에 따른 시스템 연산이 시뮬레이션에 의해 나타나는 동안, 그 모델을 시스템 자체를 나타낸다. 그러나, 시뮬레이션 결과로부터 신뢰할 수 있는 결론을 유도하는 것은 쉽지 않다. 다양한 단계들이 시뮬레이션 동안에 존재하며, 시뮬레이션의 목적 따라 달라질 수 있다. 이러한 단계들은 항상 연속적이진 않으며 불연속적인 방식으로 적용될 수도 있다. 그럼에도 불구하고, 모델의 성능을 평가하는 것은 시뮬레이션 기능의 철저한 평가 및 주기적인 수정을 요구한다. 도 4는 시뮬레이션 과정의 일례를 개략적으로 도시한 것이다.

[0038] - 시뮬레이션 유형

[0039] 3가지 유형의 시뮬레이션이 컴퓨터 과학 문헌에서 언급되어 왔다: Monte Carlo 시뮬레이션, Trace-driven 시뮬레이션 및 Discrete-event 시뮬레이션.

[0040] Monte Carlo 시뮬레이션은 정적(static) 시뮬레이션 또는 시간 축이 없는 시뮬레이션이다. 이는 특성이 시간에

따라 변하지 않는 확률적 사건들을 모델링하는 데 사용된다. 또한, Monte Carlo 시뮬레이션은 확률적 방식을 사용함으로써 비확률적인 수식들을 평가하는 데 활용된다.

[0041] Trace-driven 시뮬레이션은 시뮬레이션 과정에서 자취(trace)를 입력으로 사용한다. 자취는 실제 시스템 내 현상의 시간 순서 히스토리로 정의된다. 일반적으로, Trace-driven 시뮬레이션은 자원 관리 알고리즘들을 조율하거나 분석하는 데 사용된다.

[0042] Discrete-event 시뮬레이션은 continuous-event 시뮬레이션과는 반대로 시뮬레이션을 위한 시스템의 이산 상태 모델(discrete-state model)을 사용하고 다양한 장치들에서의 작업 수에 의해 설명되는 가변 시스템 상태에 의해 사용된다. Discrete-event 시뮬레이션에서 시간은 이산적이거나 연속적일 수 있다.

[0043] Trace-driven 시뮬레이션과 Discrete-event 시뮬레이션은 높은 성능 및 확장성(예를 들어, 정지 및 이동 센서의 가능한 개수)으로 인해 USN에서 널리 사용된다.

[0044] - 시뮬레이션 실행

[0045] 시뮬레이터들은 동기(synchronous) 또는 비동기(asynchronous) 모드 중 하나를 통해 실행된다.

[0046] 동기 시뮬레이션은 가장 간단한 시뮬레이션 방식이며 순환 기반(round-based) 기술이다: 첫째, 글로벌 시간은 프레임워크를 통해 1 단위 증가한다. 둘째, 노드들은 그들의 이동성 모델들에 따라 이동하며, 연결들은 연결성 모델에 따라 업데이트된다. 마지막으로, 이 과정은 노드들에 관하여 반복한다. 동기 시뮬레이션은 구현의 용이함, 성능 예측성 및 낮은 오버헤드를 포함하는 긍정적인 측면을 갖는다. 그러나, 이는 순환 간의 동기화 단계에 의한 통신 비용 및 약한 부하 균형(weak load balancing)을 겪는 경향이 있다. 간단하게, 동기 시뮬레이션은 짧은 계산 정도(granularities) 및 큰 순환 병렬성(parallelism)을 갖는 시뮬레이션에 적합하다.

[0047] 반면, 비동기 시뮬레이션은 대부분 이벤트를 기반으로 한다. 다수의 메시지 및 타이머 이벤트는 순서에 따라 발생해야 하는 시간 간격에 따라 정렬된다. 이벤트들은 프레임워크를 통해 반복적으로 선택되고 실행된다. 보수적인(conservative) 시뮬레이션과 낙관적인(optimistic) 시뮬레이션은 비동기 시뮬레이션의 두 가지 유형이다. 이 두 가지 시뮬레이션 모드를 비교하면, 동기 시뮬레이션은 비기능적인 노드들을 포함하는 모든 노드들에 닿기 때문에, 동기 시뮬레이션은 대게 비동기 시뮬레이션보다 느리게 실행된다. 이 조건은 비동기 시뮬레이션에는 적용되지 않는다. 이 모드에서는, 오직 메시지들과 타이머 이벤트들만이 처리되고 불필요한 순환들은 구현되지 않는다. 노드들 이동성의 연속성은 이벤트처럼 특징 지어질 수 없기 때문에 비동기 시뮬레이션 모드는 노드 이동을 지원하지 않는다.

[0048] - USN 시뮬레이션

[0049] USN 도메인에서 시뮬레이션은 무선 네트워크 프로토콜과 통신 프레임워크의 진보를 위하여, 그리고 다양한 시나리오 중 이용 가능한 것에 대한 평가를 위하여 가장 널리 쓰이는 평가 절차 중 하나이다. USN 목적을 위해 설계된 시뮬레이터들은 개발 제한들(예를 들어, 노드 및 통신)을 고려하기 위하여 흔하게 설계된다. 제한들의 성질에 기반하여, 시뮬레이션 도구들은 (1) 지향 네트워크(oriented network) 및 (2) 지향 노드(oriented node) 클래스들로 분류된다. 지향 네트워크들은 무선 네트워크 동작 및 동작(operation)의 프로토콜 스택에 초점을 둔다. 이러한 시뮬레이터들은 처음에 네트워크 시뮬레이션을 위해 설계되고 이후 USN 목적을 위해 확장된다. 이 클래스의 시뮬레이터에 대한 예시들은 OMNet++, NS-2 및 J-Sim이다. 두 번째 클래스의 시뮬레이터로서의 지향 노드는 단순 통신 모델을 포함하는 단일 노드의 기능에 대해 집중한다. 이 시뮬레이터들은 목적 노드들 및 그들의 OS에 특징적으로 집중한다. 게다가, 이 시뮬레이션 도구들은 어플리케이션을 갖는 노드의 호환성(compatibility)을 결정할 수 있다. 이 클래스의 시뮬레이터에 대한 예시들은 TOSSIM, ATEMU 및 SENS이다. 이 두 가지 클래스에 의해 두 가지의 일반적인 측면이 고려된다: (1) 시뮬레이션 모델들의 정확성(correctness) 및 (2) 모델을 구현하는 특정 도구의 적합성(suitability). 일반적으로, USN 시뮬레이터는 하기와 같은 내용을 포함하는 다중 모듈들을 포함한다:

[0050] 1) 노드는 USN 내에서 하드웨어와 소프트웨어로 구성된 장치이다. 노드 구성요소들은 액추에이터(actuator), 센서(sensor), 프로세서(processor), 송수신기(transceiver), 네트워크 프로토콜(network protocol), 에너지원(energy source) 및 어플리케이션(application)이다.

[0051] 2) 이벤트는 이벤트가 발생하는 시간을 포함하여 중요한 기능들을 나타낸다.

[0052] 3) 중간 모듈(medium module)은 노드들이 신호를 전송할 수 있게 하고 유용한(affective) 신호들과 관련하여 노

드들에게 알린다.

[0053] 4) 환경 모듈(environment module)은 모델링되어야 하는 습도, 소리, 온도 및 빛과 같은 물리적인 현상의 전파(propagation)를 가능하게 한다.

[0054] 5) 송수신기 하드웨어는 노드 전력 소모뿐만 아니라 각 센서 노드의 상태(예를 들어, 슬립(sleep), 대기(standby), 수신(receive) 및 전송(transmit))를 결정한다.

[0055] 6) 물리적인 프로토콜은 네트워크 스택의 가장 낮은 계층으로 알려져 있다. 이는 송수신기 상태 변경과 패킷 전송 및 수신과 같은 서비스들을 가능하게 한다.

[0056] 7) MAC 프로토콜은 물리적인 프로토콜 위에 존재한다. 이는 보통 노드 프로세서 소프트웨어 상에 설치된다. MAC 프로토콜은 MAC 계층 상태 변경 및 프로토콜 파라미터 정의와 같은 서비스들을 가능하게 한다.

[0057] 8) 라우팅 프로토콜은 MAC 프로토콜 상에 위치한다. 이는 메시지가 네트워크 홉 간에 라우트되는 것을 가능하게 한다.

[0058] 9) 어플리케이션 계층은 네트워크 스택의 최상단에 존재한다. 이것은 하부 계층들, 센서들 및 액추에이터들과의 연결을 통하여 USN 어플리케이션을 구현한다.

[0059] - USN 시뮬레이터 카테고리

[0060] 시뮬레이션은 오직 가장 중요한 특징들이 모델링되는 일반 시뮬레이션부터 특정 측면들이 나타난 매우 세부적인 시뮬레이션까지 다양한 추상화 수준(abstraction levels)에서 적용될 수 있다. 시뮬레이터들을 추상화 수준에 기반하여 카테고리화 한다.

[0061] 1) 일반 네트워크 시뮬레이터들은 노드 시뮬레이션보다 네트워크 시뮬레이션에 집중한다. 실제 센서 언어와는 전혀 다른 높은 수준의 언어들이 시뮬레이션 어플리케이션을 쓰기 위해 사용된다. 또한, 같은 프로그래밍 언어는 어플리케이션 및 프로토콜 코드를 위해 사용된다. 대부분의 네트워크 시뮬레이터들은 네트워크 스택의 시뮬레이션, MAC 프로토콜 및 무선 매체(radio medium)를 제공한다. 일반 네트워크 시뮬레이터들은 새로운 통신 프로토콜들을 평가하는 데 효과적이다. 그러나, 그들은 상호 운용성(interoperability) 평가 또는 소프트웨어 버그 조사에 있어서 덜 기능적이다(operational).

[0062] 2) 코드 레벨 시뮬레이터들은 실제 센서 네트워크 노드들에서 활용되는 것과 유사한 노드들을 사용한다. 따라서, 그들은 특정 OS에 대해 제공되는 네트워크 스택 실행을 가능하게 한다. 코드 레벨 시뮬레이터들은 무선 매체의 시뮬레이션을 가능하게 할 뿐만 아니라 센서 시뮬레이션도 제공한다. 그것들은 소프트웨어 버그(예를 들어, 전개형 코드(deployable code) 또는 논리적인 에러) 감지에 효과적이지만, 그것들은 하드웨어 버그(예를 들어, CPU 구조 또는 타이밍)에는 적절하지 않다.

[0063] 3) 펌웨어 레벨 시뮬레이터들은 실제 센서 네트워크에서 실행되는 펌웨어와 센서 노드 에뮬레이션 모두를 고려한다. 펌웨어 레벨 시뮬레이터들은 세부적인 시뮬레이션을 가능하게 하며 정확한 구현 결과들을 생성한다. 게다가, 그것들은 마이크로프로세서 및 무선 칩 에뮬레이션에 더하여 무선 매체 시뮬레이션을 용이하게 한다. 세부적인 시뮬레이션 때문에, 펌웨어 레벨 시뮬레이터 실행 시간은 일반 네트워크 또는 코드 레벨 시뮬레이터들의 시간보다 더 크다. 복잡도 수준에 기반하여 세 가지의 주된 카테고리로 시뮬레이터들을 분류한다.

[0064] 4) 알고리즘 레벨 시뮬레이터들은 논리, 데이터 구조 및 알고리즘의 표현식을 고려한다. 알고리즘 레벨 시뮬레이터들은 세부적인 통신 모델링보다 노드 연결들을 표현하기 위하여 그래프 데이터 구조에 집중한다. 그들은 단순한 MAC 계층 프로토콜을 갖진 않지만 거대한 네트워크 시뮬레이션을 가능하게 한다.

[0065] 5) 패킷 레벨 시뮬레이터들은 물리적인 계층 및 데이터 링크를 네트워크 스택으로 실행한다. 그러므로, 그들은 구현되어야 하는 무선 모델 및 MAC 프로토콜을 제공하고, 이것들은 전파(propagation), 충돌(collision), 페이딩(fading), 노이즈 및 파 회절(wave diffraction)에 관해 실현 가능한 것들이다.

[0066] 6) 에뮬레이터라고도 불리는 명령 레벨 시뮬레이터들은 명령 수준에서 CPU 실행 모델링을 제공한다.

[0067] - USN 시뮬레이션을 위한 조건

[0068] 탈중앙화된 통신에 멀티태스킹, 이질성(heterogeneity), 다수의 센서 노드 및 제한된 자원과 같은 USN의 다양한 특징들을 고려하면, 시뮬레이터의 설계 및 개발은 도전적인 문제이다. 본 실시예에서는 USN 시뮬레이션 도구를 위한 6개의 중요한 인자들이 고려되어야 한다.

[0069] 1) 충실도(fidelity)는 시스템 동작의 예측뿐만 아니라 시뮬레이션의 충실함(faithfulness)에 초점을 맞춘다. 이 점에서, 무선 채널을 위하여, 물리적 환경, 노드 시스템 및 정확한 모델이 개발되어야 할 필요가 있다.

[0070] 2) 확장성(scalability)은 USN 시뮬레이터에 의한 센서 노드의 밀도 및 지원된 수를 나타낸다. USN 어플리케이션들이 많은 센서 노드들의 배치를 요구하기 때문에, 시뮬레이터의 더 높은 확장성은 장점이 된다.

[0071] 3) 에너지 인식(energy aware)은 USN 시뮬레이터의 중요한 특징이다. 센서 노드들은 에너지 자원(예를 들어, 배터리 또는 솔라 셀(solar cells))을 제한하기 때문에, 전력 소모와 타이밍 정보는 센서 노드의 실제 배치 이전에 시뮬레이터들을 통하여 정확하게 모델링될 필요가 있다.

[0072] 4) 확장성(extensibility)은 사용자들이 이용 가능한 모듈들을 변형할 수 있도록 하거나 새로운 모듈들을 시뮬레이터에 넣도록 할 수 있다. 높은 모듈성(modularity)을 갖는 사용자 친화적인 인터페이스는 사용자들이 기능들을 더하거나 변경할 수 있도록 돕는다.

[0073] 5) 이질성 지원(heterogeneity support)은 USN 시뮬레이션 도구들 내 다양한 요소들의 통합을 가능하게 한다. 이것은 다양한 노드들 및 그들의 상호 연결에 대한 모델링을 포함한다.

[0074] 6) 그래픽 사용자 인터페이스(Graphical User Interface, GUL)는 네트워크 토폴로지 및 모듈들 구성의 구현을 가능하게 한다. 그것은 또한 디버깅(debugging), 트레이싱(tracing) 및 시뮬레이션 결과들의 시각화를 빠르게 할 수 있다.

[0075] - 시뮬레이션 기준 평가

[0076] 시뮬레이터를 평가하기 위한 다양한 기준이 있다. 재사용 가능성(reusability) 및 이용 가능성(availability), 성능 및 확장성, 실험 및 처리 결과를 정의하는 부유-시맨틱 스크립팅 언어(rich-semantics scripting language) 지원, 그래픽, 디버깅 및 추적(trace) 지원은 우수한 시뮬레이터 내에 존재해야 한다. 또한, 입력, 처리, 출력, 지원 및 비용 그룹들로 카테고리화 되는 시뮬레이터에 대한 다양한 핵심 특징들이 있다. 각 카테고리는 확장된 일부 기준들로 구성될 수 있으며, 설계 목표, 구조 및 어플리케이션 추상화 수준에 기반하여 이러한 특징들의 조합은 시뮬레이터 내에 존재할 수 있다.

[0077] 에뮬레이션

[0078] 시뮬레이션을 수행하기 위하여 소프트웨어와 하드웨어로 구성된 도구들은 일반적으로 에뮬레이터라고 한다. 에뮬레이터에서 시뮬레이트되는 요소들(예를 들어, 링크 및 트래픽) 쪽의 실제 하드웨어(예를 들어, 모트(motes))는 USN 어플리케이션을 위한 현실적인 성능을 제공하는 데 목표를 둔다. 에뮬레이터는 보통 일부 센서 노드들을 동시에 에뮬레이트하기 위하여 높은 확장성을 갖는다. 시뮬레이터와 비교할 때, 에뮬레이터는 실제 센서 노드 내에 구현되고 실제 어플리케이션 코드를 구동하며, 이는 그들의 성능 정밀도를 향상시킨다. 에뮬레이터들은 미세 조정 네트워크 수준 및 센서 알고리즘뿐만 아니라 센서 노드 간 타이밍 상호 작용에도 적합하다.

[0079] 에뮬레이터들은 물리적 계층 및 MAC 계층 클래스로 카테고리화 된다.

[0080] 물리적 계층 에뮬레이터에서 실제 시스템은 물리적인 계층(physical layer)을 제외한 모든 네트워크 계층들로 구성된다. 이러한 에뮬레이터들은 전파가 현실에서 마주할 수 있는 효과들을 경험하기 위하여 노드 무선 인터페이스를 통해 송출된 전파 신호들을 추출한다(rip). 반대로, 역물리적 계층 에뮬레이터들은 반대로 행동한다. 예를 들어, 네트워크 그룹의 오버헤드 부분들이 시뮬레이트되고 실제 하드웨어를 사용하는 패킷들이 전송된다.

[0081] MAC 계층 에뮬레이터에서 실제 시스템은 물리적인 계층 및 MAC 계층을 제외한 모든 네트워크 계층으로 구성된다.

[0082] - USN 시뮬레이션 및 에뮬레이션 도구에 대한 분류(taxonomy)

[0083] 노드 모델을 네트워크 시뮬레이터에 삽입하는 것은 USN 시뮬레이터의 평가 방식이다. 두 가지 유형의 노드 모델은 (1) 시뮬레이터 노드 모델 및 (2) 노드 에뮬레이터가 있다. 2)는 노드 마이크로 컨트롤러의 명령 레벨 시뮬레이터와 관련되고, 다양한 주변 장치 모델뿐만 아니라 송수신기 확장부 및 센서들로 구성된다. 노드 에뮬레이터는 네트워크 모델링과 네트워크 시뮬레이터 내 노드 모델 삽입을 가능하게 한다. 이 관점에서 USN 평가는 다음 4가지 클래스들로 카테고리화 된다.

[0084] 1) 노드 모델을 갖는 네트워크 시뮬레이터들은 이산-이벤트 타이밍, 무선 매체, 네트워크 모델링 및 더 많거나 적은 센서 노드 슬립 듀티 사이클(sleep duty cycles)에 초점을 둔다.

[0085] 2) 노드 에뮬레이터들을 갖는 네트워크 시뮬레이터들은 네트워크 시뮬레이터 및 노드 에뮬레이터 모두의 장점으로부터 이익을 얻는다. 세부적인 네트워크 모델은 네트워크 시뮬레이터를 통해 달성될 수 있다. 또한, 도구들의 정확한 타이밍 정보는 노드 에뮬레이터에 의해 얻어질 수 있다.

[0086] 3) 네트워크 모델을 갖는 노드 시스템 시뮬레이터는 SystemC와 같은 하드웨어 기술 언어들(description languages)을 통해 시스템 레벨에서 빈번하게 작동한다. 이러한 언어들은 시스템 레벨, 처리 레벨 및 레지스터 전송 레벨과 같은 다양한 세부 내용을 갖는 다양한 추상화 레벨에서 노드 하드웨어 모델링을 가능하게 한다.

[0087] 4) 네트워크 모델을 갖는 노드 에뮬레이터들은 어플리케이션 코드를 직접적으로 실행할 수 있다. 노드 에뮬레이터들은 특정 마이크로 컨트롤러를 위한 명령 세트 시뮬레이터 및 OS의 어플리케이션 코드 실행을 에뮬레이트하기 위해 설계된 에뮬레이터로 분류될 수 있다.

[0088] - 시뮬레이션 및 에뮬레이션 출력

[0089] 시뮬레이션 및 에뮬레이션 결과들은 그래프, 텍스트 파일 및 추적 파일의 애니메이션으로 제시될 수 있다. 그래프는 다수의 프로토콜 간 비교를 가능하게 한다. 그래프는 패킷 전달 양의 변동(variation), 네트워크 지연 및 스루풋(throughput), 네트워크 성능 평가를 위한 몇몇 다른 파라미터들을 설명할 수 있다. 출력 텍스트 파일은 다른 시뮬레이터 또는 프로그램에 대한 입력이 될 수 있다. 궁극적으로, 시뮬레이션 과정에서 발생하는 모든 이벤트는 추적 파일을 통해 기록될 수 있다.

[0090] 다음, USN 운영 체제(OSs)를 검토하고 비교한다.

[0091] 전술한 바와 같이, USN은 무선 네트워크를 통해 통신하는 몇몇 작은 센서 노드들로 구성된다. 물리 센서, 마이크로 프로세서/마이크로 컨트롤러, 메모리, 무선 송수신기 및 배터리와 같은 센서 노드 하드웨어의 구성요소들은 질서 있고 제어된 방법으로 동작될 필요가 있다. 이 과정은 OS를 통해 수행된다. 그러므로, 각 센서 노드는 하드웨어 제어를 위한 OS를 필요로 하고, 이는 어플리케이션 소프트웨어에 하드웨어 추상화를 제공하며, 어플리케이션과 기본 하드웨어(underlying hardware) 간의 차이를 감소시킨다. 달리 말해, OS는 어떠한 충돌 없이 자원을 정확하고 효과적으로 할당하는 것에 대한 자원 관리자로서 행동한다. USN 목적을 위해, OS는 일정한 방식으로 이종 감지 하드웨어를 추상화하고 제한된 자원으로 동작하는 능력뿐만 아니라 기본 기능, 효율적인 전력 관리 메커니즘, 필드 리프로그래밍 메커니즘(field reprogramming mechanisms) 및 설정 가능한 통신 스택을 제공해야만 한다.

[0092] 예를 들어, 구조, 실행 모델, 리프로그래밍, 스케줄링 및 전력 관리와 같은 중요한 특징들에 기반하여 USN OS에 대한 분류 프레임워크를 적용할 수 있다. 게다가, 다양한 클래스의 USN 어플리케이션을 위한 적절한 OS들이 제안되기도 하며, 5가지 USN OS의 성능 분석 및 구조가 검토된 바 있다: TinyOS, Contiki, Mantis OS, SOS 및 Microsoft.NET Micro. 이와 같이 다양한 OS들이 USN 어플리케이션 발전을 가능하게 하기 위하여 도입되고 있으며, 검증된 것들의 목록은 도 5에 제시되어 있다. 언급한 OS 외에도, 몇몇 연구들은 예를 들어, OS 신뢰성(reliability) 향상(예를 들어, t-kernel, Harbor 및 Neutron), 실시간 지원을 가능하게 하는 것(예를 들어, FIT), 프로그래밍 모델을 확장하는 것(예를 들어, protothreads 및 TOSThreads) 및 리프로그래밍 지원을 가능하게 하는 것(예를 들어, Deluge, FlexCup, Stream 및 Elon)과 같이 다양한 차원에서의 OS 능력을 향상시키려는 연구가 진행되고 있다.

[0093] 본 발명에서는 세 가지 핵심적인 측면에서 기존 검토들과 다르다. 첫째, 지금까지 생성된 그들의 파생물 및 확장물과 함께 (발견된) USN 시뮬레이션 및 에뮬레이션 환경/프레임워크 모두가 포함되도록 조사 범위를 확장시킨다. 특히, 본 발명은 시뮬레이션 및 에뮬레이션 환경의 정량적인 측면에 초점을 맞출 뿐만 아니라 널리 사용되고, 규칙적으로 업데이트되며, 다양한 기준을 기반으로 그들의 개발자들에게 잘 지원되는 것들을 정성적으로 평가한다. 또한, 일반적인 목적 기반 및 특정 목적 기반으로 이러한 도구들의 카테고리화를 제안한다. 둘째로, 본 발명에서는 시뮬레이터 및 에뮬레이터에 대한 최신 평가 기준의 일반적인 모습을 제공한다. 이는 결과적으로 어느 중요한 도구가 어떤 목적(예를 들어, 학술, 연구 또는 상업)에 적합한지 결정한다. 셋째로, 본 발명에서는 유비쿼터스 시뮬레이션 및 에뮬레이션 시나리오에서 시뮬레이터 및 에뮬레이터를 위한 많은 성능 평가 파라미터들을 제안한다.

[0094] 도 6은 본 발명의 일 실시예에 있어서 USN 성능을 평가하기 위한 장치의 내부 구성의 일례를 설명하기 위한 블록도이다.

[0095] 도 6은 본 발명의 일 실시예에 있어서 컴퓨터 시스템의 내부 구성의 일례를 설명하기 위한 블록도이다. 예를 들

어, 본 발명의 실시예들에 따른 USN 성능을 평가하기 위한 장치가 도 6의 컴퓨터 시스템(100)을 통해 구현될 수 있다. 도 6에 도시한 바와 같이, 컴퓨터 시스템(100)은 USN 성능을 평가하기 위한 방법을 실행하기 위한 구성요소로서 프로세서(110), 메모리(120), 영구 저장 장치(130), 버스(140), 입출력 인터페이스(150) 및 네트워크 인터페이스(160)를 포함할 수 있다.

[0096] 프로세서(110)는 명령어들의 시퀀스를 처리할 수 있는 임의의 장치를 포함하거나 그의 일부일 수 있다. 프로세서(110)는 예를 들어 컴퓨터 프로세서, 이동 장치 또는 다른 전자 장치 내의 프로세서 및/또는 디지털 프로세서를 포함할 수 있다. 프로세서(110)는 예를 들어, 서버 컴퓨팅 디바이스, 서버 컴퓨터, 일련의 서버 컴퓨터들, 서버 팜, 클라우드 컴퓨터, 콘텐츠 플랫폼, 이동 컴퓨팅 장치, 스마트폰, 태블릿, 셋톱 박스, 미디어 플레이어 등에 포함될 수 있다. 프로세서(110)는 버스(140)를 통해 메모리(120)에 접속될 수 있다.

[0097] 메모리(120)는 컴퓨터 시스템(100)에 의해 사용되거나 그에 의해 출력되는 정보를 저장하기 위한 휘발성 메모리, 영구, 가상 또는 기타 메모리를 포함할 수 있다. 메모리(120)는 예를 들어 랜덤 액세스 메모리(RAM: random access memory) 및/또는 동적 RAM(DRAM: dynamic RAM)을 포함할 수 있다. 메모리(120)는 컴퓨터 시스템(100)의 상태 정보와 같은 임의의 정보를 저장하는 데 사용될 수 있다. 메모리(120)는 예를 들어 USN 성능 평가를 제어하기 위한 명령어들을 포함하는 컴퓨터 시스템(100)의 명령어들을 저장하는 데에도 사용될 수 있다. 컴퓨터 시스템(100)은 필요에 따라 또는 적절한 경우에 하나 이상의 프로세서(110)를 포함할 수 있다.

[0098] 버스(140)는 컴퓨터 시스템(100)의 다양한 컴포넌트들 사이의 상호작용을 가능하게 하는 통신 기반 구조를 포함할 수 있다. 버스(140)는 컴퓨터 시스템(100)의 컴포넌트들 사이에, 예를 들어 프로세서(110)와 메모리(120) 사이에 데이터를 운반할 수 있다. 버스(140)는 컴퓨터 시스템(100)의 컴포넌트들 간의 무선 및/또는 유선 통신 매체를 포함할 수 있으며, 병렬, 직렬 또는 다른 토폴로지 배열들을 포함할 수 있다.

[0099] 영구 저장 장치(130)는 (예를 들어, 메모리(120)에 비해) 소정의 연장된 기간 동안 데이터를 저장하기 위해 컴퓨터 시스템(100)에 의해 사용되는 바와 같은 메모리 또는 다른 영구 저장 장치와 같은 컴포넌트들을 포함할 수 있다. 영구 저장 장치(130)는 컴퓨터 시스템(100) 내의 프로세서(110)에 의해 사용되는 바와 같은 비휘발성 메인 메모리를 포함할 수 있다. 영구 저장 장치(130)는 예를 들어 플래시 메모리, 하드 디스크, 광 디스크 또는 다른 컴퓨터 판독 가능 매체를 포함할 수 있다.

[0100] 입출력 인터페이스(150)는 키보드, 마우스, 음성 명령 입력, 디스플레이 또는 다른 입력 또는 출력 장치에 대한 인터페이스들을 포함할 수 있다. 구성 명령들 및/또는 USN 성능 평가와 관련된 입력이 입출력 인터페이스(150)를 통해 수신될 수 있다.

[0101] 네트워크 인터페이스(160)는 근거리 네트워크 또는 인터넷과 같은 네트워크들에 대한 하나 이상의 인터페이스를 포함할 수 있다. 네트워크 인터페이스(160)는 유선 또는 무선 접속들에 대한 인터페이스들을 포함할 수 있다. 구성 명령들은 네트워크 인터페이스(160)를 통해 수신될 수 있다. 그리고, USN 성능 평가와 관련된 정보들은 네트워크 인터페이스(160)를 통해 수신 또는 송신될 수 있다.

[0102] 이러한 도 6의 실시예는, 컴퓨터 시스템(100)의 일례일 뿐이고, 컴퓨터 시스템(100)은 도 6에 도시되지 않은 추가의 컴포넌트를 더 구비하거나, 2개 이상의 컴포넌트를 결합시키는 구성 또는 배치를 가질 수 있다. 컴퓨터 시스템(100)에 포함 가능한 컴포넌트들은 하나 이상의 신호 처리 또는 어플리케이션에 특화된 집적 회로를 포함하는 하드웨어, 소프트웨어, 또는 하드웨어 및 소프트웨어 양자의 조합으로 구현될 수 있다.

[0103] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 어플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0104] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 컴퓨터 저장 매체 또는 장치에 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

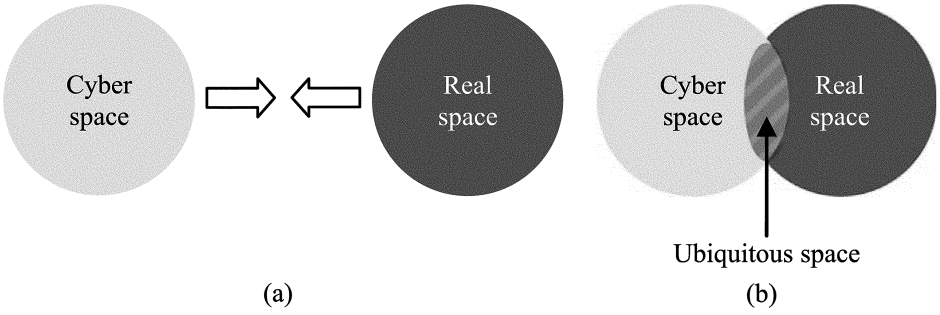
[0105] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 이때, 매체는 컴퓨터로 실행 가능한 프로그램을 계속 저장하거나, 실행 또는 다운로드를 위해 임시 저장하는 것일 수도 있다. 또한, 매체는 단일 또는 수 개의 하드웨어가 결합된 형태의 다양한 기록수단 또는 저장수단일 수 있는데, 어떤 컴퓨터 시스템에 직접 접속되는 매체에 한정되지 않고, 네트워크 상에 분산 존재하는 것일 수도 있다. 매체의 예시로는, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM 및 DVD와 같은 광기록 매체, 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical medium), 및 ROM, RAM, 플래시 메모리 등을 포함하여 프로그램 명령어가 저장되도록 구성된 것이 있을 수 있다. 또한, 다른 매체의 예시로, 어플리케이션을 유통하는 앱 스토어나 기타 다양한 소프트웨어를 공급 내지 유통하는 사이트, 서버 등에서 관리하는 기록매체 내지 저장매체도 들 수 있다.

[0106] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

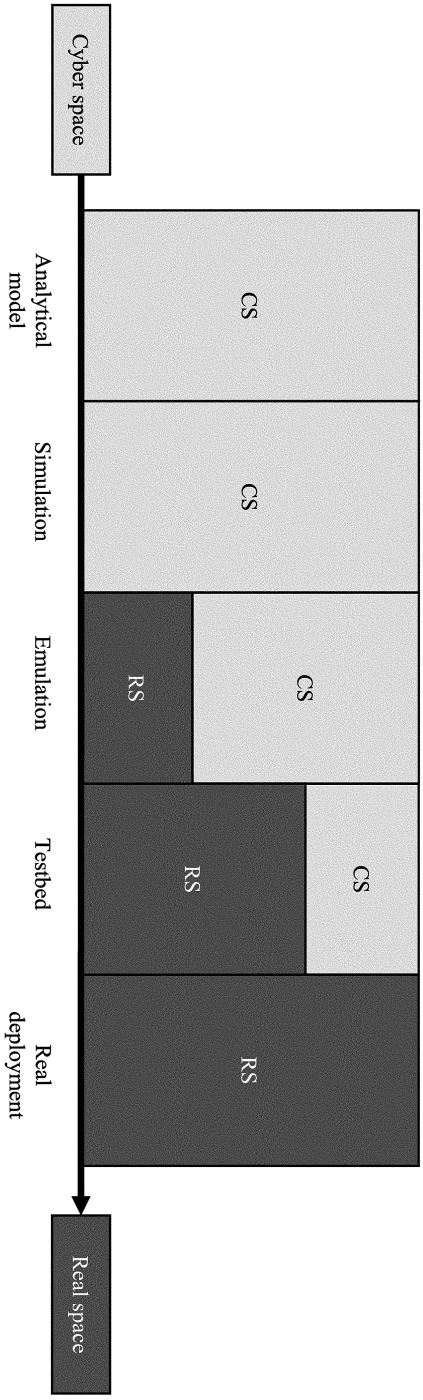
[0107] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

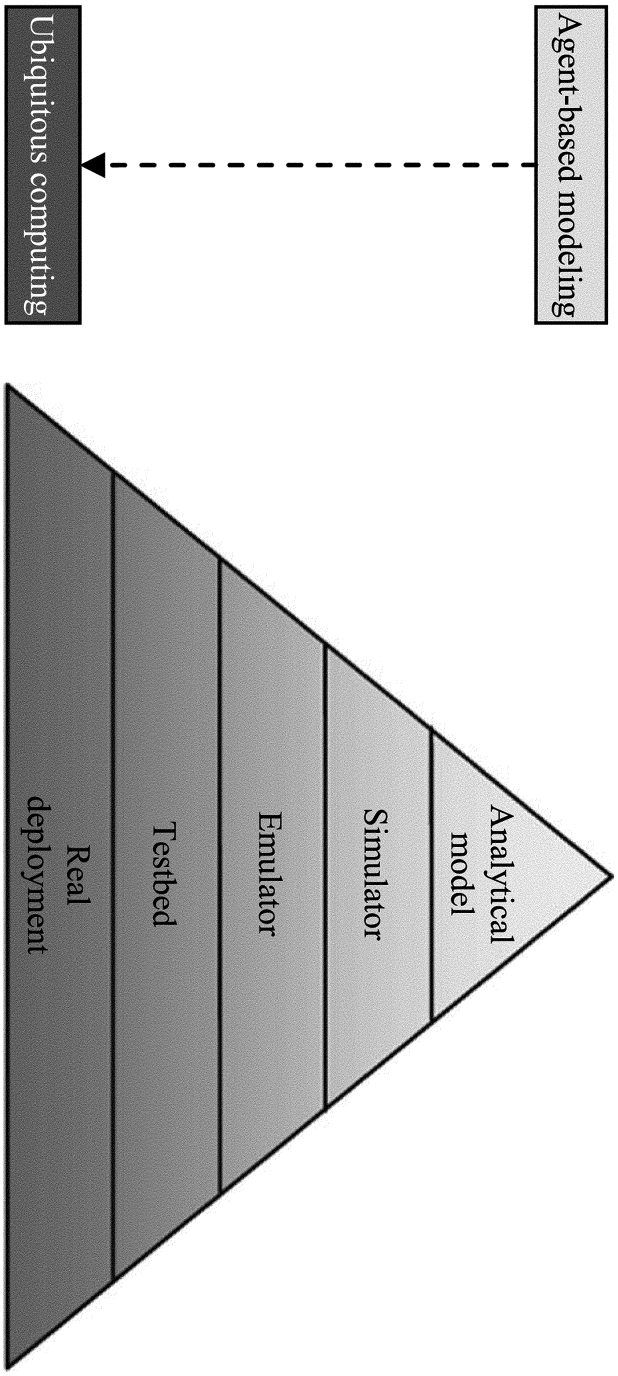
도면1



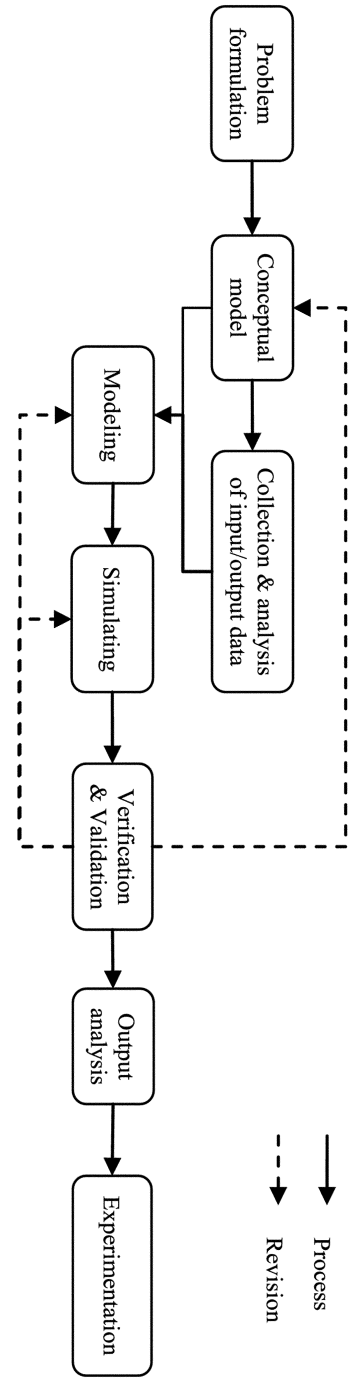
도면2


Cyber space
Analytical model
Simulation
Emulation
Testbed
Real deployment
Real space
CS
CS
CS
RS
CS
RS
RS

도면3


Agent-based modeling
Ubiquitous computing
Analytical model
Simulator
Emulator
Testbed
Real deployment

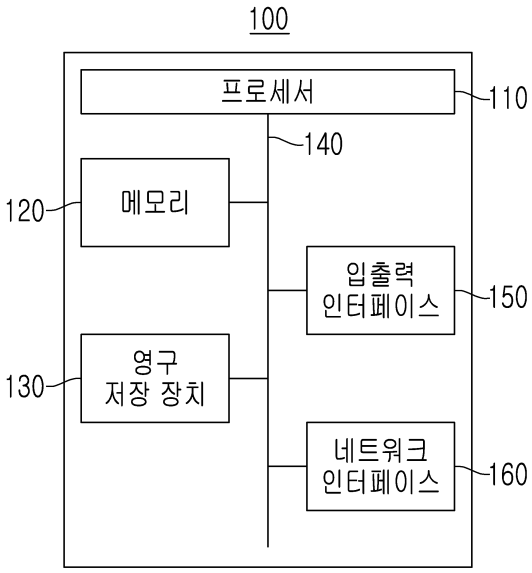
도면4


Problem formulation
Conceptual model
Collection & analysis of input/output data
Modeling
Simulating
Verification & Validation
Output analysis
Experimentation
Process
Revision

도면5

- AmbiCompVM	- Embedded Linux	- LiteOS	- NanoVM	- SenSpire OS
- AVRX	- EMERALDS	- µCOS	- OSPM	- SensorOS
- Bertha	- EYES	- MagnetOS	- OSSTAR	- SmartOS
- BTnutOS or NutOS	- FreeRTOS	- Mantis OS	- ParticleVM	- Squawk VM
- Contiki	- GenOS	- Maté	- PeerOS	- SOS
- CORMOS	- Jallad	- Microsoft .NET Micro	- PicOS	- T2
- CustomOS	- kOS	- MoteWorks	- Pixie OS	- TinyOS
- CVM	- KVM	- Nano-QPlus	- RETOS	- YATOS
- DCOS	- LORIEN	- NanoRK	- SenOS	- VMSTAR

도면6


100
프로세서
110
140
메모리
120
입출력
인터페이스
150
영구
저장 장치
130
네트워크
인터페이스
160