대규모 센서 네트워크를 위한 최적-동기식 병렬 시뮬레이션

(Optimal-synchronous Parallel Simulation for Large-scale Sensor Network)

김 방 현 † 김 종 현 ††

(Bang-Hyun Kim) (Jong-Hyun Kim)

요 약 대규모 무선 센서 네트워크의 설계 및 응용 개발을 위하여 소프트웨어 시뮬레이션이 널리 사용되고 있다. 그러한 시뮬레이션에서 네트워크의 동작과 실행시간 및 전력소모량을 가능한 한 정확히 예측하기 위해서는 시뮬레이션 정밀도가 높아야 한다. 그러나 정밀도가 높아질수록 시뮬레이션 시간은 길어지며, 센서노드의 수가 증가하면 그 시간이 더욱 길어진다. 본 연구에서는 대규모 무선 센서 네트워크 시뮬레이션에 걸리는 시간을 단축하기 위한 최적-동기식 병렬 이산-사건 시뮬레이션 방법을 제안한다. 이 방법에서는 네트워크로 연결된 여러 대의 컴퓨터들이 작업부하인 센서노드들을 분할하여 시뮬레이션 한다. 제안한 방법으로 구현한 시뮬레이터를 이용하여 실험한 결과에 따르면, 시뮬레이션 되는 센서노드의 수가 많은 경우에는 병렬 시뮬레이션에 참여하는 컴퓨터 수의 제곱에 접근하는 속도향상을 얻을 수 있다는 것을 확인하였다. 이 경우에 시뮬레이션 되는 센서노드의 수가 많아질수록 전체 시뮬레이션 시간에서 차지하는 병렬 시뮬레이션 오버헤드의 비율은 무시할 수 있을 정도로 작아지기 때문에, 컴퓨터의 수가 충분하다면 시뮬레이션 할 수 있는 센서노드의 수에는 한계가 없게 된다. 또한 LAN에 연결된 PC들을 그대로 사용하기 때문에, 병렬 시뮬레이션 환경을 저렴한 비용으로 쉽게 구축할 수 있다는 장점이 있다.

키워드 : 병렬 시뮬레이션, 센서 네트워크, 명령어-레벨 시뮬레이션

Abstract Software simulation has been widely used for the design and application development of a large-scale wireless sensor network. The degree of details of the simulation must be high to verify the behavior of the network and to estimate its execution time and power consumption of an application program as accurately as possible. But, as the degree of details becomes higher, the simulation time increases. Moreover, as the number of sensor nodes increases, the time tends to be extremely long. We propose an optimal-synchronous parallel discrete-event simulation method to shorten the time in a large-scale sensor network simulation. In this method, sensor nodes are partitioned into subsets, and each PC that is interconnected with others through a network is in charge of simulating one of the subsets. Results of experiments using the parallel simulator developed in this study show that, in the case of the large number of sensor nodes, the speedup tends to approach the square of the number of PCs participating in the simulation. In such a case, the ratio of the overhead due to parallel simulation to the total simulation time is so small that it can be ignored. Therefore, as long as PCs are available, the number of sensor nodes to be simulated is not limited. In addition, our parallel simulation environment can be constructed easily at the low cost because PCs interconnected through LAN are used without change.

Key words : parallel simulation, sensor network, instruction-level simulation

† 정 회 원 : 한국해양연구원 연수연구원
legnamai@chol.com
†† 종신회원 : 연세대학교 컴퓨터정보통신공학부 교수
jhkim34@yonsei.ac.kr
논문접수 : 2007년 4월 16일
심사완료 : 2008년 2월 13일

정보과학회논문지: 시스템 및 이론 제35권 제5호(2008.6)

1. 서 론

유비쿼터스 컴퓨팅의 인프라가 되는 무선 센서 네트워크(wireless sensor networks: 이하 WSN이라 함)는 제한된 전력량을 가진 매우 작은 하드웨어로 이루어지는 많은 수의 센서노드들로 구성된다. 이 네트워크의 토폴로지와 라우팅 방식은 그 목적에 따라 결정되어야 하며, 하드웨어 및 소프트웨어도 필요한 경우에는 변경되어야 한다. 시뮬레이션은 시스템이 구현되기 전에 시스템 동작과 성능을 예측할 수 있게 한다. 따라서 WSN을 최적으로 설계하기 위해서는 WSN 시뮬레이터가 필요하다[1].

그러한 목적을 위해 WSN 시뮬레이터는 다음과 같은 세 가지 기능들을 제공해야 한다. 첫째, 센서노드에 적재되는 응용프로그램의 실행을 가능한 한 정확하게 시뮬레이션 함으로써, WSN의 동작을 확인할 수 있어야 한다. 둘째, 센서노드들이 가진 전력량이 한정되어 있기 때문에, 응용프로그램의 실행시간 및 전력소모량을 예측할 수 있어야 한다. 마지막으로, WSN은 많은 수의 센서노드들을 시뮬레이션 할 수 있어야 한다. 그러나 WSN 시뮬레이션에 대한 지금까지의 대부분 연구들은 WSN의 동작을 확인하는 것에만 집중되어 있으며, 특정한 WSN 운영체제나 라이브러리 API, 또는 프로그래밍 언어가 필요하다는 문제점이 있다. 예를 들어, 널리 알려진 WSN 시뮬레이터인 TOSSIM[2]의 경우에는 WSN 운영체제인 TinyOS[3] 기반의 응용프로그램들에 대해서만 시뮬레이션이 가능하며, 실행시간과 전력소모량은 예측할 수 없다. 명령어-레벨의 시뮬레이션(instruction-level simulation)을 하는 Avrora[4]의 경우에는 실행시간과 전력소모량은 예측할 수 있지만, 센서노드를 하나의 쓰레드(thread)로 동작시키기 때문에 많은 수의 센서노드들을 시뮬레이션 하기가 어렵다.

명령어-레벨의 시뮬레이션은 센서노드의 시스템 클록(clock) 수준으로 시뮬레이션의 정밀도를 높임으로써, WSN 응용프로그램의 실행동작과 실행시간 및 전력소모량을 실제 센서보드와 거의 근접한 수준으로 예측할 수 있다. 또한 이 방법은 실제 센서노드에 적재되는 실행이미지(executable image)를 시뮬레이션 작업부하로 사용하여 기존의 많은 WSN 시뮬레이터들이 가진 소프트웨어 의존성 문제를 해결하였다. 그러나 명령어-레벨 시뮬레이션은 시뮬레이션 정밀도가 높기 때문에 시뮬레이션 시간이 오래 걸린다[1].

시뮬레이션 시간을 단축시키기 위해서는 병렬 이산-사건 시뮬레이션(parallel discrete-event simulation: 이하 PDES라 함) 방법을 사용할 수 있다. 그러나 일반적인 PDES 방법들[5]은 회로 시뮬레이션(logic simulation)이나 병렬 계산 시뮬레이션(parallel computational simulation)에 적합하도록 개발되어 왔기 때문에, 명령어-레벨의 WSN 시뮬레이션에 적용하기는 어렵다. 예를 들어, 동기식 시뮬레이션(synchronous simulation)[6]을 사용할 경우에는 시스템 클록 단위로 동기화 메시지를 빈번하게 발생시키기 때문에 이에 따른 통신 오버헤드(overhead)가 매우 크다는 문제점이 있다. 또한 비동기식 PDES 방법의 하나인 낙관적 시뮬레이션(optimistic simulation)[7]을 사용할 경우에는 롤백(rollback)을 위하여 저장해야 할 정보들이 매우 많다는 문제점이 있다.

따라서 본 연구에서는 명령어-레벨의 WSN 시뮬레이션에 최적화된 최적-동기식(optimal-synchronous) PDES 방법을 제안한다. WSN에서 센서노드들은 서로 독립적으로 동작하며, 다른 센서노드의 동작에 의해 영향을 받는 경우는 고주파(radio frequency: 이하 RF라 함) 통신을 수신하였을 때뿐이다. 그리고 어떤 센서노드에서 RF 송신이 시작된 후부터 다른 센서노드에서 RF 수신에 따른 동작이 발생할 때까지는 일정한 시간 지연이 존재한다. 따라서 최적-동기식 PDES 방법에서는 이러한 WSN 특성을 고려하여 여러 대의 컴퓨터에 센서노드 단위로 작업을 분배하고, RF 송신 동작이 발생하면 그것을 수신하는 모든 센서노드들에게 RF 수신 동작이 정해진 시간에 발생하도록 사건을 스케줄 함으로써, 분배된 모든 사건들이 시간 순서대로 시뮬레이션 되도록 하였다.

일반적으로 병렬 시뮬레이션에서 속도향상은 프로세서의 수보다 커질 수 없지만, 본 연구의 시뮬레이션 모델에서는 시뮬레이션 되는 센서노드의 수가 많아질수록 속도향상은 프로세서 수의 제곱에 가까워진다는 것이 수식으로 예측되었다. 이렇게 예측된 중요한 이유는 명령어-레벨 시뮬레이션은 사건의 동작을 처리하는 시간이 매우 짧아서 사건 큐(event queue)의 처리 시간이 시뮬레이션 시간에서 중요한 요인으로 작용하기 때문이다. 즉, 센서노드의 수가 많아지면 사건 큐의 길이가 길어지기 때문에, 새로운 사건(event)을 발생할 시간 순서대로 사건 큐에 삽입하기 위하여 소요되는 시간이 그 사건 루틴(event routine)을 처리하는 시간보다 더 길어진다. 이러한 경우에 병렬 시뮬레이션은 작업부하를 분할하는 것과 동시에 사건 큐도 분할하여 높은 속도향상을 제공한다.

최적-동기식 PDES 방법을 적용하여 본 연구팀에서 이전 연구에서 개발하였던 MISS(machine instruction-level sensor network simulator)[1]를 확장한 병렬 센서 네트워크 시뮬레이터(parallel sensor network simulator: 이하 PASENS라 함)는 높은 시뮬레이션 정밀도를 유지하는 동시에 시뮬레이션 시간을 크게 단축시켜

서 많은 수의 센서노드들을 포함하는 대규모 센서 네트워크를 단시간 내에 시뮬레이션할 수 있다. 특히 PASENS는 기존의 WSN 시뮬레이터들이 기억장치 용량의 한계와 긴 시뮬레이션 시간 때문에 많은 수의 센서노드들을 시뮬레이션 하기 어려운 문제점을 해결하였다. 또한 PASENS는 실행이미지를 작업부하로 사용하여 실제 WSN 응용프로그램들의 동작을 확인할 수 있고, 응용프로그램의 실행시간 및 전력소모량을 예측하여 WSN의 수명(life time)을 추정할 수 있다.

PASENS의 실험결과에 따르면, 시뮬레이션 되는 센서노드의 수가 충분히 많다면, 수식으로 예측한 바와 같이 병렬 시뮬레이션에 참여하는 컴퓨터 수의 제곱에 접근하는 높은 속도향상을 얻을 수 있다는 것을 확인하였다. 또한 PASENS는 이더넷에 연결된 PC들을 그대로 사용할 수 있기 때문에, 시뮬레이션 환경을 구축하는 것이 쉽고 병렬 시뮬레이션을 위한 비용이 추가로 발생하지 않는 장점이 있다.

본 논문의 구성은 2장에서 실행이미지를 작업부하로 사용하는 명령어-레벨의 센서 네트워크 시뮬레이션에 관련된 연구들을 설명하고, 3장에서는 구현한 시뮬레이터의 구조를 설명하였다. 그리고 4장에서는 최적-동기식 PDES 방법에 관하여 설명하고, 본 연구의 시뮬레이션 모델에서 얻을 수 있는 속도향상을 예측하였다. 5장에서는 제안한 방법으로 구현한 시뮬레이터를 사용하여 실험한 결과들을 제시하고 분석하였으며, 마지막으로 6장에서는 전반적인 결론을 요약하였다.

2. 명령어-레벨의 센서 네트워크 시뮬레이션

WSN 시뮬레이터에 대한 많은 연구가 NEST(network embedded software technology) 프로젝트[8]에서 이루어졌는데, 그 결과로서 TOSSIM(TinyOS simulator)[2], Prowler(probabilistic wireless network simulator)[9], Siesta(simple NEST application simulator)[10]가 출현하였다. 이들은 모두 NEST 프로젝트에서 개발된 TinyOS[3]가 탑재되어 있는 하드웨어 플랫폼 기반에서 동작하는 시뮬레이터들이기 때문에 TinyOS 기반에서만 동작하며, 실행시간 및 전력소모량을 예측할 수 없다. 특히 TOSSIM을 구동하기 위해서는 TinyOS에서 TOSSIM용으로 컴파일된 작업부하가 필요하고, Prowler의 경우에는 MATLAB 소프트웨어가 필요하며, Siesta는 시뮬레이션을 하기 위해서는 응용 소프트웨어가 특정한 라이브러리 API를 포함해야 하는 문제점이 있다.

WSN 시뮬레이션에서 명령어-레벨 시뮬레이션 방법은 실제 센서노드에 적재되는 실행이미지를 작업부하로 사용하며, 센서노드에서 하드웨어적으로 발생하는 동작들을 시스템 클록 수준으로 기계명령어들을 시뮬레이션한다. 이러한 특성으로 인하여 명령어-레벨 시뮬레이션 방법은 WSN 응용프로그램들의 동작과 실행시간 및 전력소모량을 실제 센서보드와 거의 근접한 수준으로 예측할 수 있다. 또한 이 방법은 WSN 운영체제나 라이브러리 API, 또는 프로그래밍 언어에 상관없이 실제 센서노드에 적재되는 실행이미지만 있으면 시뮬레이션이 가능하다. 즉, 명령어-레벨 시뮬레이션 방법은 기존의 많은 WSN 시뮬레이터들이 가진 소프트웨어 의존성 문제를 해결할 수 있다. 비록 하드웨어 의존성 문제가 발생하지만, 일반적으로 WSN에서 센서노드의 하드웨어 구조들은 소프트웨어에 비하여 상대적으로 종류가 적기 때문에 명령어-레벨 시뮬레이션은 의존성 측면에서 얻을 수 있는 장점이 단점보다 더 큰 방법이다[1].

명령어-레벨 시뮬레이션을 사용한 WSN 시뮬레이터들로는 ATEMU[11]와 Avrora[4], 그리고 본 연구의 PASENS가 있다. PASENS가 이산-사건 시뮬레이션 방법을 사용하여 명령어-레벨 시뮬레이션을 하는 반면에, ATEMU는 센서노드의 시스템 클록 간격으로 가상 센서노드들에 적재된 실행이미지를 순차적으로 시뮬레이션 하고, Avrora는 각 센서노드가 하나의 쓰레드로 동작하는 다중-쓰레드 방식으로 시뮬레이션 한다. 이 시뮬레이터들은 실행시간 및 전력소모량은 예측할 수 있지만, 높은 시뮬레이션 정밀도로 인하여 길어진 시뮬레이션 시간 때문에 많은 수의 센서노드들을 시뮬레이션 하기가 어렵다.

Avrora는 시뮬레이션 시간을 단축시키기 위하여 슬립(sleep) 상태와 타이머 등을 시뮬레이션 할 때 사건 큐를 사용하고, 쓰레드들을 관리하기 위한 효율적인 동기화 방법을 사용하였다. 또한 Avrora는 다중-쓰레드를 다중-프로세서 상에서 동작시킴으로써 시뮬레이션 시간을 단축하려 하였지만, 프로세서 수가 증가하면 전역 데이타 구조에 대한 경합(contention)과 운영체제의 쓰레드 스위칭 오버헤드 때문에 제한적인 속도향상만을 얻을 수 있었다. Avrora의 실험결과에 따르면 ATEMU는 8시간 안에 512개의 센서노드를 시뮬레이션 하지 못하였고, Avrora는 같은 수의 센서노드들을 5분 안에 시뮬레이션 하였다. 그러나 Avrora는 네 개의 센서노드를 네 개의 프로세서에서 시뮬레이션 한 경우에 2배의 속도향상을 얻은 반면에, 32개의 센서노드를 8개의 프로세서에서 시뮬레이션 한 경우에는 속도향상이 1.8배였다[4]. 이러한 결과는 쓰레드 기반의 시뮬레이션은 병렬 시뮬레이션에 부적합하다는 것을 나타낸다.

본 연구에서는 명령어-레벨 시뮬레이션을 위하여 이산-사건 시뮬레이션 방법을 사용한다. 이산-사건 시뮬레이션은 시뮬레이션 대상의 동작을 사건 단위로 정의하고, 그 동작이 발생할 시간과 함께 사건을 사건 큐에

삽입한다. 시뮬레이션은 사건 큐에 있는 사건들 중에서 가장 빨리 발생되는 사건을 사건 큐에서 꺼내면서 진행되기 때문에, 이산-사건 시뮬레이션에서는 별도의 동기화 방법이 필요가 없다. 즉, 센서노드 내부의 모듈들의 동작들에 대하여 동작의 발생 시간과 상호 작용에 따라 사건을 정의하면, 이산-사건 시뮬레이션은 이들 동작들이 실제 센서보드에서 발생되는 시간 순서와 동일하게 시뮬레이션을 수행한다. 센서노드들 사이의 동작들도 이와 동일한 이유로 인하여 동기화가 이루어진다. 또한 명령어-레벨 시뮬레이션에서 이산-사건 시뮬레이션 방법을 사용할 경우에는, 병렬 이산-사건 시뮬레이션 방법들을 이용할 수 있다. 병렬 시뮬레이션은 명령어-레벨 시뮬레이션의 문제점인 느린 시뮬레이션 속도를 향상시킴으로써, 많은 수의 센서노드를 갖고 있는 WSN을 시뮬레이션 할 수 있게 한다.

3. 시뮬레이터의 내부 구조

본 연구에서 개발한 시뮬레이터인 PASENS는 그림 1에서 보는 바와 같이 쉘 에이전트(shell agent), 통신 에이전트(communication agent), 단일 시뮬레이션 제어기(single simulation controller), 병렬 시뮬레이션 제어기(parallel simulation controller), 그리고 가상 센서노드(virtual sensor node)로 구성되어 있으며, C 언어로 구현되었다. 쉘 에이전트는 PASENS를 쉘 명령어 기반으로 제어하는 모듈이다. 쉘 명령어는 키보드를 통해 사용자로부터 입력되거나, 통신 에이전트를 통하여 같은 컴퓨터 혹은 다른 컴퓨터의 프로그램으로부터 입력된다. 쉘 에이전트는 입력된 쉘 명령어를 해석하고, PASENS의 다른 모듈들을 제어하며 해석된 명령어를 실행한다. 또한 쉘 에이전트는 다른 컴퓨터에서 실행되는 PASENS 프로그램에게 쉘 명령어를 전달하여 원격 동작을 수행할 수도 있다. 병렬 시뮬레이션에서는 이러한 방법을 통하여 쉘 에이전트가 다른 컴퓨터에서 실행되는 PASENS의 에이전트와 연동한다. 통신 에이전트는 PASENS 프로그램들 사이의 통신과 PASENS를 이용하는 다른 응용프로그램들 사이의 통신을 위하여 TCP 기반의 일대일통신(1-to-1) 혹은 UDP 기반의 멀티캐스트(multicast) 통신을 수행하는 모듈이다.

단일 시뮬레이션 제어기는 사건 큐에서 사건을 인출하여 처리하고, 그 사건에 의해 스케줄 된 새로운 사건을 사건 큐에 삽입함으로써 가상 센서노드에서 명령어-레벨의 이산-사건 시뮬레이션을 수행하는 모듈이다. 사건 큐의 제어는 이산-사건 시뮬레이션을 위한 라이브러리인 smpl[12]을 이용하였다. 병렬 시뮬레이션 제어기는 여러 개의 컴퓨터들에서 병렬 시뮬레이션을 진행할 때에 준비 작업과 동기화 작업을 수행한다.

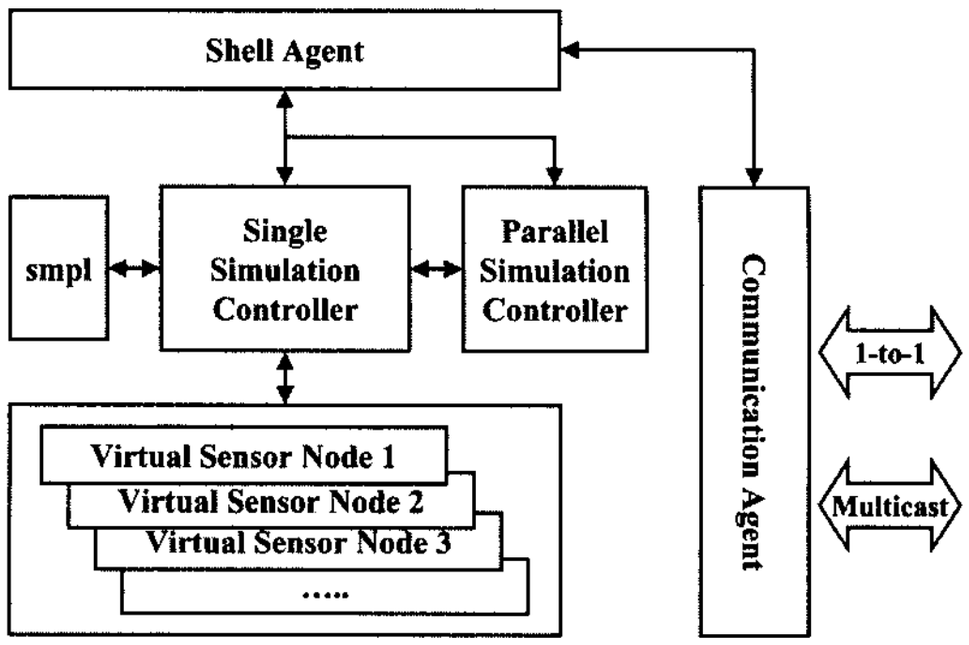


그림 1 PASENS의 구조

가상 센서노드는 센서노드의 동작을 소프트웨어적으로 시뮬레이션 하기 위한 모듈로서, 센서 네트워크 연구들[2,4,9-11]에서 많이 사용되고 있는 NEST 프로젝트[8]에서 개발된 MICAz 형식의 WSN 센서보드인 Cross-Bow 사의 MPR2400[13] 보드를 모델링 하였다. MICAz 형식의 센서보드에는 Atmel사의 ATmega128L[14] 마이크로컨트롤러(microcontroller)와 Chipcon사의 CC2420[15] RF 트랜시버가 장착되어 있다. ATmega128L은 마이크로컨트롤러의 클록이 16 ㎒에서 최대 16 MIPS의 처리율(throughput)을 가진 AVR Enhanced RISC(reduced instruction set computer) 구조의 8 비트 마이크로컨트롤러이고, CC2420은 250 Kbps로 데이타를 전송할 수 있는 2.4 GHz IEEE 802.15.4 호환 RF 트랜시버이다. 센서보드를 가상 하드웨어 단위로 설정한 이유는 WSN에서는 센서보드가 하드웨어적으로 독립된 개체이기 때문이다. 대부분의 센서보드들은 마이크로컨트롤러와 RF 트랜시버를 기본 요소로 하여 구성되기 때문에, 유사한 형태의 센서보드들은 약간의 수정만을 통하여 시뮬레이션이 가능하다.

PASENS가 사용하는 시뮬레이션 작업부하는 실제 센서보드에 적재되는 실행이미지로서, ATmega128L용으로 크로스 컴파일 되어 센서보드에 적재되는 인텔 헥스-레코드 형식(Intel's hex-record format: .hex), 모토롤라 S-레코드 형식(S-record format: .srec), 그리고 롬 이미지 형식(ROM image format: .rom)의 파일을 사용한다. 명령어-레벨 시뮬레이션 방법은 이 실행 이미지를 시뮬레이션 엔진의 가상 센서노드로 적재한 다음에, 실제 센서노드에서 프로세서 명령어가 실행되는 방법과 거의 동일하게 이루어진다. 즉, 명령어 실행동작의 시뮬레이션은 실제 센서노드에서와 같이 가상 플래시 메모리에서 현재 프로그램 카운터(program counter, PC)가 가리키는 위치의 기계명령어를 인출하고, 기계명령어의 비트 패턴을 분석하여 실행함으로써 이루어진다.

4. 병렬 이산-사건 시뮬레이션

4.1 개요

PDES는 이산-사건 시뮬레이션에서 시뮬레이션 시간을 단축시키기 위하여 사용된다. PDES에서 사건들은 논리 프로세스(logical process: 이하 LP라 함)들로 나누어져 처리되는데, 각 LP는 지역 가상시간(local virtual time: 이하 LVT라 함)을 증가시키면서 자신의 사건 큐에 할당된 사건들을 시뮬레이션 한다. 각 프로세스에 분산되어 시뮬레이션 되는 사건들은 단일 프로세스에서 시뮬레이션 되는 경우와 동일한 시간 순서대로 처리되어야 하며, 이것을 인과성 제약(causality constraints)이라고 한다. 만약 각 LP에 분산되어 시뮬레이션 되는 사건들이 하나의 LP에서 시뮬레이션 되는 경우와 동일한 시간 순서대로 처리되지 않았다면, 인과성 오류(causality error)가 발생했다고 한다. PDES의 일반적인 방법에는 동기식 시뮬레이션(synchronous simulation)과 보수적 시뮬레이션(conservative simulation), 그리고 낙관적 시뮬레이션(optimistic simulation)이 있다. 보수적 시뮬레이션과 낙관적 시뮬레이션은 비동기식 시뮬레이션 방법으로서 프로세서들 간의 LVT가 일치하지 않는 반면에, 동기식 시뮬레이션에서는 프로세서들 간의 LVT가 항상 일치한다[5].

동기식 시뮬레이션은 가장 간단한 PDES 방법으로서, 인과성 오류가 발생하지 않는 가장 짧은 시간 간격(time step)으로 전역 가상시간(global virtual time: 이하 GVT라 함)을 일정하게 증가시키면서 시뮬레이션을 진행하는 방법이다. LVT는 GVT의 복사본이기 때문에, 시뮬레이션은 일정한 시간 간격으로 증가하는 GVT(0, Δ, 2Δ, 3Δ,...)에 따라서 진행된다. 이 방식에서는 동기화를 위해 대부분의 병렬처리 환경에서 이용되는 배리어 동기화 방법들(barrier synchronization mechanisms)을 사용할 수 있다[6]. 그러나 이렇게 일정한 시간 간격으로 GVT를 증가시킬 경우에, 어떤 시간 간격 동안에는 처리할 사건들이 존재하지 않는 경우도 있기 때문에 효율이 낮아지는 문제점이 발생한다. 또한 작업균등화(load balancing)의 어려움, 빈번한 동기화 작업으로 야기되는 높은 통신 오버헤드로 인하여 LP 수에 한계가 있다는 문제점이 있다. 그러나 동기식 시뮬레이션은 비동기식 시뮬레이션에 비하여 오버헤드가 적고 구현이 용이하며, 성능을 예측할 수 있다는 장점이 있다[16]. 따라서 동기식 시뮬레이션은 기본 계산 단위(computation granularity)가 매우 작고, 많은 수의 사이클 병렬성(cycle parallelism)을 가진 시뮬레이션에 적합한 방법이다[17].

Chandy-Misra[18]와 Bryant[19]의 연구에서 제안된 보수적 시뮬레이션 방법은 Chandy-Misra- Bryant(CMB) 프로토콜이라고도 불린다. 이 시뮬레이션 방법에서는 각 LP가 원격 LP에서 수행되어질 사건을 출력채널을 통해 순서대로 다른 LP들의 입력채널로 LVT와 함께 송신한다. 외부로부터 사건을 수신한 LP는 그것을 송신한 LP로부터 이미 수신한 사건들의 LVT보다 더 적은 값의 LVT를 가지는 외부 사건을 그 이후에는 수신하지 않을 것이기 때문에, LP는 LVT의 순서대로 사건들을 안전하게 시뮬레이션 할 수 있다. 그러나 만약 어떠한 LP에 비어 있는 입력채널이 존재한다면, 그 입력채널에 마지막에 남아 있었던 사건의 LVT가 그 입력채널의 LVT로 유지된다. 이 때 그 LVT가 다른 입력채널들의 LVT나 내부의 사건 큐에 있는 사건들의 LVT보다 적을 경우, 이 LP는 다른 사건이 수행되어 LVT가 진행될 때까지 기다리는 대기 상태가 된다. 이러한 대기 상태의 LP들이 순환 형태가 되면 교착(deadlock) 상태가 발생한다. 교착 상태가 발생하지 않게 하기 위하여, 이후에는 더 이상 발생할 사건이 없다는 것을 나타내는 공백 메시지(null message)[20]를 사용할 수 있지만, 추가적인 통신 오버헤드를 초래한다.

시간 왜곡 알고리즘(Time Warp algorithm)을 기반으로 한 낙관적 시뮬레이션 방법[7]은 인과성 오류는 허용하지만 롤백(rollback)이라고 불리는 복구 방법을 통하여 인과성 제약을 만족시키는 방법이다. 시간 왜곡 알고리즘에서는 어떤 LP에 현재 LVT 이전에서 수행되어야 할 외부사건이 도착하면, 가장 최근의 상태가 저장되어 있는 시뮬레이션 히스토리(simulation history)를 이용하여 이 외부사건이 수행될 시간의 이전 상태로 복구한다. GVT는 모든 LP의 입력채널 LVT들과 사건 큐의 사건들의 최소값으로 정의되며, GVT보다 적은 LVT를 가지는 새로운 사건은 발생하지 않기 때문에 시뮬레이션 히스토리에서 더 이상 필요하지 않은 기록들을 제거하기 위하여 GVT가 사용된다. 즉, 시뮬레이션 히스토리에서 GVT보다 적은 LVT를 가지는 상태들 중에서 가장 큰 LVT를 가지는 한 상태를 제외한 모든 상태를 안전하게 제거할 수 있다. 그러나 이 방법은 시뮬레이션 히스토리를 유지하기 위한 기억장치 비용과 계산 오버헤드가 발생하는 문제가 있다.

명령어-레벨의 WSN 시뮬레이션은 높은 정밀도에 의하여 실행시간 및 전력소모량 예측과 같은 많은 기능적인 장점을 얻을 수 있는 반면에, 시뮬레이션 정밀도가 높기 때문에 시뮬레이션 시간이 길어져서 시뮬레이션 할 수 있는 센서노드 수에 한계가 발생하는 문제점이 있다. 지금까지 살펴본 일반적인 PDES 방법들은 회로 시뮬레이션이나 병렬 계산 시뮬레이션에 적합하게 연구되어 왔기 때문에, 명령어-레벨의 이산-사건 시뮬레이

션 방법을 사용하는 WSN 시뮬레이션에 적용하기는 어렵다. 동기식 시뮬레이션을 사용할 경우에는 시스템 클록 단위로 동기화 메시지를 빈번하게 발생해야 하기 때문에 이에 따른 통신 오버헤드가 매우 크다는 문제점이 있다. 또한 보수적 시뮬레이션을 사용할 경우에는 LP들 간의 동기화를 유지하기 위하여 검사해야 할 사건들의 수가 매우 많고, 낙관적 시뮬레이션을 사용할 경우에는 롤백을 위하여 저장해야 할 정보들이 매우 많다는 문제점이 있다. 따라서 본 연구에서는 시뮬레이션 시간을 단축하여 많은 수의 센서노드들을 시뮬레이션 할 수 있게 하기 위하여, 일반적인 동기식 PDES 방법을 명령어-레벨의 WSN 시뮬레이션에 적합하도록 변경한 최적-동기식 PDES 방법을 제안한다.

4.2 최적-동기식 PDES

WSN의 센서노드들은 RF 통신 동작 외에는 센서노드 별로 독립적으로 동작하기 때문에, 회로 시뮬레이션과는 달리 WSN 시뮬레이션에서는 센서노드 단위로 문제 분할이 용이하다. 또한 어떤 센서노드에서 RF 송신이 시작된 후부터 다른 센서노드에서 RF 수신에 따른 동작이 발생할 때까지는 일정한 시간 지연이 존재한다. 최적-동기식 PDES 방법은 이러한 WSN의 특성을 반영하여 WSN 시뮬레이션에 최적화한 PDES 방법이다.

최적-동기식 PDES 방법은 그림 2와 같이 GVT를 유지하면서 병렬 시뮬레이션을 전체적으로 제어하는 하나의 제어 프로세스(control process: 이하 CP라 함)와 센서노드들을 분할하여 시뮬레이션을 수행하는 여러 개의 LP들로 구성된다. 이 방법은 CP와 각 LP를 이더넷으로 연결된 각각의 컴퓨터가 하나씩 담당하여 처리하는 중앙집중식 구조이다. 병렬 시뮬레이션에 참여하는 LP의 수는 WSN 응용이나 시뮬레이터 구동 환경에 따라서 달라질 수 있기 때문에 사용자가 임의로 설정할 수 있으며, 그룹 ID를 이용하여 같은 네트워크 안에서 독립적인 여러 WSN 응용을 동시에 시뮬레이션 할 수도 있다. 이와 같이 이더넷에 연결된 PC들을 사용하는 컴퓨팅 환경은 최적-동기식 PDES 방법을 저렴한 비용으로 쉽게 구축할 수 있게 한다.

최적-동기식 PDES 방법은 동기식 시뮬레이션에 기반을 두지만 비동기식 시뮬레이션의 특징도 가지고 있는 방법이다. 동기식 시뮬레이션에 기반을 둔 이유는 명령어-레벨 시뮬레이션이 회로 시뮬레이션처럼 기본 계산 단위가 매우 작아서 사건의 동작을 처리하는 시간이 매우 짧기 때문이다. 그러나 회로 시뮬레이션과는 달리 WSN 시뮬레이션은 RF 통신 동작 외에는 센서노드 별로 독립적으로 동작하기 때문에 센서노드 단위로 문제 분할이 용이하다. 또한 어떤 센서노드에서 RF 송신이 시작된 후부터 다른 센서노드에서 RF 수신에 따른 동

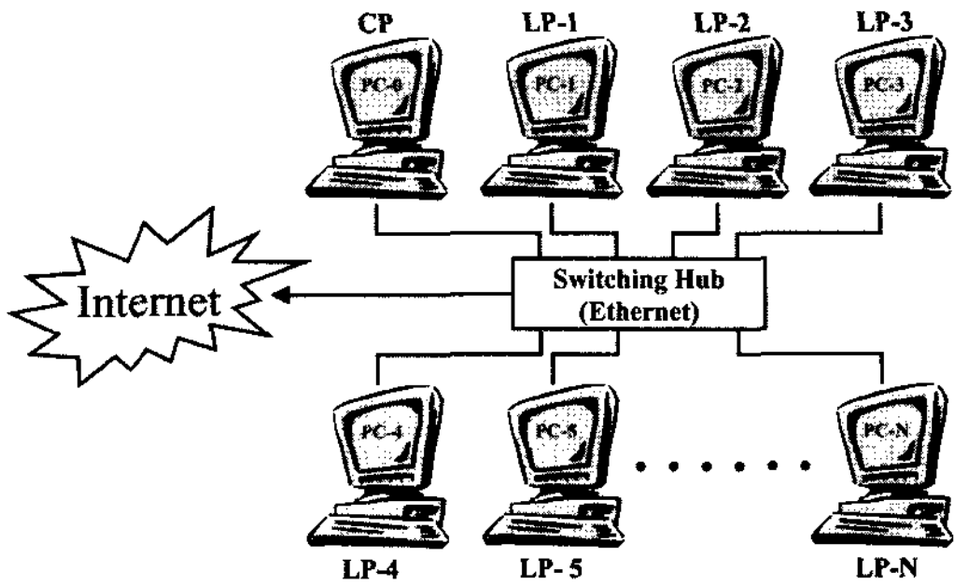


그림 2 최적-동기식 PDES의 시뮬레이션 환경

작이 발생할 때까지는 일정한 시간 지연이 존재한다. 따라서 최적-동기식 PDES 방법은 그 지연 시간 간격 단위로 동기식 시뮬레이션을 진행하고, 그 시간 간격 내에서 RF 송신 동작이 발생하면 다음 시간 간격 동안에 그것을 수신하는 모든 센서노드에서 RF 수신 동작이 발생하도록 사건을 스케줄 함으로써, 분배된 모든 사건이 시간 순서대로 수행되도록 한다. 결과적으로 최적-동기식 시뮬레이션 방법은 인과율 제약을 만족시키는 일정 시간 간격으로 GVT를 증가시키면서 동기식 시뮬레이션을 수행하지만, 그 시간 간격 내에서는 각 LP가 LVT를 증가시키면서 비동기식 시뮬레이션을 수행하게 된다.

그림 3은 CP와 LP의 최적-동기식 PDES 알고리즘을 보여준다. CP는 배리어 동기화 방법을 이용하여 GVT를 유지하면서 LP들을 제어하여 병렬 시뮬레이션을 수행한다. 시뮬레이션이 시작되면 CP는 시뮬레이션이 종료되는 시간까지 인과성 제약을 만족시키는 일정 시간 간격(Δt)으로 GVT를 증가시킨다. 이 GVT는 멀티캐스팅을 사용하여 시뮬레이션에 참여하는 모든 LP들에게 전달된다. GVT를 받은 LP들은 GVT까지 LVT를 증가시키면서 각각 시뮬레이션을 진행한다. LP는 사건 큐에서 가장 빨리 발생될 사건을 꺼내고, 그 사건이 발생되는 시간으로 LVT를 갱신한다. 사건 큐에서 꺼낸 사건의 LVT가 GVT 이하일 경우에는 사건 루틴을 실행하며, 실행 중에 RF 송신이 발생하면 다른 LP들에게 RF 메시지를 멀티캐스팅 한다. RF 메시지를 받은 LP는 이 메시지를 처리하기 위한 사건을 스케줄 한다. 만약 사건 큐에서 꺼낸 사건의 갱신된 LVT가 GVT보다 크다면, 꺼낸 사건을 다시 사건 큐에 삽입하고 LVT는 GVT로 갱신한다. 그리고 LP는 GVT까지 시뮬레이션을 완료했다는 메시지를 전송한 후에 다음 GVT를 수신할 때까지 대기한다. CP는 모든 LP들로부터 시뮬레이션 완료 메시지를 수신하면 GVT를 증가시킨다.

최적-동기식 PDES가 동기식 시뮬레이션과 구별되는 차이점은 RF 송신과 수신 사이의 지연시간으로 설정된

```
/* Δt : A time interval of GVT
   End_Time : End of simulated time */
GVT = 0;
do {
   GVT += Δt;
   if (GVT > End_Time)
     GVT = End_Time;
   Broadcast GVT to all LPs;
   Wait until receiving completion messages from all LPs;
} while (GVT < End_Time);
```

(a) CP

```
/* E_i : ith event
   LVT_i : The occurrence time of E_i */
while (1) {
   Wait for an GVT from CP;
   while (1) {
      Remove an event E_i with the earliest LVT_i from the event queue;
      LVT = LVT_i;
      if (LVT > GVT) {
         Insert E_i into an event queue;
         LVT = GVT;
         break;
      }
      Execute the event routine of E_i;
      if (RF transmission occurs by E_i)
         Broadcast the RF message to other LPs;
   }
   Insert an event to process the received message into the event queue;
   Send a completion message to CP;
}
```

(b) LP

그림 3 최적-동기식 PDES 알고리즘

GVT의 증가 간격이다. 이 시간은 동기식 시뮬레이션에서 시스템 클록 단위로 설정되는 GVT의 증가 간격에 비하여 매우 긴 시간이다. WSN의 센서노드는 다른 센서노드의 RF 통신에 의해서만 영향을 받으며, RF 통신은 송신과 수신 사이에 지연시간이 존재한다. 만약 GVT의 증가 간격 Δt가 이 지연시간보다 짧은 시간이라면 인과성 오류는 발생하지 않는다. 그림 4에서 보는 바와 같이, GVT 증가 단계가 i번째일 때에는 i×Δt 시간의 GVT가 LP들로 전송되며, GVT를 수신한 LP들은 [(i-1)×Δt, i×Δt] 구간에서 발생되는 모든 사건들을 시뮬레이션 한다. 이 구간에서 발생한 RF 송신은 Δt의 지연시간으로 인하여 [i×Δt, (i+1)×Δt] 구간에서 수신된다. 따라서 [i×Δt, (i+1)×Δt] 구간의 시뮬레이션이 시작되기 전에 RF 수신을 처리하는 사건을 수신될 시간에 발생하도록 사건 큐에 삽입하면, 모든 사건들은 시간 순서대로 시뮬레이션 된다. 즉, RF 수신 이후에 발생하는 사건들이 RF 수신을 처리하는 사건보다 먼저 시뮬레이션 되지 않기 때문에 인과성 오류는 발생하지 않는다.

MICAz 형식의 센서보드를 시뮬레이션 대상으로 설정한 본 연구에서 인과성 제약을 만족시키는 GVT의 증가 간격 Δt는 다음과 같이 결정된다. IEEE 802.15.4 호환의 RF 트랜시버 CC2420 RF 트랜시버를 사용하는 MICAz 형식의 센서보드에서, 송신 센서노드의 마이크로컨트롤러에서 송신을 발생시키는 기계명령어가 실행

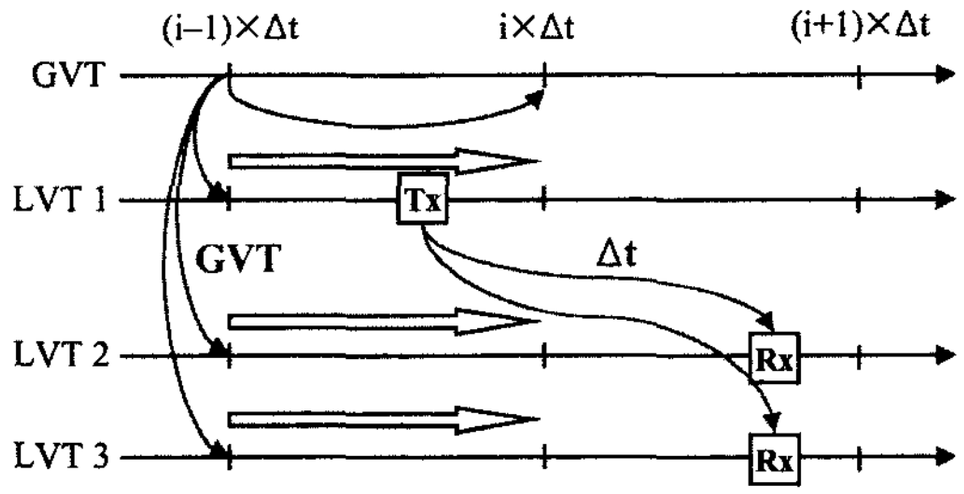


그림 4 최적-동기식 PDES의 인과성 보호

되면 CC2420에서는 송신 모드로 전환하기 위해 8 심벌(128 μs) 또는 12 심벌(192 μs)의 조정 시간(calibration time)이 지난 후에 데이타 송신이 시작된다. 그리고 수신 센서노드의 마이크로컨트롤러에서는 패킷의 SFD (start-of-frame delimiter)까지 수신이 된 후에야 SFD 신호를 통하여 수신을 인식할 수 있다. 이때까지 수신한 데이타 크기는 5 바이트이고 CC2420의 대역폭(bandwidth)은 250 Kbps이기 때문에 160 μs의 시간이 소요된다. 그 외에 송신 센서노드에서 수신 센서노드로 전달되기 위하여 미터 당 약 3 ns가 소요되는 전파 지연시간이 필요하지만, 이 시간은 가변적이고 상당히 짧은 시간이기 때문에 무시한다. 결과적으로 본 연구의 시뮬레이션 모델에서 인과성 제약을 만족시키는 GVT의 증가 간격의 최소값은 288 μs이지만, 본 연구의 실험에서 사용된 TinyOS[3]의 예제들은 RF 통신 조정 시간이 12 심벌이기 때문에 GVT의 증가 간격을 352 μs로 설정하였다.

최적-동기식 PDES 방법은 병렬 시뮬레이션에 참여하는 LP의 수가 증가하더라도 오버헤드 시간은 거의 증가하지 않는다. 최적-동기식 PDES 방법에서 오버헤드 시간은 LP들 사이의 동기화를 유지시키기 위한 시간이다. 즉, 오버헤드 시간은 CP가 LP들에게 일정 시간 간격으로 GVT를 포함하는 메시지를 전송하는 시간, GVT까지 시뮬레이션을 완료한 LP가 CP로 시뮬레이션 완료 메시지를 전송하는 시간, RF 송신이 발생한 LP가 다른 LP들에게 RF 메시지를 전송하는 시간, 그리고 먼저 시뮬레이션을 완료한 LP가 다른 LP들이 시뮬레이션을 완료할 때까지 기다리는 시간의 합이다. 오버헤드 시간의 대부분을 차지하는 메시지 통신에 필요한 네트워크 대역폭은 LP의 수가 증가할수록 커지지만, 이 대역폭이 100 Mbps의 이더넷 대역폭보다 작기 때문에 LP의 수가 증가하더라도 오버헤드 시간이 거의 증가하지 않는다. 특히 LP가 시뮬레이션 하는 센서노드의 수가 많아질수록 전체 시뮬레이션 시간에서 오버헤드 시간이 차지하는 비율이 매우 낮아지기 때문에, 센서노드의 수가 많은 경우에 오버헤드 시간은 전체 시뮬레이션 시간

에 거의 영향을 주지 않는다.

예를 들어, GVT의 증가간격이 352 μs인 경우에는 1초의 GVT동안 2,840번의 Δt 증가가 발생한다. 이 때 배리어 동기화를 위해 CP와 LP들로부터 통신 메시지가 각각 2,840개가 발생된다. 만약 LP의 수가 40개이고 RF 전송 메시지가 4개가 발생한다면, 모두 116,444개의 통신 메시지가 발생한다. 최적-동기식 PDES에서 통신 메시지의 크기가 150 바이트 미만이기 때문에 이 경우에 최대로 요구되는 네트워크 대역폭은 17 Mbps에 불과하다. 또한 많은 센서노드를 시뮬레이션 함으로써 전체 시뮬레이션 시간이 10초가 걸렸다면, 1초의 시뮬레이션 시간동안 요구되는 네트워크 대역폭은 1.7 Mbps로 감소한다. 결과적으로, 최적-동기식 PDES에서는 시뮬레이션 되는 센서노드의 수가 많아질수록 전체 시뮬레이션 시간에서 차지하는 오버헤드 시간의 비율은 무시할 수 있을 정도로 작아진다.

PASENS는 그림 5와 같이 프로세스의 상태를 변화시키면서 병렬 시뮬레이션을 진행한다. PASENS가 구동되면 단일 시뮬레이션을 할 수 있는 초기(Init) 상태로 전이(transition)된다. 이 상태에서 "cpstart" 쉘 명령어를 실행하면 PASENS는 CP로 동작하게 되며, 멀티캐스트 메시지를 사용하여 같은 네트워크에 있는 CP들이 사용하지 않는 그룹 ID를 찾아서 자신의 그룹 ID로 설정한 후에 수집(Collect) 상태로 전이된다. 만약 PASENS가 초기 상태에서 "lpstart" 쉘 명령어를 실행하면 LP로 동작하게 되며, 시뮬레이션에 참여하지 않는 유휴(idle) 상태로 전이된다. 수집 상태의 CP는 "lpcollect [number]" 쉘 명령어를 실행하여, 설정된 그룹 ID로 유휴(idle) 상태의 LP 노드들에게 참여 요청 메시지(Join Request Message)를 멀티캐스팅 한다. CP로부터 그 메시지를 수신한 LP들은 그룹 ID를 설정하고, 참여 확인 메시지(Join Acknowledge Message)를 CP로 전송한 후에 대기(Wait) 상태로 전이된다. 참여 확인 메시지를 수신한 CP는 지정된 시간 안에 최대 'number'개의 LP 수만큼 선착순으로 LP ID를 1부터 순서대로 해당 LP에 전송한 후에 마스터(Mater)로 전이된다. LP ID를 부여받은 LP는 슬레이브(Slave) 상태로 전이되어 CP의 원격 쉘 명령어를 수행하게 되며, LP ID를 부여받지 못한 대기 상태의 LP들은 일정 시간 후에 다시 유휴 상태로 전이된다. 마스터 상태의 CP는 쉘 명령어를 사용하여 슬레이브 상태의 LP들을 원격 동작시켜 병렬 시뮬레이션을 진행한다. "reset" 쉘 명령어는 마스터 상태의 CP 또는 슬레이브 상태의 LP를 각각 수집 상태와 유휴 상태로 되돌린다. 그리고 "cpend"와 "lpend" 쉘 명령어들은 수집 상태의 CP와 유휴 상태의 LP를 각각 초기 상태로 되돌린다.

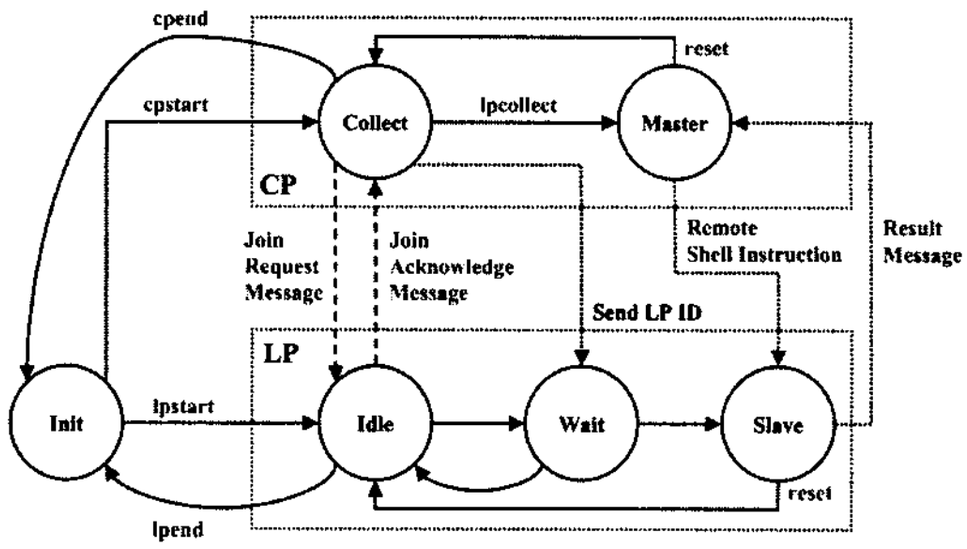


그림 5 PASENS의 프로세스 상태 흐름도

4.3 속도향상 예측

일반적으로 이산-사건 시뮬레이션 방식에서 새로이 스케줄 된 사건을 사건 큐에 삽입할 때는 예정된 사건 발생 시간 순서에 맞추어 정렬시켜야 한다. 기존의 PDES 방법이 적용된 대부분의 응용들은 사건 큐의 평균 길이가 비교적 짧고, 사건의 동작을 처리하는 시간이 사건의 큐를 관리하는데 걸리는 시간보다 매우 길기 때문에, 병렬 시뮬레이션의 성능을 예측[16]할 때 사건 큐를 관리하는데 걸리는 시간은 고려되지 않았다.

그러나 명령어-레벨의 이산-사건 시뮬레이션 방법과 최적-동기식 PDES 방법을 사용하는 본 연구의 시뮬레이션 모델에서는 사건 큐를 관리하는데 걸리는 시간은 시뮬레이션 시간을 예측하는데 중요한 요인이 된다. 명령어-레벨 시뮬레이션에서는 사건의 동작을 처리하는데 걸리는 시간(사건 루틴의 수행 시간)이 매우 짧기 때문에, 센서노드의 수가 많아지면 사건 큐의 길이가 길어져서 새로운 사건을 사건 큐에 삽입하는데 걸리는 시간이 전체 시뮬레이션 시간에 크게 영향을 준다. 즉, 작업부하의 크기가 증가할수록 사건 큐의 길이가 길어져서 한 사건을 사건 큐에 삽입하는데 걸리는 시간은 증가하는 반면에, 각 사건의 동작을 시뮬레이션 하는데 걸리는 시간은 일정하다. 따라서 본 연구의 시뮬레이션 모델에서 병렬 시뮬레이션으로 얻을 수 있는 속도향상을 사건 큐를 관리하는데 걸리는 시간을 고려하여 다음과 같이 예측할 수 있다.

한 컴퓨터를 사용하여 단위 시간까지 시뮬레이션을 진행하는데 걸리는 단일 시뮬레이션 시간 T_s는 식 (1)과 같이 사건 큐를 관리하는데 걸리는 시간 T_{queue}와 사건의 동작을 처리하는데 걸리는 시간 $T_{process}$의 합이 된다. 단위 시간은 사건 큐의 평균 길이의 사건들이 모두 시뮬레이션 되는 가상시간(virtual time: 이하 VT라 함)으로 설정하였다.

$$T_s = T_{queue} + T_{process} \quad (1)$$

사건 큐를 한번 검색하는데 걸리는 시간을 a, 새로운 사건을 발생 시간 순서대로 정렬된 위치에 삽입하기 위

하여 사건 큐를 검색하는 평균 횟수를 s, 단위 시간동안 센서노드 당 시뮬레이션 되는 평균 사건 수를 e, 그리고 시뮬레이션 되는 센서노드의 수를 N이라고 하면, T_{queue}는 다음과 같다.

$$T_{queue} = aseN$$

즉, 단위시간 동안 시뮬레이션 되는 사건의 수가 eN개이며, 이 수는 사건 큐의 평균 길이를 나타낸다. 본 연구의 시뮬레이션 모델에서는 사건 큐의 관리 알고리즘으로 순차 검색(sequential search)에 기반을 두는 삽입 정렬(insertion sort)을 사용하는 *smpl*[12]을 이용한다. 따라서 s는 근사값으로 $\frac{eN}{2}$이 되고, T_{queue}는 다음과 같다.

$$T_{queue} = \frac{ae^2}{2}N^2$$

$T_{process}$는 하나의 사건 동작을 처리하는데 걸리는 평균 시간을 b라고 하면 다음과 같다.

$$T_{process} = beN$$

결과적으로, 한 컴퓨터를 사용하여 단위 시간까지 시뮬레이션을 진행하는데 걸리는 단일 시뮬레이션 시간 T_s는 식 (2)와 같이 정리될 수 있다.

$$T_s = \frac{ae^2}{2}N^2 + beN \quad (2)$$

P개의 LP를 사용하여 병렬 시뮬레이션 하는 경우에는, 각 LP마다 N/P개의 센서노드가 분배된다. 앞 절에서 설명한 바와 같이 최적-동기식 PDES에서 오버헤드 시간은 LP의 수에 영향을 받지 않고 거의 일정하기 때문에, 오버헤드 시간을 v라고 하면 병렬 시뮬레이션 시간 T_p는 식 (3)과 같다.

$$T_p = \frac{ae^2}{2}(\frac{N}{P})^2 + be\frac{N}{P} + v \quad (3)$$

P개의 LP를 사용한 병렬 시뮬레이션에서 얻을 수 있는 속도향상 S_p는 단일 시뮬레이션 속도 T_s와 병렬 시뮬레이션 속도 T_p의 비율로 계산되며, 이 결과를 센서노드의 수 N을 기준으로 정리하면 식 (4)와 같다.

$$S_p = \frac{T_s}{T_p} = \frac{P^2 + \frac{2b}{ae}\frac{P^2}{N}}{1 + \frac{2b}{ae}\frac{P}{N} + \frac{2v}{ae^2}\frac{P^2}{N^2}} \quad (4)$$

식 (4)에서 $\frac{2b}{ae}$와 $\frac{2v}{ae^2}$는 상수이기 때문에, 센서노드의 수 N이 커질수록 속도향상은 식 (5)와 같이 P^2에 근접해간다는 것을 예측할 수 있다.

$$\lim_{N\to\infty} S_p = P^2 \quad (5)$$

일반적으로 병렬 시뮬레이션에서 속도향상이 프로세서의 수보다 커질 수는 없다는 점을 생각하면, 식 (5)에서 예측된 최대 속도향상은 특이한 현상이다. 이러한 속도향상은 센서노드의 수(작업부하의 크기) N에 대해서 사건 큐를 검색하는 시간은 $O(N^2)$으로 증가하지만, P개의 프로세서를 사용하여 병렬 시뮬레이션을 수행하면 작업부하의 크기가 N/P로 감소하여 그 시간이 $O(N^2/P^2)$으로 단축되기 때문이다. 결과적으로, 본 연구의 시뮬레이션 모델에서는 사건 큐를 관리하는데 걸리는 시간이 시뮬레이션 시간에 중요한 요인으로 작용한다는 것을 알 수 있다.

만약 사건 큐의 관리를 위하여 다른 검색 알고리즘을 사용하면 얻을 수 있는 속도향상은 달라질 수 있지만, 어떠한 검색 알고리즘을 사용하더라도 센서노드의 수가 사건 큐의 길이에 영향을 주기 때문에 속도향상은 프로세서의 수보다 커진다. 이러한 속도향상의 가장 큰 이유는 센서노드의 수가 증가할수록 사건 큐의 길이가 길어져서 한 사건을 사건 큐에 삽입하는데 걸리는 시간은 증가하는 반면에, 각 사건의 동작을 시뮬레이션 하는데 걸리는 시간은 일정하기 때문이다. 예를 들어, 평균 검색시간이 매우 빠른 이진 검색(binary search)을 사용할 경우에는 센서노드의 수 N에 대하여 사건 큐를 관리하는데 걸리는 시간이 $O(Nlog_2N)$의 비율로 증가하며, 이 비율은 $O(N)$보다 높다. 따라서 P개의 프로세서를 사용하여 병렬 시뮬레이션을 수행하면, 사건 큐를 관리하는데 걸리는 시간이 $O(\frac{N}{P}log_2\frac{N}{P})$ 비율로 감소하기 때문에 속도향상은 앞에서와 달리 P^2보다는 낮지만 P보다는 훨씬 더 커지게 된다.

5. 실 험

실험에서 시뮬레이션 대상인 센서노드는 MICAz 형식을 가지는 CrossBow 사의 MPR2400 보드[13]로 설정하였으며, 이 센서보드의 마이크로컨트롤러 시스템 클록은 7.37 MHz이다. 센서노드들은 반경 10 m의 3차원 공간에 임의적으로 위치하게 하고 RF 통신의 유효거리를 10 m로 설정함으로써 모든 센서노드들이 서로 통신이 가능하도록 하였으며, RF 충돌(collision)을 제외한 통신 오류는 없는 것으로 가정하였다. 그리고 센서노드의 동작은 시뮬레이션 시작과 동시에 전원이 켜지고 부팅되어 시작되는 것으로 가정하였고, 1초의 VT동안 시뮬레이션을 진행하였다. 실험은 100 Mbps 이더넷으로 연결된 41대의 PC들을 사용하여 진행하였으며, 모든 PC들은 동일한 하드웨어와 운영체제를 가진다. 각 PC의 프로세서는 펜티엄-4 1.7 GHz이고, 주기억장치의 용량은 256 MB이며, 운영체제는 MS 윈도우즈 XP이다. 한 대의 PC에서 동작하는 PASENS는 CP로 동작하고,

나머지 40대의 PC에서는 병렬 시뮬레이션이 시작하기 전까지는 유휴 상태의 LP로 대기하도록 하였다. 즉, 각 PC는 하나의 CP 또는 LP로서 동작한다.

시뮬레이션 작업부하로는 TinyOS[3] 1.1.7에 포함되어 있는 "CntToRfm"과 "RfmToLeds" 예제를 MICAz 형식으로 크로스 컴파일한 실행이미지를 사용하였다. "CntToRfm"은 4 Hz 간격으로 증가된 카운터 값을 RF 통신으로 전송하는 예제이고, "RfmToLeds"는 RF 통신으로 수신된 카운터 값의 하위 세 비트를 LED로 출력하는 예제이다. 두 개 이상의 센서노드를 실험할 경우에는 헤드 센서노드(head sensor node)가 나머지 센서노드들에게 주기적으로 브로드캐스팅 하는 상황과 유사하도록, 한 개의 센서노드에는 "CntToRfm.srec"를 적재하고 나머지 센서노드에는 "RfmToLeds.srec"를 적재하였다. 병렬 시뮬레이션에서는 GVT의 증가 간격은 352 μs로 설정하였고, 가능한 한 같은 수의 센서노드들을 각 LP가 시뮬레이션 하도록 하였다. 만약 균등하게 센서노드들을 분배할 수 없을 경우에는 마지막 LP에 할당되는 센서노드의 수를 조정하였다.

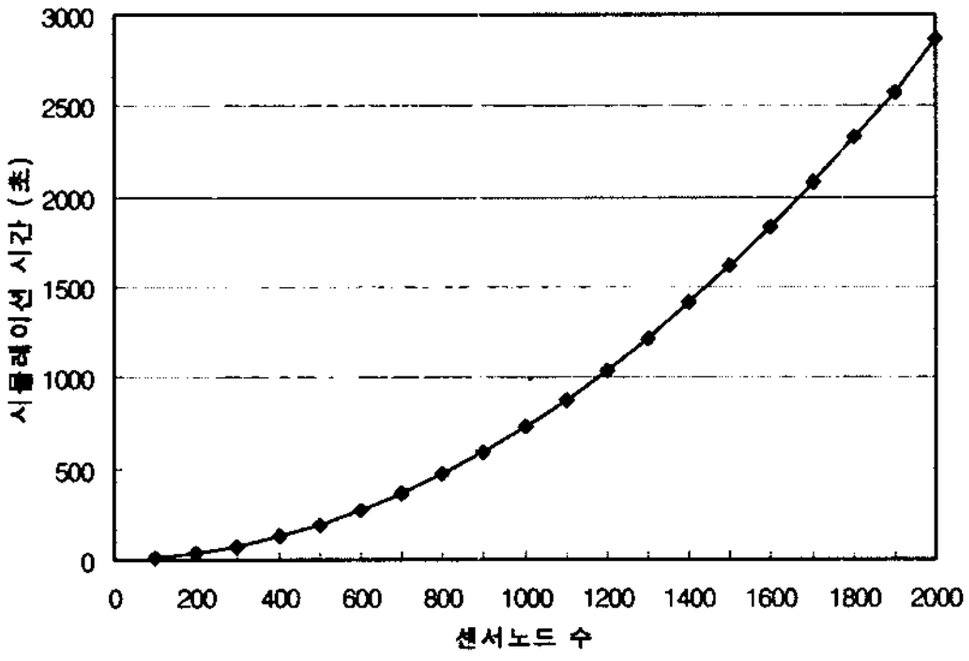
시뮬레이션이 정확히 수행되었는지를 검증하기 위하여 실제 센서보드에 실험에 사용한 예제들의 실행이미지를 적재하여 실행시키고, JTAG(joint test action group) 인터페이스를 사용하여 마이크로컨트롤러와 RF 트랜시버의 레지스터의 값을 시스템 클록 단위로 검침하였다. 검침된 값들은 PASENS의 시뮬레이션 결과값과 시스템 클록 단위로 정확히 일치한다는 것을 확인하였으며, LCD나 RF 통신도 정상적으로 동작한다는 것을 확인하였다.

5.1 단일 시뮬레이션에서 작업부하 크기의 영향

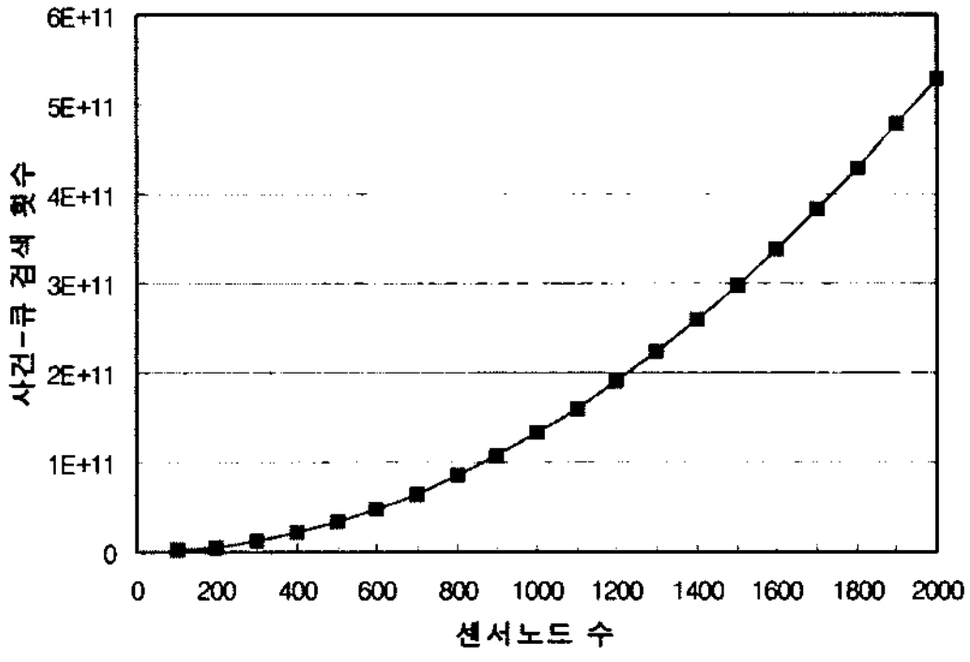
WSN 시뮬레이션에서 작업부하의 크기는 센서 네트워크의 크기, 즉 센서노드의 수에 비례한다. 실험 1은 한 대의 컴퓨터에서 시뮬레이션을 진행할 때에 작업부하의 크기가 시뮬레이션 시간에 미치는 영향을 확인하기 위하여, 시뮬레이션 되는 센서노드 수를 100개부터 2,000개까지 증가시키면서 1초의 VT동안 진행하였다. 그림 6(a)의 결과에서 보면, 센서노드 수가 증가함에 따라 시뮬레이션 시간이 급속히 길어지는 것으로 나타났다. 예를 들어, 시뮬레이션 되는 센서노드 수가 200개일 경우에는 소요된 시뮬레이션 시간이 32초이지만, 센서노드의 수가 10배 증가한 2,000개일 경우에는 시뮬레이션 시간이 2,864초가 되어 89배 증가하였다. 즉, 한 컴퓨터를 사용한 경우에 시뮬레이션 시간을 나타내는 식 (2)에서 예측한 것과 같이, 센서노드의 수가 N으로 증가하면 시뮬레이션 시간은 $O(N^2)$의 비율로 길어졌다.

명령어-레벨의 이산-사건 시뮬레이션에서는 사건 루틴을 처리하는 시간보다 그 사건을 사건-큐에 발생할 시간 순서대로 삽입하기 위하여 사건 큐를 검색하는 시간이 더 길다. 따라서 센서노드의 수가 많아질수록 그 동작들을 시뮬레이션 하기 위하여 스케줄 되는 사건들의 수가 많아지고, 그 사건들을 사건 큐에 삽입하기 위하여 사건 큐를 검색하는 시간이 점점 더 길어진다. 즉, 센서노드의 수 N에 대해서 사건의 동작을 시뮬레이션 하는데 걸리는 시간은 $O(N)$의 비율로 증가하지만, 새로이 발생되는 사건을 사건 큐에 삽입하기 위하여 사건 큐를 관리하는데 걸리는 시간은 $O(N^2)$의 비율로 증가한다.

그림 6(b)는 실험 1에서 사건 큐의 검색 횟수를 측정한 결과를 보여주고 있는데, 센서노드의 수가 N배 증가하면 사건 큐의 검색 횟수가 $O(N^2)$배 많아진다는 것을 확인할 수 있다. 예를 들어, 센서노드 수가 200개일 경우에는 사건 큐의 검색 횟수가 5.3×10^9이지만, 센서노드의 수가 10배 증가한 2,000개일 경우에는 사건 큐의 검색 횟수가 5.3×10^{11}이 되어 100배 증가하였다. 이 결과는 명령어-레벨 시뮬레이션 방법을 사용할 경우에 사건을 처리하는 시간보다는 새로운 사건을 사건 큐에 삽입



(a) 시뮬레이션 시간



(b) 사건-큐의 검색횟수

그림 6 단일 시뮬레이션에서 작업부하 크기의 영향

할 때 걸리는 시간이 전체 시뮬레이션 시간에 큰 영향을 미친다는 것을 입증해주고 있다.

5.2 LP 수에 따른 속도향상

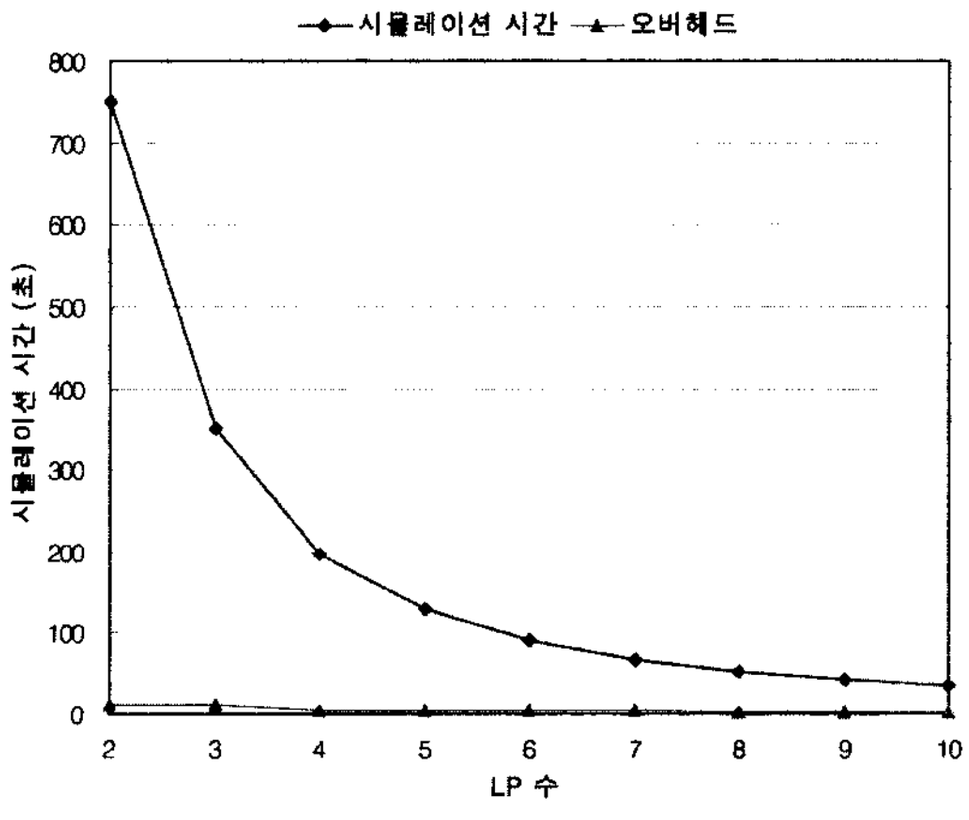
실험 2는 병렬 시뮬레이션에서 LP의 수(PC의 수)에 따른 속도향상을 확인하기 위하여 센서노드의 수를 2,000개로 고정하고, 시뮬레이션에 참여하는 LP의 수를 2개부터 10개까지 증가시키면서 진행하였다. 그림 7(a)의 결과에서 보면, LP의 수가 증가함에 따라 시뮬레이션 시간이 크게 단축되었고 병렬 시뮬레이션을 위해 부가적으로 소요된 오버헤드 시간은 거의 일정하다는 것을 알 수 있었다. 예를 들어, LP의 수가 두 개일 때 소요된 시뮬레이션 시간은 750초로서, 단일 시뮬레이션의 2,864초보다 크게 단축되었다. 그리고 LP의 수가 10개로 증가하면 시뮬레이션 시간이 36초까지 단축되어서, LP의 수가 많아질수록 병렬 시뮬레이션 효과가 매우 높아진다는 것을 알 수 있었다.

실험결과에서 LP의 수가 증가하더라도 오버헤드 시간이 길어지지 않는 이유는 오버헤드 시간의 대부분을 차지하는 LP들 간의 통신에 필요한 네트워크 대역폭이 LP의 수가 증가할수록 커지지만, 이 대역폭이 실험환경의 네트워크 대역폭보다 작기 때문이다. 그리고 LP가 시뮬레이션 하는 센서노드의 수가 많아질수록 전체 시뮬레이션 시간에서 오버헤드 시간이 차지하는 비율이 매우 낮아지기 때문에, 센서노드의 수가 많은 경우에 오버헤드 시간은 전체 시뮬레이션 시간에 거의 영향을 주지 않는다.

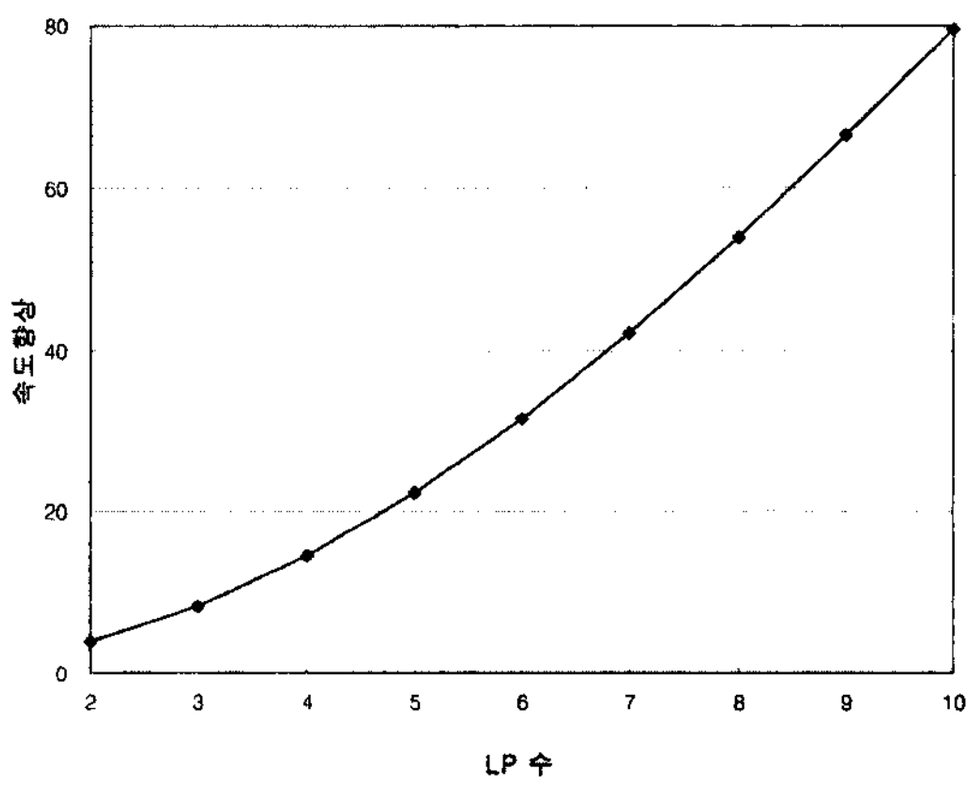
그림 7(b)는 실험 2의 결과를 속도향상으로 나타낸 것이다. 결과에 따르면, LP의 수가 두 개일 때와 10개일 때 속도향상은 각각 3.8배와 80배로 나타났다. 일반적으로 병렬 시뮬레이션에서 속도향상이 프로세서의 수(LP의 수)보다 커질 수 없다는 점을 생각하면, 이러한 속도향상 결과는 특이한 현상이다. 이 결과는 사건 큐의 처리 시간이 시뮬레이션 시간에서 중요한 요인으로 작용하기 때문이다. 즉, 병렬 시뮬레이션은 작업부하를 분할하는 것과 동시에 사건 큐도 분할하여 높은 속도향상을 제공한다. 만약 N개의 센서노드들을 P개의 LP를 사용하여 병렬 시뮬레이션을 수행하면 각 LP가 시뮬레이션 하는 센서노드의 수가 N/P로 감소하기 때문에, $O(N^2)$ 비율로 증가한 사건 큐를 검색하는 시간이 $O(N^2/P^2)$ 비율로 감소한다.

5.3 작업부하 크기에 따른 속도향상

실험 3은 병렬 시뮬레이션으로 얻을 수 있는 속도향상을 예측한 식 (5)를 검증하기 위하여, LP의 수를 2개와 10개로 각각 고정시키고, 센서노드의 수를 100개부터 2,000개까지 증가시키면서 시뮬레이션을 진행하였다. 그림 8의 결과에서 보면, 센서노드의 수가 많아질수록 속도향상은 병렬 시뮬레이션에 참여하는 LP 수의 제곱에 접근하는 것으로 나타났다. 예를 들어, LP의 수가 두 개인 경우에는 센서노드의 수가 600개 이상이 되면 속도향상이 4배에 근접하게 수렴하는 것을 그림 8(a)에서 볼 수 있다. 이 경우에 각 LP가 시뮬레이션 하는 센서노드의 수가 300개보다 많아지면 속도향상이 최대로 나타났다. 또한 LP의 수가 10개인 경우에는 속도향상이 100배에 접근해가는 것을 그림 8(b)에서 확인할 수 있다. 이 결과는 본 연구의 시뮬레이션 모델에서 시뮬레이션 되는 센서노드의 수가 많아질수록, 병렬 시뮬레이션으로 얻을 수 있는 속도향상이 LP 수의 제곱에 가까워진다는 것을 예측한 식 (5)가 타당하다는 것을 나타낸다.

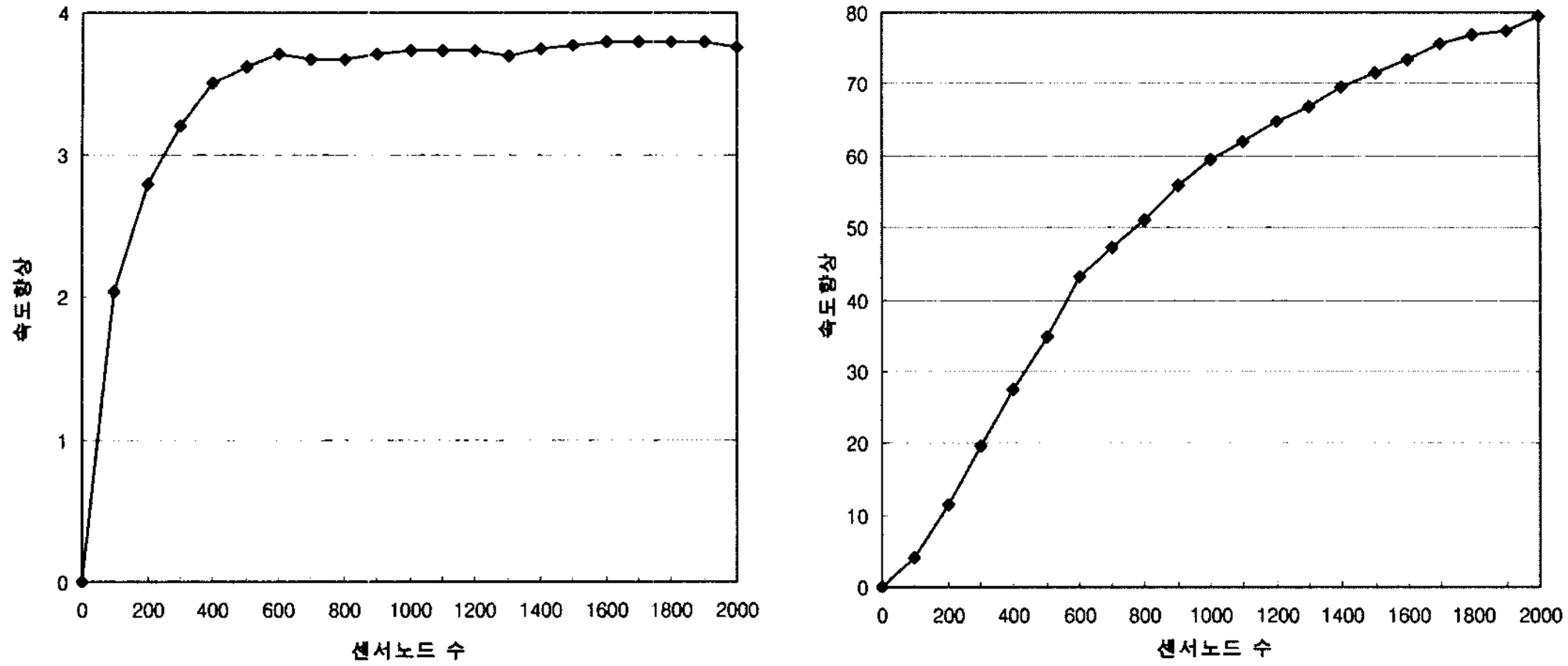


(a) 시뮬레이션 시간



(b) 속도 향상

그림 7 병렬 시뮬레이션에서 LP 수의 영향



(a) LP 수 = 2 (b) LP 수 = 10

그림 8 센서노드 수에 따른 속도향상

5.4 LP 수의 확장성

실험 4는 LP 당 동일한 작업부하에서 LP 수의 확장성을 분석하기 위하여, LP당 시뮬레이션 되는 센서노드의 수를 50개, 250개, 500개로 각각 고정하고, LP의 수를 5개부터 40개까지 증가시키면서 시뮬레이션을 진행하였다. 즉, 시뮬레이션 되는 전체 센서노드의 수는 250개부터 20,000개까지이다. 그림 9의 결과에서 보면, LP 당 동일한 작업부하를 가지고 있을 경우에는 LP 수가 전체 시뮬레이션 시간에 영향을 주지 않는다는 것을 알 수 있다. 이 결과는 LP의 수가 증가할 경우에 배리어 동기화와 RF 전송을 위한 통신 메시지의 수는 증가하지만, 통신량이 네트워크 대역폭보다 적어서 통신 지연이 발생하지 않기 때문이다. 또한 전체 시뮬레이션 시간이 오래 걸리는 경우에는 1초의 시뮬레이션 시간동안 요구되는 네트워크 대역폭은 더 감소한다. 예를 들어, 전체 센서노드의 수가 20,000개인 경우에 시뮬레이션 시간이 약 200초이기 때문에, 1초의 시뮬레이션 시간에 요구되는 네트워크 대역폭은 1/200로 감소한다.

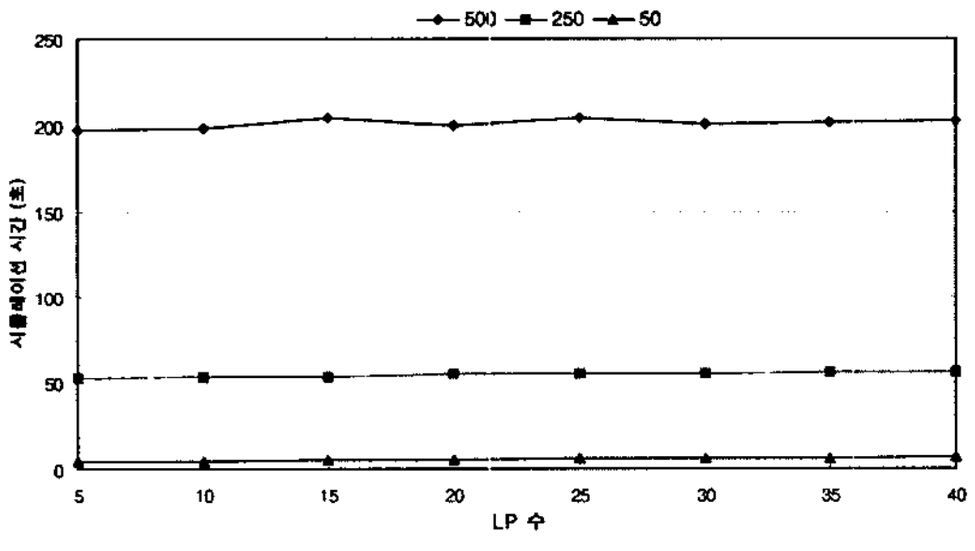


그림 9 LP 수의 확장성

5.5 다른 WSN 시뮬레이터들과의 비교

PASENS는 센서 노드에 실제로 적재되는 실행이미지를 작업부하로 사용하는 병렬 센서 네트워크 시뮬레이터로는 최초이기 때문에, 다른 WSN 연구와 성능을 직접적으로 비교하기가 어렵다. 따라서 병렬 시뮬레이션을 하지는 않지만 유사한 실행이미지 형식을 작업부하로 사용하는 연구들인 TOSSIM[2], Avrora[4], ATEMU[11]의 실험결과와 본 연구의 결과를 비교하기 위하여 Avrora[4]의 실험과 유사한 환경에서 실험 5를 진행하였다.

Avrora의 실험에서는 3.06 GHz의 듀얼 Xeon 프로세서와 4 GB의 주기억장치, 그리고 리눅스 2.4.20 운영체제의 컴퓨터를 사용한 반면에, 본 실험에서는 2.8 GHz의 펜티엄-4 프로세서와 1 GB의 주기억장치, 그리고 MS 윈도우즈 XP 운영체제의 컴퓨터를 사용하였다. 그리고 센서노드의 수를 1개부터 1,024개까지 증가시키면서, 절반의 센서노드들에는 “CntToRfm”를, 나머지 절반의 센서노드들에는 “RfmToLeds”를 나누어 적재시키고 10초의 VT동안 시뮬레이션 하였다. 그림 10은 PASENS의 실험 결과를 Avrora[4]에서 제시한 실험 결과와 함께 보여주는 것으로서, PASENS-1은 단일 시뮬레이션 결과를 나타내고 PASENS-8은 8개의 LP를 사용하여 병렬 시뮬레이션을 한 결과이다.

그림 10의 결과에서 보면 PASENS-1의 시뮬레이션 시간은 ATEMU[11]보다는 짧지만 Avrora[4]나 TOSSIM[2]에 비해서는 더 길다는 것을 알 수 있다. 시뮬레이션 정밀도가 낮은 TOSSIM은 TinyOS의 컴포넌트 단위로 이산-사건 시뮬레이션을 수행하기 때문에 시뮬레이션 시간이 가장 짧고, ATEMU는 슬립 상태에서도 시스템 클록 단위로 시뮬레이션을 진행하기 때문에 시

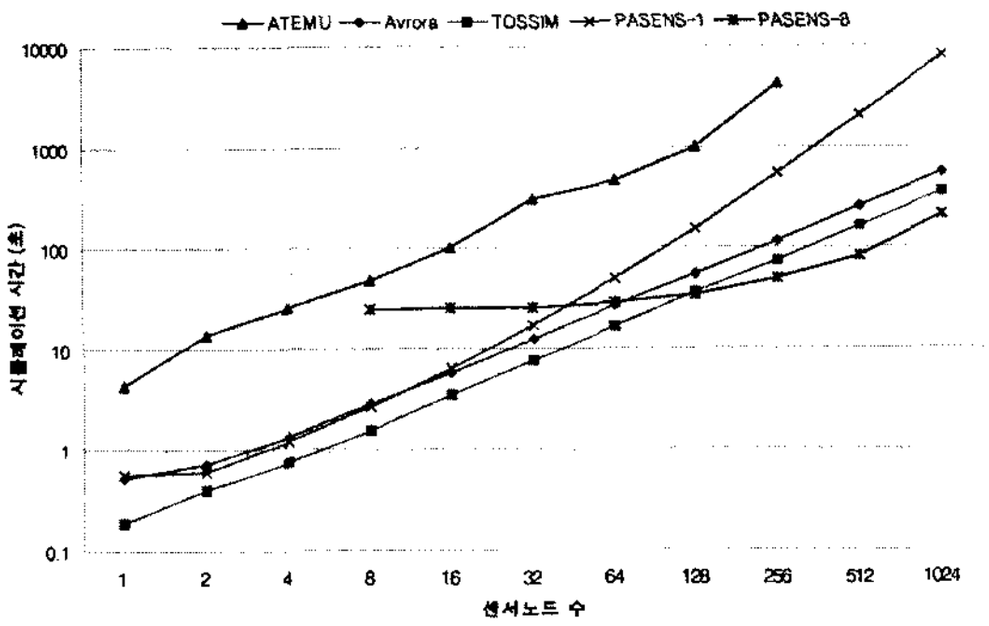


그림 10 PASENS와 다른 WSN 시뮬레이터들과의 확장성 비교

뮬레이션 시간이 가장 길다. 그리고 Avrora가 PASENS-1에 비하여 시뮬레이션 시간이 짧은 이유는, 센서노드를 하나의 쓰레드로 동작시키는 Avrora를 다중-쓰레드 동작이 용이한 하이퍼-쓰레드 기술(hyper-threading technology)이 적용된 듀얼 Xeon 프로세서를 사용하여 실험하였기 때문이다. 즉, Avrora는 4개의 프로세서에서 실험한 결과인 반면에, PASENS-1은 한 프로세서에서 실험한 결과라고 볼 수 있다. 8개의 LP를 사용한 PASENS-8은 센서노드의 수가 적을 때에는 병렬 시뮬레이션의 오버헤드로 인하여 TOSSIM이나 Avrora에 비하여 시뮬레이션 시간이 길었다. 그러나 센서노드의 수가 128개 이상으로 증가하면 PASENS-8이 가장 짧은 시뮬레이션 시간을 나타내었고, 센서노드의 수가 1024개일 경우에는 PASENS-8의 시뮬레이션 시간이 TOSSIM에 비해서는 1.6배, 그리고 Avrora에 비해서는 2.6배 빠른 결과를 얻을 수 있었다.

6. 결 론

본 연구에서는 대규모 WSN을 시뮬레이션 하기 위하여 명령어-레벨의 이산-사건 시뮬레이션 방법과 최적-동기식 PDES 방법을 사용하여 병렬 시뮬레이터인 PASENS를 개발하였다. 명령어-레벨 시뮬레이션은 센서노드의 시스템 클록 수준으로 시뮬레이션의 정밀도를 높임으로써, 센서노드의 동작과 실행시간 및 전력소모량을 실제와 거의 근접한 수준으로 예측할 수 있게 한다. 또한 이 방법은 실제 센서노드에 적재되는 실행이미지를 시뮬레이션 작업부하로 사용하여 기존의 많은 WSN 시뮬레이터들이 가진 소프트웨어 의존성 문제를 해결하였다. 그러나 명령어-레벨 시뮬레이션은 정밀도가 높기 때문에 시뮬레이션 시간이 현저히 길어져서 시뮬레이션 할 수 있는 센서노드의 수에 제한이 있다는 문제점이 있다. 최적-동기식 PDES 방법은 그러한 문제점을 해결하여 시뮬레이션 시간을 크게 단축시킴으로써 대규모의 WSN을 시뮬레이션 할 수 있게 한다. PASENS를 사용하여 실험한 결과에 따르면, 시뮬레이션 되는 센서노드의 수가 많은 경우에는 병렬 시뮬레이션에 참여하는 컴퓨터 수의 제곱에 접근하는 속도향상을 얻을 수 있다는 것을 확인하였다. 이 결과는 본 연구의 시뮬레이션 모델에서 병렬 시뮬레이션으로 얻을 수 있는 속도향상을 예측한 수식이 타당하다는 것을 나타낸다.

일반적으로 병렬 시뮬레이션에서 속도향상이 프로세서의 수보다 커질 수는 없다는 점을 생각하면, 본 연구의 속도향상 결과는 특이한 현상이다. 이러한 현상이 발생하는 이유는 센서노드의 수가 많아질수록 그 동작들을 시뮬레이션 하기 위하여 스케줄 되는 사건들의 수가 많아지고, 그 사건들을 사건 큐에 시간 순서대로 삽입하기 위하여 사건 큐를 검색하는 시간이 점점 더 길어지기 때문이다. 다시 표현하면, 작업부하의 크기가 증가할수록 사건 큐의 길이가 길어져서 한 사건을 사건 큐에 삽입하는데 걸리는 시간은 증가하는 반면에, 각 사건의 동작을 시뮬레이션 하는데 걸리는 시간은 일정하다. 결과적으로, 작업부하의 크기 N에 대해서 사건의 동작을 시뮬레이션 하는데 걸리는 시간은 $O(N)$의 비율로 증가하지만, 사건 큐를 관리하는데 걸리는 시간은 $O(N^2)$의 비율로 증가한다. 그러나 P개의 프로세서를 사용하여 병렬 시뮬레이션을 수행하면 작업부하의 크기가 N/P로 감소하기 때문에, 사건 큐를 검색하는 시간은 $O(N^2/P^2)$ 비율로 감소한다. 결과적으로 병렬 시뮬레이션을 이용하면 속도향상이 프로세서의 수보다 커질 수 있는 것이다. 예를 들어, 센서노드의 수가 2000개인 경우의 실험 결과에 따르면, LP의 수가 두 개일 때와 10개일 때 속도향상은 각각 3.8배 및 80배로서, P^2에 접근하는 것으로 나타났다.

본 연구의 시뮬레이션 모델에서는 사건 큐의 관리 알고리즘으로 순차 검색에 기반을 두는 삽입 정렬을 사용하기 때문에 높은 속도향상을 얻을 수 있었다. 만약 다른 검색 알고리즘을 사용하면 얻을 수 있는 속도향상은 달라질 수 있지만, 어떠한 검색 알고리즘을 사용하더라도 사건 큐의 길이에는 센서노드의 수가 영향을 미치기 때문에 속도향상은 프로세서의 수보다 커진다.

실제 WSN에서는 센서노드의 수가 수백 혹은 수천 개가 되는 경우도 있지만, 그러한 대규모의 WSN을 단일 컴퓨터에서 시뮬레이션 하는 경우에는 기억장치 용량의 한계와 긴 시뮬레이션 시간 때문에 시뮬레이션 할 수 있는 센서노드의 수에 한계가 있다. 최적-동기식 PDES 방법을 이용하면 병렬 시뮬레이션에 참여하는 컴퓨터의 수가 증가하더라도 오버헤드는 증가하지 않기 때문에, 컴퓨터의 수가 충분하다면 시뮬레이션 할 수 있는 센서노드의 수에는 한계가 없게 된다. 본 연구의 실험에서는 20,000개의 센서노드까지 시뮬레이션을 하였지

만, 더 많은 수의 센서노드들의 시뮬레이션도 가능하다. 특히, 명령어-레벨 시뮬레이션과 같이 사건을 처리하는 시간이 짧은 경우에는 병렬 시뮬레이션에 참여하는 컴퓨터 수보다 높은 속도향상 효과를 기대할 수 있다. 또한 이 방법은 이더넷에 연결된 PC들을 사용하기 때문에, 병렬 시뮬레이션 환경을 구축하기가 쉽고 비용이 저렴하다는 장점도 있다. PASENS 프로그램의 실행 파일과 프로그램 소스는 'http://caeagle.yonsei.ac. kr/pasens/'에서 제공된다.

참 고 문 헌

[1] 김방현, 김태규, 정용덕, 김종현, "전력소모량 및 실행시간 추정이 가능한 센서 네트워크 시뮬레이터의 개발", 한국시뮬레이션학회 논문지, 15(1): 35-42, 2006년 3월.

[2] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," In Proceedings of 1st ACM Conference on Embedded Networked Sensor Systems, 2003.

[3] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D.E. Culler, and K.S. Pister, "System Architecture Directions for Networked Sensors," In Proceedings of International Conference or. Architectural Support for Programming Languages and Operating Systems, 2000.

[4] B.L. Titzer, D.K. Lee, and J. Palsberg, "Avrora: Scalable Sensor Netwcrk Simulation with Precise Timing," In Proceedings of The 4th IEEE International Symposium on Information Processing in Sensor Networks, 2005.

[5] A. Ferscha, Parallel and Distributed Simulation of Discrete Event Systems, In Handbook of Parallel and Distributed Computing, McGraw-Hill, 1995.

[6] J.K. Peacock, J.W. Wong, and E.G. Manning, "Distributed Simulation using a Network of Processors," Computer Networks, 3(1):44-56, 1979.

[7] D.A. Jeerson. "Virtual Time," ACM Transactions on Programming Languages and Systems, 7(3): 404-425, July 1985.

[8] The NEST Project, http://webs.cs.berkeley.edu/.

[9] G. Simon, P. Volgyesi, M. Maroti, and A. Ledeczi, "Simulation-based Optimization of Communication Protocols for Large-scale Wireless Sensor Networks," In Proceedings of the IEEE Aerospace Conference, March 2003.

[10] A. Ledeczi, M. Maroti, and I. Bartok, "Simple Nest Application Simulator," Draft, Institute for Software Integrated Systems, Vanderbilt University, October 2001.

[11] J. Polley, D. Blazakis, J. McGee, D. Rusk, J.S. Baras, and M. Karir, "ATEMU: A Fine-grained Sensor Network Simulator," In Proceedings of The First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004.

[12] M. H. MacDougall, Simulating Computer Systems: Techniques and Tools, MIT Press, July 1987.

[13] CrossBow, MPR/MIB Users Manual, April 2005.

[14] Atmel, ATmega128(L) Complete, March 2006.

[15] Chipcon, SmartRF CC2420 Preliminary Datasheet 1.2, June 2004.

[16] J. Xu and M.J. Chung, "Predicting the Performance of Synchronous Discrete Event Simulation," IEEE Transaction on Parallel and Distributed Systems, 15(12):1130-1137, Dec. 2004.

[17] J. Xu and J. Zhang, "Efficiently Unifying Parallel Simulation Techniques," Proceedings of the 44th Annual Southeast Regional Conference ACM-SE'06, March, 2006.

[18] K.M. Chandy and J. Misra, "Distributed Simulation: A Case Study in Design and Verification of Distributed Programs," IEEE Transactions on Software Engineering, SE-5(5):440-452, September 1979.

[19] R.E. Bryant, "A Switch-Level Model and Simulator for MOS Digital Systems," IEEE Transactions on Computers, C-33(2):160-177, February 1984.

[20] J. Misra, "Distributed Discrete-Event Simulation," ACM Computing Surveys, 18(1):39-65, March 1986.

김 방 현

1996년 연세대학교 전산학과 이학사. 2001년 연세대학교 전산학과 이학석사. 2007년 연세대학교 컴퓨터정보통신공학부 이학박사. 2008년~현재 한국해양연구원 연수연구원. 관심분야는 병렬처리, 시뮬레이션, 센서 네트워크, 임베디드 시스템

김 종 현

1976년 연세대학교 전기공학과 공학사 1981년 연세대학교 전기공학과 공학석사 1988년 Arizona State University 전기 및 컴퓨터공학과 공학박사. 1976년~1982년 국방과학연구소 연구원. 1988년~1990년 한국전자통신연구소 실장 역임. 1990년~현재 연세대학교 컴퓨터정보통신공학부 교수. 1996년 Oregon State University 전산학과 방문교수. 2003년 Florida State University 전산학과 방문교수. 관심분야는 컴퓨터구조, 병렬처리, 시뮬레이션, 센서 네트워크