한국시뮬레이션학회 논문지
Vol. 17, No. 4, pp. 113-122 (2008. 12)

Ptolemy Tool을 지원하는 무선 센서네트워크 Actor의 설계 및 구현

안기진[1] · 주현철[2] · 오형래[2] · 김영덕[3†] · 양연모[4] · 송황준[2]

Design and Implementation of Sensor Network Actors Supporting Ptolemy Tool

Ki Jin An · Hyunchul Joo · Hyung Rai Oh · Young Duk Kim · Yeon Mo Yang · Hwangjun Song

ABSTRACT

Emerging of wireless sensor networks results in several modeling and design issues on the network simulations. In previous works, most researchers have used three evaluation approaches such as analysis method, computer simulation and real test-bed measurement in order to verify the performance of wireless sensor networks. Among these approaches, analysis method is widely used since the other approaches have significant drawbacks such as limitation of network power, computational problem of distribute processing and complication of debugging process. However, the analysis method also shows poor performance when it deals with complex operations in huge wireless sensor networks. Thus, in this paper, we design and develop SMAC and AODV protocols by using Ptolemy tool in order to improve the simulation performance. The developed Ptolemy actor can be easily adapted to compare and evaluate the various protocols for wireless sensor networks.

Key words : Ptolemy, Wireless sensor networks, Simulator

요 약

최근, 무선 센서 네트워크의 출현은 네트워크 디자인에 많은 이슈를 가져왔다. 전통적으로, 유/무선 네트워크의 성능 측정을 위해서 분석적 접근, 컴퓨터 시뮬레이션과 실제 테스트베드상의 측정 3가지 방식을 이용하였다. 그렇지만 네트워크의 파워 제한, 분산된 처리와 장애 처리등과 같은 제한 때문에 알고리즘이 복잡하여 일반적으로 분석적 접근을 이용하여 성능을 측정하였다. 하지만 이러한 분석적 접근은 센서 네트워크의 복잡한 동작환경에서 성능측정에 한계를 가지고 있다. 이러한 제약을 극복하기 위해 본 논문에서는 Ptolemy Tool를 이용하여 효과적인 무선 센서 네트워크 시뮬레이션의 성능평가를 위하여 MAC계층의 SMAC와 네트워크계층의 AODV 를 개발하고 실험하였다. 구현된 Actor는 센서네트워크내 다양한 프로토콜들의 성능비교에 용이하게 활용될 수 있다.

주요어 : Ptolemy, 무선 센서 네트워크, 시뮬레이터

1. 서 론

우리나라에서는 정보통신 기술 수준에 대한 제2의 도약과 관련 산업의 경기활성화 및 해외수출 시장의 확대 개척을 위해 정부 중심으로 IT839전략이 추진되었다. IT839 전략은 8대 신규 서비스와 3대 첨단인프라, 9대 신 성장 동력을 뜻하며 서비스를 창출, 인프라 구축, 기술개발을 상호 보완적으로 연계하여 미래 성장 동력을 창출하기 위함이다. 3대 인프라는 광대역 통합망 (BcN), u-센서네트워크(USN), IPv6를 의미하며, 8대 서비스 및 9대 신성장 동력에 대해 직, 간접적으로 기술적/기능적인 실행을 지원하고 미래에 출현할 새로운 정보통신 창출을 돕는 역할을 한다. 이중에서 u-센서네트워크는 많은 수의 센서 노드들로 구성되며, 각 센서 노드의 하드웨어는 매우 작은 규모이다. 또한 센서 네트워크는 구축 목적에 따라 네트

* 본 연구는 교육과학기술부에서 지원하는 대구경북과학기술연구원 기관고유사업비로 수행되었음.
2008년 10월 23일 접수, 2008년 12월 9일 채택
[1] 포항공과대학교 정보통신학과
[2] 포항공과대학교 컴퓨터공학과
[3] 대구경북과학기술연구원 미래산업융합기술연구부
[4] 금오공과대학교 전자공학부
주 저 자 : 안기진
교신저자 : 김영덕
E-mail; ydkim@dgist.ac.kr

워크 토폴로지 및 라우팅 방식이 결정되어야 하고, 이와 더불어 센서 노드의 하드웨어와 소프트웨어도 필요에 따라 다양하게 변경되어야 한다. 따라서 센서 네트워크가 구현되기 전에 시스템 동작과 성능을 예측할 수 있는 센서 네트워크 시뮬레이터가 필요하다. 그러므로 본 논문에서는 JAVA기반의 Ptolemy Tool[1]을 이용하여 GUI기반의 센서네트워크 시뮬레이터를 개발하고자 한다.

본 논문에서 수행한 연구의 구체적인 범위는 Ptolemy 상에서 WSN에 필요한 각 Actor의 콤포넌트를 설계 및 구현하는 것이다. MAC계층에서는 대표적인 프로토콜로써 SMAC을 구현하였으며, 네트워크계층에서는 라우팅 프로토콜로써 AODV 프로토콜을 구현하였다.

본 논문을 통하여 얻을 수 있는 결과는 무선 센서 네트워크상에서 기존의 Ptolemy Tool을 적용하여 초기 설계 단계에서부터 시뮬레이션 단계, 마지막으로 소스코드 생성단계까지 하나의 Tool을 이용하여 완성할 수 있음이다. 또한 무선 센서 네트워크 및 애드혹 네트워크에서 범용적으로 사용되는 MAC, Routing 프로토콜을 제공함으로써 무선 통신을 위한 개발 플랫폼을 제공하게 되며, HILS (Hardware loop in the simulation) 개념에 따라 개발 전 단계에서 오류 수정을 용이하게 한다. 특히, Ptolemy를 이용한 센서네트워크 시뮬레이터의 선행연구는 아직까지 미진하며, Ptolemy를 개발한 UCB에서도 하이브리드 시뮬레이션 Tool에 대한 선행연구로써 본 연구 결과를 유용하게 활용할 수 있으리라 예상된다.

2. 관련 연구

2.1 Ptolemy Tool

Ptolemy Tool은 U.C. Berkeley의 Chess(Center for Hybrid and Embedded Software Systems) 프로젝의 일환으로 개발된 오픈소스기반의 임베디드 소프트웨어 모델링 툴이다. 현재는 Ptolemy II라는 이름으로 발전되었으며, 이기종 시스템의 모델링, 시뮬레이션, 그리고 병렬 시스템의 디자인에 널리 활용되고 있다. 특히, Actor 중심의 디자인을 지원함으로써, 모델내의 다른 Actor들과 효과적으로 통신을 할 수 있으며, 연산의 효율을 높인다. 이는 기존의 객체 지향 디자인과는 달리, 콤퍼넌트 사이의 동시수행성과 통신 기능을 향상시켰음을 의미한다. 그림 1은 GUI기반의 Ptolemy Tool을 이용한 Actor의 디자인 예를 보인다.

그림 1에서 Actor는 일반적인 객체와 마찬가지로 컴포넌트 인터페이스를 가지며, 이를 통하여 내부의 상태를 추

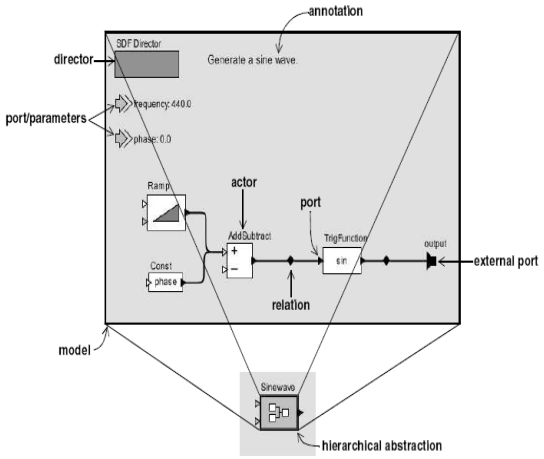


그림 1. Ptolemy Tool의 Actor 디자인

상화 하고, Actor의 동작을 기술하게 된다. 또한 외부의 다른 Actor와 통신하는 방법을 제공함으로써 이질적인 환경에서도 쉽게 모델링을 지원할 수 있도록 한다. 더불어, 인터페이스는 통신의 시작점을 나타내는 포트(port)와 Actor의 동작을 명세하는 파라미터를 포함한다. 이러한 컴포넌트의 상호작용을 디자인하고 모델링함으로써 실행가능한 모델을 빠르게 도출할 수 있다.

2.2 Viptos

Viptos(Visual Ptolemy and TinyOS)[2]는 TinyOS 기반의 무선 센서 네트워크를 위하여 통합된 시뮬레이션 개발 환경을 제공한다. 모든 모델링 과정이 GUI(Graphical User Interface)를 기반으로 이루어지며, 사용자는 실제 TinyOS 프로그램의 인터럽트 레벨의 실험을 할 수 있다. 즉, 사용자는 Ptolemy상의 다른 계산 모델들을 사용하면서 패킷 레벨의 네트워크 시뮬레이션을 동시에 운용할 수 있으며, 이는 상위 계층의 프로그램을 손쉽게 물리계층으로 시뮬레이션을 진행할 수 있음을 의미한다.

또한 Ptolemy 및 Viptos로 구현된 Actor들은 간단한 컴파일 과정을 통하여 TinyOS에 사용 가능한 NesC 코드로 생성할 수 있다는 장점이 있으며, TinyOS, TOSSIM과 더불어 기존의 상용 프로그램과는 달리 오픈소스기반으로 널리 활용될 수 있다는 장점을 가진다.

2.3 국내외 시뮬레이터 연구

국내에서는 MISS (Machine Instruction-level Sensor Network)[3]라는 특정 운영체제에 의존하지 않으면서 전력소모량과 실행시간 추정기능도 포함하는, 기계명령어-

레벨의 센서 네트워크 시뮬레이터를 구현하였다. 특히 시뮬레이터 구현에 있어서 이산-사건 시뮬레이션 기법을 이용함으로써 센서 노드 내부 모듈간 및 노드들 간의 시간 동기화가 가능하게 하였고, 기계명령어-레벨의 시뮬레이션 작업부하를 사용하여 시뮬레이션의 정밀도를 높이는 동시에 전력소모량과 실행시간 추정이 가능하게 하였다.

국외의 연구로서는 U.C. Berkeley 대학 중심의 NEST (Network Embedded Software Technology) 프로젝트에서 개발된 TOSSIM(TinyOS Simulator)[4], Prowler(Probabilistic wireless network simulator)[5], Siesta(Simple NEST application simulator)[6], Ashut(Acoustic simulator for urban terrain)[7], 그리고 Rmase(Routing modeling application simulation environment) 등이 있으며, 이들은 NEST 프로젝트에서 개발된 TinyOS[8]가 탑재되어 있는 하드웨어 플랫폼 (mote 또는 센서노드)기반에서 동작하는 시뮬레이터들이다. TOSSIM[4]은 센서 노드에 탑재된 TinyOS와 응용프로그램의 동작과 상호작용을 시뮬레이션 하는 운영체제 시뮬레이터이며, Prowler는 무선 네트워크 알고리즘을 평가할 수 있는 네트워크 시뮬레이터이고, Siesta는 NEST 응용 프로그램 및 미들웨어의 기능을 검증할 수 있는 미들웨어 시뮬레이터이다. 그리고 Ashut는 주어진 환경에서 직접 전달되는 음향과 반사 전달되는 음향의 도착 시간을 계산할 수 있는 시뮬레이터이고, Rmase는 네트워크 토폴로지 및 응용 시나리오를 생성하여 센서 네트워크의 라우팅 기능을 평가하는 Prowler 기반의 라우팅 시뮬레이터이다. 그러나 Berkley에서 제작한 이와 같은 시뮬레이터들은 TinyOS상에서만 동작하며, 전력소모량 및 실행시간에 대한 부분은 고려되지 않았다.

JSIM [9]은 West Bohemia 대학에서 개발한 네트워크 시뮬레이터로써 JAVA 기반으로 제작되었고. Discrete Time Process-Oriented Simulation을 위한 Object-Oriented 라이브러리를 지원하며, Best Effort, Integrated Service 및 DiffServ 프로토콜을 구현하였다. 한편, GloMoSim은 UCLA에서 제작하였으며 OSI 7 Layer의 구조와 기능적으로 비슷하게 구현되어 있으며, 각 레이어별로 API를 이용하여 제작하였고 다양한 프로토콜을 지원한다. 또한 JAVA 기반의 Visualization Tool을 지원한다.

이 외에도 NS2(Network Simulator-2)[10], Sense[11], OPNET[12]등 객체지향 개념의 시뮬레이터들이 많지만, 실제 포팅이 가능한 소스코드를 생성하는 오픈 소스 기반의 Ptolemy와의 연동이 어려우며, Ptolemy가 제공하는 동기화 작업, 에너지 모델의 적용, 실시간 신호 처리 등에서 모델링 지원이 부족한 면이 있다. 또한 Ptolemy를 사용함으로써 FSM(Finite Stats Machine) 기반의 유동적이며 계층적인 모델링을 수행할 수 있다. 특히, 현재까지 무선 센서 네트워크에서 HILS 기능을 제공하는 시뮬레이터는 존재하지 않으므로, 본 논문에서 구현한 Ptolemy 기반의 시뮬레이션 Actor를 활용함으로써, 센서네트워크 모델링 및 구현에 큰 역할을 할 것으로 예상된다.

3. MAC Layer Actor 구현

센서 네트워크에서 매체접근을 위한 MAC(Medium Access Control) 프로토콜의 대표적인 예로 SMAC(Sensor MAC) [13] 프로토콜을 설계 및 구현하였다. MAC 계층의 프로토콜을 Ptolemy Actor로 구현함으로 이를 기반으로 하는 응용프로그램 및 네트워크 토폴로지, 그리고 여러 가지 통신 시나리오를 재현할 수 있다. 3.1에서 SMAC Actor의 구체적인 설계방법과 구현내용을 살펴보고 3.2에서 구현한 Actor를 통하여 실험한 예제를 설명한다.

3.1 SMAC Actor 구현

SMAC의 기본 동작은 파워 소비를 최소화하기 위한 주기적인 Listen/Sleep을 하며, 다른 센서 노드들이 전송 중일때 NAV를 이용하여 Sleep 상태로 들어간다. 또한 효율적인 채널의 예약을 위하여 RTS/CTS(Request To Send/Clear To Send) 기법을 이용해서 Hidden terminal 문제를 해결하고 있으며, Message passing 기법 사용하여 한번의 RTS/CTS로 여러 개의 DATA 패킷을 전송한다. 이는 SMAC에서 처리율을 높이기 위한 방법이 된다. NAV(Network Allocation Vector)는 모든 RTS/CTS/DATA 패킷 내에 포함되며, 모든 DATA 패킷은 ACK를 요구한다.

그림 2는 실제 Ptolemy를 이용하여 제작한 센서 네트워크 시뮬레이션 환경의 한 예를 나타내고 있다. 싱크노드와 센서노드 사이의 일대일 통신 상황으로 모든 센서 노드들에 적용되는 파라미터들의 값과 각 센서 노드들의 위치를 사용자 임의로 변경 및 배치할 수 있다. 여기서, 각 센서 노드 주위의 원은 각 센서 노드의 통신 반경을 의미한다.

그림 3은 무선 통신 채널로 사용된 PowerLossChannel의 파라미터를 나타낸다. 무선 채널 상으로 패킷을 전송함에 있어, 손실 확률, 전파지연 및 거리에 따른 파워 감소에 대한 세부적인 파라미터 설정이 가능하다.

그림 4는 센서 노드의 내부 구현을 나타낸다. 센서 노드는 크게 Collision Indicator, SMAC, Tester Actor와 in, out 포트로 구성된다. 각 구성요소의 세부적인 사항은

그림 2. 센서 네트워크 시뮬레이션 환경

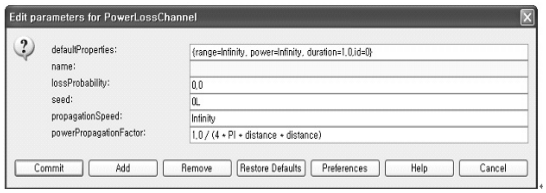


그림 3. PowerLossChannel의 파라미터

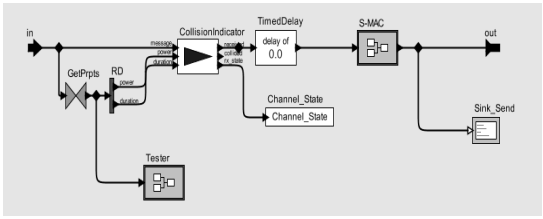


그림 4. SMAC 노드의 내부 구현

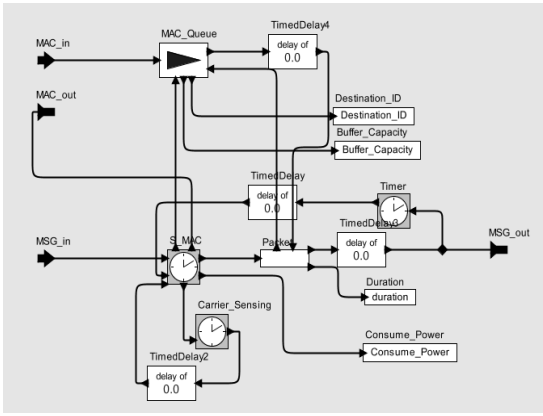


그림 5. SMAC Actor의 구현

다음과 같다. Collision Indicator Actor는 수신된 패킷의 파워와 duration 정보를 기반으로 하여, 각 상황의 맞게 수신된 패킷을 적절히 처리하고, Channel_State 파리미터 값을 갱신하는 역할을 한다. Collision Indicator Actor는 수신된 패킷을 Receive, Collision, Drop으로 판별한 후 해당 출력 포트로 패킷을 전달한다. Receive는 수신된 패킷의 파워가 미리 정해진 임계값 보다 높고 다른 노드로부터 수신된 패킷들의 영향을 받지 않은 경우로, 이 경우에만 SMAC Actor에 수신 패킷을 중계하게 된다. Collision은 수신된 패킷의 파워가 미리 정해진 임계값 보다 높으나 다른 노드로부터 수신된 패킷들의 영향을 받은 경우, 패킷 충돌 상황으로 판단하여 단순히 패킷을 폐기시킨다. Drop은 수신된 패킷의 파워가 미리 정해진 임계값 보다 낮은 경우로 역시 패킷을 폐기시킨다.

SMAC Actor는 프로토콜의 핵심이 되는 부분으로써 단일 채널을 사용하는 경쟁기반 프로토콜을 통한 시간을 프레임 단위로 나누고, 이 프레임을 Listen 구간과 Sleep 구간으로 나눈다. Sleep 구간에서는 노드가 무선 통신을 위한 부분의 전원을 끄고 에너지 소모를 거의 하지 않는 상태이다. Listen 구간의 Duty Cycle을 줄이는 방법으로 전력 소모를 감소 시킬 수 있다. 서브 Actor 중 Tester Actor는 패킷 송수신 관련 상황을 실시간으로 보여주기 위한 역할을 한다. 입출력 포트로써 in, out 포트가 있으며 무선 채널상으로 패킷을 송수신하기 위한 역할을 한다.

그림 5는 SMAC Actor의 내부 구현을 나타낸다. SMAC Actor는 MAC_Queue, Timer, Packet, Carrier_Sensing, S_MAC Actor로 구성되며, 각 컴포넌트의 주요 역할은 다음과 같다. MAC_Queue Actor는 상위 레이어에서 전송된 패킷들을 저장하고 Destination_ID, Buffer_Capacity 파라미터 값을 갱신하는 역할을 한다. Timer Actor는 송신 노드에서 무선 채널상으로 RTS 또는 DATA 패킷을 전송 시, 해당 타이머를 설정하는 역할을 한다. 이 중 RTS 타이머는 RTS 패킷을 실제 무선 채널 상에 전송 후, 미리 정해진 일정 시간을 대기, 정해진 일정 시간 동안 해당 노드로부터의 CTS 패킷을 수신하지 못한 경우, RTS 패킷 재전송 처리를 위한 신호를 S_MAC Actor에 전송한다. 또한 DATA 패킷을 실제 무선 채널 상에 전송 후, DATA 타이머를 설정하는데, 미리 정해진 일정 시간을 대기하도록 한다. 이 기간동안 해당 노드로부터의 ACK 패킷을 수신하지 못한 경우, DATA 패킷 재전송 처리를 위한 신호를 S_MAC Actor에 전송하게 된다.

Packet Actor는 MAC_Queue Actor와의 정보교환을 통해 실제 컨트롤 및 데이터 패킷을 생성하여 무선 채널 상으로 전송하고, duration 파라미터를 갱신하는 역할을 한다. 생성된 패킷 구조는 그림 6과 같다.

3.2 SMAC Actor의 실험

실험에서 구성된 센서 네트워크 시뮬레이션 환경은 그림 2와 같으며, SMAC 프로토콜을 사용하는 송신 노드와 수신 노드 사이의 일대일 통신 상황을 나타낸다. SMAC은 노드가 비활성시, 수신 상태에서 소비하는 에너지를 줄이기 위해서 주기적으로 Listen 구간과 Sleep 구간을 반복하며, Listen 구간 내에서 신뢰성 있는 데이터 전송을 위해서 송신 노드와 수신 노드는 RTS/CTS/DATA/ACK의 순서를 따른다. 그림 7 및 그림 8은 실제 송신 노드와 수신 노드가 전송한 메시지를 Collision Indicator Actor를 통하여 나타내고 있다.

다음은 송신 노드가 (100) 수신 노드에게(1) 전송하는 메시지 로그를 나타낸다.

• {Dest_ID = 1, End_Packet = true, Generation_Time = 2.9227032218, Msg_Duration = 1.0E-4, Msg_Type = “RTS”, NAV_Time = 0.00235, Network = “-”, Source_ID = 100}

• {Dest_ID = 1, End_Packet = true, Generation_Time = 2.9229532218, Msg_Duration = 0.002, Msg_Type = “DATA”, NAV_Time = 1.5E-4, Network = “-”, Source_ID = 100}

아래는 수신 노드가 (1) 송신 노드에게 (100) 전송하는 메시지를 나타낸다.

• {Dest_ID = 100, End_Packet = true, Generation_Time = 2.9228032218, Msg_Duration = 1.0E-4, Msg_Type = “CTS”, NAV_Time = 0.0022, Network = “-”, Source_ID = 1}

• {Dest_ID = 100, End_Packet = true, Generation_Time = 2.9250032218, Msg_Duration = 1.0E-4, Msg_Type = “ACK”, NAV_Time = 0.0, Network = “-”, Source_ID = 1}

한편, 적절한 네트워크 토폴로지의 재구성을 통하여 실험에서 패킷의 충돌을 재현할 수 있다. SMAC 프로토콜을 사용하는 두 개의 송신 노드와 한 개의 수신 노드 사이의 통신 상황을 가정하고, 두 개의 송신 노드들은 동시에 경쟁에 참여하게 되면 두 송신 노드가 전송한 RTS 메시지가 충돌을 발생시킨다. 이 후, 두 개의 송신 노드들은 일정 기간 동안 수신 노드로부터 어떠한 응답 메시지도 전송되지 않으므로 인해 RTS 메시지 전송이 실패하였음을 인식하고, 백오프 값을 재설정한 후 재전송을 한다. 두 송신 노드 중 식별자 200 노드가 먼저 RTS 메시지를 재전송하고, 수신노드와 통신을 재개한다. 이 때, 식별자 100의 송신 노드는 RTS 메시지의 오버히어링을 통해 라디오를 차단한다. NAV 타이머가 만료된 이후, 대기한 송신 노드는 다시 수신 노드와 통신을 재개한다. 그림 9는 시뮬레이션 환경하에서 송신 노드들과 수신 노드 사이의 메시지 전송 절차를 나타내며, 아래 로그 메시지는 식별자 100의 송신노드가 수신노드로 패킷을 전송하는 순서를 보여준다. Generation_Time에서 나타내듯이, 두 번째 RTS는 약 30msec의 NAV 타이머 동작 후 통신을 재개함을 알 수 있다.

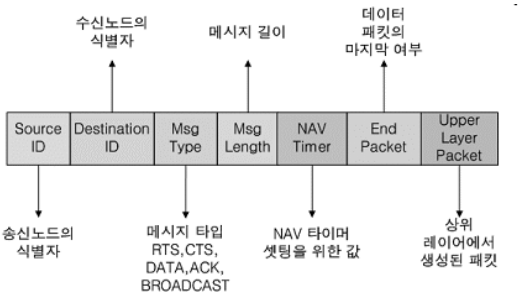


그림 6. Packet Actor에서 생성된 패킷 구조

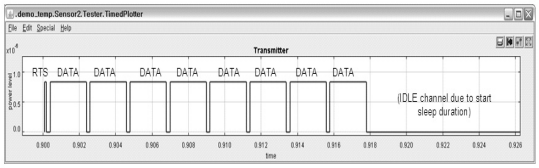


그림 7. 송신 노드의 메시지

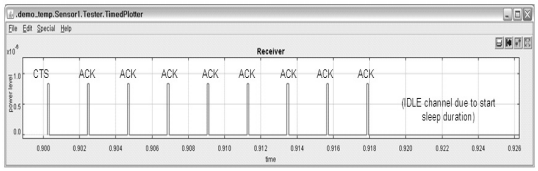


그림 8. 수신 노드의 메시지

• {Dest_ID = 1, End_Packet = true, Generation_Time = 14.9178883371, Msg_Duration = 1.0E-4, Msg_Type = “RTS”, NAV_Time = 0.00235, Network = “-”, Source_ID = 100}

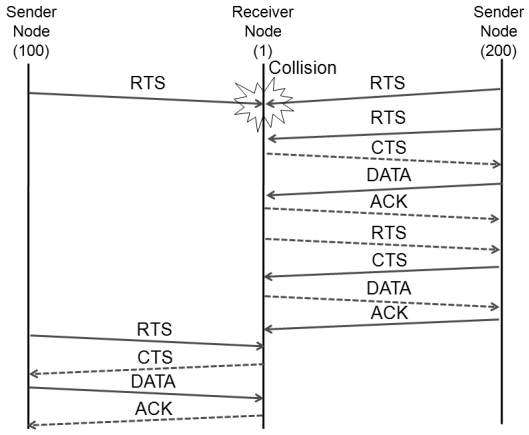


그림 9. 송신 노드들과 수신 노드의 패킷충돌 시나리오

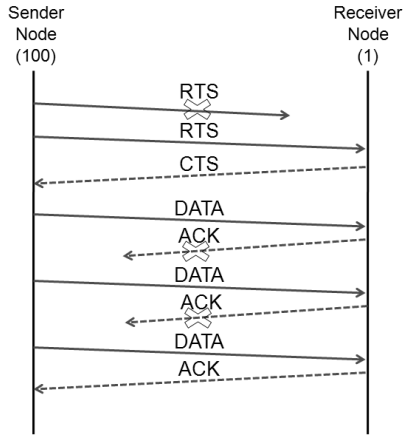


그림 10. RTS 혹은 DATA패킷의 재전송 절차

• {Dest_ID = 1, End_Packet = true, Generation_Time = 14.9527239237, Msg_Duration = 1.0E-4, Msg_Type = “RTS”, NAV_Time = 0.00235, Network = “-”, Source _ID = 100}
• {Dest_ID = 1, End_Packet = true, Generation_Time = 14.9529739237, Msg_Duration = 0.002, Msg_Type = “DATA”, NAV_Time = 1.5E-4, Network = “-”, Source _ID = 100}

SMAC의 통신에서 패킷의 충돌로 인한 재전송 이외에, 채널 에러 혹은 간섭 등으로 인한 패킷의 손실이 발생할 수 있다. 이를 지원하기 위하여 SMAC의 노드는 무선 채널상으로 RTS 또는 DATA 패킷을 전송시, 해당 타이머를 설정하며, 해당 타이머가 만료되기 전까지 수신 노드로부터 어떠한 응답도 수신하지 못한 경우, 그림 10과 같은 재전송 과정을 수행한다. 이 때, 실험상의 패킷 손실 발생을 위해 무선 채널 상의 패킷 손실율은 20%로 설정하였다. 다음은 실제 송신 노드와 수신 노드가 전송한 메시지를 나타내고 있다.

• {Dest_ID = 1, End_Packet = true, Generation_Time = 7.9091953612, Msg_Duration = 1.0E-4, Msg_Type = “RTS”, NAV_Time = 0.00235, Network = “-”, Source _ID = 100}
• {Dest_ID = 1, End_Packet = true, Generation_Time = 7.9194167853, Msg_Duration = 1.0E-4, Msg_Type = “RTS”, NAV_Time = 0.00235, Network = “-”, Source _ID = 100}
• {Dest_ID = 1, End_Packet = true, Generation_Time = 7.9196667853, Msg_Duration = 0.002, Msg_Type = “DATA”, NAV_Time = 1.5E-4, Network = “-”, Source _ID = 100}
• {Dest_ID = 1, End_Packet = true, Generation_Time = 7.9228667853, Msg_Duration = 0.002, Msg_Type = “DATA”, NAV_Time = 1.5E-4, Network = “-”, Source _ID = 100}
• {Dest_ID = 1, End_Packet = true, Generation_Time = 7.9260667853, Msg_Duration = 0.002, Msg_Type = “DATA”, NAV_Time = 1.5E-4, Network = “-”, Source _ID = 100}

즉, 패킷 손실로 인하여 RTS가 두 번 전송되어지며, ACK 패킷을 2번 수신하지 못함으로 인하여, 동일한 DATA 패킷이 3번 전송됨을 보여준다.

4. Network Layer Actor 구현

센서 네트워크에서 경로설정 및 통신 토폴로지 유지를 위한 라우팅 프로토콜의 대표적인 예로 AODV (Ad hoc On-Demand Distance Vector) [14] 프로토콜을 설계 및 구현하였다. MAC계층의 SMAC 프로토콜의 구현과 마찬가지로 네트워크 계층의 Ptolemy Actor로 구현함으로써, 이를 기반으로 하는 응용프로그램을 쉽게 모델링할 수 있으며, 네트워크 토폴로지, 그리고 여러 가지 통신 시나리오를 재현할 수 있다. 4.1에서 AODV Actor의 구체적인 설계방법과 구현내용을 설명하고, 4.2에서 구현한 Actor를 통하여 실험한 예제를 분석한다.

4.1 AODV Actor 구현

AODV는 애드 혹 네트워크에서 통신에 참여하고자 하는 노드들이 자가적이고 동적인 방법으로 다중 홉 라우팅을 가능하도록 한다. AODV는 모바일 노드가 빠르게 새로운 목적지의 경로를 찾도록 하며, 현재 사용되는 경로만을 유지하도록 한다. AODV의 특징은 각 경로 엔트리마다 Destination Sequence Number를 사용한다는 것이다. Destination Sequence Number는 목적지 노드에 의해 생성되며 경로요청을 하는 모든 노드들의 경로정보에 포함된다. 이를 통해 라우팅 테이블내 경로설정의 무한 루프를 방지할 수 있으며, 노드들이 최신의 경로를 유지할 수 있도록 한다. 만약 어떤 목적지로의 경로가 여러 개 존재할 때, 노드는 Sequence Number가 큰 경로를 선택하게 된다. AODV Actor의 전체 내부 구성도는 그림 11과 같다.

AODV Actor는 입출력 포트를 갖는데, 입력포트는 Packet 혹은 AODV_Packet의 입력을 받아 AODV Actor에 전달하게 된다. 출력포트는 보내려는 패킷의 다음 홉에 대한 주소 정보와 Packet 혹은 AODV_Packet를 Record-Token형식으로 MAC에 전달하게 된다. Packet과 AODV_Packet은 ptolemy.data.Token 클래스를 상속받으며, 출발지와 목적지의 주소정보 등을 포함한다. AODV_Packet은 RREQ(Route Request), RREP(Route Reply), HELLO, RERR(Route Error) 패킷들 중 하나를 형식으로 갖는다.

AODV Actor의 동작과정은 경로 테이블에 목적지 노드 주소의 경로가 있는 경우, MAC단에 다음 홉의 주소정보와 Packet을 전달하여 즉각적인 전송이 이루어지도록 한다. 경로 테이블에 목적지 노드의 경로가 없는 경우, AODV_Packet을 이용하여 경로 요청 과정을 수행한다. 또한 목적지에 대한 경로를 얻어오기 전에 도착한 패킷들은 라우팅 큐에 저장되며, 이 후 RREP 패킷을 받으면, 라우팅 큐의 패킷을 순서대로 다음 홉으로 전달한다.

PacketGenerator Actor를 통해 AODV Actor를 테스트 할 수 있다. 이 Actor로부터 생성된 패킷의 목적지 노드의 경로를 모르는 경우, On-Demand 방식으로 AODV Actor를 통해 경로를 요청을 하게 된다. 경로가 설정된 이후에 각 노드들은 라우팅 정보를 이용하여 목적지 노드까지 이 패킷을 전송한다.

PacketGenerator Actor의 파라미터 설정은 그림 12와 같으며, 각 파라미터를 설명은 다음과 같다. time은 처음 Packet의 생성시간을 나타내며, value 파라미터는 Packet에 사용자가 특정 값을 입력할 수 있 수 있도록 한다. source는 출발 노드의 IP 주소, destination는 목적지 노드

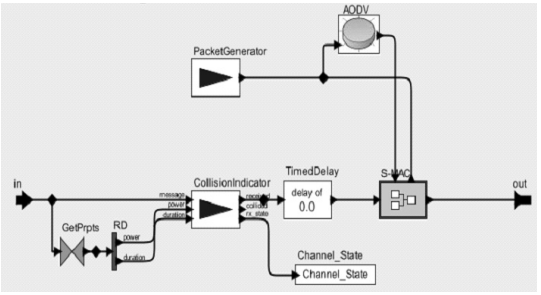


그림 11. AODV Actor의 내부

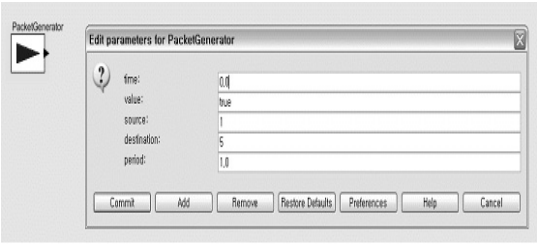


그림 12. PacketGenerator Actor의 파라미터

의 IP 주소를 나타낸다. period는 Packet의 생성 주기이며 time 시간 이후부터 이 주기마다 새로운 패킷이 생성된다. 또한 입출력 포트로써 output는 파라미터 값을 반영한 Packet을 내보내는 포트이다.

4.2 AODV Actor의 실험

그림 13은 AODV Actor와 SMAC Actor을 연동한 모습과 각종 콘트롤 패킷의 임계값 및 네트워크 파라미터를 설정한 결과를 보여준다. 두 Actor가 데이터를 주고받기 위해 입출력포트가 각각 연결돼 있다. AODV Actor는 다음 홉에 대한 주소와 패킷정보를 RecordToken 형식으로 전달해주고, SMAC은 이 정보를 받아 다음 홉까지 이 패킷을 전달해준다. SMAC Actor는 받은 패킷의 주소를 판별하여 AODV에게 패킷을 전달하게 된다. PacketGenerator Actor를 통해 패킷 전송 이벤트를 AODV Actor로 발생시키고 있다.

AODV Actor와 SMAC Actor를 사용하여 7개의 센서를 가지고 그림 14의 토폴로지 환경에서 애드 혹 네트워크 전송 실험을 하였다. 시뮬레이션 시작과 동시에 센서 1은 센서 5로의 보낼 Packet을 PacketGenerator로부터 생성한다. 센서 1은 센서 5로의 경로를 가지고 있지 않기 때문에 센서 5로 경로요청을 하게 된다. 경로 요청과 경로 응답과정을 통해 1→2→3→4→5의 경로가 설정되고 이 경로로 Packet이 전송된다.

전송 시나리오를 통해 발생한 경로요청 및 경로응답과정

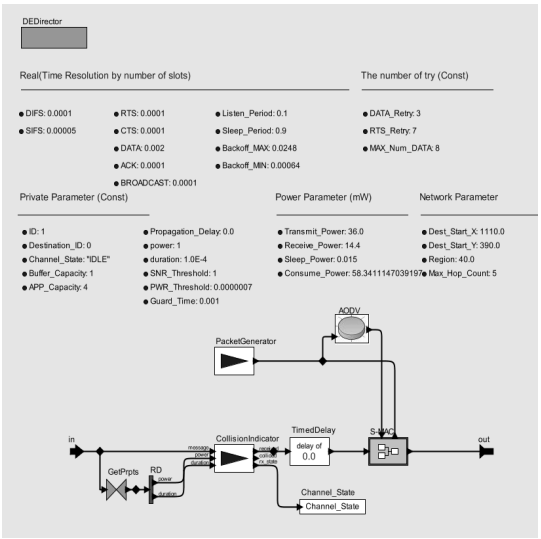


그림 13. AODV Actor와 SMAC Actor의 연동 모습

과 응답을 통해 얻은 경로로의 전송 모습을 AODV Actor의 입출력 포트로 확인할 수 있다.

AODV_Packet은 Data와 Type의 RecordToken 형식 {Data=[AODV_Packet의 정보], Type="AODV_Packet"}이고, AODV_Packet 정보의 출력 사항은 그림 15와 같다. Hello Packet과 같이 브로드캐스트 형태로 전송 되는 경우 AODV_Packet의 Source ID와 Destination ID 부분이 0으로 설정된다.

그림 16 및 그림 17는 릴레이 노드 센서 3에서의 AODV Actor의 입출력 메시지를 나타내었다. 센서 3에서는 RREQ 메시지를 수신하였을 때, 목적지 5에 대한 주소를 가지고 있지 않기 때문에 이전 홉 노드에서 받은 RREQ 메시지를 다시 브로드캐스트 한다. 또한 RREP 메시지를 받아서 출발 노드 1에게 전달하는 역할을 수행하고 있다. 또한 경로가 설정된 이후에는 경로 테이블 정보를 이용하여 목적지 5로의 다음 홉 주소 4의 정보와 Packet을 RecordToken 형식으로 SMAC Actor로 전달해주는 모습을 볼 수 있다.

5. 결 론

본 연구의 결과물인 Ptolemy를 이용한 SMAC, AODV Actor들은 현재 존재하는 센서 네트워크 시뮬레이터들이 가지는 GUI지원 문제, 실제 센서 네트워크로의 적용을 위한 TinyOS 호환성 문제 등을 해결하였다. 즉, Ptolemy를 이용한 Actor들은 GUI기반의 시뮬레이터로써 사용자가 사용하기 편리하며, 시뮬레이션 결과 역시 분석하기

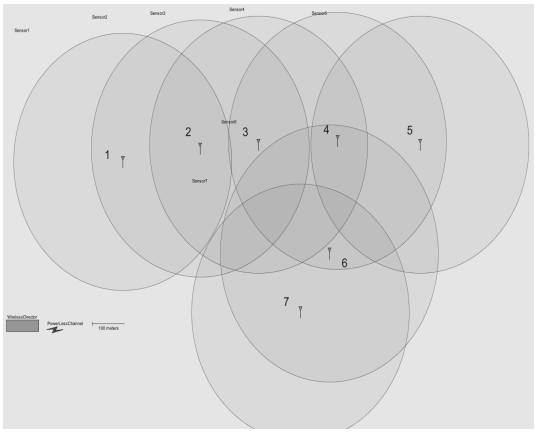


그림 14. AODV 경로 설정 시나리오

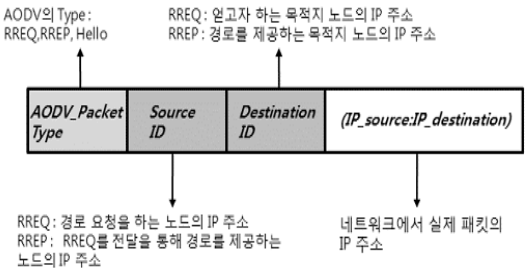


그림 15. AODV_Packet의 출력정보

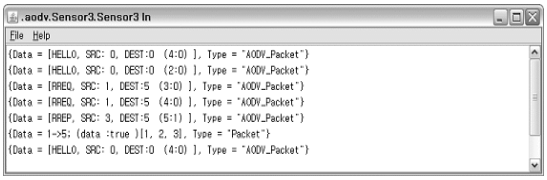


그림 16. 릴레이 노드 센서 3의 AODV Actor 입력

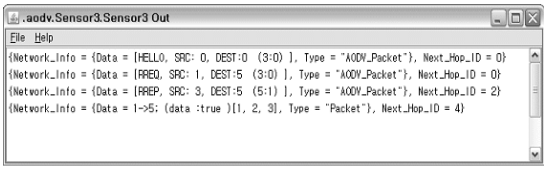


그림 17. 릴레이 노드 센서 3의 AODV Actor 출력

쉽게 디자인 되었다. 그리고 Ptolemy와 TinOS를 연결하는 Viptos를 이용함으로써 TinyOS기반에서 실행 가능한 소스코드를 바로 생성할 수 있는 장점을 가진다. 특히 장애물이나 노드의 움직임을 줄 수 있으므로 다른 시뮬레이터보다 좀 더 다양하고 정확한 환경에서의 시뮬레이션이 가능

하므로 센서 네트워크 알고리즘의 설계와 성능비교에 요구되는 시간과 노력을 상당부분 줄일 것으로 예상된다. 향후 연구과제로서는 ZigBee 프로토콜의 기반이 되는 IEEE 802.15.4 MAC 프로토콜과 위치인식 기반의 라우팅 프로토콜들을 구현하고 다양한 실험을 진행해나갈 것이다.

참 고 문 헌

1. Ptolemy Project,URL:http://ptolemy.eecs.berkeley.edu/.
2. Elaine Cheong, Edward A. Lee, Yang Zhao, "Viptos: a graphical development and simulation environment for TinyOS-based wireless sensor networks", international conference on Embedded networked sensor systems, 2005.
3. 김방현, 김태규, 정용덕, 김종현, "전력소모량 및 실행시간 추정이 가능한 센서 네트워크. 시뮬레이터의 개발", 한국시뮬레이션학회 논문지, 2006.
4. Philip Levis, Nelson Lee, Matt Welsh, David Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications", SenSys'03, November, 2003.
5. Prowler:Probabilistic Wireless Network Simulator, URL: http://www.isis.vanderbilt.edu/projects/nest/prowler.
6. A. Ledeczi, M. Maroti, and I. Bartok, SIESTA - Simple NEST Application Simulator Siesta v1.0, Institute for Software Integrated Systems, Vanderbilt Univ., 2001.
7. NEST Project, http://webs.cs.berkeley.edu.
8. D. Gay, P. Levis, and D. Culler, "Software Design Patterns for TinyOS", LCTES, June. 2005.
9. J-SIM (JavaSim), URL:http://www.j-sim.org.
10. Sensor, URL:http://www.ita.cs.rpi.edu/sense/index.html.
11. S.McCanne and S. Floyd, "NS network simulator," URL: http://www.isi.edu/nsnam/ns.
12. OPNET, URL:http://www.opnet.com.
13. Wei Ye, John Heidemann and Deborah Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," INFOCOM, 2002.
14. C. E. Perkins, "Ad Hoc On-Demand Distance Vector (AODV) Routing", IETF MANET Working Group, draft-ietf-manet-aodv-12.txt, November 2002.

안 기 진 (kijin@postech.ac.kr)

2007 서울시립대학교 전자전기컴퓨터공학부 학사
2007～현재 포항공과대학교 정보통신학과 석사

관심분야 : 네트워크 모델링&시뮬레이션, Ad hoc 네트워크

주 현 철 (chul1978@postech.ac.kr)

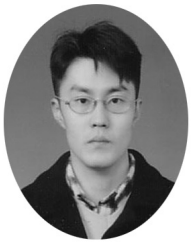
2005 한양대학교 컴퓨터공학과 학사
2007 포항공과대학교 컴퓨터공학과 석사
2007～현재 포항공과대학교 컴퓨터공학과 박사과정

관심분야 : 네트워크 모델링&시뮬레이션, IPTV

오 형 래 (raibest@postech.ac.kr)

2003 홍익대학교 전기전자공학과 학사
2005 홍익대학교 전기전자공학과 석사
2005～현재 포항공과대학교 컴퓨터공학과 박사과정

관심분야 : 모델링&시뮬레이션, QoS 알고리즘, 무선 네트워크



김 영 덕 (ydkim@dgist.ac.kr)

2004 성균관대학교 컴퓨터교육과 학사
2006 포항공과대학교 정보통신학과 석사
2006～현재 DGIST 미래산업융합기술연구부 연구원

관심분야 : 모바일 컴퓨팅, 네트워크 시뮬레이션

양 연 모 (yangym@kumoh.ac.kr)

1990 KAIST 전기전자공학과 학사
1999 GIST 메카트로닉스 석사
2006 GIST 메카트로닉스 공학박사
2006～2008 DGIST 선임연구원
2008～현재 금오공과대학교 전자공학부 교수

관심분야 : EPONs, 센서네트워크, 임베디드시스템, 네트워크 시뮬레이션

송 황 준 (hwangjun@postech.ac.kr)

1990 서울대학교 제어계측공학과 학사
1992 서울대학교 제어계측공학과 석사
1999 University of Southern California EE-Systems 공학박사
2005～현재 포항공과대학교 컴퓨터공학과 부교수

관심분야 : 멀티미디어 통신, 통방융합기술, 네트워크 시뮬레이션