무선센서네트워크에서 성능측정을 통한 전송방식의 문제점 분석 및 개선

임동선* · 이좌형** · 정인범***

Performance Evaluation and Enhancement of Transmission Technique in Wireless Sensor Networks

Dong-sun Lim* · Joa-hyoung Lee** · In-bum Jung***

요 약

주변환경에서 정보를 수집하는 센서네트워크는 다양한 분야에서 사용되고 있다. 정밀한 분석을 위하여 매우 짧은 주기로 정보를 수집할 필요가 있는 어플리케이션에서는 네트워크를 구성하는 노드의 수가 증가하고 정보를 수집하는 주기가 짧아질수록 생성되는 데이터의 양도 증가하기 때문에 노드 간에 지연을 최소화하여 효율성을 높일 필요가 있다. 본 논문에서는 네트워크 구성시 데이터 전송률에 영향을 미치는 전송주기와 센서 노드간의 거리, 전송 패킷의 크기에 대한 실험을 실시하여 실험한 결과를 바탕으로 주기적 전송 기법의 한계와 문제점을 분석하고 패킷 전송 작업이 완료되었음을 알려주는 전송 완료 이벤트를 이용하여 연속적인 패킷 전송시 지연을 줄여주는 SET(SendDone Eventbased Transmission Technique)기법을 제안한다. SET에서는 패킷 전송이 완료된 시점에 즉시 다음 패킷을 전송하기 때문에 패킷을 전송하는데 소요되는 시간에 상관없이 패킷 전송간에 지연이 발생하지 않는다. 따라서 연속적으로 대량의 패킷을 전송할 때 높은 전송율을 제공할 수 있다.

ABSTRACT

Sensor network is used to obtain sensing data in various area. The interval to sense the events depends on the type of target application and the amounts of data generated by sensor nodes are not constant. Many applications exploit long sensing interval to enhance the life time of network but there are specific applications that requires very short interval to obtain fine-grained, high-precision sensing data. If the number of nodes in the network is increased and the interval to sense data is shortened, the amounts of generated data are greatly increased and this leads to increased amount of packets to transfer to the network. To transfer large amount of packets fast, it is necessary that the delay between successive packet transmissions should be minimized as possible. In this paper, we propose SET(SendDoneEventbasedTransmission Technique)which reduces the delay between successive packet transmissions by using SendDone Event which informs that a packet transmission has been completed. In SET, the delay between successive packet transmissions is shortened very much since the transmission of next packet starts at the time when the transmission of previous packet has completed, irrespective of the transmission time. Therefore SET could provide high packet transmission rate given large amount of packets.

키워드

센서네트워크, 이벤트 기반, 연속적 패킷 전송, 지연, Timer Event, SendDone Event

Key word

sensor network, event base. Successive packet transmission, delay, Event, SendDone Event

* 한국전자통신연구원 (dslim@etri.re.kr)

** 강원대학교 (jinnie4u@kangwon.ac.kr)

*** 강원대학교 (교신저자, ibjung@kangwon.ac.kr)

접수일자 : 2009. 10. 13

심사완료일자 : 2009. 10. 28

I. 서 론

최근 MEMS와 마이크로프로세서 그리고 무선 통신 기술의 발전으로 센서 노드들을 이용하여 넓은 지역에 걸쳐 정확한 정보를 얻고자 하는 센서네트워크가 널리 보급되기 시작했다. 센서네트워크는 주위환경에서 발생하는 이벤트 정보를 수집하여 전달하는 것을 기본 목적으로 한다. 센서네트워크에서 사용되는 운영체제도 발생하는 이벤트를 효율적으로 처리하도록 이벤트 기반으로 동작하도록 설계된다. 센서네트워크를 구성하는 센서 노드들은 데이터를 센싱하는 센서 모듈과 센싱된 데이터를 가공 처리하는 MCU모듈 그리고 데이터를 다른 노드나 외부네트워크로 전송하기 위한 전송모듈로 구성된다. 센서 노드들은 이벤트에 따른 단순한 작업을 반복적으로 수행하기 때문에 이러한 모듈들은 매우 낮은 사양으로 구성된다. 이벤트 기반 운영체제에서 저사양의 모듈들을 어떻게 효율적으로 사용할 것인가에 대한 많은 연구들이 이루어지고 있다.

센서네트워크에서 센싱 주기는 매우 길게 설정하는 것이 일반적이기는 하지만 매우 짧은 센싱 주기를 필요로 하는 어플리케이션들도 있다. 빌딩이나 댐과 같은 건축물들에 대한 안전 진단을 목적으로 하는 어플리케이션이 대표적인 예가 될 수 있다[1,2]. 빌딩이나 교량과 같은 건축물에 대한 안전을 진단하기 위하여 유선으로 연결된 센서들을 이용하여 데이터를 수집하는 것이 일반적인 방법이다. 하지만 센서들을 유선으로 연결하는 경우 설치비용이 많이 들며 설치장소에 따라 설치가 용이하지 않은 곳도 있을 수 있다. 최근에는 이러한 문제점들을 해결하기 위하여 무선센서를 이용하는 방법이 연구되고 있다. 유선센서에 비해 무선센서는 선을 필요로 하지 않기 때문에 설치비용이 절감되며 설치가 용이하다. 이러한 어플리케이션에서는 건물에 발생하는 이벤트를 감지하기 위하여 가속도와 같은 정보를 수집한다. 매우 긴 센싱 주기를 가지는 동식물의 생태를 모니터링하는 것과 같은 어플리케이션들에 비해 구조물의 안전을 진단하는 어플리케이션에서는 매우 짧은 센싱 주기를 필요로 한다. 생태 모니터링의 경우 한 시간당 한번 정도의 센싱 주기도 충분하지만 구조물 모니터링에서는 초당 100Hz이상의 데이터를 센싱하여 외부네트워크로 전달한다. 네트워크상에 노드의 수가 많아질수록 전송해야할 데이터의 양도 기하급수적으로 증가할 수 있다[4,5].

문제는 센서네트워크에서 통신수단으로 사용하는 무선통신이 매우 낮은 대역폭을 가지며 가변적이고 에러율이 높다는 것이다. 많은 양의 데이터를 낮은 대역폭을 가지는 불안정한 무선네트워크로 전송하기 위해서는 많은 시간이 소모될 수 있다. 많은 데이터를 좀더 빠르게 전송하기 위해서는 패킷 하나하나를 전송하는데 필요한 시간을 줄일 필요가 있다. 매우 짧은 센싱주기를 갖는 어플리케이션을 목적으로 하는 센서네트워크에서는 효율적으로 연속적인 데이터를 전송하는 기법에 대한 연구가 있어야 한다[7,8,9,10,11,12,13].

본 논문에서는 네트워크 구성시 데이터 전송률에 영향을 미치는 전송주기와 센서 노드간의 거리, 전송 패킷의 크기에 대한 실험을 실시하여 실험한 결과를 바탕으로 주기적 전송 기법의 한계와 문제점을 분석하고 패킷 전송 작업이 완료되었음을 알려주는 전송 완료 이벤트를 이용하여 연속적인 패킷 전송시 지연을 줄여주는 SET(SendDone Event based Transmission Technique)기법을 제안한다. SET에서는 연속적으로 데이터를 전송할 때 타이머 이벤트를 이용하는 것이 아니라 패킷 전송을 완료하였다는 이벤트에 기반하여 데이터를 전송하도록 한다. 하나의 패킷 전송이 완료되는 시점에 바로 다음 패킷을 전송하도록 하여 패킷 전송 간에 발생할 수 있는 지연을 줄일 수 있다. 매우 많은 양의 데이터를 전송해야하는 경우 패킷 전송 간의 지연을 조금이라도 줄이는 것이 매우 큰 효과를 가져올 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문과 관련된 연구에 대하여 간단히 살펴본다. 3장에서는 실험을 통해 센서네트워크에서 사용되는 센서모트의 성능을 측정하고 4장에서는 센서네트워크용 이벤트 기반 운영체제인 Tinyos의 특징에 관하여 설명하고. 5장에서는 기존방식의 문제점을 지적하고 제안하는 기법인 SET에 대하여 설명한다. 6장에서는 성능을 측정하고 결과를 분석한다. 마지막으로 6장에서 본 논문의 결론을 맺는다.

Ⅱ. 관련연구

센서네트워크에서 데이터 전송에 관한 연구는 크게 에너지 효율적인 면을 고려하는 연구와 멀티홉 라우팅에 관한 연구가 주를 이루고 있다. 센서노드들이 배터리를 통하여 에너지를 공급받기 때문에 노드의 수명을 연장하기 위하여 에너지를 고려하는 것은 필수적이다. 센서노드에서 에너지 소모가 가장 큰 부분이 라디오를 이용하여 데이터를 전송하는 부분이기 때문에 에너지 효율을 고려한 데이터 전송에 관한 연구가 많이 진행되고 있다. 이러한 연구들에서는 여러 노드들의 데이터를 융합하거나 압축기법을 이용하여 데이터 전송횟수를 줄이기 위해 노력한다. 일반적으로 센서네트워크상에서 센싱되는 데이터들은 지역적으로나 시간적으로 유사한 값을 가지기 때문에 데이터 융합이나 압축을 통하여 데이터의 중복성을 많이 줄일 수 있다. 데이터의 양이 줄어들면 다른 노드로의 전송횟수도 줄기 때문에 에너지 측면에서 많은 이득을 볼 수 있다. 하지만 구조물 모니터링과 같은 일부 어플리케이션의 경우 데이터의 융합과 같은 작업을 수행하기 어렵다는 문제점이 있다. 또한 데이터 융합이나 압축을 통하여 각각의 노드에서 전송해야할 데이터의 양을 줄일 수 있더라도 데이터의 최종 목적지인 베이스스테이션 주변의 노드들이 전송해야할 데이터의 양은 매우 많아질 수 있다. 센싱주기가 짧아서 발생하는 데이터의 양이 많은 경우 효율적으로 많은 데이터를 연속적으로 전송하는 기법이 필요하다[14,15,16].

많은 양의 데이터를 효율적으로 전송하기 위한 기법으로 LRX기법이 제안되었다[4][9]. LRX는 구조물 모니터링을 위한 어플리케이션을 개발하면서 발생한 문제점을 해결하기 위해 제안된 기법이다. 구조물 모니터링을 위해 테스트베드를 구성하여 실험한 결과 매우 많은 양의 데이터가 발생하는데 전송모듈에서 이를 효율적으로 처리 못하는 문제점이 발견되었다. LRX는 센싱된 데이터를 메모리상에 블록단위로 저장하고 블록단위로 전송하도록 한다. 노드간에 링크상태나 수신노드의 처리량에 따라 전송할 수 있는 데이터양이 가변적이기 때문에 윈도우개념을 사용하여 전송하는 데이터의 양을 조절한다. LRX를 이용할 경우 윈도우 사이즈에 따라 전송율이 달라짐을 실험을 통하여 증명하였다. LRX는 전체 데이터의 전송에 관한 기법이라면 SET는 패킷 전송 사이에 발생할 수 있는 지연을 줄이고자 하는 기법으로 LRX와 SET를 함께 사용할 경우 보다 높은 성능을 기대할 수 있을 것이다.

Ⅲ. 센서노드의 네트워크 성능 측정

실험은 패킷전송을 위한 센서노드와 패킷수신을 하는 하나의 게이트웨이 노드, 패킷을 검사하고 DB 저장 기능을 위한 호스트PC로 이루어진다. 실험의 소프트웨어 구성도는 그림 1와 같다. 센서노드에는 CRC에러 측정을 위한 checkCRC가 각각 설치되며 게이트웨이에 패킷을 전송하는 동작을 한다. 게이트웨이 노드에는 TOSBase를 설치하였다. 센서노드로부터 패킷을 수신하는 기능을 하는 TOSBase는 패킷 수신여부를 확인하기 위해 Ack를 이용한다.

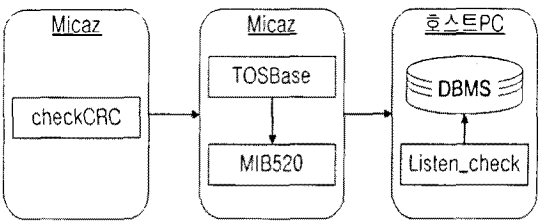

그림 1. 센서모트 성능 측정 시스템 구성도.
Fig 1. Test System Configuration

호스트PC와의 통신은 프로그래밍 보드인 MIB520를 통한 UART 방식으로 이루어진다[3]. 호스트PC에서 실행되는 Listen_check는 센서노드에서 전송한 패킷수와 호스트PC에서 수신한 패킷수, CRC 에러수, 비트에러패턴을 DB에 저장하는 동작을 한다.

Micaz 모트의 거리에 따른 패킷전송률 결과는 그림 2와 같다. 1개와 2개의 모트를 이용한 패킷 전송률은 서로 비슷한 결과를 보여 다. 이는 두 개의 모트를 이용한 실험에서는 패킷의 충돌로 인한 성능 저하가 크지 않다는 것을 보여준다.

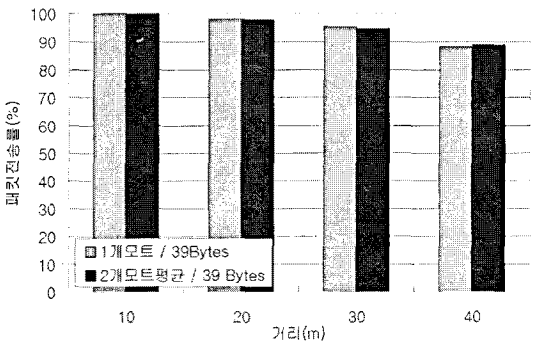

그림 2. 거리에 따른 패킷 전송율 변화.
Fig 2. Packet transmit rate with distance

하지만 그림 3와 같이 패킷의 크기가 115Bytes인 센서 노드의 최대전송률 실험에서는 39Bytes 크기의 패킷을 사용할 때보다 전송률이 낮았으며 거리가 멀어질수록 전송률의 저하가 크게 나타났다. 39Bytes 패킷 사용보다 115Bytes 크기의 패킷 사용에서 CRC 에러율이 높게 나타났으며 특히 먼 거리 통신에서는 큰 크기의 패킷을 사용하는 것이 좋지 않음을 알 수 있다.

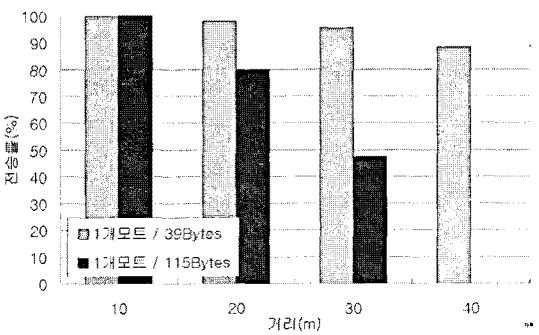

그림 3. 패킷 크기에 따른 전송율 변화
Fig 3. packet transmit rate with size.

Ⅳ. 이벤트 기반 운영체제

본 논문에서 제안하는 SET (SendDone Event based Transmission Technique)기법은 센서네트워크용 이벤트 기반 운영체제인 Tinyos상에서 연속적인 패킷 전송을 효율적으로 하고자 개발되었다. 본 장에서는 Tinyos의 특징에 대해 살펴보고 이를 바탕으로 다음 장에서 제안하는 SET기법에 대해 설명한다[17].

4.1 Tinyos의 특징

센서네트워크용으로 개발된 Tinyos는 이벤트 기반으로 동작하는 운영체제이다. Tinyos는 컴퍼넌트로 구성되며 컴퍼넌트 간에 통신 수단으로 command와 event를 사용한다[17,18]. 그림 4는 tinyos 의 구조를 보여준다. 하나의 컴퍼넌트에서 다른 컴퍼넌트로 작업을 요청할 때 사용하는 것이 command이며 작업의 결과나 특정한 사건의 발생을 다른 컴퍼넌트로 알려줄 때 사용하는 것이 event이다. Event는 하드웨어적 인터럽트와 소프트웨어적 인터럽트 모두를 포함한다. 라디오모듈로 새로운 패킷이 전송되거나 센서에서 새로운 값이 감지되는 것과 같은 것들이 하드웨어적 인터럽트에 포함된다. 소프트웨어적 인터럽트는 컴퍼넌트 상에서 작업의 결과를 알리거나 특정한 사건이 발생하였다는 것을 다른 컴퍼넌트로 통보할 때 발생한다. 하드웨어적 인터럽트는 여러 단계를 거치면서 소프트웨어적 인터럽트화 된다. Tinyos에서는 용어 Event가 하드웨어적 인터럽트와 소프트웨어적 인터럽트 모두를 지칭한다[20].

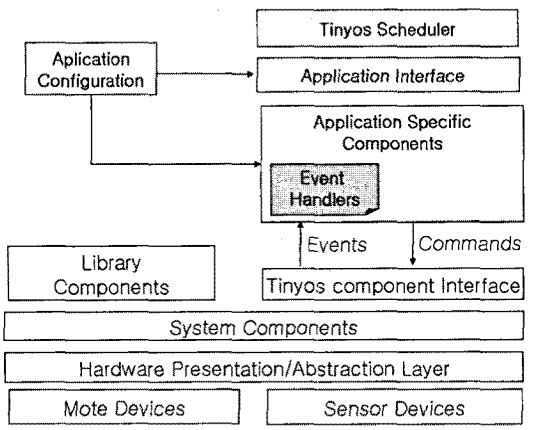

그림 4. TinyOs의 계층도
Fig 4. Layer of TinyOS

한 컴퍼넌트가 다른 컴퍼넌트로 작업을 요청한 경우 작업을 수행하는 컴퍼넌트에서 작업의 결과를 Event로 알려준다면 작업을 요청하는 컴퍼넌트는 Event를 처리

할 Event Handler를 가지고 있어야만 한다. 컴퍼넌트상에는 소프트웨어적 인터럽트와 하드웨어적 인터럽트 모두를 처리할 수 있는 Event Handler가 있어만 한다. 운영체제의 스케줄러는 하드웨어적으로나 소프트웨어적으로 발생하는 이벤트를 적절하게 처리하는 임무를 담당한다. 그림 5는 Tinyos의 스케쥴러를 보여준다. 스케줄러는 인터럽트 정보에 해당하는 인터럽트 벡터를 가지고 있으며 인터럽트가 발생하는 경우 인터럽트 벡터에서 인터럽트에 대한 인터럽트 서비스 루틴(ISR)을 찾아 수행한다.

Tinyos에서는 Event Handler가 인터럽트 서비스 루틴에 해당한다. Event Handler나 Command에서 처리해야 할 작업이 많은 경우나 긴 시간이 필요한 경우 task를 생성하여 작업을 뒤로 미룰 수 있다. 하나의 작업을 처리하는데 너무 긴 시간이 소모되면 다른 작업들을 처리하는데 지연이 발생할 수 있다. 실시간 처리를 필요로 하는 센서노드에서 작업에 지연이 생기면 중요한 데이터가 손실될 수 도 있다. Tinyos에서는 event나 command에서 시간이 오래 걸리는 작업을 처리하기 위해 Task를 사용한다. Task는 linux의 커널에서 인터럽트 처리시 top half와 bottom half로 나누어 처리하는 것과 비슷한 개념이다. Tinyos에서는 모든 event(interrupt)들을 처리한 후에 task를 처리한다.

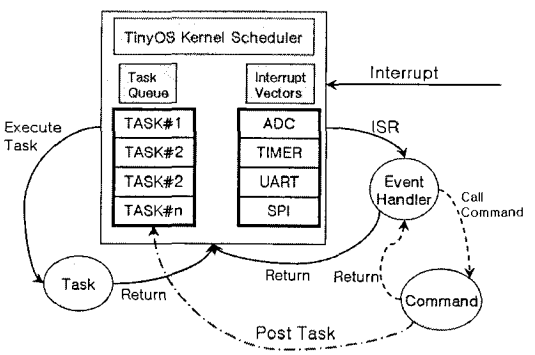

그림 5. TinyOS의 스케줄러
Fig 5. Scheduler of TinyOS

4.2 Split Phase

ADC와 같은 I/O장치에 입출력을 수행하거나 Flash Memory에 대한 접근을 수행하는 경우 CPU와의 속도 차이로 인해 많은 지연이 발생할 수 있다. 이 경우 그림 6의 (A)에서처럼 블로킹 방식을 사용한다면 ADC나 Flash Memory를 담당하는 컴퍼넌트에 작업을 요청한 컴퍼넌트는 이후의 작업을 처리하지 못하고 대기하게 된다. 그림 6의 (A)에서 Component1이 Component2에서 제공하는 command인 goCmdX()를 실행하는 경우 goCmdX에서 처리해야할 작업이 많다면 Component1은 goCmdX가 작업을 마무리할 까지 대기하여야 한다 [20].

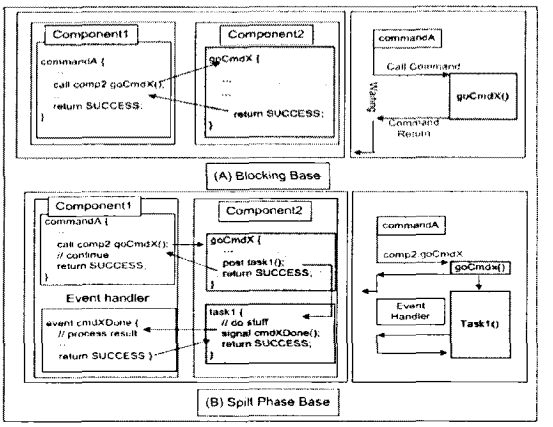

그림 6. 블록킹 베이스와 스플릿 패이즈 베이스
Fig 6. Blocking base and Split Phase base

Tinyos에서는 Spilt Phase개념을 사용하여 문제를 해결한다. Spilt phase는 두 개의 작업을 나누어서 처리한다는 개념이다. Spilt phase는 시간이 오래 걸리는 작업은 나중에 실행하도록 task를 생성하고 작업을 요청한 컴퍼넌트로 결과를 알려주는 방식으로 작업을 요청한 컴퍼넌트의 지연을 줄인다. Split phase를 사용하는 경우 요청한 작업에 대한 결과가 두번 리턴된다. 첫번째는 요청한 작업을 수행할 수 있는지 여부에 대한 결과값이다. 그림 6의 (B)에서 Component1에서 Component2에 있는 goCmdX를 실행하는 경우 goCmdX는 작업을 모두 실행하고 결과를 리턴하는 것이 아니라 task를 생성(post task1부분)하고 작업을 시작했다는 결과를 리턴해 준다. goCmdX에서 리턴되는 SUCCESS의 의미는 작업을 모두 완료하였다는 것이 아니라 작업을 성공적으로 시작했다는 의미이다. Component1는 결과값을 리턴받은 후 남은 일을 다시 시작할 수 있기 때문에 지연이 줄어든다.

V. SET

Tinyos에서는 주기적인 작업을 수행하기 위해 Timer를 사용한다. Timer는 설정된 주기로 Timer Fired Event를 발생시킨다. 10분 주기로 주위의 온도를 센싱하는 어플리케이션이 실행되는 경우 Timer는 10분 간격으로 Timer Fired Event를 발생시키며 Timer Fired Event Handler에서 온도를 센싱하여 네트워크로 전송하는 작업을 수행한다. 무선네트워크 상에서 하나의 패킷을 전송하는데 어느 정도의 시간이 걸릴지 예측하기는 매우 어렵다. 무선환경이 변화가 매우 심하며 매체를 공유하여 노드간에 경쟁이 필요하기 때문에 환경의 변화가 심하고 경쟁하는 노드의 수가 증가할수록 패킷을 전송하는 시간도 증가한다. 패킷을 전송하는데 소요되는 시간을 예측하기 어렵기 때문에 Timer Event를 이용하여 여러 개의 패킷을 연속적으로 보내고자할 때 Timer Event의 발생주기를 얼마로 설정하여야 위와 같은 문제가 발생하지 않을 것인가를 결정하기 어렵다. 주기를 너무 길게 설정하면 위와 같은 문제는 발생하지 않겠지만 패킷 전송 사이에 지연이 증가할 수 있다.

Timer Event에 기반하여 연속적으로 패킷을 전송할 때에 지연이 발생하는 이유는 패킷을 전송하는데 소요되는 시간을 정확하게 예측하기 어렵다는 것이다. 주변 노드수나 혼잡정도에 따라 전송이 완료될 때까지 걸리는 시간은 매우 가변적이다. 이러한 상황에서 패킷을 전송할 최적에 주기를 찾는 것은 매우 어려운 일이다. 패킷 전송간에 지연을 없애기 위해서는 패킷 전송이 완료되는 시점에 다음 패킷을 전송하도록 하는 것이 최선에 방법일 것이다. SET에서는 이 문제를 해결하기 위해 SendDone Event를 이용한다. SendDone Event는 패킷 전송을 완전하게 마쳤을 때 발생하는 이벤트이다. 즉 Send 컴퍼넌트는 패킷 전송을 끝마치는 시점에서 패킷 전송을 완료하였다는 것을 상위 어플리케이션으로 통보하기 위해 SendDone Event를 발생시킨다. SendDone Event가 발생하였다는 것은 패킷 전송이 완료되었다는 것을 의미하며 다음 패킷을 전송할 수 있다는 것이다. SendDone Event를 이용할 경우 패킷 전송을 완료하는데 소요되는 시간이 아무리 길더라도 지연없이 바로 다음 패킷을 전송할 수 있다. 전송시간이 짧은 경우에도 다음 패킷에 대한 전송요청을 바로 하기 때문에 지연이 발생하지 않는다.

그림 7은 SET에서 SendDone Event를 이용하여 연속적으로 패킷을 전송하는 시퀀스타이어그램이다. Timer Fired Event가 발생하면 어플리케이션의 컴퍼넌트는 Send컴퍼넌트로 Send작업을 요청한다. Send 컴퍼넌트는 패킷을 전송할 수 있는지 여부를 판별하고 가능한 경우 task를 생성하고 SUCCESS를 리턴하여 작업을 시작하였음을 통보한다. 이부분까지는 Timer Event를 이용하여 패킷을 전송하는 경우와 동일하다. 하지만 SET에서는 이후의 패킷 전송을 위해 Timer Event를 주기적으로 발생시키지 않는다. 다음 패킷을 전송할 시점을 SendDone Event가 알려주기 때문이다.

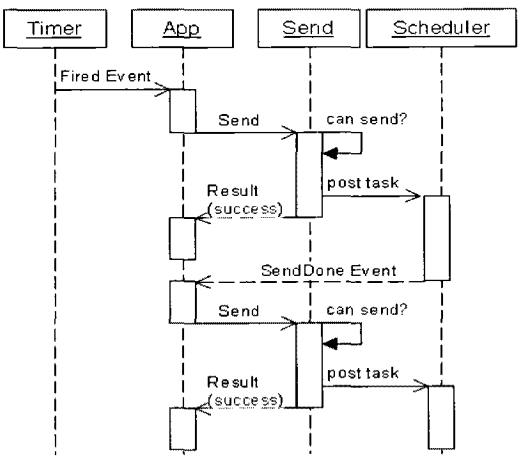

그림 7. 패킷 전송 시퀀스 다이어그램
Fig 7. Packet transmission sequence diagram

Send컴퍼넌트에서 전송작업을 마치면 SendDone Event를 발생시키고 어플리케이션의 컴퍼넌트의 SendDone Event에 대한 Event Handler가 실행된다. Timer Event에 의존하는 경우 SendDone Event Handler에서 별다른 작업을 수행하지 않지만 SET에서는 다음 패킷에 대한 전송을 요청한다. 패킷전송을 완료하였다는 SendDone Event가 발생한 직후이기 때문에 Send작업을 요청하면 전송중에 있는 패킷이 있는 경우가 없다. 즉 Send 컴퍼넌트가 리턴하는 값은 항상 SUCCESS인 것이다. 따라서 SET에서는 전송요청이 실패하는 경우가 발생하지 않으며 즉각적으로 패킷을 전송하기 때문에 지연이 거의 발생하지 않는다.

Timer Event와 SendDone Event를 이용하여 연속적으로 패킷을 전송할 때 가장 큰 차이는 패킷 전송 간의 지연에 있다. Timer Event를 이용하는 경우 전송작업을 수행하는데 걸리는 시간을 예측하기 어렵기 때문에 Timer Event의 주기에 따라서 지연이 가변적일 수 있다. 이에 비해 SendDone Event를 이용하는 경우에는 하나의 전송작업이 끝난 직후에 바로 다음 전송작업을 시작하기 때문에 지연이 거의 발생하지 않는다. 따라서 연속적인 패킷 전송시에는 Timer Event보다는 SendDone Event를 이용하는 것이 더 효과적이라 할 수 있다.

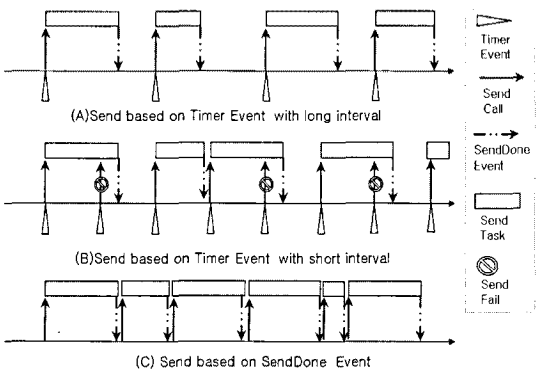

그림 8. 이벤트 사용비교
Fig 8. Event usage comparison

그림 8은 Timer Event와 SendDone Event를 이용하는 경우에 시간상의 차이를 보여준다. 그림 8의 (A)와 (B)는 Timer Event를 사용하는 경우이며 (C)는 SendDone Event를 사용하는 경우이다. (A)는 Timer Event를 사용할 때 Event간 간격을 길게 주었을 때, 즉 이벤트 발생 주기를 길게 하였을 경우를 나타내며 (B)는 간격을 짧게 주었을 경우를 보여준다. (A)를 보면 Timer Event의 간격이 Send작업에 필요한 시간보다 길게 설정되었기 때문에 Timer Event에서 Send작업 요청시 다른 전송작업이 진행중인 경우가 없다. 따라서 Send컴퍼넌트는 항상 SUCCESS를 리턴하고 작업 요청이 실패하는 경우는 없다. 하지만 그림에서 보듯이 Timer Event간의 간격이 너무 길기 때문에 Send작업이 끝나고 다음 Send작업이 시작될때까지 지연이 많이 발생하는 문제점이 있다. Send작업이 빠르게 진행될수록 전송 간에 발생하는 지연시간은 증가한다.

반대로 (B)에서처럼 Timer Event의 발생주기를 너무 짧게 설정하는 경우에는 Send작업이 마무리되기 전에 다음 Send작업을 요청하는 일이 발생하기 때문에 FAIL이 리턴되는 경우가 발생한다. 즉 Timer Event간의 간격이 Send작업을 수행하는데 필요한 시간보다 짧은 경우에 Send작업 요청은 실패한다. 이로 인하여 전송작업이 지연되는 현상을 볼 수 있다. (B)에서 Send작업이 오래 걸리는 경우와 짧게 걸리는 경우에 차이를 볼 수 있다. Send 시간이 Timer Event 발생주기보다 긴 경우에는 전송 도중에 Timer Event가 발생하여 Send요청이 실패한다. Send작업이 완료된 후에 다음 Timer Event까지의 시간만큼 전송에 지연이 발생한다. 만약 Timer Event 발생주기가 매우 짧다면 전송완료 후에 다음 Timer Event가 발생할 때까지 시간차이가 별로 없을 것이다. 하지만 Timer Event가 너무 자주 빠르게 발생하면 Send작업이 완료되기 전에 Timer Event가 빈번하게 발생하여 Send작업을 요청하기 때문에 타이머 이벤트를 처리하기 위한 이벤트 핸들러의 반복적인 실행으로 불필요한 오버헤드가 증가할 수 있다. 반대로 Send작업에 걸리는 시간이 Timer Event보다 짧은 경우에는 (A)에서처럼 Send작업이 끝나고나서 Timer Event가 발생하기 때문에 전송요청이 실패하는 경우가 줄어들며 지연도 줄어든다. 하지만 Send작업에 걸리는 시간이 매우 가변적인 상황에서는 두 가지 경우 모두 발생할 수 있기 때문에 적합한 Timer Event주기를 설정하기란 매우 어려운 일이다.

이에 반해 SET에서처럼 SendDone이벤트를 이용하는 경우에는 Send 작업에 소요되는 시간에 상관없이 Send작업 완료 후에 바로 다음 Send작업을 시작하기 때문에 그림 12의 (C)에서 보듯이 지연이 대폭 줄어든다. SendDone Event가 발생한 시점에는 Send작업이 완료된 상태이기 때문에 Send작업을 요청하였을 때 Send작업이 즉시 수행가능하다. 따라서 Send컴퍼넌트는 항상 SUCCESS를 리턴하며 Send작업이 끝났을 때 SendDone Event가 발생하면 다시 Send작업을 요청할 수 있다. 따라서 Send작업을 수행하는데 걸리는 시간이 길거나 짧은 것에 상관없이 Send작업 요청 사이에 지연은 매우 짧아진다.

VI. 실험평가

6.1 실험환경 및 평가요소

본 논문에서 제안하는 SET기법의 성능을 평가하기 위하여 센서네트워크용 이벤트 기반 운영체제인 Tinyos에서 Timer Event와 SendDone Event를 이용하여 연속적으로 패킷을 전송하였을 때의 성능을 비교하였다. 실험에 사용된 센서노드용 모트는 Crossbow사에서 개발한 Micaz 모트로 표1의 사양을 가진다[22]. Micaz은 센서네트워크용으로 개발된 센서노드로서 8MHz의 저사양의 MCU를 가지며 4KByte의 메모리 공간을 가진다. 무선통신에 사용되는 RF Transceiver로는 Chipcon CC2420칩을 사용하여 최대 250Kbps의 대역폭을 제공한다.

SET의 성능을 측정하기 위하여 노드간 다이렉트 통신에서의 전송율을 측정하였다. 비교대상인 Timer Event를 사용하는 경우에 이벤트의 발생 주기별로 Send 작업 요청시 Fail이 리턴되는 횟수를 측정하였다. Send 작업을 수행하는데 소요되는 시간에 따라 전송율이 어떻게 변하는지 측정하기 위하여 패킷을 전송하는 노드의 수를 1대에서 3대까지 증가시키면서 전송 성공율을 측정하였다. Tinyos에서는 CSMA/CA방식을 사용하여 다른 노드가 패킷을 전송하고 있으면 임의의 시간동안 backoff를 하고 나서 다시 전송이 가능한지 여부를 검사한다. 따라서 전송에 참여하는 노드의 수가 증가할수록 무선통신상에서 경쟁이 심해지기 때문에 패킷 전송에 소요되는 시간은 노드의 수에 비례하여 증가한다. 최종적으로 Timer Event를 이용하는 방식과 SendDone Event를 사용하는 방식에서의 가능한 최대 패킷 전송율을 측정하여 비교하였다

6.2 실험결과

Timer Event를 사용하는 경우 Send작업에 소요되는 시간과 Timer Event의 발생 주기에 따라 Send작업을 요청하였을 때 Fail이 리턴되어 작업 요청이 실패하는 경우가 발생할 수 있다. 그림 9는 노드간 1 : 1 통신에서 Timer Event를 이용하여 연속적으로 패킷을 전송하는 경우에 Timer Event의 발생주기에 따라 Send요청이 실패하는 경우 (Fail이 리턴되는 경우)의 비율을 나타낸다. 그래프에서 가로축은 Timer Event의 발생주기로 1ms에서부터 100ms까지 증가한다. 세로축은 전체 Send요청중 Fail이 리턴되는 비율을 나타낸다. Tinyos상에서 제공하는 Timer는 최대 3ms이상으로 주기를 설정해야 간격을 보장하며 2ms이하는 정확성을 보장하지 못한다. 따라서 그래프상에서 1~2ms정도의 주기에서는 전체 요청의 50~70%정도가 실패하는 것을 확인할 수 있다.

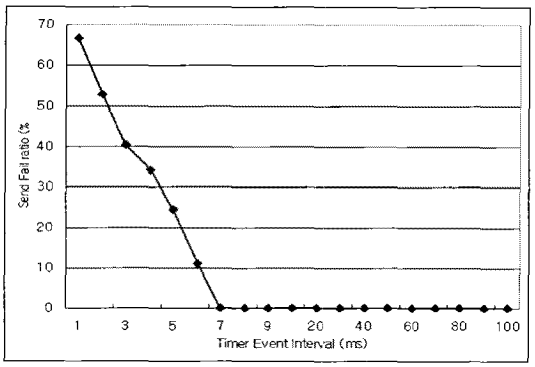

그림 9. 전송 주기에 따른 전송 실패율
Fig 9. Packet transmit failure ratio.

Send작업에 소요되는 시간이 연속적인 패킷 전송에 영향을 미치는지 확인하기 위하여 패킷 전송에 참여하는 모트에 수를 증가시켜서 전송 성공율을 측정하여 비교하였다. 그림10은 모트에 수를 3대까지 증가시키면서 측정한 Send작업 요청의 성공율을 보여준다. 그림 10을 보면 모트의 수가 증가할수록 Send요청에 대한 성공율이 급격히 낮아지는 것을 확인할 수 있다. 모트가 1개일 경우는 그림9와 같은 결과를 보여주고 있다. 모트가 2개일 경우와 3개일 경우에는 모트가 1개일 경우보다 낮은 성공율을 보이고 있다. 모트가 1개일 때에는 7ms정도의 주기로 이벤트가 발생하면 100%의 성공율을 나타내고 있으나 모트가 2개일 경우에는 4배가 넘는 30ms정도의 주기를 가져야만이 90%이상의 성공율을 얻을 수 있었다. 모트가 3개인 경우에는 성능이 더욱 나빠져서 30ms정도의 주기까지는 대부분의 요청이 실패하는 것으로 나타났으며 50ms이상으로 이벤트가 발생할 때 80%정도의 성공율을 얻을 수 있었다. 이는 모트가 1개일 경우보다 7배 이상의 주기를 필요로 하면서도 낮은 성공율을 가지는 것이다.

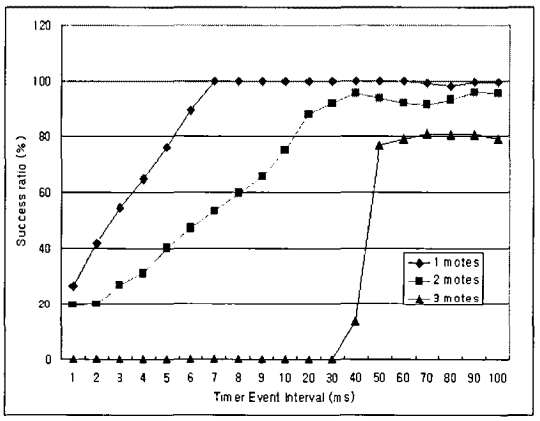

그림 10. 노드수에 따른 패킷 전송 성공률
Fig 10. Packet transmit ratio with motes

Timer Event를 사용하는 것에 비해 SendDone Event를 사용하는 경우 어느 정도의 성능향상을 가져오는지 확인하기 위해 패킷 크기를 변화시키면서 패킷 전송율을 측정하였다. 그림 11은 Timer Event의 발생주기를 4ms로 설정하여 주기적으로 패킷을 전송하도록 하는 경우와 SendDone Event가 발생하였을 때 다음 패킷을 전송하도록 하는 경우의 초당 패킷 전송율을 보여준다. 패킷의 크기를 변화시키면서 전송율을 측정하여 전송시간이 전송율에 미치는 영향을 분석할 수 있도록 하였다.

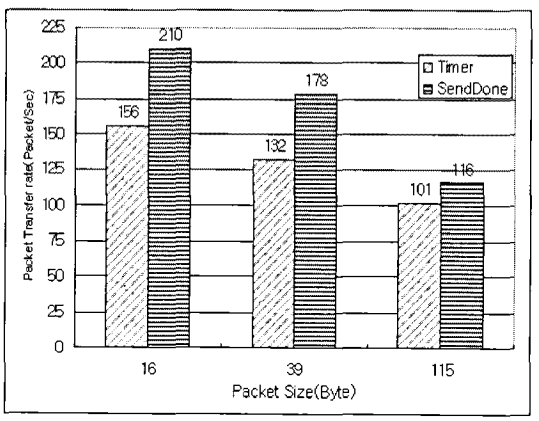

그림 11. 패킷 크기에 따른 전송율
Fig 11. Packet transmit ratio with size

그림 11을 보면 패킷 크기가 커질수록 전송율은 낮아지는 것을 볼 수 있다. 패킷 길이가 길어질수록 전송하는데 소요되는 시간이 증가하기 때문에 전송율이 낮아진다. 패킷 길이가 길어지면 전송율이 낮아지기는 하지만 전체적으로 Timer Event를 이용하는 것보다 SendDone Event를 이용하는 것이 높은 전송율을 나타내는 것을 확인할 수 있다.

Ⅶ. 결론

센서네트워크가 다양한 분야에서 사용되면서 시스템에 대한 요구사항도 다양해지고 있다. 적용되는 어플리케이션에 따라 이벤트를 감지하는 주기가 매우 길거나 반대로 매우 짧아질 수 있으며 이에 따라 생성되는 데이터의 양도 달라진다. 매우 정밀한 감지를 위하여 이벤트를 감지하는 주기가 매우 짧아지면 생성되는 데이터의 양이 증가하며 네트워크로 전송해야하는 패킷수도 증가한다. 많은 양의 데이터를 빠르게 연속적으로 전송하기 위해서는 패킷 전송 사이에 지연을 최소화시켜야만 한다.

일반적으로 패킷을 연속적으로 전송하기위해 주기적으로 발생하는 Timer Event를 사용한다. Timer Event를 이용하면 보낼 패킷의 수만큼 이벤트를 발생하여 연속적으로 패킷을 전송하도록 할 수 있다. 하지만 패킷을 전송하는데 소요되는 시간이 매우 가변적이기 때문에 전송요청이 실패하거나 전송간에 지연이 발생할 수 있다. 본 논문에서는 센서네트워크용 이벤트기반 운영체제에서 연속적인 패킷 전송을 효율적으로 하기 위한 방안으로 SET기법을 제안하였다. SET는 하나의 패킷을 전송하고 다음 패킷을 전송하는 것을 SendDone Event Handler에서 수행하도록 하여 패킷 전송간에 지연을 줄이고자 하였다. 센서네트워크용으로 사용되는 모트를 이용한 성능평가를 통하여 Timer Event를 이용하는 방법의 문제점을 확인하였고 SendDone Event를 이용하는 것이 Timer Event를 이용하는 것보다 15~35%정도 많은 전송율을 제공할 수 있음을 보였다.

참고문헌

[1] N. Xu et al., "A Wireless Sensor Network for Structural Monitoring," Proc. ACM Symp. Networked Embedded Sensing (Sensys), ACM Press, 2004, pp.13 - 24.

[2] J. Paek et al., "A Wireless Sensor Network for Structural Health Monitoring Performance and Experience," Proc. 2nd IEEE Workshop on Embedded Networked Sensors,IEEECSPress,2005

[3] S. Madden et al., TAG, "A Tiny Aggregation Service for Ad Hoc Sensor Networks," Proc. Usenix Symp. Operating Systems Design and Implementation, Usenix Assoc., 2002

[4] S. Kim, D. Culler, and J. Demmel, "Structural health monitoring using wireless sensor networks," Berkeley Deeply Embedded Network System Course Report, 2004

[5] Kottapalli, Venkata A., Kiremidjian, Anne S., Lynch, Jerome P., Carryer, Ed, Kenny, Thomas W., Law, Kincho H., Lei, Ying, "Two-tiered wireless sensor network architecture for structural health," monitoring SPIE, Volume 5057, pp.8-19 (2003).

[6] J. A. Stankovic, T. Abdelzaher, C. Lu, L. Sha, J. Hou, "Real-Time Communication and Coordination in Embedded Sensor Networks," Proceedings of the IEEE, 91(7): 1002-1022, July 2003.

[7] C. Wan, A. Campbell, and L. Krishnamurthy. "A reliable transport protocol for wireless sensor networks." WSNA2002.

[8] S. Kim, R. Fonseca, and D. Culler. "Reliable Transfer on Wireless Sensor Networks." SECON2004.

[9] S. Rangwala , R. Gummadi, R. Govindan, K. Psounis, "Interference-aware fair rate control in wireless sensor networks", ACM SIGCOMM 2006, Pages: 63 - 74

[10] W. Hu, Nirupama Bulusu, S. Jha, "A communication paradigm for hybrid sensor/actuator networks," in Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC2004).

[11] A. Boukerche, R. Werner Nelem Pazzi, R. Borges de Araujo, "A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications." MSWiM 2004: 157-164

[12] Y. Zhu, R. Sivakumar, "Challenges: communication through silence in wireless sensor networks." MOBICOM 2005: 140-147

[13] A. S., Lochmann, S., "Distinct Enlargement of Network Size or Measurement Speed for Serial FBG Sensor Networks utilizing SIK-DS-CDMA", Proc. of Sensors and their Applications, London, England, 6-8 Sep 2005

[14] T. He, J. A. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A stateless protocol for real-time communication in sensor networks," In Proc. of the 23rd International Confernece on Distributed Computing Systems (ICDCS-23), Providence, RI, USA, May 2003.

[15] W. A., Culler, D."A Transmission Control Scheme for Media Access in Sensor Networks." International Conference on Mobile Computing and Networking, (Mobicom), p.221-235, July 2001.

[16] S. Dulman, L. v. Hoesel, T. Nieberg, P. Havinga: "Collaborative Communication Protocols for Wireless Sensor Networks", Workshop on European Research on Middleware and Architectures for Complex and Embedded Systems, Pisa, Italy, April. 2003.

[17] http://www.tinyos.net

[18] D. Gay, P. Levis, and D. Culler, "Software Design Patterns for TinyOS," Proceedings of the ACMS IGPLAN/SIGBED 2005 Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'05), Chicago, June 2005.

[19] 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," Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI2004), San Francisco, March 2004

[20] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler," The nesC Language: A Holistic

Approach to Networked Embedded Systems," Proceedings of Programming Language Design and Implementation (PLDI), San Diego, June 2003.

[21] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson, "Wireless Sensor Networks for Habitat Monitoring,"2002 ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, September 2002.

[22] Crossbow Inc, http://www.xbow.com/

저자소개

임동선(Dong-Sun Lim)

1986년 숭실대학교 전자계산학과 (학사)
1995년 한국과학기술원(석사)
2005년~현재 강원대학교 컴퓨터 정보통신공학과(박사수료)
1986 ~ 현재 한국전자통신연구원 책임연구원
※관심 분야: 멀티미디어 시스템, 센서 네트워크

이좌형(Joa-Hyoung Lee)

2003년 강원대학교 정보통신공학과(공학사)
2005년 강원대학교 컴퓨터정보통신 공학과(공학석사)
2005년~현재 강원대학교 컴퓨터정보통신공학과 (박사과정)
※관심 분야: 멀티미디어 시스템, 센서 네트워크

정인범(In-Bum Jung)

1985년 고려대학교 전자공학 학사.
1985년~1995년 ㈜ 삼성전자 컴퓨터 시스템사업부 선임 연구원.
1992년~1994년 한국과학기술원 정보통신공학과 공학석사
1995년~2000년 8월 한국과학기술원 전산학과 공학박사
2001년~현재 강원대학교 컴퓨터정보통신공학전공 교수
※관심분야: 멀티미디어 시스템, 센서네트워크