무선 센서 네트워크에서 신속한 코드 전송 기법
(A Fast Code Propagation Scheme in Wireless Sensor Networks)

이 한 선 † 정 광 수 ††
(Hansun Lee) (Kwangsue Chung)

요 약 무선 센서 네트워크를 구성하는 센서 노드는 한번 배치되면 사람의 간섭 없이 오랜 기간 동안 동작하는데 실행중인 소프트웨어를 수정 또는 추가를 할 필요가 있다. 그러나 센서 노드를 회수하기 어려운 경우가 있기 때문에 원격 코드 업데이트 기법이 필요하게 되고, 이를 위한 신뢰성 있는 코드 전송 프로토콜에 대한 연구가 활발하게 진행되고 있다. 하지만 신뢰성만을 고려한 코드 전송 프로토콜은 코드를 안정적으로 전송하기만을 고려하기 때문에 코드를 신속하게 전송한다는 관점에 대한 고려가 부족하다는 한계를 갖는다. 그 결과 긴 코드 전송시간에 의해 불필요한 에너지 소모를 발생함으로써 센서노드의 에너지 효율을 저하시키게 된다.

본 논문에서는 기존의 코드 전송 프로토콜들이 가지는 한계를 극복하는 FCPP(Fast code propagation protocol)을 제안하였다. FCPP는 신뢰성 있는 전송뿐만 아니라 신속함을 고려한 접근 방법을 제시하고 있다. 새로 제안한 알고리즘은 RTT기반의 전송률 조절과 NACK 억제 기법으로 네트워크 상태를 반영한 전송률 조절과 에러복구에 의한 불필요한 전송지연을 피하도록 하여 네트워크의 사용률을 최대화하여 신속한 코드 전송을 가능하게 한다. 또한 ns-2 시뮬레이터를 이용한 실험을 통해 제안한 FCPP가 센서 네트워크의 코드 전송에서 신뢰성 및 신속함을 모두 만족시킬 수 있음을 확인하였다

키워드 : 무선 센서 네트워크, 네트워크 프로그래밍, 코드 전송, 무선 네트워크

Abstract Once the sensor node in wireless sensor networks is installed, it usually operates without human intervention for a long time. The remote code update scheme is required because it is difficult to recall the sensor node in many situations. Therefore, studies on the reliable and efficient transport protocol for code propagation in wireless sensor networks have been increasingly done. However, by considering only the stability aspect of transmission, most of previous works ignore the consideration on the fast code propagation. This results the energy inefficiency by consuming *unnecessary energy due to the slow code propagation.*

In this paper, in order to overcome limitation of the previous code propagation protocols, we propose a new code propagation protocol called "FCPP(Fast Code Propagation Protocol)". The FCPP aims at improving the reliability as well as performance. For this purpose, the FCPP accomplishes the fast code propagation by using the RTT-based transmission rate control and NACK suppression scheme, which provides a better the network utilization and avoids a unnecessary transmission delay. Based on the ns-2 simulation result, we prove that the FCPP improves significantly both reliability and performance.

Key words : Wireless Sensor Networks, Network Programming, Code Propagation, Wireless Networks

· 본 연구는 건설교통부 첨단도시기술개발사업 - 지능형국토정보기술혁신 사업과제의 연구비지원(07국토정보C03)에 의해 수행되었습니다.

† 학생회원 : 광운대학교 전자통신공학과
hslee@adams.kw.ac.kr
†† 종신회원 : 광운대학교 전자통신공학과 교수
kchung@kw.ac.kr

논문접수 : 2006년 2월 2일
심사완료 : 2007년 10월 25일

정보과학회논문지: 정보통신 제35권 제1호(2008.2)

1. 서 론

최근 정보통신 기술의 비약적인 발전으로 기존 계산기로써의 컴퓨터는 정보단말로써 발전하여 우리의 생활에 밀접한 영향을 주고 있다. 정보통신 기술의 진보는 유비쿼터스 컴퓨팅(Ubiquitous computing)이라는 새로운 정보통신 혁명을 야기하게 되었고, 사회 발전의 흐름과 끊임없이 환경을 인간 친화적으로 바꾸고 싶어 하는 인간의 욕구와 맞물려 무선 센서 네트워크의 필요성이 제기되고 있다.

무선 센서 네트워크란 센서가 달려 있어 센싱이 가능하고 센싱된 정보를 가공할 수 있는 프로세서가 달려 있으며 이를 전송할 수 있는 무선 송수신기를 갖춘 소형 장치, 즉 센서 노드로 구성된 네트워크를 의미한다. 기존의 네트워크와 다르게 의사소통의 수단이 아니라 다수의 센서 노드를 이용하여 환경의 변화, 수질 오염, 지진 활동, 건물의 구조적 상태 등에 대한 정보를 수집하는 것을 목적으로 한다[1].

센서 노드는 한번 배치되면 사람의 간섭 없이 오랜 기간 동안 동작하지만 수집된 환경 정보가 완벽하지 않아 회수하지 않고는 완벽한 동작의 수행이 불가능한 경우가 존재한다. 그러나 실제로는 나무 꼭대기처럼 물리적으로 닿기 힘든 곳이나 둥지 안과 같이 센서 노드를 회수하는 과정에서 정보 수집에 피해를 주게 되는 다수의 시나리오가 있다. 이런 어려움에도 실행중인 소프트웨어를 수정 또는 추가할 필요가 있기 때문에 원격 코드 업데이트 기법이 필요하게 된다. 원격 코드 업데이트는 무선 센서 네트워크의 확장성을 제공하고 소프트웨어의 유지보수와 디버깅을 가능하게 한다. 부가적으로, 대규모의 네트워크에서도 멀티 홉 전송기법을 사용하여 인위적인 코드 업데이트 과정을 자동화할 수 있다[2].

원격 코드 업데이트 과정에서 패킷 손실은 코드 업데이트의 실패를 야기하기 때문에 신뢰성 있는 전송 프로토콜이 필요하게 되고, 이를 위한 연구가 활발하게 진행되고 있다[2-7]. 이러한 연구들은 신뢰성 있는 코드 전송을 위해 손실된 패킷에 대한 안정적인 재전송을 목적으로 하지만 코드 업데이트에 소모되는 시간에 대한 고려가 부족하다는 한계를 갖는다. 모든 센서 노드는 원격 코드 업데이트 과정에서 동작 상태로 에너지를 소모하게 되므로 긴 코드 업데이트 시간은 많은 에너지를 소모하여 네트워크의 수명을 단축시키게 된다. 이러한 문제점을 개선하기 위해 신속한 코드 업데이트에 대한 연구가 요구된다.

본 논문에서는 무선 센서 네트워크 환경에서 신속하고 신뢰적인 코드 전송을 위한 새로운 프로토콜인 FCPP (Fast code propagation protocol)를 제안한다. FCPP는 신속한 코드 전송을 위해 네트워크 상태에 따른 전송률 조절과 NACK(Negative acknowledgment) 기반의 에러복구에 의한 불필요한 전송지연을 피하도록 설계되었다. 그럼으로써 신뢰성 있는 코드 전송뿐만 아니라 신속한 코드 전송을 가능하게 한다.

본 논문의 2장에서는 무선 센서 네트워크 환경에서 코드 전송을 위한 고려사항과 관련 연구에 대하여 기술하였고, 3장에서는 기존 연구들의 단점을 개선하기 위해 새롭게 제안한 FCPP의 알고리즘에 대해 기술하였으며, 4장에서는 시뮬레이터를 이용하여 제안한 알고리즘의 성능을 검증하고 기존 코드 전송 프로토콜과의 성능 비교 결과를 기술하였다. 마지막으로 5장에서는 결론을 맺었다.

2. 관련연구

본 장에서는 무선 센서 네트워크 환경에서 코드 전송을 위한 고려 사항과 관련 연구에 대해서 기술한다. 먼저 코드 전송 프로토콜을 설계하기 위해 무선 센서 네트워크의 특성과 기존 전송 프로토콜의 문제점을 지적하여 코드 전송을 위한 고려사항을 살펴보고 코드 전송을 위한 기존의 관련 연구에 대해 간략한 소개와 문제점을 기술하였다.

2.1 코드 전송을 위한 고려 사항

무선 센서 네트워크에서 코드 업데이트를 위해 전송된 코드 패킷의 일부분이 손실될 경우 나머지 코드 패킷은 쓸모없게 되기 때문에 코드 전송을 위한 신뢰성 있는 전송 프로토콜이 요구된다. 현재 유선 네트워크에서 사용되는 신뢰성 있는 전송 프로토콜인 TCP(Transmission control protocol)는 패킷 손실의 원인을 네트워크의 혼잡(Congestion)으로 보고 혼잡 제어(Congestion control)에 초점을 두고 있다. 그러나 무선 센서 네트워크에서 패킷 손실은 무선 채널간의 간섭(Interference)이나 낮은 전력에 의해 발생하기 때문에 이를 고려한 신뢰성 있는 전송 프로토콜이 필요하게 된다.

센서 노드는 전원 공급을 배터리에 의존하기 때문에 한정된 에너지를 가지고 동작한다. 그렇기 때문에 에너지를 효율적으로 사용하여 에너지 소모를 최소화하여야 하는데 이를 위해 전송 프로토콜 관점에서는 전송되는 패킷 수를 줄이는 것을 고려해야 한다. 신뢰성 있는 전송 프로토콜에서 데이타 패킷은 확실히 전송되어야 하기 때문에 데이타 패킷보다는 제어 패킷의 수를 최소화하는 것이 중요하다. 그래서 전송 데이타 패킷에 대해 응답을 받는 ACK(Acknowledgment) 기반의 전송 프로토콜보다는 손실된 데이타 패킷에 대해서만 알려주는 NACK 기반의 전송 프로토콜이 고려될 수 있다.

신뢰성 있는 전송을 위해서는 손실된 패킷을 복구하

는 재전송 기법이 요구되는데 크게 두 가지로 종단간(End-to-end) 에러복구 방법과 홉간(Hop-by-hop) 에러복구 방법으로 나눌 수 있다. 종단간 에러복구 방법은 송신 노드와 수신 노드만 에러복구에 관여한다. 패킷이 손실되었을 경우, 수신 노드는 복구 요청을 역 경로를 통하여 송신 노드로 전송한다. 그러나 무선 환경은 높은 에러율을 가지기 때문에 복구 요청을 전송하는 과정에서 중계 노드에서 손실될 수 있고 수신 노드는 송신 노드로 복구 요청을 재전송하게 된다. 따라서 전송되는 패킷수가 증가하게 된다는 단점을 가지고 있다.

반면 홉간 에러복구 방법은 송신 노드와 수신 노드뿐만 아니라 중계(Intermediate) 노드도 에러복구에 관여한다. 패킷이 전송되는 경로의 중계 노드는 수신된 패킷을 저장하고 패킷이 손실되었을 경우, 이전 노드로 복구 요청을 한다. 만약 중계 노드가 요청한 패킷을 저장하고 있을 경우, 복구를 요청한 노드로 패킷을 재전송하게 된다. 따라서 패킷 손실률이 높은 무선 환경에서 종단간 에러복구 방법보다 적은 패킷을 전송한다. 그림 1은 두 방법을 비교한 것으로 10개의 패킷을 10홉에 걸쳐 전송하였을 때 전송된 패킷 수를 나타낸다. 홉당 전송 성공률이 낮아짐에 따라 홉간 에러복구에 비해 종단간 에러복구는 현저하게 많은 패킷을 재전송하는 것을 확인할 수 있다. 재전송이 증가함에 따라 센서 노드의 에너지 효율은 떨어지게 되고 센서 네트워크의 수명을 줄이는 원인이 된다. 따라서 코드 전송을 위해서는 종단간 에러복구 방법 보다는 홉간 에러복구 방법이 선호된다.

원격 코드 업데이트 과정에는 신뢰성 있는 전송뿐만 아니라 신속한 코드 전송도 고려될 필요가 있다. 모든 센서 노드는 원격 코드 업데이트 과정에서 동작 상태로 유휴 청취(Idle listening)를 하게 되고, 긴 코드 업데이트 시간은 많은 에너지를 소모하여 네트워크의 수명을 단축시키게 된다. 유휴 청취 상태에서 엿듣기(Over-hearing)에 의한 에너지 소모를 줄이기 위해 Active/Sleep 과정을 반복하여 네트워크 수명을 최대화 하는 기법 등이 제안되고 있지만 긴 전송 지연시간을 갖는

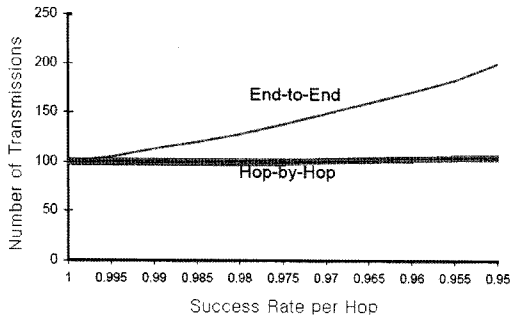


그림 1 홉간 및 종단간의 전송 패킷 수 비교

단점을 갖는다. 따라서 센서 노드의 동작 상태를 최소화하기 위한 신속한 코드 업데이트 방법이 필요하다.

코드 업데이트에서 센서 노드는 수신된 코드 패킷을 저장하기 위해 충분한 메모리 공간을 확보해야 한다. 그렇지 못할 경우 수신된 코드 패킷은 손실되게 되고 코드 업데이트 과정을 수행하지 못하게 된다. 코드 전송을 위한 메모리 공간이 보장되어있기 때문에 수신 버퍼의 크기로 인한 패킷 손실은 고려하지 않아도 무방하다.

2.2 코드 전송을 위한 관련 연구

무선 센서 네트워크 환경에서 코드를 전송하기 위해, 2.1절에서 기술한 사항을 고려한 코드 전송 프로토콜로 PSFQ[3], Deluge[4], XNP[5] 등 많은 연구들이 수행되어 왔다.

2.2.1 PSFQ(Pump slowly, fetch quickly)

PSFQ는 무선 센서 네트워크를 위한 OS(Operating system)인 TinyOS[8]에 구현되어 멀티 홉을 통한 코드 업데이트 전송 기능을 제공한다.

PSFQ는 NACK 기반의 신뢰성 있는 전송 프로토콜로 데이타 저장을 통해 순차적인 패킷 전송을 보장한다. 느리게 데이타를 전송(Pump slowly)함으로써 손실된 패킷을 복구하기 위한 충분한 시간을 보장하고, 빠르게 손실된 패킷을 복구(Fetch quickly)함으로써 신뢰적인 재전송을 한다. Pump 동작을 위해서 T_{min}과 T_{max}의 전송 타이머 값을 갖고, 둘 사이의 값으로 패킷을 전송 간격을 결정한다. Fetch 동작을 위해서는 $T_r(<T_{max})$의 에러복구 타이머 값을 가지며 에러복구를 실패하여 타이머가 완료되면 NACK를 재전송한다. 에러복구를 위한 타이머 값은 Pump/Fetch의 비율로 결정된다.

그림 2는 PSFQ의 동작과정을 보여준다. 3번 패킷이 손실된 경우, 노드 A는 4번 패킷을 전송한 후, 전송 타이머를 T_{min}과 T_{max} 사이의 임의의 값으로 설정한다. 노드 B는 4번 패킷을 비순차적으로 수신함으로써 패킷이 손실됨을 인지하고 손실된 패킷을 복구하기 위해 Fetch 동작을 수행하여 NACK를 전송한다. 이 때, 노드 B는 순차적으로 패킷을 전송하기 위해 비순차적으로 수신한 4번 패킷을 저장하고 노드 C로 전송을 하지 않는다. 에러복구 타이머 Tr 동안 NACK에 대한 응답이 없을 경우, NACK를 재전송한다. 노드 A는 NACK를 수신하여 3번 패킷을 재전송하고 전송 타이머가 완료된 후, 다음 패킷을 전송한다. 노드 B는 손실된 3번 패킷을 재전송 받은 후, 순차적으로 이웃 노드 C로 전송하게 된다.

PSFQ는 고정된 전송 타이머 값과 에러복구 타이머 값을 가지게 되는데, 네트워크의 상태에 상관없이 전송 간격으로 T_{min}이상의 값을 가지게 된다. 식 (1)은 PSFQ에서 코드의 최소 전송 시간 D(n)을 나타내는데, T_{min}은 최소 전송간격, n은 코드의 분할 수, h는 홉 수를 의

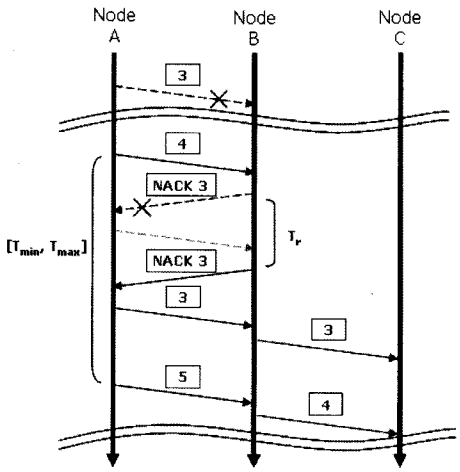


그림 2 PSFQ 동작

미한다. 코드의 분할 수나 홉 수가 증가함에 따라 코드의 전송 시간도 증가하게 된다. 또한 고정된 타이머 값을 사용함으로써 충분한 네트워크 자원을 사용하지 못한다는 단점을 가지고 있다.

$$D(n) = T_{\min} \times n \times h \quad (1)$$

2.2.2 Deluge

Deluge는 코드를 페이지 단위로 분할하고 이를 다시 패킷 단위로 분할하여 전송하며 이로 인해 전체 코드의 일부만 가진 노드가 코드를 전송하는 소스 노드가 되어 동작할 수 있게 한다. 분할된 코드 패킷은 파이프라인(Pipelining)으로 전송되기 때문에 코드 전송을 완료하는 데까지 걸리는 시간이 감소하게 된다. Deluge는 신뢰할 수 있는 코드 전송을 위해서 주기적으로 페이지 번호와 패킷 번호와 같은 자신의 코드 상태를 이웃 노드로 통지(Advertisement)하고 이를 수신한 노드는 필요한 코드 패킷이 있을 경우, 전송을 요청(Request)함으로써 데이타(Data)를 전송하는 3가지 단계를 가진다. 모든 노드는 코드 패킷을 매 주기마다 보낼 수 있으며 만약 특정 주기에 코드 패킷을 보내기 전에 자신이 보내려는 코드 패킷과 비슷한 내용의 코드 패킷을 엿들으면(Overhearing) 자신이 보내려던 코드 패킷의 전송을 중단함으로써 통신량을 줄이고 효율적으로 에너지를 사용하게 한다.

Deluge는 Advertisement-Request-Data의 3가지 단계로 동작하기 때문에 PSFQ와 같이 Data-NACK의 2가지 단계로 동작하는 방식보다 많은 제어 패킷을 전송하게 되어, 결과적으로 많은 에너지를 소모한다는 문제를 가지고 있다.

2.2.3 XNP(In-network programming)

XNP는 TinyOS[8]에 포함되어 있으며 TinyOS의 응용들을 무선 환경을 통해 원격으로 코드를 업데이트하기 위해 개발되었다. XNP의 동작방법은 BS(Base Station)가 자신의 전송 영역안의 센서 노드들에게 분할된 코드를 브로드캐스팅하고 코드의 전송이 끝난 후, 센서 노드들은 BS로 수신하지 못한 코드 패킷에 대한 정보를 NACK 패킷으로 전송하여 손실된 패킷을 복구하게 한다. 그렇지만 XNP는 멀티 홉을 통한 코드 전송을 지원하지 않고 단일 홉을 통한 코드 전송만 지원하기 때문에 센서 노드들이 배치된 무선 센서 네트워크 환경에서 실제로 사용할 수 없다는 단점을 가지고 있다.

2.3 기존 연구의 문제점

2.3.1 네트워크 상태에 대한 고려

무선 센서 네트워크 환경에서 코드 전송을 위한 대다수의 연구들은 신뢰성 있는 코드 전송에 초점을 두고 있다. 특히 기존 연구에서 PSFQ는 안정적인 전송을 위해, 길고 고정된 패킷 전송간격을 사용하여 에러복구를 위한 시간을 보장하기 때문에 긴 코드 전송시간을 가지게 된다. 그러나 모든 센서 노드는 코드를 전송하는 동안 에너지를 소모하기 때문에 긴 코드 전송시간은 불필요한 에너지 소모를 발생함으로서 센서노드의 에너지 효율을 저하시키게 된다. 따라서 신뢰성 있는 코드 전송뿐만 아니라 신속한 코드 전송이 필요하게 된다.

신속한 코드 전송을 위해서는 네트워크 상태를 고려할 필요가 있다. 무선 센서 네트워크에서 패킷 손실은 간섭 또는 낮은 전력에 의해 발생하기 때문에 전송률을 적절하게 조절하여 간섭의 영향을 최소화 하고 네트워크의 사용률을 최대로 하여 신속하게 코드 전송을 한다.

2.3.2 NACK를 이용한 에러복구

무선 센서 네트워크에서 신뢰성 있는 전송 프로토콜은 제어 패킷의 수를 줄이기 위해 NACK 기반으로 동작한다. NACK 기반의 코드 전송 프로토콜은 손실 이벤트 전달을 방지하기 위해 데이타 저장을 통한 순차적인 전송을 한다. 손실 이벤트는 패킷 손실로 인해 비순차적인 패킷을 수신함으로써 발생하게 되는데 손실된 패킷 이후에 수신된 패킷을 이웃 노드로 전송함으로써 손실 이벤트가 전달되게 된다. 그림 3과 같이 노드 A에서 4개의 패킷을 전송 할 때 3번 패킷이 손실되었을 경우, 노드 B는 2번 패킷 수신 후 4번 패킷을 수신함으로써 손실 이벤트를 발생시키고 손실된 3번 패킷을 복구하기 위해 NACK를 전송하게 된다. 노드 B가 수신된 4번 패킷을 이웃 노드로 전송하게 되면 이후의 노드 C, D도 손실 이벤트를 발생시키게 되고 NACK를 전송한다. 그러나 NACK를 수신한 노드 B, C는 3번 패킷을 가지고 있지 않기 때문에 에러복구를 위한 패킷을 전송하지 못하고 NACK를 버리게 된다. 이런 불필요한 NACK의 전송을 방지하기 위해 그림 4와 같이 노드 B

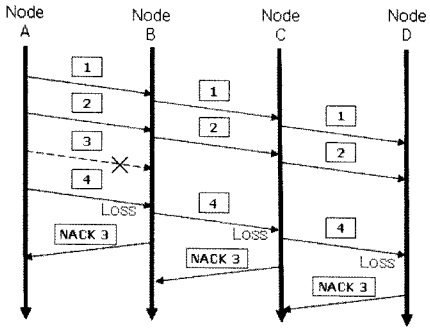


그림 3 손실 이벤트 전달

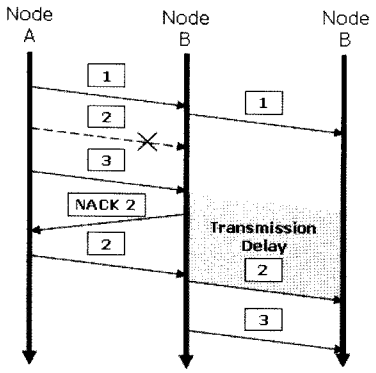


그림 4 에러복구에 의한 전송 지연

에서 2번 패킷이 손실되고 3번 패킷을 수신하였을 경우, 이웃 노드로 3번 패킷을 바로 전송하지 않고 손실된 2번 패킷을 복구한 후 순차적으로 이웃노드로 전송한다.

NACK를 이용한 에러복구 기법에서는 에러복구 동안 수신된 패킷을 전송하지 못하고 에러복구까지 기다리게 됨으로써 불필요한 전송지연을 유발하게 되는 문제점을 갖는다.

3. 신속한 코드 전송 프로토콜 설계

본 장에서는 2.3절에서 지적한 문제점들을 개선하기 위해서 새롭게 제안한 FCPP 알고리즘에 대해 기술한다. 제안한 FCPP는 NACK 기반의 환경에서 엿듣기를 통한 전송률 조절 알고리즘과, 기존의 NACK 기법의 문제점을 해결하기 위한 NACK 억제 기법을 기반으로 설계되었다. 이를 통해 네트워크 상황에 알맞은 트래픽 전송을 통해 전송률을 개선하고, 에러복구 과정에서 불필요한 전송지연을 피하여 신속한 코드 전송 동작을 수행하도록 한다.

3.1 전송률 조절 알고리즘

FCPP는 네트워크의 상태에 따라 전송률을 조절하기 위해 RTT(Round Trip Time)를 기반으로 전송간격을 조절한다. ACK 기반의 전송 프로토콜에서는 데이타 패킷을 전송하고 ACK 패킷을 반송하는 시간의 차이로써 RTT를 계산하게 되는데, NACK 기반의 전송 프로토콜에서는 손실된 패킷에 대해서만 NACK를 수신할 수 있기 때문에 ACK 기반의 전송 프로토콜과 같은 RTT 측정 방법이 불가능하다. 그렇지만 코드 업데이트 과정에서 모든 센서 노드는 동작 상태에 있고, 이웃 노드가 전송하는 패킷을 수신하기 위해 유휴 청취(Idle listening)를 하기 때문에 엿듣기(Overhearing)가 가능하다는 무선 환경의 특성으로 노드에서 패킷을 전송한 시간과 이웃 노드에서 해당 패킷을 재전송하는 시간의 차를 이용하여 RTT를 측정할 수 있다. 또한 엿듣기를 통하여 손실된 패킷이 수신될 경우 손실된 패킷의 재전송 요구 없이 수동적으로 복구 가능하다.

그림 5는 엿듣기를 통하여 RTT를 측정하는 방법을 나타내고 있다. T_{send}는 노드 A에서 패킷을 전송하는 시간을 나타내며 $T_{overhear}$는 이웃노드 B에서 전송하는 패킷을 엿듣는 시간을 의미한다. 엿듣기를 통한 RTT 측정 방법은 식 (1)과 같이 두 시간의 차로 측정할 수 있다. 하지만 무선 네트워크 환경은 불안정하기 때문에 RTT의 값은 큰 변동 폭을 가지게 되므로 식 (2)와 같은 Lowpass filter를 적용한 RTT 값을 사용한다.

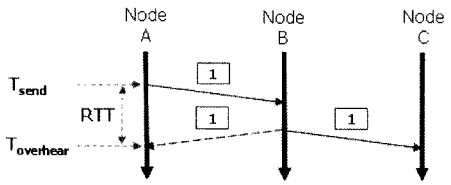


그림 5 엿듣기를 통한 RTT 측정

$$RTT_{estimate} = T \tag{1}$$

$$RTT_{n+1} = \alpha \times RTT_n + (\alpha - 1) \times RTT_{estimate} \tag{2}$$

계산된 RTT를 기반으로 전송률을 조절하기 위해 식 3.3과 같이 홉수 h에 RTT의 1/2을 곱한 값으로 전송간격을 위한 전송 타이머의 값을 설정한다. RTT의 1/2은 OTT(Oneway trip time)로, 한 노드에서 다음 이웃 노드까지 패킷을 전송하는데 걸리는 시간을 의미하게 되고 홉수 h를 곱함으로써 h 홉까지 패킷을 전송하는데 걸리는 시간을 나타낸다. 따라서 식 (3)과 같은 전송간격으로 패킷을 전송함으로써 h 홉까지 패킷을 전송한 후 다음 패킷을 전송하도록 한다.

$$T_{interval} = h \times \frac{RTT}{2} \approx h \times OTT \tag{3}$$

무선 네트워크 환경에서 전송 및 간섭 영역은 그림 6과 같이 나타낼 수 있는데 실선과 점선으로 된 원은 각

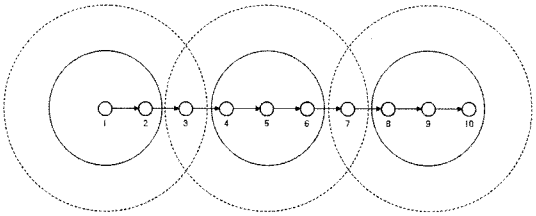


그림 6 전송 및 간섭 영역

각 전송 영역과 간섭 영역을 의미한다. 노드 1에서 패킷을 전송하게 되면 노드 3에 간섭을 일으켜 노드 4와의 전송을 동시에 할 수 없게 되고, 노드 4에서의 패킷 전송은 노드 2의 패킷 전송과 충돌을 일으키게 한다. 따라서 패킷 전송에 의한 충돌이나 간섭을 최소화하기 위해서는 4홉 이상의 노드에서 패킷을 전송해야 하고 채널 사용률을 최대로 하기 위한 전송 홉 간격은 4홉이 된다[9].

이와 같은 이유로 FCPP에서 무선 채널의 충돌과 간섭에 의해 패킷이 손실되는 것을 최소화하며 패킷을 신속하게 전송하기 위해서 전송 간격을 위한 식 (3)의 홉수, h는 4로 결정한다.

3.2 NACK 억제 기법

NACK 기반의 전송 프로토콜은 손실 이벤트 전달을 방지하기 위해 순차적인 전송을 보장하고 불필요한 전송 지연을 유발하게 된다. FCPP는 불필요한 전송 지연을 제거하여 코드 전송을 신속하게 수행하기 위해 비순차적으로 코드를 전송하게 한다. 비순차적으로 코드를 전송할 경우 손실 이벤트가 이웃 노드로 전달되기 때문에 이를 방지하기 위한 방법으로 NACK 억제 기법을 제안한다.

NACK 억제 기법은 패킷을 이웃 노드로 전송할 때 노드에서 순차적으로 수신을 희망하는 패킷의 순차번호를 명시한다. 이 번호는 노드에서 손실된 패킷 없이 완전하게 받은 패킷의 번호를 의미하고 이웃 노드에게 손실된 패킷을 복구할 수 있는 최대 패킷 번호를 알려주는 기능을 한다. 노드에서 패킷을 비순차적으로 수신하였을 경우, 손실된 패킷을 복구하기 위한 동작을 하게 된다. 패킷의 희망 순차번호가 노드에서 수신을 희망하는 순차번호보다 크게 되면 해당 패킷을 전송한 노드에 손실된 패킷을 저장하고 있기 때문에 NACK를 전송하여 손실된 패킷을 복구하게 된다. 패킷의 희망 순차번호가 노드에서 수신을 희망하는 순차번호보다 작다면 해당 패킷을 전송한 노드는 손실된 패킷을 저장하지 않고 있기 때문에 NACK를 전송하지 않는다.

그림 7은 NACK 억제 기법의 동작과정을 보여준다. 노드 A는 1, 2, 3번 패킷을 가지고 있을 때, 패킷을 전송하기 위해 수신을 희망하는 순차번호 4를 명시하여 이웃노드 B로 전송한다. 2번 패킷이 손실 되었을 경우

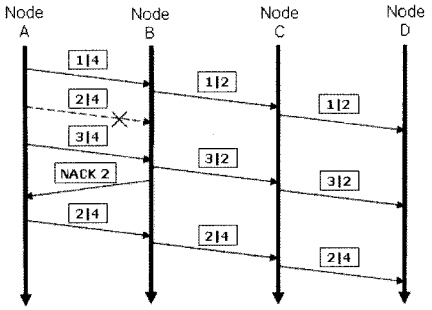


그림 7 NACK 억제 기법의 동작

노드 B는 3번 패킷을 비순차적으로 수신한 후 노드 A가 수신을 희망하는 4번 패킷이 노드 B가 수신을 희망하는 2번 패킷보다 크기 때문에 NACK를 발생하여 에러복구를 요청한다. 그리고 노드 B는 자신이 수신받기 희망하는 2번 패킷을 명시하여 이웃노드 C에게 전송한다. 노드 C는 노드 B에 의해 명시된 수신을 희망하는 패킷번호가 자신보다 크지 않기 때문에 비순차적인 패킷을 수신하였음에도 NACK를 발생하지 않는다.

그림 8은 지금까지 설명한 NACK 억제 기법의 알고리즘을 정리한 것이다. 패킷을 수신하였을 경우 비순차적으로 패킷을 수신하였는지 판단하고 비순차적으로 패킷을 수신하였다면 수신 희망 순차번호를 비교하여 에러복구 여부를 결정하고 자신의 수신 희망 순차번호를 명시하여 이웃노드로 전송한다.

```
Receive (SeqnNo, Expected_SeqnNo):
        IF (Out of Sequence)
                IF (Expected_SeqnNo > My_Expected_SeqnNo)
                        Packet Loss Recovery

        Forward Packet with My_Expected_SeqnNo
```

그림 8 NACK 억제 알고리즘

3.3 FCPP 동작 알고리즘

본 절에서는 3.1절과 3.2절에서 설계한 네트워크 상태에 따른 전송률 조절 알고리즘과 NACK 억제 기법을 이용하여 제안한 FCPP를 구성하고 전체적인 동작을 서술한다. FCPP는 크게 코드전송과 에러복구로 동작으로 나누어 볼 수 있다.

3.3.1 코드 전송

코드 전송을 위해 BS는 코드를 분할하고 분할된 코드는 코드 패킷으로 이웃 노드로 브로드캐스팅하여 전달한다. FCPP에서 코드 전송을 위한 패킷은 그림 9와 같이 파일 ID, 파일 길이, 순차번호, 희망 순차번호 그리고 분할된 코드 부분의 5개 필드로 정의된다.

FCPP에서 BS는 모든 노드로 코드를 전송하기 위해 Flooding[8] 기법을 사용한다. 그렇기 때문에 BS에서

file ID	file length	sequence number	expected sequence number	payload

그림 9 코드 패킷의 구조

패킷 전송간격을 조절함으로써 전체 네트워크의 전송률을 조절하게 된다. 전송률 조절 방법으로는 3.1절에서 설계한 전송률 조절 알고리즘을 사용한다.

BS는 수신을 희망하는 순차번호에 코드의 분할 개수를 명시하여 코드 패킷을 이웃 노드로 브로드캐스팅한다. 이때 전송 간격은 임의의 값으로 하고 이웃 노드에서 전송한 패킷을 BS가 엿들음으로써 RTT를 측정하고 전송률을 조절하게 된다. 중계 노드에서는 패킷을 수신하면 자신이 수신을 희망하는 순차번호를 명시하여 이웃 노드로 브로드캐스팅한다.

3.4.2 에러 복구

FCPP에서 에러복구는 NACK 기반으로 동작한다. 에러복구를 위한 NACK 패킷은 그림 10과 같이 분할된 코드 필드 없이 파일 ID, 파일 길이 그리고 손실윈도우의 3개 필드로 정의한다.

file ID	file length	loss window

그림 10 NACK 패킷의 구조

에러복구는 비순차적으로 패킷을 수신하였을 경우 발생하게 되는데 마지막 패킷과 같이 더 이상 패킷 수신이 일어나지 않게 되면 에러 복구를 하지 않게 된다. 이런 경우를 위해 식 (4)와 같은 타임아웃 값으로 에러복구 타이머를 설정하여 일정 시간동안 패킷을 수신하지 못하였을 경우, 능동적으로 NACK를 전송하여 에러복구를 수행하게 한다.

$$T_{timeout} = \beta \times T_{interval} \quad (4)$$

노드에서 패킷을 수신하였을 때, 비순차적으로 패킷을 수신하였다면 해당 패킷의 희망 순차번호가 자신의 희망 순차번호보다 큰지 검사하고, 만약 크다면 NACK를 전송하여 손실된 패킷을 복구한다. 만약 수신된 코드 패킷의 희망 순차번호가 자신의 희망 순차번호가 크지 않다면, NACK를 발생하지 않고 다음 패킷을 기다린다. 에러복구 타이머 $T_{timeout}$동안 패킷을 수신하지 못하였다면 능동적으로 NACK를 발생하여 손실된 패킷을 복구한다.

4. 실험 및 성능 평가

본 장에서는 새로 제안한 FCPP의 성능 평가를 위해 LBNL(Lawrence berkely national laboratory)의 ns-2 (Network simulator)[16]를 사용하여 다양한 실험을 수행하였다.

4.1 실험 환경

제안한 FCPP의 성능을 평가하기 위해서 그림 11과 같은 실험 환경을 구성하여 성능 실험을 수행하였다.

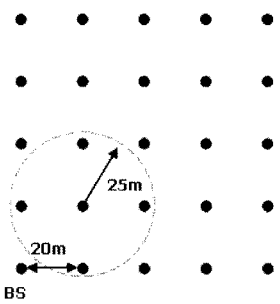


그림 11 실험 환경

본 실험은 100m × 100m 크기의 네트워크 구역을 가정하고, 20m 간격으로 5 × 5 그리드 토폴로지를 구성하였다. 코드 전송 프로토콜로 제안한 FCPP와 성능을 비교하기 위한 기존 프로토콜로 신뢰성 있는 전송 프로토콜인 PSFQ를 사용하였다. BS에서는 2.5Kbytes의 코드를 50Bytes 씩 50개의 패킷으로 나누어 전송한다. 무선 구간은 ns-2의 에러 모델을 사용하여 무선 링크에서 송수신되는 패킷에 대하여 일정하게 0~50%의 패킷 에러율을 적용하였고, 25m의 전송범위와 2Mb의 대역폭을 갖는 Simple CSMA/CA(Carrier sense multiple access/ collision avoidance) MAC(Media access control)으로 설정하였다. PSFQ의 Pump 동작을 위한 T_{min}과 T_{max}의 전송 타이머 값은 각각 100ms, 50ms로 설정하였다.

4.2 FCPP 성능 실험

본 실험은 FCPP의 기본 성능으로 신뢰성 있는 코드 전송을 평가하기 위한 것이다. FCPP는 신속한 코드 전송을 위해 제안되었지만 코드 전송 프로토콜의 근본적인 목적은 코드의 신뢰성 있는 전송이기 때문에 다음과 같은 성능 평가실험을 수행하였다.

4.2.1 네트워크 크기에 따른 코드 전송 시간 실험

패킷 에러율이 0%일 때, N × N 그리드 토폴로지에서 직경을 2개의 노드에서 20개의 노드까지 변화시키면서 FCPP에서 코드 전송 시간을 측정한 결과는 그림 12와 같다. 네트워크의 직경이 증가함에 따라 선형적으로 코드 전송시간이 증가하는 것을 확인할 수 있다. 직경의 노드 수가 증가함에 따라 전체 노드 수는 지수 적으로 증가하기 때문에 코드 전송을 완료하는 시간의 차이가 커지게 된다. 또한 분할된 패킷 수가 증가함에 따라 코드 전송을 완료하는 시간은 패킷 수에 비례적으로 증가하게 된다. 이를 통해 FCPP가 안정적으로 코드 패킷을 전송하는 것을 확인할 수 있다.

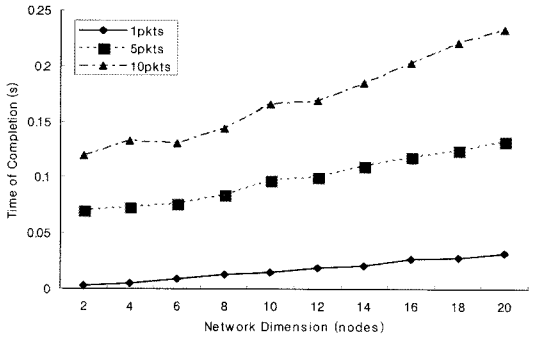


그림 12 네트워크 크기에 따른 코드 전송시간

4.2.2 패킷 수에 따른 코드 전송시간 실험

그림 13은 에러율이 0%일 때, 5 × 5 그리드 토폴로지에서 코드 패킷의 수를 1개에서 10개로 변화시키면서 FCPP에서 코드 전송 완료한 시간을 측정한 결과이다. 코드 패킷의 수가 증가함에 따라 코드 전송을 완료한 시간은 선형적으로 증가하는 것을 결과를 통해 확인할 수 있다. 이로써 FCPP가 안정적으로 코드 패킷을 전송하는 것을 확인할 수 있다.

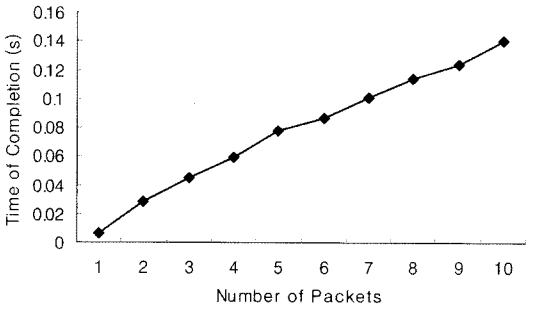


그림 13 패킷 수에 따른 코드 전송시간

4.3 기존 프로토콜과 성능 비교 실험

본 실험은 기존의 코드 전송 프로토콜 관련 연구와 제안하는 FCPP의 성능을 비교하기 위한 것이다. 성능 비교 대상으로 2장에서 설명한 PSFQ를 사용하였다.

4.3.1 패킷 에러율에 따른 코드 전송 시간

FCPP는 네트워크 상황에 따른 전송률 조절과 NACK 억제 기법으로 코드 패킷을 신속하게 전송하도록 한다. 따라서 기존의 코드 전송 프로토콜인 PSFQ와 코드 전송을 완료하는 시간을 비교함으로써 코드 전송의 신속성을 검증 할 필요가 있다. 그림 14는 FCPP와 PSFQ에서 패킷 에러율에 따른 코드 전송시간을 보여준다. 패킷 에러율이 30% 이하일 경우, FCPP의 코드 전송시간은 PSFQ의 20% 정도의 전송시간을 가진다. FCPP가 네트워크 상태에 따라 전송률을 조절하고 에러 복구를 위한 전송 지연시간을 제거함으로써 보다 빠르게 코드를 전송할 수 있게 하기 때문이다. 하지만 패킷 에러율이 30% 이상의 경우 FCPP의 코드 전송시간이 급격하게 증가하는 것을 볼 수 있다. 노드에서 마지막 코드 패킷이 손실되었을 경우 비순차적이 패킷수신이 일어나지 않아 에러 복구를 하지 못하고 에러 복구 타이머가 완료되어 NACK를 전송하여 손실된 코드 패킷을 재전송 받기 때문에 패킷 에러율이 증가할수록 코드 전송을 완료하는 시간이 길어지게 된다.

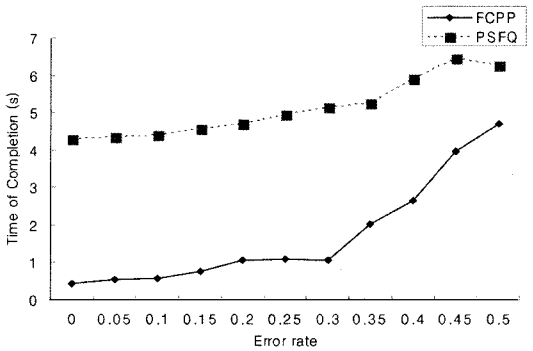
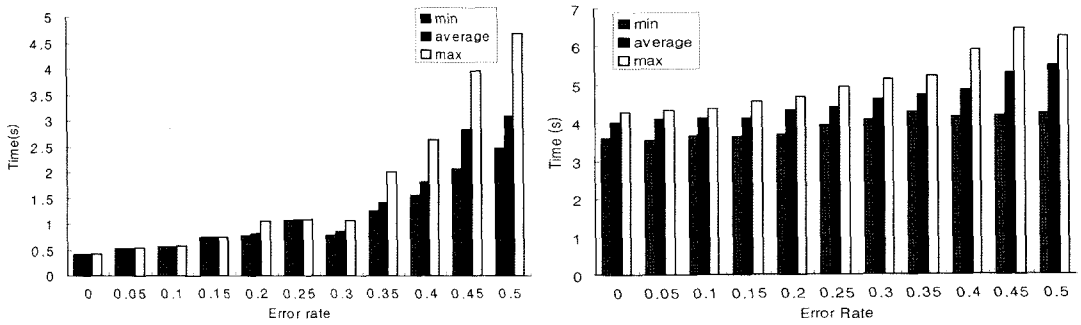


그림 14 패킷 에러율에 따른 코드 전송시간 비교

그림 15는 FCPP와 PSFQ의 최대, 최소, 평균 코드 전송시간을 분석한 결과이다. 그림 15(a)를 보면 FCPP는 NACK 억제 기법을 사용하여 비순차적인 패킷을 수신함으로써 발생하는 전송지연을 제거하고 신속하게 패킷을 전송하게 되기 때문에 패킷 에러율이 30% 이하인 경우 FCPP의 최대, 최소의 코드 전송시간의 차이가 거의 없는 것을 볼 수 있다. 그러나 패킷 에러율이 50%로 증가함에 따라 최대, 최소의 코드 전송시간의 차이가 2초 정도로 PSFQ와 비슷하지만 최대 코드 전송시간은 여전히 적은 값을 가진다. 그림 15(b)를 보면 PSFQ는 고정된 전송 타이머 값으로 전송 간격을 정하므로 일정한 최대, 최소의 코드 전송시간의 차이를 보이는 것을 확인할 수 있다. 이를 통해 FCPP가 신속하게 코드 패킷을 전송하는 것을 확인할 수 있다.

4.3.2 패킷 에러율에 따른 전송 패킷 수

신속한 코드 전송만을 목적으로 할 경우 프로토콜의 안정적인 코드 전송이 결여될 수 있기 때문에 FCPP의 안정성을 검증하기 위해 PSFQ와 전송 패킷 수를 비교하였다. 그림 16은 패킷 에러율이 증가함에 전송 패킷 수는 선형적으로 증가하는 것을 보여준다. PSFQ는 안정적인 코드 전송을 목적으로 전송간격을 길게 하여 에러 복구를 위한 시간을 보장하기 때문에 적은 제어 패킷을 사용하게 한다. FCPP는 무선 환경의 간섭과 충돌을 최소화하며 네트워크의 사용률을 최대로 이용하기 때문에 PSFQ와 비슷한 전송 패킷 수를 보여준다. 이를



(a) FCPP (b) PSFQ

그림 15 패킷 에러율에 따른 코드 전송시간 비교

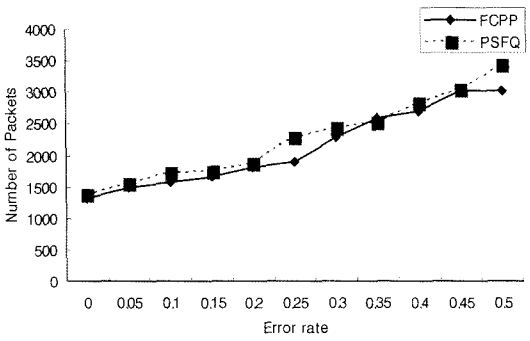


그림 16 패킷 에러율에 따른 전송 패킷 수 비교

통해 FCPP가 안정적으로 코드 패킷을 전송하는 것을 확인할 수 있다.

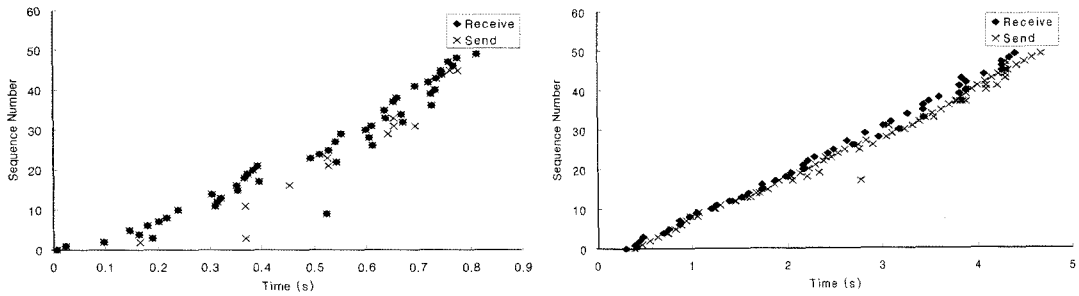
4.3.4 순차번호 분석

순차번호를 분석하기 위해서 5×5 그리드 토폴로지의 정 가운데 노드에서 시간에 따른 순차번호의 변화를 확인해 보았다. 그림 17에서 FCPP와 PSFQ가 코드 패킷 전송을 완료한 시간은 각각 0.81초와 4.68초이다. 그림 17(a)는 FCPP에서 시간에 따른 순차번호를 분석한 것으로 NACK 억제 기법을 사용하여 에러 복구에 의한 전송 지연을 제거하기 위해 수신된 패킷을 바로 전송하여 비순차적으로 패킷을 전송하는 것을 볼 수 있다. 그림 17(b)는 PSFQ에서 시간에 따른 순차번호를 분석한 것으로 패킷이 손실되었을 경우 손실된 패킷을 복구하기 전까지 패킷 전송을 하지 않는 것을 확인할 수 있다.

5. 결론 및 향후 과제

무선 센서 네트워크를 구성하는 센서 노드는 한번 배치되면 사람의 간섭 없이 오랜 기간 동안 동작하는데 실행중인 소프트웨어를 수정 또는 추가를 할 필요가 있다. 그러나 센서 노드를 회수하기 어려운 경우가 있기 때문에 원격 코드 업데이트 기법이 필요하게 되고, 이를 위한 신뢰성 있는 코드 전송 프로토콜에 대한 연구가 활발하게 진행되고 있다. 하지만 신뢰성만을 고려한 코드 전송 프로토콜은 코드를 안정적으로 전송하기만을 고려하기 때문에 코드를 신속하게 전송한다는 관점에 대한 고려가 부족하다는 한계를 갖는다. 그 결과 긴 코드 전송시간에 의해 불필요한 에너지 소모를 발생함으로써 센서노드의 에너지 효율을 저하시키게 된다.

본 논문에서는 기존의 코드 전송 프로토콜들이 가지는 한계를 극복하는 FCPP(Fast code propagation protocol)을 제안하였다. FCPP는 신뢰성 있는 전송뿐만 아니라 신속함을 고려한 접근 방법을 제시하고 있다. 새



(a) FCPP (b) PSFQ

그림 17 시간에 따른 순차번호

로 제안한 알고리즘은 RTT기반의 전송률 조절과 NACK 억제 기법으로 네트워크 상태를 반영한 전송률 조절과 에러복구에 의한 불필요한 전송지연을 피하도록 하여 네트워크의 사용률을 최대화하여 신속한 코드 전송을 가능하게 한다.

시뮬레이터를 이용한 실험을 통해서 FCPP의 기본적인 신뢰성 있는 코드 전송 성능을 검증하고 기존의 PSFQ(Pump slowly, fetch quickly)와 성능 비교를 통하여 신속한 코드 전송뿐만 아니라 안정적인 전송 측면에서도 대등한 결과를 보였다. 실험 결과를 통해 제안하는 FCPP가 센서 네트워크의 코드 전송에서 신뢰성 및 신속함을 모두 만족시킬 수 있음을 확인하였다.

본 논문에서는 매우 단순화한 네트워크 토폴로지 모델을 사용하여 성능을 평가하였다. 하지만 실제 환경에서는 노드 밀도의 증가로 인한 심한 충돌로 패킷이 손실될 수 있는 개연성을 고려하지 않았다. 따라서 향후 연구 과제로 노드 밀도의 증가에 따른 패킷손실을 고려한 연구가 필요하다. 또한 제안한 FCPP이 실제 센서 네트워크에서 코드 전송을 위해 동작할 수 있는 구현 연구가 수행되어야 할 것이다.

참 고 문 헌

[1] 채동현, 한규호, 임경수, 안순신, "센서 네트워크의 개요 및 기술동향", 정보과학회 논문지, 제22권, 제12호, pp. 5-12, 2004. 12.

[2] T. Stathopoulos, J. Heidemann and D. Estrin, "A Remote Code Update Mechanism for Wireless Sensor Networks," Technical Report CENS-TR-30, November 2003.

[3] C. Wan, A. Campbell, and L. Krishnamurthy, "PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks," In Proceedings of ACM WSNA, pp. 1-11, July 2002.

[4] J. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," In Proceedings of ACM SenSys, pp. 81-94, November 2004.

[5] Crossbow Technology Inc. Mote in-network programming user reference, http://webs.cs.berkeley.edu/tos/tinyos-1.x/doc/xnp.pdf

[6] F. Stann and J. Heidemann, "RMST: Reliable Data Transport in Sensor Networks," In Proceedings of the First International Workshop on Sensor Net Protocols and Applications, pp. 102-112, April 2003.

[7] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz, "ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks," In Proceedings of MobiHoc, June 2003.

[8] TinyOS, http://tinyos.net/

[9] R. Jiang, V. Gupta, and C. Ravishankar, "Interactions Between TCP and the IEEE 802.11 MAC Protocol," In Proceedings of IEEE DISCEXi, April 2003.

[10] C. Ho, K. Obraczka, G. Tsudik and K. Viswanath, "Flooding for Reliable Multicast in Multi-Hop Ad Hoc Networks," In Proceedings of ACM DialM, August 1999.

[11] N. Reijers and K. Langendoen, "Efficient Code Distribution in Wireless Sensor Networks," In Proceedings of ACM WSNA, pp. 60-67. September 2003.

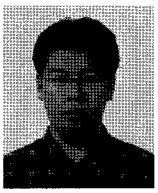
[12] S. Ni, Y. Tseng, Y. Chen and J. Sheu, "The broadcast storm problem in a mobile ad hoc network," In Proceedings of ACM Mobicom, pp. 151-162. August 1999.

[13] K. Tang and M. Gerla, "MAC Reliable Broadcast in Ad Hoc Networks," MILCOM, pp. 1008-1013, October 2001.

[14] IEEE Std. 802.11, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," November 1997.

[15] L. Feeney and M. Nilsson, "Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment," In Proceedings of IEEE INFOCOM, pp. 1548-1557, April 2001.

[16] The network simulator ns-2, http://www.isi.edu/nanam/ns/



이 한 선

2004년 광운대학교 전자물리학과 학사 2006년 광운대학교 전자통신공학과 석사 2006년~현재 광운대학교 전자통신공학과 박사과정. 관심분야는 센서네트워크, Ad-Hoc 네트워크, 임베디드시스템

정 광 수

1981년 한양대학교 전자공학과 학사 1983년 한국과학기술원 전기 및 전자공학과 석사. 1991년 미국 University of Florida 전기공학과 박사(컴퓨터공학전공). 1983년~1993년 한국전자통신연구원 선임연구원. 1991년~1992년 한국과학기술원 대우 교수. 1993년~현재 광운대학교 전자공학부 교수(정보통신 연구원). 관심분야는 인터넷 QoS, 유.무선 비디오 스트리밍, 센서네트워크