묵시적 응답 및 간접 복구를 이용한 무선 센서 네트워크에서의 신뢰성 있는 멀티캐스팅

(Reliable Multicasting with Implicit ACK and Indirect Recovery in Wireless Sensor Networks)

김 성 훈 † 양 현 †† 박 창 윤 †††
(Sung Hoon Kim) (Yang Hyun) (Chang Yun Park)

요 약 센서네트워크가 다양하고 동적인 영역으로 발전해감에 따라 임무 갱신과 같은 기능을 위해 신뢰성 있는 멀티캐스팅 기술이 새롭게 요구되고 있다. 기존의 연구들은 NAK 기반 방식을 사용하고 있지만, 마지막 패킷 문제 등을 안고 있다. 본 논문은 묵시적 ACK와 간접 복구를 이용하는 ACK 기반 오류 제어 기법인 RM2I를 제안한다. 묵시적 ACK는, 송신 노드로부터 패킷을 받은 수신 노드가 다음 노드로 포워딩할 때 송신 노드도 이를 간접적으로 받게 되는데, 이 수신 내용을 ACK로 해석하는 것을 말한다. 간접 복구는 어떤 노드가 상위 노드가 아닌 이웃노드로부터 패킷을 간접적으로 받았을 때 이를 상위 노드로부터 받은 것으로 해석하여 오류 복구에 활용하는 것을 말한다. NS-2 시뮬레이터를 이용하여 다양한 환경 및 인자에서 RM2I의 에너지 성능을 분석 및 검증하였다. 실험 결과, 묵시적 ACK는 ACK의 수를 줄여 제어 부하를 감소시키고, 간접 복구는 재전송 횟수를 줄여 데이타 전송 부하를 감소시켰다. 또한, NAK 기반 오류 복구 기법의 이론적 에너지 성능 상한선을 계산하여 이와 비교하였다. 비교 결과, 에지 노드에서의 부하를 제외하면 어떤 NAK 기반 기법과도 견줄 수 있는 에너지 효율성을 제공한다는 것을 알 수 있었다.

키워드 : 센서네트워크, 신뢰성 있는 멀티캐스팅, 에너지 절약

Abstract As sensor networks are used in various and dynamic applications, the function of sink-to-sensors reliable multicasting such as for task reprogramming is newly required. NAK-based error recovery schemes have been proposed for energy efficient reliable multicasting. However, these schemes have incompleteness problems such as the last packet loss. This paper introduces an ACK-based error recovery scheme, RM2I(Reliable Multicast with Implicit ACK and Indirect Recovery). It utilizes wireless multicast advantage in which a packet may be delivered to all of its omni-directional neighbor nodes. When a sender overhears a packet which its receiver forwards to the next nodes, it may interpret it as an ACK from the receiver. We call it an Implicit ACK. In Indirect Recovery, when a node receives a packet from neighbor nodes which are not its direct upstream node, it saves and utilizes it for error recovery. Using NS-2 simulator, we have analyzed their effects. We have also compared RM2I with the NAK-based error recovery scheme. In results, RM2I shows comparable performances to the ideal NAK-based scheme, except where Implicit ACK and Indirect Recovery do not occur at the edges of the networks.

Key words : Sensor Networks, Reliable Multicasting, Energy Saving

---

· 본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업의 일환으로 수행하였음. [2006-S-040-01, Flash Memory 기반 임베디드 멀티미디어 소프트웨어 기술 개발]

† 정 회 원 : 중앙대학교 컴퓨터공학과
kimsh@kisa.or.kr
†† 학생회원 : 중앙대학교 컴퓨터공학과
yanghyun@cnlab.cse.cau.ac.kr
††† 종신회원 : 중앙대학교 컴퓨터공학과 교수
cypark@cau.ac.kr

논문접수 : 2008년 2월 26일
심사완료 : 2008년 3월 4일

정보과학회논문지: 정보통신 제35권 제3호(2008.6)

1. 서 론

센서 네트워크의 응용 분야가 광범위해지면서, 신뢰성 있는 전송에 대한 요구도 높아지고 있다. 대표적인 예로는 임무 변경을 위한 네트워크 재프로그래밍을 들 수 있다. 싱크는 새로운 임무를 수행하는 프로그램을 소스 노드들에게 다운로드하고, 소스 노드는 이 프로그램으로 재구동하여 새로운 임무를 수행한다. 이 과정에서 전송되는 프로그램의 순결성 유지를 위한 신뢰성 있는 전송이 필요하다.

신뢰성을 제공하기 위해서는 에너지 소모를 동반하는 오류제어를 수행하여야 한다. 따라서 에너지 소모를 최소화하여야 하는 센서노드 입장에서는 전력 효율적인 오류제어 기법이 절대적으로 필요하다. 본 논문은 센서 네트워크에서 신뢰성 있는 멀티캐스트를 제공하면서 에너지 효율성을 높이는 기법을 개발하는 것을 목적으로 하고 있다. 본 연구가 개발하는 오류 제어는, 인터넷의 SRM[1] 프로토콜과 같이, 멀티캐스트 라우팅 위에서 독립적으로 동작하면서 다수의 수신자에게 신뢰성 있는 전송을 제공하는 기능을 담당한다.

멀티캐스팅에서는, 복수의 수신 노드가 동시에 응답을 보내는 문제를 피하기 위해 고전적인 ARQ 대신에, 오류가 발생한 패킷에 대해서만 부정응답(NAK: Negative ACKnowledgement)을 보내는 NAK 기반 오류 복구 기법들이 많이 사용되어 왔다. 이 방식은 전송 받은 모든 패킷에 대하여 ACK를 보내는 것보다 전력 효율 면에서도 유리할 수 있으므로 에너지 절약이 중요한 센서 네트워크에서도 유력한 방법으로 인식되어 왔다. 따라서 센서 네트워크에서 신뢰성 있는 멀티캐스트에 관한 기존의 연구들은 NAK 기반 오류 복구 기법을 주로 채택하여 왔다.

그러나 NAK 기반 오류 복구 방식은 몇 가지 문제점을 가지고 있다. 우선 ACK와 마찬가지로 여러 수신자가 같은 NAK를 송신자에게 보내려고 할 경우 NAK 폭주 문제가 발생한다. 두 번째로 NAK만 가지고는 수신자의 정확한 상태를 알기 어려워 흐름 제어가 어렵다[2]. 세 번째로 마지막 패킷 손실, NAK 손실 등과 같은 문제에 대한 대책이 필요하다. 끝으로 오류율이 높은 경우 수신자들이 보내는 NAK가 많아지므로 에너지 효율 개선도 낮아진다.

본 논문에서는 두 가지 현상에 착안하여 새로운 오류 제어 기법을 제안한다. 첫째, 무선 네트워크에서는 전송 전파가 사방으로 퍼져서 1-홉 내에 있는 노드들은 모두 이를 수신하게 된다. 이 현상을 부정적으로 보면 원하지 않는 노드들도 불필요한 수신에 에너지를 소모하는 overhearing 문제로 볼 수 있지만, 복수의 노드들이 동시에 정보를 교환할 수 있는 "Wireless Multicast Advantage (WMA)"로 이용될 수 있다. 둘째, 대부분의 센서네트워크 응용에서 처리량은 크게 중요하지 않다는 점이다. 따라서 시간을 희생하더라도 에너지를 절약할 수 있다면 효과적인 방법이 될 수 있다.

이에 기초하여 본 연구는 묵시적 ACK(Implicit ACK)와 간접 복구(Indirect Recovery)를 적용하는 ACK 기반의 신뢰성 있는 멀티캐스트 기법으로 RM2I(Reliable Multicast with Implicit ACK and Indirect Recovery)를 제안한다. 묵시적 ACK란, 멀티캐스팅 과정에서 어떤 노드가 수신한 패킷을 다음 노드로 포워딩할 때 이 패킷을 송신한 노드도 이를 수신하게 되는데, 이를 수신 노드가 패킷을 잘 받았다는 ACK로 인식하는 것을 말한다. 수신 노드가 패킷을 수신하고 이를 다음 노드로 포워딩할 때 ACK보다 포워딩을 먼저 한다면("Forward before ACK"), 묵시적 ACK 효과를 얻을 수 있고, 결국 ACK의 전송 부하를 줄여서 에너지를 절약할 수 있다. 간접 복구는, 오류 복구를 송신 노드의 재전송에만 의존하지 않고, 주변 노드들의 전송 내용을 간접적으로 받았을 때 이를 무시하지 않고 챙겨 두어서 오류 복구에 활용하는 것을 말한다. 간접 복구는 송신 노드의 재전송 부하를 줄여서 에너지 손실을 줄일 수 있다.

묵시적 ACK와 간접 복구 모두, WMA에 기초하여, 주변 노드들의 전송 동작 중에서 송신과 수신 노드 사이의 직접적인 오류제어 동작인 ACK와 재전송을 대체할 수 있도록 활용하려는 기법이다. 실질적인 대체 효과를 얻으려면 직접적인 오류제어 동작이 지연되어야 하는데, 처리 속도에 상대적으로 민감하지 않은 센서네트워크 응용에서는 이를 수용할 수 있다고 판단된다.

본 연구의 기법이 주변에서 들려오는 통신 내용을 활용하는 것이다 보니, 수신에 소모되는 에너지에 대한 우려가 있다. 이에 대한 본 연구의 입장은 어차피 수신할 것이라면 이를 수신해서 활용하자는 것이다. 신뢰성 있는 전송을 위해서는 어떠한 오류 제어 기법도 신뢰성 있는 패킷 전송이 완료되는 기간에는 무선 인터페이스를 활동(active)시켜야 하며, 이 기간 동안 도착하는 모든 프레임을 일단 수신하게 된다. 사이사이에 인터페이스를 휴면(idle)시키는 기법이 있다면, 본 연구에도 원칙적으로 적용이 가능할 것이다. 본 연구의 기법에서는 각 노드가 전송하는 데이타 및 제어 패킷의 횟수를 줄이므로 주변 노드의 인터페이스가 수신하는 양도 줄어드는 효과가 있다.

실험은 송신은 물론 수신에 소모되는 에너지를 포함시켜 에너지 절약 효과를 검증한다. NS-2 시뮬레이터를 이용하여 토폴로지, 응용 트래픽의 유형, 윈도우 크기 및 재전송 타임아웃 등을 변화시키면서 실험하였다. 또

한 NAK 기반 오류 제어 기법의 부하 하한선과도 비교하여 에너지 절약 효과를 검증하였다.

본 논문의 구성은 다음과 같다. 2장에서 센서네트워크에서의 신뢰성 문제와 신뢰성 있는 멀티캐스트 프로토콜에 대해 소개한다. 3장에서는 핵심 기술인 묵시적 ACK와 간접 복구 기법에 대해 자세히 설명한다. 4장에서는 위의 두 기법들을 적용한 프로토콜 RM2I의 설계에 대해 살펴보고, 5장에서는 제안 기법의 에너지 효율성을 검증하기 위해 실시한 다양한 실험들의 결과를 제시, 분석 평가한다. 마지막으로 6장에서 향후 연구를 제시하며 결론을 맺는다.

2. 배경 및 관련연구

센서네트워크에서 신뢰성 있는 전송이 필요한 경우는 실행 파일, 구성 등에 관한 제어 메시지, 코드 업데이트 등을 할 때이다[3-5]. 또한, 가스 검출 감지나 군사적 목적 센서네트워크와 같이 노드가 감지하는 대상이 매우 중요해서 내용을 반드시 싱크 노드에게 전달해야만 하는 경우에도 신뢰성 있는 전송이 요구된다[4].

보통 센서 노드들은 크기가 작고 제약적인 연산 처리 능력을 가지기 때문에 유선환경에서 사용하던 신뢰성 있는 데이타 전송 기법은 센서네트워크에 적용하기에는 적합하지 않다. PSFQ(Pump Slowly Fetch Quickly)는 하나의 송신 노드에서 여러 개의 소스 노드로 데이타를 신뢰성 있게 전송하기 위해 설계 되었다[6]. PSFQ는 크게 세 가지의 연산으로 구성된다. Pump 연산은 패킷을 하나씩 전파시키는 것이며 전파되는 패킷에는 시퀀스 넘버가 붙는다. 이 때, 패킷 전파는 1-홉 거리의 브로드캐스트로 이루어진다. 패킷 전송 간의 시간 간격은 비교적 느리게 이루어진다. Fetch 연산은 수신되지 않은 패킷에 대하여 재전송을 요청하기 위해 곧 바로 NAK를 보낸다. NAK를 보내는 노드는 1-hop 이전의 노드에 의해 해당 패킷이 복구될 때까지 다음 패킷을 전송하지 않는 순서적 전송을 한다. Report 연산은 싱크 노드의 요청으로 시작되며 노드들은 자신의 주소와 수신/미수신 패킷에 대한 정보를 리포트 패킷에 실어 보낸다.

PSFQ에서는 Pump 연산이 매우 느리게 이루어지기 때문에 진행 속도는 느리지만 전송에 실패한 패킷에 대해서만 NAK를 보내기 때문에 에너지를 절약할 수 있다. 또한 오류 복구가 홉단위로 이루어지므로 복구 부하가 전체 네트워크로 분산되는 효과도 있다. 그러나 마지막 패킷 문제와 같은 NAK 관련 문제를 그대로 안고 있다.

GARUDA도 본 연구 및 PSFQ와 같은 목적으로 개발된 알고리즘이다[7]. 우선 싱크 노드는 블록의 첫 번째 패킷을 포함해서 주기적으로 짧은 길이의 펄스(pulse)들을 높은 에너지로 발생시킨다. 싱크가 발생시킨 펄스를 인지한 이웃 노드들은 자신들도 펄스를 발생시킨다. 이러한 연쇄적인 펄스 발생은 에지 노드까지 이어져서 네트워크 전체의 연결을 구성하면서 중간 중간에 코어 노드를 선정한다. 구성이 완료된 후, 싱크 노드는 패킷을 보내기 시작하며, 패킷을 받은 소스 노드들은 펄스 발생을 중단하고 받은 패킷을 전파시킨다. 오류가 발생한 패킷에 대해 NAK를 보내서 재전송을 요청한다. 그러나 재전송은 코어 노드에 의해 이루어지며, 패킷 복구가 이루어지는 동안에도 다음 노드로 계속해서 전달하는 비순서적 전달 정책을 취하고 있다.

GARUDA는 블록의 첫 번째 패킷을 신뢰성 있게 보내기 위해 비교적 많은 에너지를 소모한다. 그러나 이를 기반으로 오류 복구를 할 때마다 해당 패킷을 복구해 줄 코어 노드를 생성하며, 코어 노드는 매번 다르게 선정되기 때문에 오류 복구에 대한 부하를 네트워크 전체로 분산시켜 네트워크 전체의 에너지를 절약할 수 있다.

RMST는 Directed Diffusion 기반에 신뢰성 있는 전송 기법을 추가한 것이다[8]. 따라서 PSFQ나 GARUDA와는 다르게 센서 노드들로부터 싱크 노드로 신뢰성 있는 전송을 보장하는 것을 목적으로 하고 있다. 서로 다른 여러 싱크들이 동일한 이벤트에 관심이 있는 경우에는 신뢰성 있는 멀티캐스트가 요구된다. RMST는 캐시 모드와 비 캐시 모드로 나누어진다. 캐시 모드에서는 싱크 노드와 강화된 경로 상에 있는 모든 중간 노드들이 패킷들을 캐싱한다. 그리고 이 중간 노드들은 미 수신된 패킷이 있는지 주기적으로 자신의 캐시를 검사한다. 만약 미 수신된 패킷이 있다면 강화된 경로를 타고 소스 방향으로 NAK를 전송 한다. 이렇게 보내진 NAK의 이동은 미 수신된 패킷을 가지고 있는 중간 노드를 만날 때까지 계속되고 미 수신된 패킷을 가지고 있는 중간 노드를 만나면 이 노드에 의해 복구가 단대단(end-to-end)으로 이루어진다. 비 캐시 모드에서는 소스 노드만이 패킷 복구를 할 수 있다.

RMST의 캐시 모드에서는 패킷 복구를 담당하는 노드가 따로 지정되어 있지 않아서 복구 부하가 경로 상에 있는 모든 중간 노드들로 분산되지만 비 캐시 모드에서는 패킷 복구를 담당하는 노드가 소스 노드뿐이기 때문에 소스 노드로 복구 부하가 집중되어 소스 노드에서의 막대한 에너지 소모를 초래할 수 있다.

센서 네트워크에서 대부분의 신뢰성 있는 멀티캐스트 기법들은 모든 패킷에 대해 ACK를 보내는 것이 낭비가 되기 때문에 NAK 기반의 오류 복구 정책을 취한다. 그러나 본 논문에서는 WMA를 이용한 묵시적 ACK를 통해 ACK 부하 문제를 극복하였다. 또한 주요 목적이 신뢰성 있는 다운스트림이므로 무선의 멀티 홉 브로드캐스

트 특성을 통한 간접 복구를 이용하여 오류 복구를 위한 재전송을 줄임으로써 에너지 효율성을 극대화 시켰다. 그리고 홉단위 오류복구로 복구 부하를 분산시켰다.

3. 에너지 개선 기법

3.1 센서 네트워크에서의 ACK 전송 방법

본 연구에서는 오류 복구를 위해 홉 단위의 ACK를 필요로 한다. 이 때 ACK를 링크 수준에서 어떻게 보내는지에 따라 Multicast-ACK와 Unicast-ACK의 두 가지 접근 방법이 있다. Unicast-ACK는 패킷을 보내온 노드를 목적지로 하여 ACK를 보내는 것을 말하고, Multicast-ACK는 송신 노드를 포함하는 멀티캐스트 주소를 목적지로 하여 수명이 1 홉인 ACK를 보내는 것을 말한다.

대표적인 무선 네트워크인 802.11 MAC에서 Unicast-ACK는 MAC 수준의 유니캐스트 방식인 RTS/CTS/DATA/ACK 방식을 사용해서 전달된다. 따라서 최대 4번의 MAC 프레임 전송이 필요하다. 반면, Multicast-ACK는 MAC 수준의 멀티캐스트 방식으로 구현되므로 한 번의 프레임 전송으로 이루어진다. 단, MAC 수준에서 오류제어는 이루어지지 않으며, ACK의 수신 여부는 확인할 수 없다.

MAC 수준의 프레임 전송 중 오류가 발생하지 않는다면, Multicast-ACK 방식이 에너지를 절약할 수 있다. Unicast-ACK에서는 송신자만이 ACK를 수신하지만, Multicast-ACK에서는 1 홉 전송 범위에 있는 모든 멀티캐스팅 참여자가 ACK를 수신하게 된다. 그러나 무선 인터페이스에서 프레임 수신에 소모되는 에너지는 유니캐스트 경우에도 마찬가지이므로(즉, overhearing 발생), 실제로 추가되는 에너지 소모는 불필요한 ACK를 오류제어 프로토콜까지 올리고 이를 버리는데서 발생된다. 연산에 소모되는 에너지는 전송에 소모되는 에너지에 비해 매우 적으므로, 프레임 전송 횟수를 줄이는 Multicast-ACK 방식이 에너지 면에서 유리하다고 할 수 있다.

프레임 전송 중에 오류가 발생하면 문제는 복잡해진다. Unicast-ACK에서는 MAC 수준의 재전송을 거쳐 복구를 하지만 Multicast-ACK 방식에서는 바로 ACK 전송의 실패가 된다. 그러나 ACK는 트랜스포트 수준 오류제어 메시지 일종이고, ACK의 손실도 데이타 손실의 경우와 마찬가지로 복구 처리가 되므로 신뢰성을 유지하는 데에는 문제가 없다. 결국, ACK 메시지 손실을 MAC 수준에서 복구하는 것과 트랜스포트 수준에서 복구하는 것의 차이이며, 어느 쪽이 에너지 면에서 효율적인 가는 많은 요소가 고려되어야 할 것이다.

본 연구에서는 실험을 통해 두 방법의 에너지 효율성을 점검하였으며, 실험 결과는 5장에 자세히 기술하였다. 실험 결과, Multicast-ACK가 Unicast-ACK에 비해 오류율에 관계없이 보다 효율적이라는 것을 보여 주었으며, 따라서 본 연구에서는 Multicast-ACK를 사용하였다.

3.2 묵시적 ACK를 이용한 제어 부하 감소

무선 통신에서 송신 노드로부터 전송된 패킷을 받은 수신 노드가 다음 노드로 포워딩 혹은 릴레이 할 때, 송신 노드도 이 패킷을 간접적으로 수신하게 된다. 대칭형 신호 전달 환경에서 멀티캐스팅 통신을 하는 상황이라면 이 현상은 별도의 처리 없이 자연스럽게 발생한다. 묵시적 ACK란 이처럼 ACK를 기다리던 송신노드가 수신 노드의 포워딩 동작을 패킷을 잘 받았다는 ACK로 인식하는 것을 말한다.

본 연구에서는 수신 노드가 패킷을 받게 되면 송신 노드로 ACK를 전송하는 것은 뒤로 미루고 다음 노드로 패킷 포워딩을 먼저 실시하는 "Forward before ACK" 방식을 제안한다. 그림 1은 묵시적 ACK의 동작을 나타내고 있다. 송신 노드는 수신 노드가 포워딩한 패킷을 자동으로 간접 수신하게 되며 이를 통해 수신 노드에게 패킷이 잘 도달했다는 것을 알아낼 수 있다. 수신 노드는 ACK를 전송하지 않아도 되기 때문에 그만큼 에너지를 절약할 가능성이 있다.

묵시적 ACK 알고리즘에서 오류가 한 번도 발생하지 않는다면 ACK 역시 한 번도 보낼 필요가 없고 묵시적 ACK 만으로도 신뢰성 있는 전송이 가능하다. 오류가 발생하는 경우는 크게 두 가지이다. 하나는 송신 노드가 보낸 패킷이 오류 난 경우이고, 다른 하나는 수신 노드의 포워딩, 즉, 묵시적 ACK가 오류 난 경우이다. 앞의 오류를 처리하기 위해서는 송신 노드에서 타임아웃은 필수이다. 두 번째 오류도 송신 노드에서의 타임아웃에 이은 재전송으로 복구가 이루어지지만, 수신 노드가 중

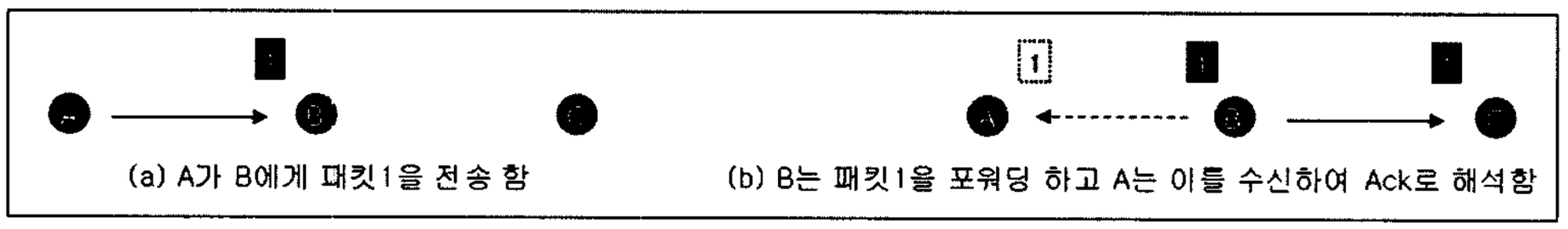

그림 1 묵시적 ACK 기반 패킷 포워딩

복 포워딩을 하지 않으면서 수신 상황을 알리기 위해서는 별도의 ACK를 보내는 명시적 ACK 전송이 필요하다. 포워딩을 하지 않는 에지 노드에서도 명시적 ACK가 사용된다.

종합해 보면, 본 연구의 묵시적 ACK 기법은 ACK를 사용하지 않는 것이 아니라 기존 ARQ의 ACK를 사용하되 포워딩 시점, ACK 전송 시점, 재전송 타임아웃 시점 등을 조정하여 ACK의 부하를 최대한 줄이는 방식이다. 묵시적 ACK의 효과가 극대화되는 환경은 오류율이 낮은 경우이다. 한편, 에지 노드들이 많은 환경에서는 묵시적 ACK의 효과가 크게 발휘되지 못할 수도 있다.

3.3 간접 복구를 이용한 재전송 에너지 절약

간접 복구는, 무선 통신에서 오류가 발생한 경우 송신자의 재전송에 의해서 오류를 복구하는 대신에, Wireless Multicast Advantage 특성에 의해 간접적으로 수신한 이웃 노드들이 패킷 내용을 이용해서 오류 복구에 활용하는 것을 말한다. 그림 2는 간접 복구를 보여주고 있다.

그림 2에서 멀티캐스팅 경로 상으로 A의 다음 노드는 노드 B와 노드 D이다. 그림 2(a)에서처럼 A가 보낸 패킷 1이 B에는 잘 도착했으나 D에는 도착하지 못하였다면, 보통의 경우 타임아웃 동안 D로부터의 ACK를 기다린 후 재전송을 할 것이다. 한편, B는 패킷 1을 노드 C로 포워딩을 할 것이다. 그런데 D는 B가 C로 전송하는 패킷 1을 간접적으로 수신할 수 있으며, 이를 A로부터 패킷을 수신한 것이라 간주할 수 있다면 노드 A에게 패킷 1에 대한 ACK를 보낼 수 있게 된다. 이 과정이 노드 A에서 패킷 1에 대한 타임아웃이 발생하기 전에 이루어진다면 오류 복구 작업을 피할 수 있고 그만큼 에너지를 절약할 수 있다.

1-홉 반경 내에 이웃 노드들이 많은 밀집된 네트워크를 가정한다면, 타임아웃 시간이 간접 복구의 성패를 좌우한다. 송신 노드의 재전송 타임아웃 시간이 간접 복구가 이루어질 수 있을 만큼 길면 재전송 횟수를 줄일 수 있기 때문에 에너지를 절약할 수 있지만 진행 속도는 느려지게 된다. 센서 네트워크에서는 보통 센서 노드들이 조밀하게 구성되는 경우가 많고 높은 처리량이 필요한 경우는 많지 않으므로, 간접 복구를 센서 네트워크에 적용하는 것은 적합하다고 볼 수 있다.

앞에서 살펴본 바와 같이 묵시적 ACK는 오류율이 적은 환경에서 높은 에너지 효율성을 기대할 수 있으며, 간접 복구는 이웃 노드가 많고 오류율이 높은 환경에서 에너지 효율성이 발휘될 수 있다. 두 기법은 직교적이기 때문에 같이 사용해도 각 기능의 효과를 그대로 반영할 수 있다. 다음 장에서는 묵시적 ACK와 간접 복구를 혼합하여 오류 제어를 수행하는 RM2I 프로토콜을 설명한다.

4. RM2I 설계

4.1 주요 설계 결정 사항

본 논문에서 제안하는 오류 제어 알고리즘은 멀티캐스트 라우팅 계층 위에서 동작하는 것을 가정한다. 오류 복구는 멀티캐스팅 홉단위로 이루어진다. 멀티캐스팅에서 종단간 오류 복구는 개념적으로는 간단하지만 에너지 소비가 많은 것으로 인식되고 있다. 또한 홉단위 오류 복구는 오류 복구를 담당하는 노드들을 네트워크 전체로 분산시키는 효과가 있어서 네트워크의 수명을 연장시킬 수 있다. 본 논문에서는 이 프로토콜을 RM2I (Reliable Multicast with Implicit ACK and Indirect Recovery)라고 부른다. 제안 알고리즘의 프로토콜 계층 관계도는 그림 3과 같다.

RM2I는 슬라이딩 윈도우를 기초로 동작한다. 이는 간접복구 시 비순서적인 패킷들을 유지할 수 있는 공간을 제공하기 위함이다. 오류 복구 정책으로는 오류가 발

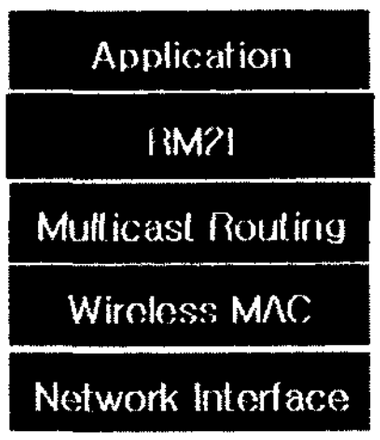

그림 3 제안 알고리즘의 프로토콜 계층 관계도

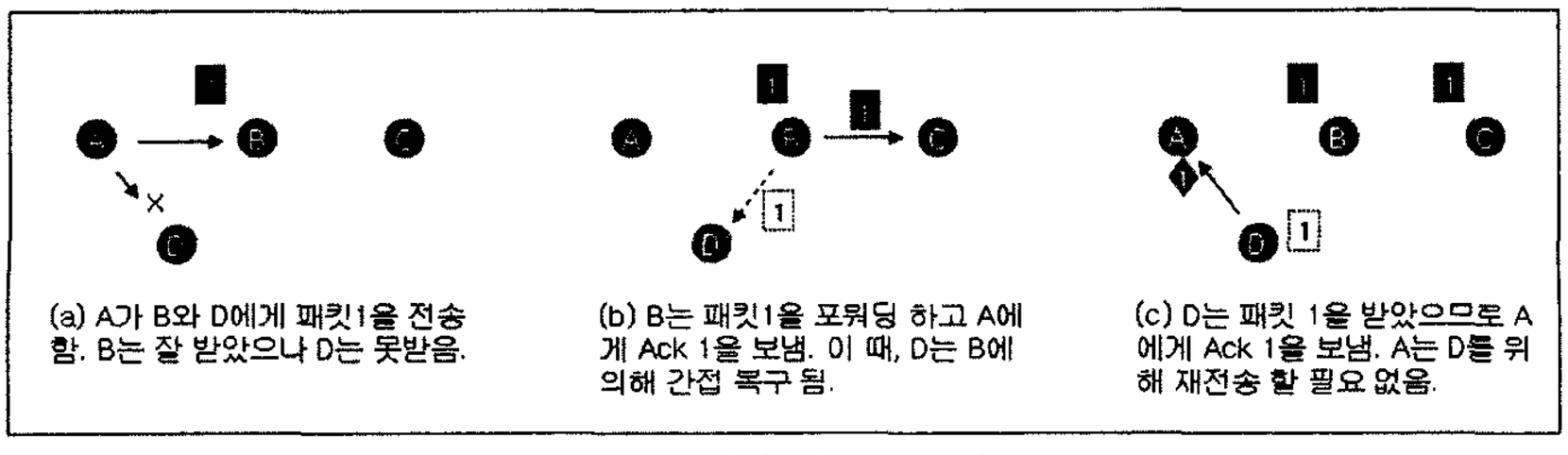

그림 2 간접 복구의 동작

생한 패킷만을 재전송해서 에너지 소모를 줄여야 하므로 selective repeat ARQ를 사용하도록 한다. 이 같은 고전적인 오류 제어 프로토콜에 묵시적 ACK와 간접 복구 기능을 추가시켜서 불필요한 제어 메시지와 데이타 전송을 제거하여 에너지를 절약하도록 한다.

RM2I는 멀티캐스트 라우팅에 독립적인 동작을 추구하기 때문에 멀티캐스트 라우팅의 라우팅 정보에 의존하지 않도록 설계하였다. 이를 위해 RM2I는 그림 4와 같은 이웃 노드 테이블을 구축한다. 이웃 노드들이란 오류 복구 측면에서 정의되는 것으로서 자신에게 의지하여 오류 복구를 하는 다음 노드들과 자신에게 재전송을 해주는 이전 노드를 말한다. RM2I 이웃 노드는 1홉 범위 내의 송수신 가능 노드들로 구성되며, 멀티캐스트 라우팅 상의 이웃 노드들과는 다를 수 있다.

노드 1의 이웃 노드는 노드 0, 노드 2, 노드 3 이다. 이 때, 노드 0은 노드 1의 이전 노드가 되고 노드 2와 노드 3은 노드 1의 다음 노드가 된다. 노드 1은 노드 2와 노드 3에게 오류 복구를 위한 재전송을 해 줄 의무가 있다. 그래서 노드 1은 노드 2와 노드 3에 대한 송신 윈도우를 유지한다. 반대로 오류 복구 시 노드 2와 노드 3은 노드 1에 대한 수신 윈도우를 유지한다.

네트워크 전체적인 토폴로지 관점에서 봤을 때 노드 0은 송신 노드, 노드 1은 중간 포워드 노드, 노드 2와 3은 에지 노드가 된다. 송신 노드는 패킷을 발생시키는 노드를 말하며, 다음노드로 패킷을 전달하기만 한다. 중간 노드는 패킷을 포워딩하는 노드를 말하며, 이웃 노드 테이블에 이전 노드와 다음 노드를 모두 가진다. 에지노드는 토폴로지의 가장 외곽에 있는 노드로서 이전 노드로부터 패킷을 받기만 한다. 에지 노드의 이전노드는 오로지 하나만 존재하고 다음 노드는 갖지 않는다.

전통적인 슬라이딩 윈도우 프로토콜은 하나의 송신자/수신자 쌍에 대해 동작하므로 수신 상태와 버퍼링을 동시에 관리하여도 무방하였다. 그러나 RM2I에서는 홉단

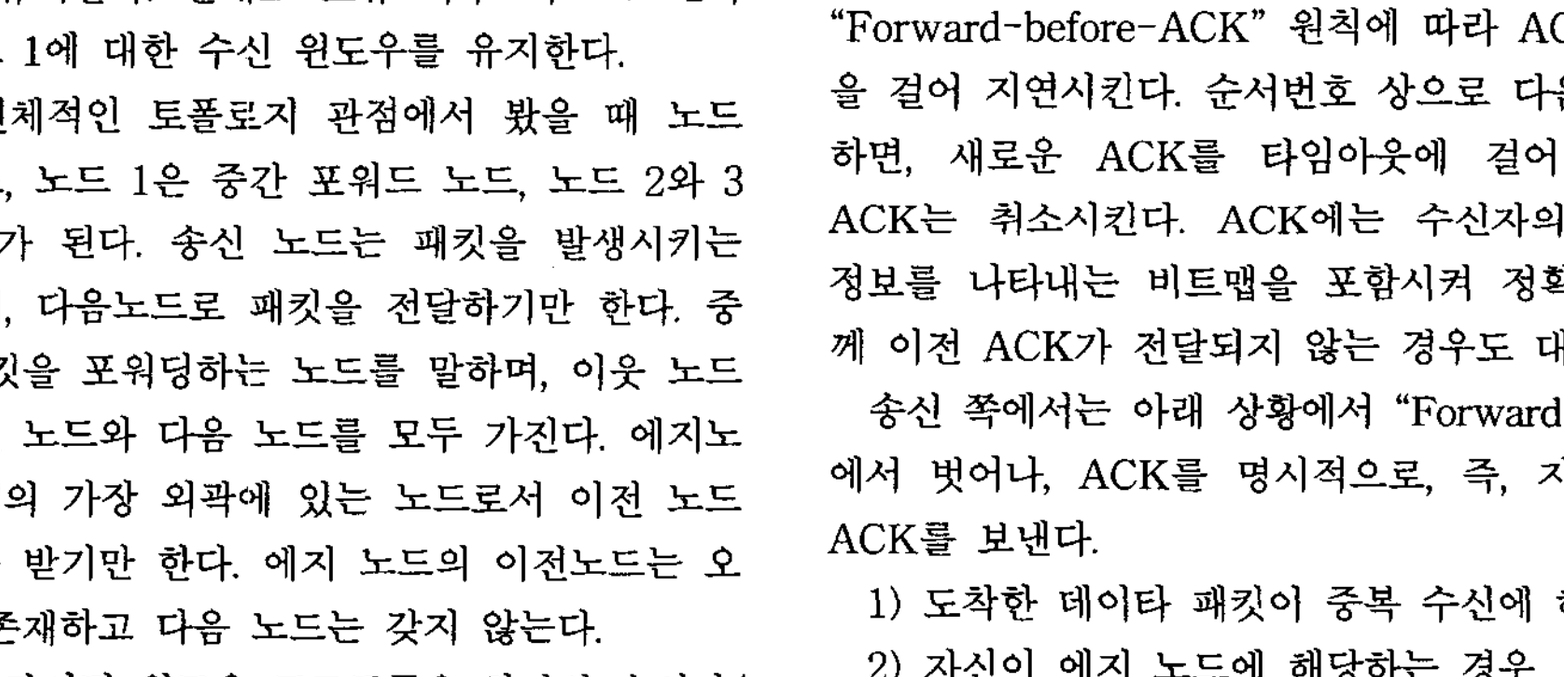

그림 4 이웃노드 테이블의 구성

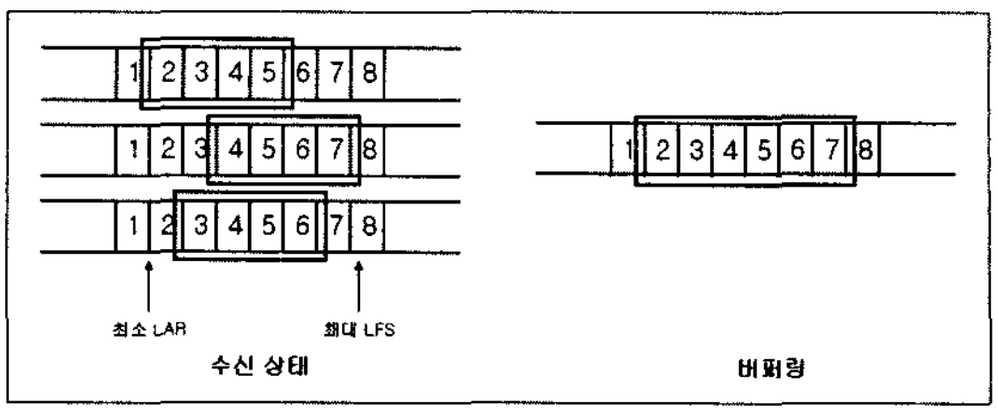

그림 5 RM2I의 송신 윈도우 정책

위에서 1:N의 전송이 이루어지므로 송신 노드 하나당 여러 개의 수신 노드가 존재할 수 있다. 이 때문에 송신 윈도우는 여러 개가 될 수 있으며, 각 노드별로 진행이 다를 수 있기 때문에 수신 상태와 재전송 버퍼를 분리할 필요가 있다. 따라서 송신 윈도우들 중에 최소 LAR (Last ACK Receive)과 최대 LFS(Last Frame Sent) 사이의 패킷을 버퍼에 유지하게 된다. 수신 상태와 버퍼링의 분리는 그림 5와 같이 이루어진다.

4.2 묵시적 ACK 및 간접 복구의 구현

RM2I는 기본적으로 selective repeat 슬라이딩 윈도우 프로토콜로 동작한다. 묵시적 ACK에서 송신 쪽은 "Forward-before-ACK" 원칙에 따라 ACK를 타임아웃을 걸어 지연시킨다. 순서번호 상으로 다음 패킷이 도착하면, 새로운 ACK를 타임아웃에 걸어 놓고 이전의 ACK는 취소시킨다. ACK에는 수신자의 윈도우 상태 정보를 나타내는 비트맵을 포함시켜 정확한 상태와 함께 이전 ACK가 전달되지 않는 경우도 대비한다.

송신 쪽에서는 아래 상황에서 "Forward-before-ACK"에서 벗어나, ACK를 명시적으로, 즉, 지연 없이 즉시 ACK를 보낸다.

1) 도착한 데이타 패킷이 중복 수신에 해당하는 경우
2) 자신이 에지 노드에 해당하는 경우
3) 주기적으로

1)의 경우는 묵시적 ACK가 제대로 전달되지 않은 상

태에서 포워딩을 할 필요는 없으므로 ACK를 보내야 한다. 에지 노드는 다음 노드로 포워딩하는 경우가 없으므로 묵시적 ACK는 성립하지 않고, 따라서 즉시 ACK를 보내야 하며, 2)의 경우가 이에 해당한다. RM2I는 이웃 노드 테이블의 구축 및 안전한 오류 제어 동작을 위해 중간 노드의 경우에도 주기적으로 명시적 ACK를 보내도록 하였다.

수신 쪽은 수신된 패킷이 ACK의 의미를 갖는지 확인하고, 해당하는 경우 슬라이딩 윈도우의 ACK 수신 경우에 따라 동작하도록 한다. RM2I로 올라온 패킷은 모두 목적지 주소가 지정된 멀티캐스팅 주소이고 발신자는 소스이다. 따라서 자신이 다음 노드로 보낸 패킷의 순서번호와 같다고 해서 다음 노드가 포워딩한 패킷이라고 단정하여 묵시적 ACK로 처리할 수는 없다. RM2I는 헤더에 포워딩, 즉, 중계하는 노드의 ID를 넣어서 수신 RM2I가 해당 패킷을 포워딩한 노드를 인식할 수 있게 하였다. 한편, 묵시적 ACK로 동작하기 위해서는 비트맵 정보가 필요하므로 이를 패킷에 포함되도록 하였다. RM2I의 헤더 구조는 그림 6과 같다.

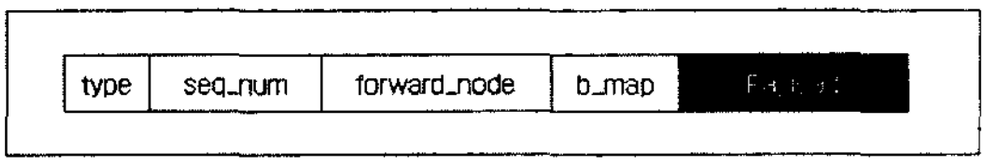

그림 6 RM2I 헤더 구조

RM2I 헤더 구조에서 type은 데이타가 패킷인지 ACK인지 구분해 주고, forward_node는 현재의 패킷을 포워딩한 노드를 나타낸다. seq_num은 순서번호를 나타내고, b_map은 forward_node나 수신 쪽의 버퍼 상태를 나타내는 비트맵으로 묵시적 ACK 및 ACK의 내용이 된다.

간접 복구는 정의에서 알 수 있듯이 이전 노드가 아닌 노드가 보낸 데이타 패킷도 수신하여 슬라이딩 윈도우 오류 제어에 사용하는 것이다. 유일한 차이는 수신한 패킷의 포워드 노드 ID(forward_node)에 관계없이 ACK는 이웃노드 테이블 상의 이전 노드에게 보낸다는 것이다. RM2I는 기본적으로 간접 복구와 묵시적 ACK를 동시에 적용하므로, 실제적으로는 위의 설명과 같이 다음 노드로 포워딩하여 묵시적 ACK를 보내거나, 즉시 명시적 ACK를 보낸다.

5. 실험 및 평가

RM2I의 에너지 효율성을 입증하기 위해 NS-2 시뮬레이터[9]에서 실험을 하였다. 멀티캐스트 라우팅 계층으로는 MAODV[10]를 채택하여 Carleton 대학의 MAODV 구현물[11]을 사용하였다. 기타 Application, 802.11, Network Interface, 오류 및 에너지 모델 등은 NS-2에서 기본적으로 구현되어 있는 것들을 사용하였다. 802.11에 구현되어 있는 대역폭 및 에너지 소비 전력은 표 1과 같다.

표 1 모의실험에 사용되는 대역폭 및 에너지 소비 전력

매개변수	값
802.11 MAC 대역폭	1.6Mbps
유휴 전력	35mW
수신 전력	395mW
송신 전력	660mW

표 2 모의실험에 사용되는 데이타의 크기

데이타 종류	크기(byte)
전송 패킷	200
ACK	40

센서 네트워크에서는 비교적 작은 크기의 패킷을 전송한다. 패킷과 ACK의 크기는 표 2와 같이 설정하였다. 실험에 주로 사용한 트래픽 타입은 CBR(constant bit rate)이다. CBR은 어플리케이션에서 일정한 주기로 패킷을 발생시키기 때문에 실험 결과를 분석하기에 용이하다. 제안 알고리즘이 어플리케이션 및 패킷 발생 분포에 독립적이라는 것을 보이기 위한 Telnet 응용에 대한 실험도 실시하였으며, 뒤에서 설명한다.

에너지 효율성을 측정하기 위한 지표로는 모든 노드에서의 평균 소모 에너지(J)를 사용하였다. 평균 소모 에너지는 NS-2에 구현되어 있는 에너지 모델을 이용하여 측정하였다. 그러나 이렇게 측정된 에너지는 멀티캐스트 라우팅이나 무선 MAC과 같은 다른 계층에 의해 발생된 에너지 소모도 포함하고 있다. 따라서 제안 알고리즘만의 에너지 소모를 측정하기 위해 전체 측정한 에너지에서 다른 계층에서 사용한 에너지 소모를 빼는 과정을 거쳤다.

다양한 토폴로지에서의 동작을 관찰하기 위해 그림 7과 같은 일반 토폴로지, 거대 밀집 토폴로지, 선형 토폴로지에서 실험하였다. 일반 토폴로지는 Sink-to-Sensors 통신 상황을 대변하는 토폴로지로 선택하였다. 거대 밀집 토폴로지는 밀집도가 높아 이웃 노드가 많은 토폴로지로서 제안 기법의 확장성을 확인하기 위해 선택하였다. 선형 토폴로지는 이웃 노드 수가 한정되는 반면, 에지노드 수가 적은 특징을 갖는다.

5.1 Multicast-ACK와 Unicast-ACK의 비교

본 실험 목적은 트랜스포트 수준의 ACK를 링크 수준에서 어떻게 보내는 것이 총량적으로 에너지 효율적인가를 확인하는 것이다. 무선 MAC 계층에서의 동작 및 에너지 효과를 관측하여야 하므로, 통신 환경은 가장 간단하게 설정하였다. 실험에 사용되는 인자 값들은 표 3과 같다.

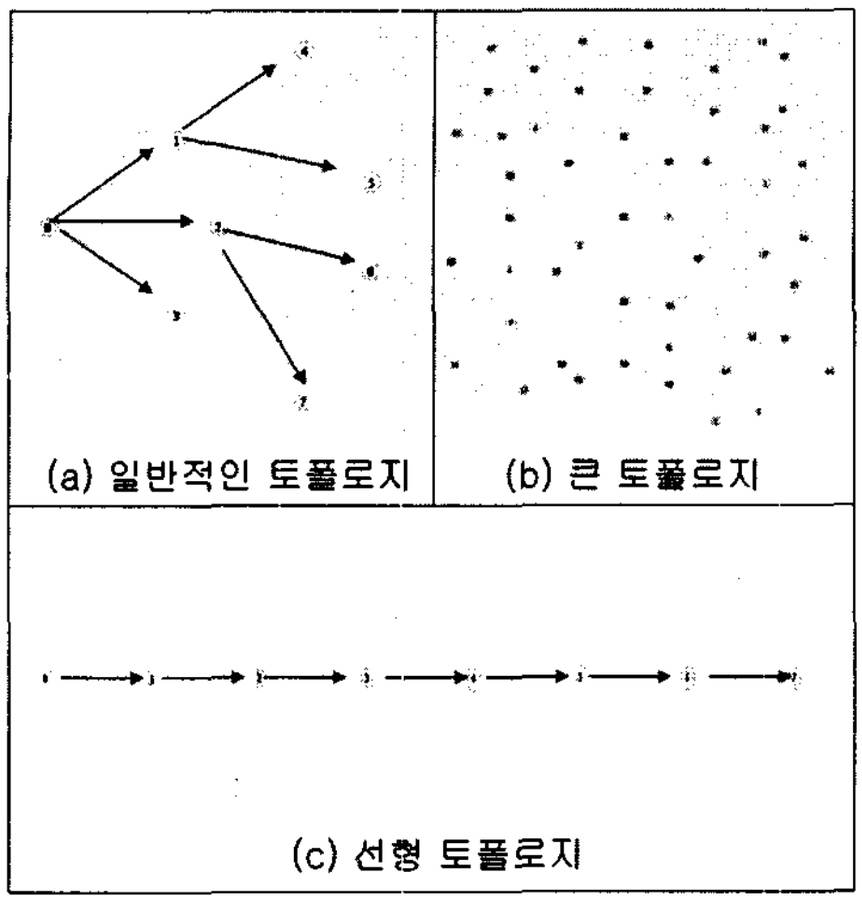

그림 7 실험에 사용된 토폴로지

표 3 Multicst-ACK와 Unicast-ACK를 비교 실험하는 데 사용되는 인자 값

인자	값	
토폴로지	타입	일반적인 토폴로지
통신 응용	타입	CBR
	패킷 생성 주기	5.0초
	재전송 타임아웃	1.0초
	Window 크기	1
	실험시간	1000초
오류율	0, 5, 10%	

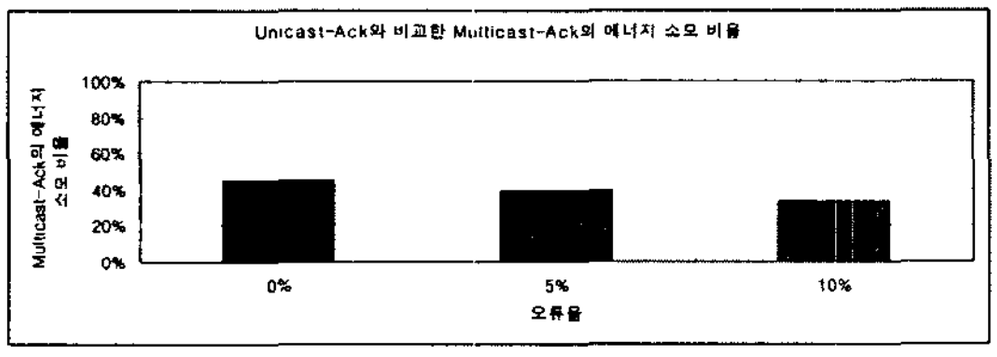

그림 8 Unicast-ACK 대비 Multicast-ACK의 에너지 소모 비율

전통적인 Unicast-ACK에 대한 본 연구의 Multicast-ACK의 평균 소모 에너지(J)의 비를 평가 지표로 한 실험 결과는 그림 8과 같다. 실험 결과, 모든 오류율에서 50% 이하의 값을 보이고 있으며, 이는 Multicast-ACK를 사용하는 것이 각각 44/38/38 바이트 길이인 RTS/CTS/ACK를 사용하지 않기 때문에 에너지 절약 효과가 있다는 것을 보여주고 있다.

5.2 묵시적 ACK와 간접 복구 기법의 에너지 성능 분석

묵시적 ACK 및 간접 복구 기법을 selective repeat과 비교 실험하였다. 실험에 사용되는 패킷 생성 주기는 2.0초, 재전송 타임아웃은 5.0초로 설정하였다. 이는 패킷 오류 시 재전송 타임아웃을 길게 하여 묵시적 ACK의 효과가 나타나기에 충분한 시간을 제공하기 위함이다. 윈도우 크기는 8, 실험시간은 500초, 오류율은 0%, 10%로 설정하였다.

표 4 묵시적 ACK 및 간접 복구 기법의 토폴로지 및 오류율별 에너지 소모(J)

	오류율 0%	오류율 10%		
	selective repeat	selective repeat	묵시적 ACK	간접 복구
일반 토폴로지	0.50	0.70	0.62	0.55
선형 토폴로지	0.36	0.49	0.31	0.49

먼저, 일반 토폴로지에서 묵시적 ACK의 동작을 확인하였다. 표 4에서 보는 바와 같이 일반적인 토폴로지에서 selective repeat의 경우 10%의 오류가 발생하였을 때, 오류가 발생하지 않았을 때 보다 약 42%의 전송 에너지를 더 소모하였다. 반면 묵시적 ACK는 약 25%의 전송 에너지를 더 소모하여 selective repeat보다 에너지 효율적인 오류 복구를 수행한다는 것을 알 수 있다.

다음으로 선형 토폴로지에서 묵시적 ACK 알고리즘의 효과를 확인하였다. 선형 토폴로지에서는 overhearing 문제도 거의 발생하지 않아서 이웃 노드가 많은 일반 토폴로지보다 전체적인 평균 소모 에너지가 적다. 또한 시작과 끝 노드를 제외한 모든 노드가 중간 노드로 동작하므로 묵시적 ACK 효과가 거의 모든 노드에서 발생한다. 표 4에서 보는 바와 같이 전통적인 selective repeat의 경우 10%의 오류가 발생하였을 때, 오류가 발생하지 않았을 때 보다 약 37%의 전송 에너지를 더 소모하는 반면, 묵시적 ACK는 오류가 발생하지 않은 selective repeat에 비해 오히려 약 13%의 전송 에너지를 더 절약하였다. 이는 오류가 발생하여 재전송에 소모되는 에너지보다 ACK를 줄여 절약하는 에너지가 더 많음을 나타낸다.

이어서 간접 복구 기법만을 적용하여 한 RM2I의 에너지 효과를 분석하였다. 일반 토폴로지에서 간접 복구를 적용하였을 때는 10%의 오류 복구를 위해 약 11%의 전송 에너지를 더 소모하며, 이는 selective repeat의 42%, 묵시적 ACK의 25%보다 에너지 절약의 효과가 더 뛰어나다는 것을 나타낸다.

선형 토폴로지에서는 간접 복구에 참여하는 이웃 노드들이 없기 때문에 에너지 절약을 기대하기 힘들다. 실험 결과, 간접 복구에서 10%의 오류가 발생하였을 때 오류가 발생하지 않은 selective repeat에 비해 약 37%의 에너지를 더 소모하였다. 이는 간접 복구의 효과가 거의 없었다는 것을 뜻한다.

5.3 RM2I의 에너지 성능

묵시적 ACK와 간접 복구를 적용한 RM2I의 성능을

여러 실험 인자에 대해 실험하여 에너지 성능을 검증하였다. 먼저 일반 토폴로지에서 윈도우 크기에 따라 각 기법들이 어떠한 효과를 내는지 실험하였다. 나머지 인자들은 앞의 실험과 동일하고 윈도우 크기를 2에서 16까지 변화시키면서 실험하였다.

그림 9에서처럼 selective repeat의 경우, 오류가 없을 때는 윈도우 크기에 상관없이 일정한 양의 전송 에너지를 소모하며, 10%의 오류가 발생하면 윈도우 크기에 따라 에너지 소모량이 약간 감소하다가 평준화 된다는 것을 알 수 있다. 이는 selective repeat의 누적 ACK의 효과 때문에 전송되는 ACK 수가 감소해서 나타나는 결과이다. 10%의 오류 상황에서 묵시적 ACK도 윈도우 크기에 따라 에너지 소모량이 감소하다가 평준화되며, 전반적으로 selective repeat보다 적은 양의 에너지를 소모하였다. 윈도우의 크기가 증가하면서 중간 노드가 포워딩하는 기회도 많아지고, 이에 따라 묵시적 ACK의 효과도 증가한다. 실제로 중간노드에서는 명시적으로 ACK를 보내는 경우는 거의 발생하지 않으며, 에지 노드만이 명시적 ACK를 보낸다.

간접 복구에서도 10%의 오류가 발생한 경우 유사한 결과를 보인다. 간접 복구에서는 윈도우 크기가 커질수록 전송되는 패킷들이 많아지므로 이를 간접적으로 수신하는 노드들이 그만큼 많은 패킷을 쌓아두었다가 오류 복구에 사용할 수 있다. 묵시적 ACK와 간접 복구를 같이 사용하면 두 기법의 효과들이 각각 반영되어 오류가 없는 selective repeat보다도 에너지를 절약한다. 이는 묵시적 ACK의 효과로 ACK 부하가 줄어들고, 간접 복구의 효과로 손실 패킷에 대한 재전송의 수가 줄었기 때문이다.

다음은 재전송 타임아웃 시간이 각 기법들에 어떻게 영향을 미치는지에 대해 실험하였다. 일반 토폴로지에서 재전송 타임아웃을 RTT의 2배, 4배, 8배, 12배, 16배, 20배로 증가시켜가면서 실험을 진행하였다. 윈도우 크기는 8로 설정하였다. 패킷 생성 주기는 RTT의 4배와 8배 사이의 시간과 비슷한 값인 0.15초로 설정하였다. 이로 인해 재전송 타임아웃이 패킷 생성 주기보다 크거나 작을 때 각 기법들이 어떤 효과를 내는지 관찰할 수 있었다. 실험 결과는 그림 10과 같다.

처리량의 경우와는 달리, 원칙적으로 재전송 타임아웃이 길어지더라도 에너지 소모 면에서 손해는 없다. selective repeat은 오류가 발생하지 않은 경우 거의 일정하게 에너지를 소모하였으며, 10%의 오류에서는 재전

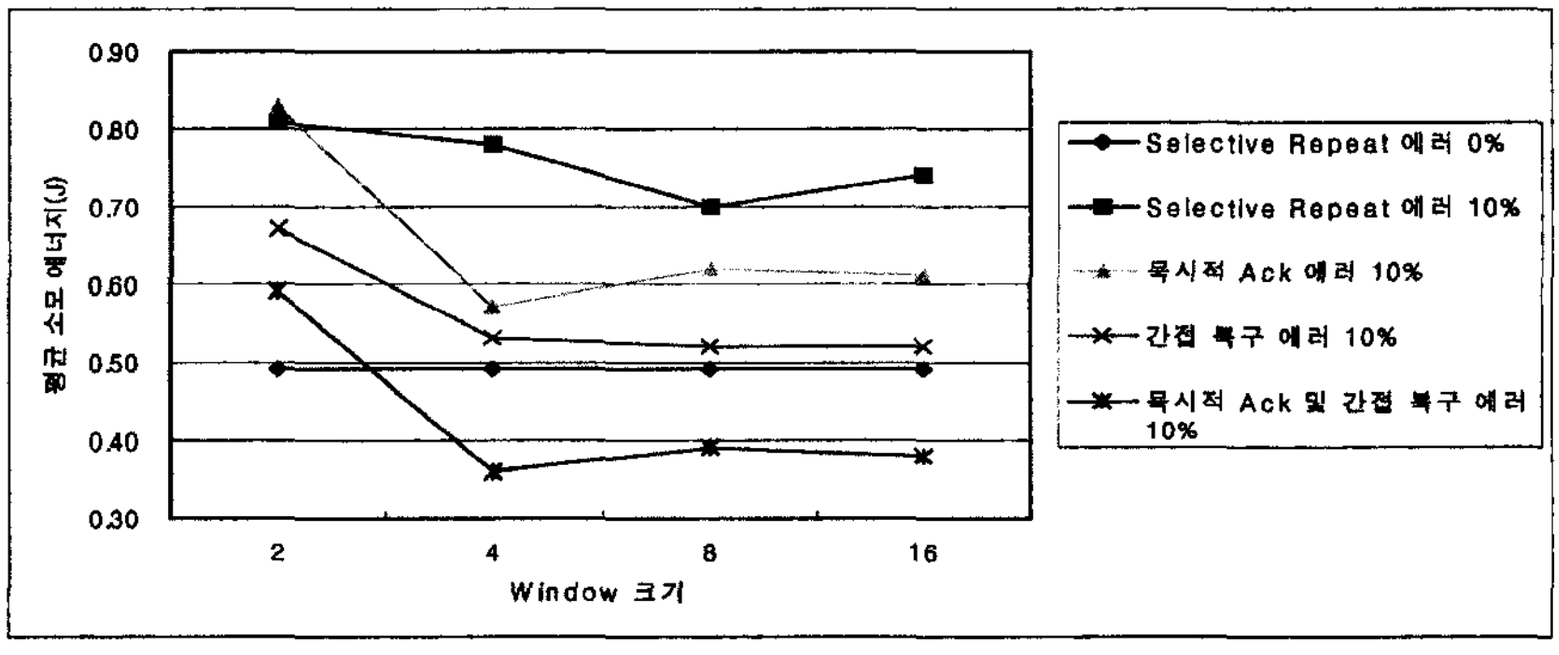

그림 9 윈도우 크기에 따른 평균 전송 에너지 소모 비교

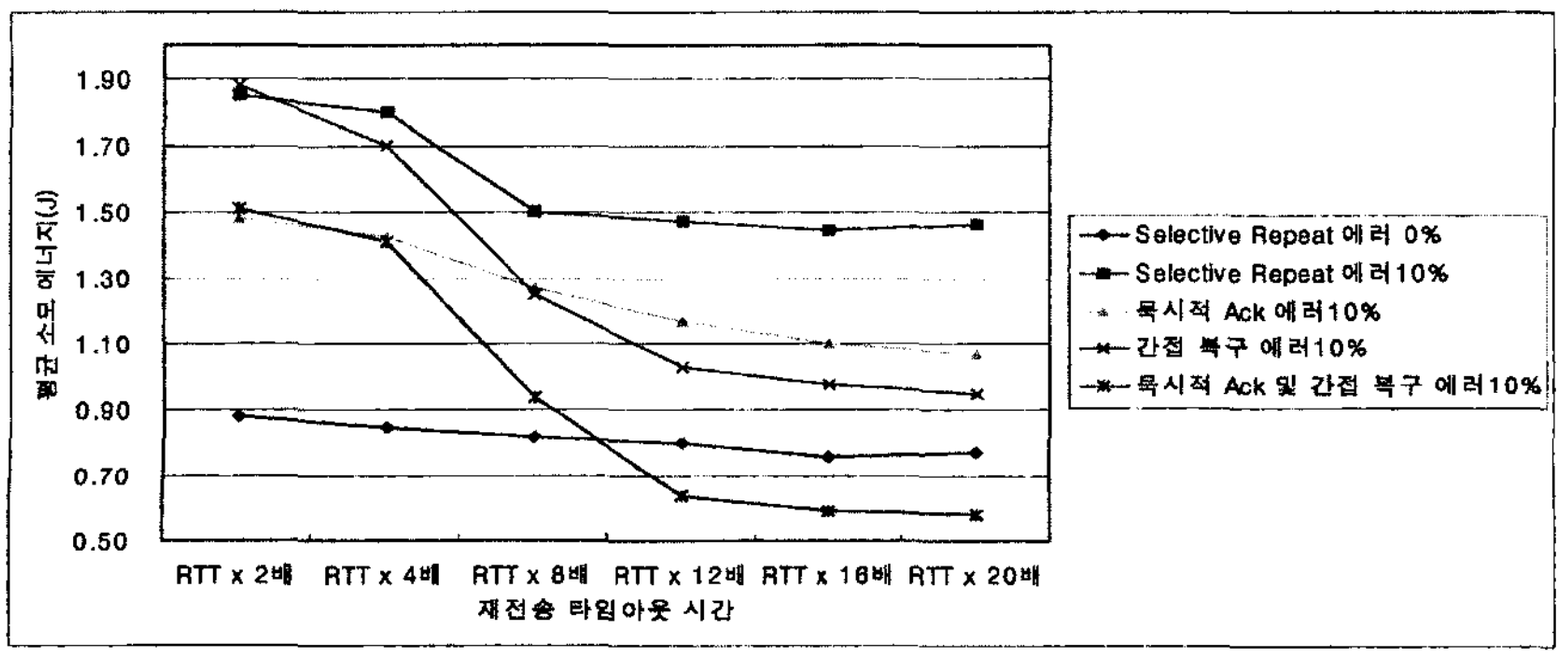

그림 10 재전송 타임아웃 시간에 따른 평균 전송 에너지 소모 비교

송 타임아웃 시간이 RTT의 2배를 넘어서면서부터 에너지 소모를 조금씩 줄이는 면모를 보였다.

묵시적 ACK의 경우, 재전송 타임아웃 시간에 따라 비교적 일정한 비율로 에너지 절약을 보여주었다. 이는 송신 노드가 묵시적 ACK를 감지할 시간이 충분하기만 하면 재전송 타임아웃 시간에 큰 영향을 받지 않는다는 것을 알려준다. 간접 복구는 재전송 타임아웃 시간에 민감한 반응을 보였다. 재전송 타임아웃 시간이 RTT의 2배일 때는 10%의 오류가 발생한 selective repeat과 유사하게 에너지를 소비했지만 재전송 타임아웃 시간이 길어질수록 그 차이가 벌어졌다. 이는 재전송 타임아웃 시간이 길어질수록 그 사이에 더 많은 패킷들을 챙겨두어서 오류 복구에 사용한다는 것으로 해석된다.

10%의 오류가 발생했을 때 묵시적 ACK와 간접 복구를 모두 사용한 경우에서는 위 두 기법들의 에너지 효율적인 측면이 모두 반영되어 상당한 에너지 절약을 가져다주었다. 재전송 타임아웃 시간이 늘어남에 따라 절약하는 에너지도 많아졌고 RTT의 12배를 넘어서면서부터는 오류가 발생하지 않은 selective repeat보다도 더 많이 에너지를 절약하는 면모를 보였다.

이상의 실험 결과를 보면, RM2I는 충분한 윈도우 크기와 비교적 긴 타임아웃에서 에너지 절약 효과를 보인다고 할 수 있다. 윈도우 크기가 커지면 노드에 버퍼가 많이 필요하고 버퍼 내에 있는 패킷 등을 검색하거나 정렬하는 등의 연산 측면에서도 추가적인 오버헤드가 들어간다. 그러나 센서 노드 하드웨어의 발달 속도를 감안할 때, 수 KB 정도의 버퍼는 충분히 지원될 수 있다고 예상된다. 또한 연산에 소모되는 에너지 부하는 전송 오버헤드에 비해 미비하기 때문에 전체적인 에너지 절약 효과를 기대할 수 있다.

지금까지의 실험들은 결과 분석을 쉽게 하기 위해 주기적인 패킷 생성을 하는 CBR을 응용으로 사용하였다. RM2I가 응용의 트래픽 특성에 독립적으로 동작하는지를 확인하기 위해 패킷 생성 주기를 갖지 않는 Telnet을 사용하여 실험하였다. 일반 토폴로지에서 실험한 결과는 표 5와 같다. 두 기법 모두 selective repeat보다 좋은 에너지 성능을 보였지만, 간접 복구 기법은 CBR 경우에 비해 효과가 줄어들었다. 이는 Telnet의 패킷 생성 간격이 불규칙해지면서 시간에 민감한 간접 복구가 영향을 받은 것으로 해석된다. 묵시적 ACK와 간접 복구를 모두 사용하면 오류가 발생하지 않은 selective repeat과 유사한 성능을 보인다. 따라서 RM2I는 응용에 독립적으로 동작한다고 할 수 있다.

RM2I의 확장성을 검증하기 위해 노드 수가 50개인 거대 밀집 토폴로지에서 실험을 하였다. 표 6에서 보는 바와 같이 노드의 개수가 많은 상황에서도 RM2I의 에너지 효율성은 유사하게 나타났다. 밀집 토폴로지에서는 이웃 노드들이 많기 때문에 간접 복구의 효과가 비교적 뛰어나게 나타났다. 10%의 오류 환경에서 묵시적 ACK와 간접 복구를 같이 사용하였을 때 오류가 발생하지 않은 selective repeat보다 약 53% 정도의 에너지를 절약하였다.

5.4 NAK 기반 기법의 에너지 부하 하한선과의 비교

ACK 기반의 RM2I가 NAK 기반의 다른 오류 제어 기법과 어떤 에너지 경쟁력을 갖는지 확인하기 위해 간단한 성능 비교를 하였다. 특정 NAK 기반 기법을 NS-2에 구현하여 실험하는 대신에, NAK 기반 기법이 이론적으로 얻을 수 있는 최상의 에너지 성능 한계를 계산하고 이를 RM2I와 비교하였다. 구체적으로, 오류가 발생한 패킷에 대해서 한 번의 NAK와 한 번의 재전송만으로 오류가 복구된다는 가정 하에 식 (1)과 같이 데이타 패킷 수와 NAK 수를 계산하고, 이를 RM2I 시뮬레이션 결과와 비교하였다. NAK 기반에 대한 예상치는 실제로는 이루어질 수 없는 이상적인 수치이다. 따라서 NAK 기반 기법에 대한 오류 복구 부하의 안전한 하한선이라고 할 수 있다.

패킷 전송 수 = 오류가 발생하지 않았을 때의 패킷 전송 수 * (1 + 오류율)

Nak 전송 수 = 오류가 발생하지 않았을 때의 패킷 전송 수 * 오류율

식 (1) NAK 기반 기법의 패킷 및 NAK 전송 수에 대한 확률적 예상치

일반 토폴로지에서 비교 실험한 결과를 그림 11에 나

표 5 어플리케이션 Telnet을 사용하였을 때의 평균 소모 에너지

	오류율 0%	오류율 10%			
	selective repeat	selective repeat	묵시적 ACK	간접 복구	묵시적 ACK 및 간접 복구
일반 토폴로지	0.67	1.05	0.75	0.96	0.68

표 6 거대한 토폴로지에서의 평균 소모 에너지

	오류율 0%	오류율 10%			
	selective repeat	selective repeat	묵시적 ACK	간접 복구	묵시적 ACK 및 간접 복구
일반 토폴로지	2.02	3.85	3.02	1.90	0.93

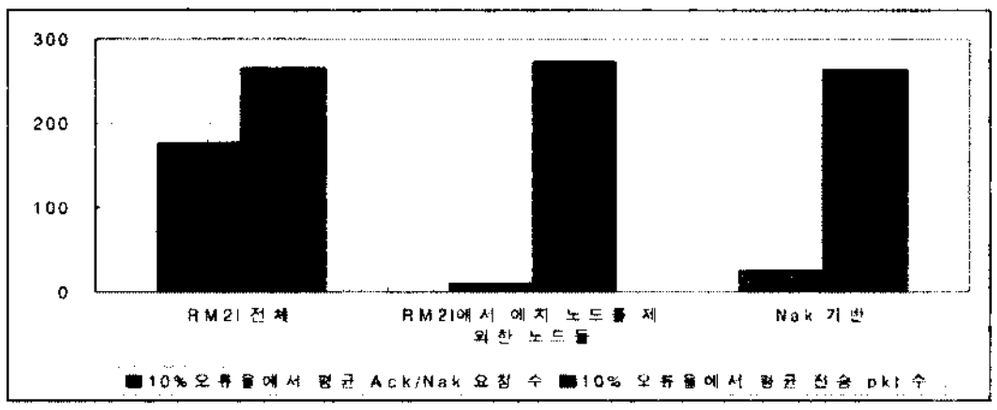

그림 11 일반적인 토폴로지에서 RM2I와 NAK 기반 기법의 비교

표 7 선형 토폴로지에서 RM2I와 NAK 기반 기법의 비교

	RM2I 전체	Nak 기반
10% 오류율에서 평균 Ack/Nak 요청 수	107	24
10% 오류율에서 평균 패킷 전송 수	284	264

타내었다. RM2I 기법에서 전체 노드들에 대해 발생된 데이타 패킷 전송 수는 NAK 기반에서 발생된 데이타 패킷 전송 수에 매우 근접하다. 그러나 RM2I의 ACK 전송 수는 NAK 기반의 NAK 전송 수보다 훨씬 많았다. RM2I에서 ACK는 대부분 에지 노드에서 전송되므로 에지 노드를 제외한 노드들에 대해 발생된 ACK만을 측정해보니 NAK 전송 수보다 작았다.

이를 보다 명확히 분석하기 위해 선형 토폴로지에서 비교해 보았다. 표 7에서 보는 바와 같이 여전히 RM2I 기법에서 전체 노드들이 발생시킨 ACK의 수가 NAK 기반에서 발생시킨 NAK의 수보다 많기는 하지만 일반 토폴로지 실험에서 보다 차이가 많이 줄어들었다. 이는 선형 토폴로지에서는 중간 노드들이 많아서 묵시적 ACK가 많이 이루어지기 때문이다. 그러나 이웃 노드들의 수가 작기 때문에 간접 복구의 효과가 줄어들어서 데이타 패킷 전송 수의 차이는 약간 늘어났다.

결론적으로, NAK 기반 부하에 대한 예상치가 실제로 얻기는 어려운 안전한 하한선이라는 점을 고려할 때, RM2I는 어떠한 NAK 기반 오류 제어 기법과 견줄 수 있는 에너지 성능을 보인다고 할 수 있다. 또한, ACK 전송 부하 대부분이 에지 노드에 의해 발생하므로, 향후 연구로 에지 노드에 NAK를 적용하는 것으로 RM2I를 확장하면 에너지 효율성을 더욱 높일 수 있을 것으로 예상된다.

6. 결론 및 향후 연구

센서 네트워크에서 신뢰성 있는 멀티캐스트에 대한 요구가 늘어가고 있다. 기존에 유선에서 사용하던 알고리즘들이 많이 있지만 에너지 효율적인 측면도 같이 고려해야 하는 센서 네트워크에서 이런 알고리즘들은 적합하지 않다. 현재까지 개발된 센서 네트워크에서의 신뢰성 있는 멀티캐스트 기법들은 주로 NAK 기반의 오류 복구 기법들을 사용하였다. 그러나 이 기법은 마지막 패킷에 대한 신뢰성보장이 어렵다는 등의 문제가 남아 있다. 본 논문에서는 모든 패킷에 대해 완벽히 신뢰성을 보장하고도 에너지 효율성을 증대시킬 수 있는 ACK 기반의 오류 복구 기법을 제안하였다.

먼저 묵시적 ACK를 제안하였다. 무선 통신에서 묵시적 ACK는 송신 노드로부터 전송된 패킷을 받은 수신 노드가 그 패킷을 포워딩 혹은 릴레이 할 때, 송신 노드도 이 패킷을 간접적으로 수신하게 되는데, 이 때 수신한 패킷을 수신 노드가 패킷을 잘 받았다는 것으로 간주하여 ACK로 인식하는 것을 말한다. ACK 응답에 대한 부하를 줄임으로써 에너지를 절약할 수 있었다.

다음으로 간접 복구를 제안하였다. 무선 통신에서 간접 복구는 어떤 노드가 자신의 상위 노드가 아닌 이웃 노드들로부터 패킷을 간접적으로 받았을 때 이를 무시하지 않고 챙겨 두어서 오류 복구에 활용하는 것을 말한다. 송신 노드의 재전송에 대한 부하를 줄임으로써 에너지 효율성을 높일 수 있었다. 이 두 기법은 무선 통신에서 기존의 연구들이 회피하려고 했던 overhearing 문제를 멀티캐스팅이라는 사용 목적에 맞추어 적극적으로 활용하려는 시각에서 나온 정책이라고 할 수 있다.

네트워크 시뮬레이터 NS-2를 통하여 제안 알고리즘의 성능 개선 효과를 확인하였다. 묵시적 ACK는 네트워크 경계에 있는 에지 노드를 제외한 노드에서 ACK 전송에 소모되는 제어 부하를 줄여서 성능 개선을 이루었다. 간접 복구는 이웃 노드가 많고 재전송 타임아웃 시간이 비교적 긴 환경에서 데이타의 재전송에 소모되는 부하를 줄이는 효과를 발휘한다는 것을 확인하였다. 묵시적 ACK와 간접 복구를 혼합하였을 때에는 성능 개선이 극대화 되어 10%의 오류 복구를 신뢰성 있게 달성하고도 오류가 발생하지 않았을 때의 selective repeat보다 오히려 에너지 소모가 줄어드는 등의 결과를 보여주었다.

본 연구의 기법이 주변에서 들려오는 통신 내용을 활용하는 것이다 보니, 수신에 소모되는 에너지에 대한 우려가 있다. 이에 대한 본 연구의 입장은 다음과 같다. 수신 에너지를 근본적으로 줄이는 방법은 overhearing 해결 기법들이 공통적으로 제시하는 무선 인터페이스를 휴면(sleep)시키는 방법이다. 그러나 신뢰성 있는 전송을 위해서는 송신자는 수신 응답을 수신자는 재전송을 받아야만 하므로, 어떠한 오류 제어 기법도 신뢰성 있는 패킷 전송이 완료되는 기간에는 무선 인터페이스를 활동(active)시켜야 한다. 결국, 오류 제어 기법에 따라 활동 기간 동안 각 노드의 무선 인터페이스가 수신하는

양에 차이가 있을 뿐이지 본 연구의 기법이 특별히 수신을 많이 하는 것은 아니다. 인터페이스 활동 기간을 줄이는 오류 제어 기법이 개발된다면, 본 연구에도 이를 적용될 수 있을 것이다. 요약을 한다면, 어차피 수신할 것이라면 이를 수신해서 활용하자는 것이다. 활용을 위한 처리에 소모되는 에너지는 전송을 줄여서 얻는 에너지 절약에 비하면 미미하므로 문제가 되지 않는다. 본 연구의 기법은 각 노드가 전송하는 데이타 및 제어 패킷의 횟수를 줄이므로 주변 노드의 인터페이스가 수신하는 양도 줄어드는 효과가 있다. 실험의 에너지 소모양은 송신은 물론 수신에 소모되는 에너지도 고려한 것이므로, 본 연구의 에너지 절약 효과는 유효하다고 할 수 있다.

향후 연구로 에지 노드에서 NAK 기법을 적용하도록 RM2I를 확장시킬 것이다. 실험 결과, 제안 기법에서의 ACK 부하 중 대부분이 에지 노드에 의해 발생되는 것을 알았다. 따라서 이 에지노드에 NAK 기반 오류 복구 기법을 도입하면 ACK에 소모되는 제어 부하를 줄여서 에너지 효율성을 더욱 개선시킬 수 있을 것이다.

참 고 문 헌

[1] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," *IEEE/ACM Transactions on Networking*. November. 1996.

[2] 김용진 외, "멀티캐스트 전송을 위한 오류제어기법의 분류", *전자통신동향분석 제14권 제3호*, 1996년, 6월.

[3] T. Stathopoulos, J. Heidemann, D. Estrin, "A Remote Code Update Mechanism for Wireless Sensor Networks," *CENS Technical Report #30*, UCLA, Department of Computer Science, USC, Information Sciences Institute, 2003.

[4] S-J. Park and R. Sivakumar, "Sink-to-Sensors Reliability in Sensor Networks," *Extended Abstract to appear in Proceedings of ACM MobiHoc*, Annapolis, MD, June 2003.

[5] Millennial Net "Maximizing Data Reliability in Wireless Sensor Networks," *A Millennial Net White Paper*, www.millennialnet.com, 2005.

[6] C-Y. Wan, A. Campbell, and L. Krishnamurthy, "PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks," in *Proc. ACM Int. Workshop on Sensor Networks and Architectures*, Atlanta, Sept. 2002.

[7] Seung-Jong Park, et al., "A Scalable Approach for Reliable Downstream Data Delivery in Wireless Sensor Networks," Proc. of MobiHoc 2004, pp. 78-89, Tokyo, Japan, 24-26, May 2004.

[8] F. Stann and J. Heidemann, "RMST: Reliable Data Transport in Sensor Networks," *Appearing in 1st IEEE International Workshop on Sensor Net Protocols and Applications (SNPA)*, May 2003.

[9] NS-2 Simulator, http://www.isi.edu/nanam/ns/

[10] Royer, E. M. and Perkins, C. E.; "Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing," *IETF, Internet Draft: draft-ietf-manet-maodv-00.txt*, 2000.

[11] Y. Zhu, T. Kunz, "MAODV Implementation for NS-2.26," *System and Computing Engeering, Carleton University*, Technical Report SCE-04-01, January, 2004.

[13] 이윤희, "신뢰성 있는 멀티캐스트 프로토콜에서 NAK 메시지의 기능 분리를 통한 지역적 오류 복구", *중앙대학교 제 94회 석사학위 논문*, 2000년 11월.

김 성 훈
정보과학회논문지 : 정보통신
제 35 권 제 1 호 참조

양 현
정보과학회논문지 : 정보통신
제 35 권 제 1 호 참조

박 창 윤
정보과학회논문지 : 정보통신
제 35 권 제 1 호 참조