이동 센서 네트워크를 위한 적응적 FEC 코드 제어 알고리즘

이영수° 홍승욱 안종석
동국대학교 컴퓨터공학과
{diwkd°, swhong, jahn}@dgu.edu

An Adaptive FEC Code Control Algorithm for Mobile Sensor Networks

Young-Su Lee° Seung-Wook Hong Jong-Suk Ahn
Dept. of Computer Engineering, Dongguk University

요 약

전파(propagation) 오류가 빈번한 무선 이동 네트워크에서는 전송 성능을 향상하기 위해 FEC(Forward Error Correction) 알고리즘을 채택한다. 그러나 정적인 FEC 코드 제어 방식은 연속적으로 변화하는 전파 오류율에 알맞은 정정 코드(check code)를 적용하지 못해 성능이 저하된다. 일례로 측정한 고 오류 무선 센서 네트워크에서는 초단위 평균 BER(Bit Error Rate) 또는 분단위 평균 BER이 0에서 최대 10^{-3}까지 연속적으로 변화한다. 이러한 무선 환경에서 전파 오류를 100% 복구하기 위한 정정 코드를 채택하는 경우에는, 불필요한 정정 코드량은 전체 데이터에 최대 20%를 차지한다. 본 논문에서는 무선 채널의 BER을 직접 측정하지 않고 패킷 전송 성공 여부에 따라 정정 코드의 량, 즉 FEC단계를 동적으로 변화하는 AFECCC (Adaptive FEC Code Control) 알고리즘을 소개한다. AFECCC는 트레이스 기반(trace-driven) 시뮬레이션에서 정적 FEC 방식에 비해 최대 5% 이상, 또한 실제 센서 네트워크에서는 정적 FEC 알고리즘에 비해 최대 15% 성능이 향상되었다.

1. 서 론

무선 채널의 평균 BER은 약 10^{-6}~10^{-3}으로 평가되고 있어, 전파 오류 방지나 복구 방법을 채택하지 않는 경우에는 대부분 패킷들이 전파 오류에 의해 손실된다. 본 연구에서 사용되는 센서 네트워크[1]에서는 신호 간섭이 있는 환경에서 90% 이상의 패킷이 손상되는 것으로 관찰되었다. 또한 무선 채널 BER은 송수신자의 이동 또는 주위 환경 변화에 따라서 큰 폭으로 변화하기 때문에, 이러한 변화에 정적으로 FEC정정 코드를 결정하는 것은 비효율적이다.

FEC는 정정 코드를 이용하여 수신 측에 데이터를 재전송하지 않고 오류를 복구할 수 있으나, FEC 정정 코드량을 무선 채널 BER에 알맞게 결정하기 어렵다는 문제가 있다. 연속적으로 BER이 변화하는 무선 채널의 전송 성능을 개선하기 위해서는 채널 상태에 따라 FEC 부하를 동적으로 조절해야 한다.

본 논문에서는 이러한 무선 채널 상태 변화에 알맞은 수신측의 채널 상태에 대한 정보 없이 응답(ack) 패킷의 도착 유무에 따라 적절한 FEC 단계를 결정하는 적응적 FEC 코드 제어 알고리즘, AFECCC(Adaptive FEC Code Control)를 제안한다. AFECCC는 ack가 도착하지 않으면 사용하는 FEC 단계를 상향하며 일정 기간동안 ack의 분실이 없으면 FEC 단계를 하향하는데, 이때 상/하향 알고리즘으로는 오류율이 크고 빠르게 변화하는 환경에서 성능 향상을 보이는 MIAD(Multiplicative Increase Additive Decrease) 방식을 채택하였다. 실험 결과, 트레이스 기반 시뮬레이션에서 AFECCC는 정적 FEC 제어 알고리즘에 비해 성능 향상을 보였으며, 실제 센서 네트워크에는 최대 15%까지 성능을 향상하였다.

본 논문의 구성은 다음과 같다. 2절에서 적응적 알고리즘의 적용 가능성을 실험적으로 살펴보며, 3절에서는 AFECCC 구조를 설명한다. 4절에서는 트레이스 기반 시뮬레이션과 실제 센서네트워크를 통해 AFECCC의 성능을 측정하며, 5절에서는 결론 및 향후 연구 과제를 요약한다.

2. 센서 네트워크에서 채널 오류 특성 분석

AFECCC 알고리즘의 성능 향상 가능성은 두 가지 수치, 즉 BER 변화의 지속 시간과 BER 변화 폭에 의해 결정된다. BER이 변화하지 않거나, 측정된 채널 상태에 알맞은 FEC 단계로 적응하기 위해 필요한 최소 시간보다 BER이 빠르게 변화한다면 AFECCC를 적용할 수 없다. 성능 향상을 위해서는 BER의 변화 지속 시간이 알맞은 FEC 코드로 적응하기 위한 지연 시간보다 수 십배 이상은 커야 할 것이다.

그림 1은 TR-거리(송수신 노드 사이의 거리)가 6~13m로 증가할 때, NCBPP(the Number of Corrupted Bytes Per Packet)의 표준편차의 분포를 도식한 것이다. 각 수직선은 센서 네트워크에서 4시간 동안 100바이트의 패킷을 3.2kbps의 속도로 계속적으로 전송하는 방식으로 10일 동안 수집한 것이다. 센서 네트워크는 8bit CPU, 128Kbyte 플래쉬 램, 8Kbyte SDRAM로 제작된 Mica Mote로 구성되었으며 Mica Mote는 최대 전송 출력이 90mW인 저출력 900Mz 라디오파를 이용한다.

그림 1에서 TR거리에 따른 평균 BER은 거리 11미터 이내에서는 2바이트 정도로 서서히 증가하나 11미터에서는 11바이트로 급격히 증가한다. 또한 표준편차 값은 거리가 증가할수록 2에서 10바이트까지 변화한다. TR거리가 증가할수록 평균 NCBPP가 증가하는 것은 전송 거리가 증가함에 따라 수신 신호 출력이 감소하는 LSF(Large Scale Fading) 현상으로, NCBPP의 표준 편차의 분포가 넓어지는 것은 다중경로 간섭으로 인한 SSF(Small Scale Fading) 현상으로 설명될 수 있다. 그림 1은 센서 네트워크에서 수신자가 이동하는 경우 또는 수신 출력이 급격히 저하되는 거리에 고정된 경우에도 AFECCC가 필요함을 보여준다. 일례로 센서 노드가 2에서 13미터 반경에서 이동할 때 최대 18바이트 오류를 복구하기 위해서 정적인 RS(Reed-Solomon)[2] FEC 방식을 이용하는 경우, 각 패킷마다 36바이트 정정 코드를 첨부해야 하는데, 대부분의 거리에서는 6바이트의 오류만이 발생하므로 24바이트 정정 코드가 낭비된다.

그림 2-(1), 2-(2)는 AFECCC가 센서 채널에 잘 적응할 수 있다는 것을 보여주는데, 송수신자 거리가 11미터일 때 측정한 오류 트레이스에서 평균 단위 시간을 1초, 10초로 했을 때 평균 NCBPP의 변화를 측정한 것이다. 그림 2에서 알 수 있듯

* 본 연구는 한국과학재단 특정기초연구과제(과제번호: 1NK0402702)의 지원을 받아 수행하였습니다.

이 평균 단위 시간을 달리하여도 평균 NCBPP가 큰 폭으로 변화하며, 700~1200초 사이에서는 거의 변화가 없는 구간도 나타난다. 즉, 이것은 10초 단위의 BER 변화에 맞게 FEC 단계를 변화하여도 성능 향상을 할 수 있다는 것을 의미하며, 거의 변화가 없는 구간에서는 AFECCC가 오류율에 적응하여 높은 성능 향상을 가져올 수 있다는 것을 의미한다.

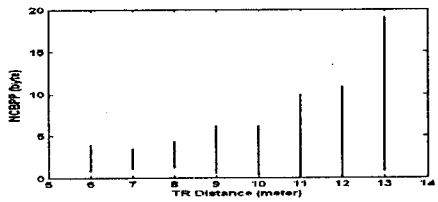
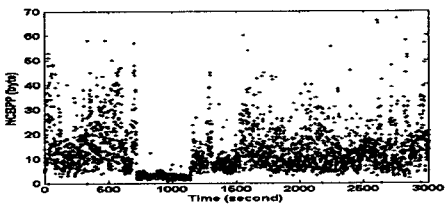
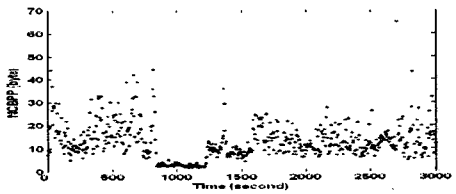


그림 1 TR거리에 따른 NCBPP 분포



2-(1) 초당 평균 NCBPP의 변화



2-(2) 10초당 평균 NCBPP의 변화

그림 2 단위시간별 평균 NCBPP의 변화

그림 3은 센서 채널에서 수집된 트레이스에 네 가지 정적 FEC 알고리즘을 적용하였을 때 이론적으로 낭비되는 시간을 비율로 계산한 것이다. 정적 FEC 알고리즘은 추가적인 FEC 전송으로 인한 대역폭 낭비와 높은 오류율에 비해 적은 FEC를 사용했을 경우 발생하는 패킷 손실에 의한 대역폭 낭비를 갖는다. 그림 3의 네 가지 FEC 단계는 각각 6, 10, 20, 40바이트의 RS 코드를 사용하는데, 각각 3, 5, 10, 20바이트의 에러를 정정한다. 각 FEC 코드량은 수집된 패킷에서 90% 이상의 오류가 2에서 20바이트에 집중된 것을 기초로 결정한 것이다.

그림 3에서 FEC를 사용하지 않는 802.11b를 사용한 경우 70%이상의 대역폭을 패킷 손실로 인한 재전송에 낭비하게 된다. TR거리가 6m일 때, FEC4와 FEC1은 각각 37%와 4%의 대역폭을 부가적인 FEC 코드 전송에, 0.2%와 8%를 재전송에 낭비하고 있다. 반면에 TR거리가 13m일 때는 부가적인 FEC 코드 전송에 25%와 2%, 재전송에 10%와 78%의 대역폭을 낭비하는 것을 알 수 있다. 그림 3에서 측정된 대역폭 낭비를 최소화 하기 위해서는 [6, 10]구간에서 FEC2, [10,12]구간에서 FEC3, 마지막으로 13m구간에서는 FEC4를 적용해야한다.

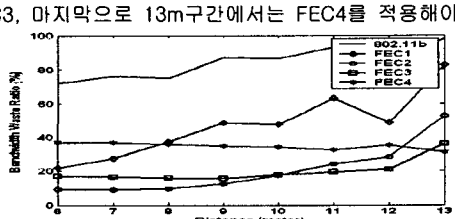


그림 3 정적 FEC 알고리즘의 대역폭 낭비 비율

3. AFECCC 알고리즘

AFECCC는 무선 채널의 상태에 따라 여러 개의 FEC 단계 중에서 채널 상태에 알맞은 단계를 적응적으로 결정한다. 무선 채널의 상태는 ack 패킷의 도착으로 파악하는데, ack의 도착은 현재 채택하는 FEC 단계가 BER보다 높거나 알맞은 것을, ack의 분실은 낮은 것을 의미한다. AFECCC는 알맞은 FEC 단계를 점진적으로 결정하게 되는데, 이것은 무선 망의 수신자단에서 패킷의 손상된 바이트 수를 정확하게 계산할 수 없고 또한 SNR(Signal-to-Noise Ratio)과 같은 수치를 수신자가 측정하고 이를 송신자에게 알려줄 수는 있으나, 이를 이용해서 손상된 바이트 수를 계산하기는 어렵기 때문이다.

AFECCC는 바로 전에 전송한 패킷이 손실되었을 때 즉각적으로 상위 단계 FEC 단계를 채택한다. 이는 FEC 코드 부하보다는 패킷 손실로 인한 전송 시간의 낭비가 크기 때문이다. 그러나 계속적인 FEC 코드 증가는 성능 저하를 야기하므로, 일정 시간동안 패킷 손실이 발생하지 않는다면 과중한 FEC 코드 부하를 줄이기 위해 낮은 FEC 단계로 하향해야 한다.

AFECCC에는 하위 단계로 하향하는 적절한 탈퇴시간을 결정하기 위해, 모든 FEC 단계에 DT(Drop Timer)가 있고 이 DT는 지수적 백오프(exponential back-off) 알고리즘에 의해 관리된다. 새로운 FEC 단계를 채택할 때마다 AFECCC는 그 단계의 DT의 최대 타임아웃 값에 가중치 α(>1)를 곱하여 최대 Tmax까지 지수적으로 증가시킨다. 즉 AFECCC가 어느 단계를 빈번히 채택하면, 이 단계의 타임아웃 시간이 증가되어 이 단계로 안정화된다. 이때, α와 Tmax의 값은 사용 중인 FEC 단계에서 하향하여 채널 상태가 개선되었는지를 검색하는 검색 빈도수를 결정한다. α값이 작으면 현재 사용하는 FEC 단계에서 빈번히 하향되며, 크면 천천히 하향된다.

또한 AFECCC에는 현재 채택되지 않은 모든 FEC 단계들의 DT을 지수적으로 감소시키는 한 개의 PT(Polling Timer)가 있다. 즉, DT를 증가하는 것은 학습에 해당되며, 다른 FEC 단계들의 DT값을 감소하는 것은 과거의 학습 내용을 잊는 것에 해당된다. PT가 매 Tp을 초과할 때마다, 현재 사용 중인 FEC 단계를 제외한 다른 단계의 DT를 붕괴 가중치(decay factor), β(<1)만큼 지수적으로 감소하여 최소 Tmin까지 감소시킨다. 그림 4는 이러한 AFECCC의 FEC 단계 변화를 도식한 것이다.

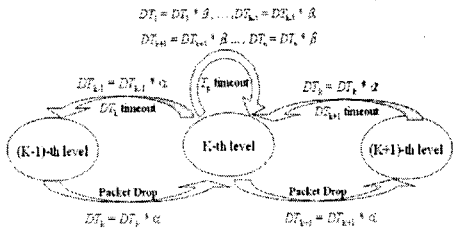


그림 4 AFECCC의 상태 전이 다이어그램

AFECCC 알고리즘에서는 네 개의 변수, α, Tmax, Tmin, β를 조절할 수 있는데, 이 네 변수가 작은 값을 가질수록 AFECCC는 빈번히 FEC 코드의 크기를 줄이게 된다. 본 논문에서는 인접한 두개의 FEC 레벨간의 FEC 코드 크기의 차이가 패킷 크기의 1/h일 때 경험적으로 Tp, Tmin, Tmax 값을 각각 RTT, h * RTT, h * Tmin 으로 설정하였다.

구체적으로는 AFECCC는 하나의 패킷 전송이 성공적으로 수행된 시간 RTT마다 DT의 타임아웃 값을 조절한다. 또한 Tmin을 h개 패킷을 전송하기 위해 소요되는 시간 h * RTT으로 결정한다. 참고로, h * RTT 시간은 FEC 코드를 전송하기 위해 낭비되는 시간 1/h * (h * RTT)와 하나의 패킷 손실로 낭비되는 시간이 같아지기 위해 소요되는 시간이다.

4. 성능 평가

4.1 AFECCC 시뮬레이션 성능평가

그림 5는 실제 센서 네트워크에서 수집한 거리별 트레이스를 기반으로 ARQ(Automatic ReQuest)와 네 가지 정적 FEC (RS(104,100), RS(108,100), RS(120,100), RS(140,100)), 그리고 AFECCC 알고리즘의 성능을 평가한 것이다. 센서 네트워크의 거리별 트레이스는 송신노드가 수신노드로 100바이트의 데이터를 전송할 때 TR 거리를 1m~13m 까지 변화 하면서 수집하였다. 또한 이동하는 센서노드의 오류율 변화를 반영하기 위해 거리별 트레이스를 10분 단위로 혼합하는 방법을 이용하였다. 세 개의 정적 FEC 알고리즘이 예상된 임계 BER 이전부터 성능이 저하되는 것을 볼 수 있는데, 이것은 인공적으로 가공한 트레이스의 에러 분포가 분석적 오류모델의 오류 분포와는 달리 비정규적이기 때문이다. 반면에 AFECCC의 성능은 모든 정적 알고리즘에 비해 좋은 성능을 보이는데, 이것은 10분 단위로 혼합한 트레이스의 오류율 변화가 AFECCC 알고리즘이 적응하는데 충분하기 때문이다.

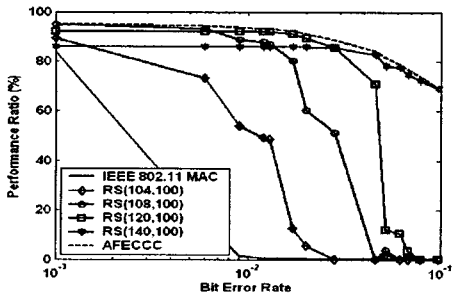


그림 5 트레이스 기반의 AFECCC 성능 평가

4.2 AFECCC 구현

본 절에서는 무선채널 트레이스에 사용한 Mica 센서 노드에서 사용하는 SMAC(Sensor-MAC)[3]에 AFECCC를 구현[4]하는 방법을 설명한다. SMAC은 802.11b와 유사한 MAC계층 프로토콜로서, 전력을 절약하기 위해 수신자가 수신할 데이터가 없을 때에는 슬립(sleep) 모드로 전환되는 프로토콜이다.

AFECCC는 그림 6과 같이 네 개의 이벤트, DT 타임아웃, PT 타임아웃, 재전송 타임아웃, 그리고 패킷 전송 완료를 처리하기위한 핸들러를 SMAC 계층에 구현하며, RS 코드의 변/복조 기능은 물리계층에 구현한다.

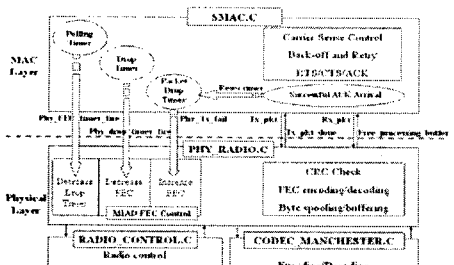


그림 6 SMAC 프로토콜 스택에서 AFECCC 알고리즘의 구현

그림 7은 AFECCC가 동작하기 위해 SMAC의 네 가지 프레임, RTS, CTS, DATA, 그리고 ACK의 헤더를 확장한 것을 도식화한 것이다. RTS, CTS, ACK와 같은 제어 프레임은 상대적으로 짧기 때문에 고정된 길이의 FEC 코드를 사용한다고 가정하였으며, 반면에 DATA 프레임은 FEC 코드량을 명시하기 위한 필드와 FEC 코드를 저장하기 위한 필드를 추가하였다.

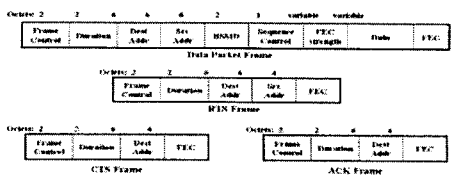


그림 7 AFECCC를 적용한 SMAC 프레임의 네 가지 확장 헤더 구조

그림 8은 Mica Mote로 구성된 송수신 노드가 2시간 동안 80 바이트의 패킷을 연속적으로 전송했을 때 AFECCC 알고리즘의 성능 대비 네 가지 정적 알고리즘의 성능을 나타낸 것으로 TR 거리를 1~11m로 변화하면서 측정하였으며, 네 개의 그래프의 한 점은 3번의 AFECCC 트레이스를 이용하여 계산한 평균값이다. 모든 알고리즘의 성능을 공정하게 비교하기 위해 AFECCC를 사용하여 얻은 트레이스에서 오류 비트 수를 추출하여 정적 FEC를 사용한 경우 패킷 손실과 부가적 FEC 코드 전송으로 인한 부하를 계산하였다.

그림 8에서 보듯이 RS(84,80) 을 사용하는 정적 FEC는 TR 거리가 3m이하에서는 다른 세 가지 정적 FEC 보다 좋은 성능을 보이나 TR거리가 4m 이상 증가하였을 때 심각한 성능 저하를 보인다. 또한 AFECCC는 대부분의 구간에서 모든 정적 FEC 알고리즘보다 좋은 성능을 보이며, TR 거리가 11m인 경우에는 RS(92,80)과 같은 성능을 보인다.

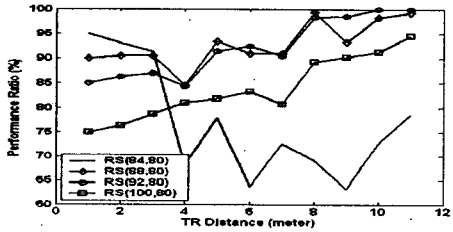


그림 8 실제 센서 네트워크에서 AFECCC 성능 대비 정적 FEC의 성능 비율

5. 결론 및 향후 연구 과제

본 논문은 채널의 BER 변동이 빈번한 저출력 센서네트워크에서 전송성능을 향상하기 위해 FEC 코드량을 동적으로 결정하는 방안으로 AFECCC 알고리즘을 제안하였다. AFECCC는 패킷 전송의 성공 여부에 따라 무선 채널의 BER 변화를 판단하고 FEC 코드량을 조절하는 알고리즘으로 다양한 분석적 오류 모델과 트레이스 기반 시뮬레이션을 통해 성능을 분석하고 실제 센서 네트워크에 구현하여 기존의 정적 알고리즘에 비해 성능 향상을 입증하였다. 향후에는 AFECCC 알고리즘의 다양한 성능 변수들을 네트워크 환경에 알맞게 자동으로 결정하는 방안을 연구할 것이다.

참고문헌

[1]J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building Efficient Wireless Sensor Networks with Low-Level Naming http://ww.cs.ucsd.edu/sosp01/papers/heidemann.pdf SOSP01, October 2001

[2]W. Peterson and E. Weldon, Jr. Error-Correcting Codes, 2nd Edition, The Massachusetts Institute of Technology

[3]Wei Ye, John Heidemann, and Deborah Estrin. An Energy-Efficient MAC protocol for Wireless Sensor Networks. In Proceedings of the IEEE Infocom, pp.1567-1576. New York, NY, USA, USC/Information Sciences Institute, IEEE. June, 2002

[4]http://network.dongguk.ac.kr/publication/AFECCC/AFECCC.html