싱크 주변 노드를 위한 센서 네트워크 혼잡 제어 기술

문성현° 이성민 차호정
연세대학교 컴퓨터과학과
{shmoon°, sulee, hjcha}@cs.yonsei.ac.kr

A Congestion Control Technique for the Bottleneck Nodes in Wireless Sensor Networks

SungHyun Moon° Sung-Min Lee HoJung Cha
Dept. of Computer Science, Yonsei University

요 약

본 논문은 기존의 혼잡 제어 기술이 전체 노드들을 위해 설계 되어 싱크 주변에서 생성되는 특별한 혼잡을 해결 할 수가 없기 때문에 싱크 주변 노드를 위한 혼잡 제어 기술을 제안한다. 싱크 주변에서 일어나는 'Bottle Neck' 문제가 일어났을 경우 데이터의 분석에 의한 우선순위결정과 타임 스케줄을 이용한 순차적인 데이터 전송으로 혼잡을 줄이고 그로 인해 데이터의 의미 있는 전송과 신뢰성을 동시에 만족시킨다. 또한 이 기술은 멀티 싱크 또는 모바일 싱크에도 적용 시킬 수 있고, 기존의 전체 노드를 위한 혼잡 제어 기술과 제시하는 기술을 혼용하여 사용할 수도 있다. 본 논문이 제안하는 기법은 혼잡 제어를 하지 않은 상태에서의 데이터 전달 방식에 비해 데드라인 만료 비율이 1.5%이상 감소하였고 프레임 손실율 또한 28%이상 감소하였다.

1. 서 론

최근 많은 센서 네트워크 응용들의 실용화 가능성이 입증됨에 따라 관련 기술 연구가 활발하게 이루어지고 있다. 센서 네트워크는 무선을 이용해서 통신을 하고, 노드의 수가 기존의 Ad-hoc보다 많고 밀집도가 높을 수 있는 특징을 지니고 있다. 또한 에너지 소스, 메모리, 프로세서 등의 능력이 한정 되어 있기 때문에 센서 네트워크에 특화된 혼잡 제어 기술이 필요로 한다. 특히 위치 추적과 같은 실시간 응용에서 혼잡제어 없이는 수행능력이 기대치에 미달하는 상황이 가능하다.

현재 센서 네트워크를 위한 혼잡 제어 기술들이 개발되어 존재한다. PSFQ[1]는 신뢰성있는 retasking/reprogramming 기법을 제안하였다. PSFQ[1]는 보내야할 패킷은 천천히 보내고, 패킷 손실이 일어났을 때의 hop-by-hop 회복은 빠르게 수행하는 방식으로 센서 네트워크의 신뢰성을 보장하도록 하였다. 그러나 이러한 기법은 혼잡의 원인을 제거하지 못하고 손실된 패킷의 회복 과정에서 혼잡을 더 증가시킨다. RMST[2]는 노드에서 싱크로의 신뢰성을 보장하기 위해 센서네트워크 전송계층이 지녀야 할 기능들을 제안하고 있다. 여러 방식의 nack와 arq 기능을 조합하여 실험을 하였고, 최적의 조합으로 selective-arq기능과 함께 nack 기반의 전송 계층을 제안하고 있다.

혼잡제어기능에 좀더 초점을 둔 ESRT[3]는 버퍼의 사용량을 모니터링하여 혼잡을 탐지하고, 소스 노드에게 제어신호를 브로드캐스팅하여 소스 노드의 reporting rate를 조절한다. [3]는 모든 노드의 reporting rate이 같다고 가정하기 때문에 혼잡의 원인이 되는 노드가 여러 개일 경우 성능이 보장되지 못하고, 브로드캐스팅으로 인한 에너지 소모가 크다는 문제점이 있다. CODA[4]는 이러한 [3]의 문제점들을 해결하기 위해, open-loop hop-by-hop backpressure, close-loop multi-surce regulation 기법을 제안하였다. 전자는 [3]와 비슷한 방식으로 소스노드의 전송율을 조절하는 기법이고, 후자는 reporting rate가 다른 여러 소스 노드들의 혼잡을 제거하기 위한 기법으로 싱크로부터 ack를 받지 못했을 경우 소스 노드가 스스로 전송율을 감소시키는 방법이다.

위에서 언급한 기법들은 싱크 주변에서 생기는 bottle neck문제를 고려하지 않고 있기 때문에 싱크 주변에서 일어나는 혼잡에 의한 데이터 손실을 해결 할 수 없다. 센서 네트워크에서 센싱된 데이터는 대부분 싱크로 전송되어 의미를 가지기 때문에, 싱크 주변에서의 Bottle neck을 해결하는 것은 매우 중요하다. Bottle neck은 혼잡문제의 특이한 형태로서 싱크 주변에 주로 일어날 수 있는 현상이다. 싱크에게 많은 노드들이 많은 양의 데이터를 보내려고 하거나 싱크 주변노드가 싱크에게 전송해야 할 많은 데이터를 받을 때를 뜻한다. 이 때 충돌, 버퍼 오버플로우 등으로 데이터 전송이 불가능해지고, 전송된다 하더라도 fairness가 무시됨으로서 데이터의 의미가 없어 질 수도 있다. 또한 기존의 ESRT와 같은 기법은 데이터를 발생시키는 노드에게 컨트롤 메시지를 보내는 방법을 사용하고 있지만 움직이는 소스 노드나 노드 자체가 위치 정보를 발생 시키는 것이 아니라 자연적인 이벤트일 경우 쓸 수가 없게 된다. 따라서 real-time 응용에 적합하지 않다. 제안하는 기법은 싱크 주변 노드에 특화되어 동작하는 기술로서 기존의 혼잡제어가 불가능한 상황에서도 부분적으로 혼잡제어를 제공하고, 기존의 기법과 같이 사용함으로써 성능 향상을 기대할 수 있다. 이 기법을 통해서 의미 있는 데이터 전송과 데이터 전송 시 fairness와 충돌 회피를 최대한 만족 시키는 방법으로 bottle neck이라는 특수한 혼잡상황을 제어했다. 제안하는 혼잡제어 기법은 실제 센서 노드에서 구현되어져 실험했고, 그 결과 혼잡도를 낮추며 데이터를 전송하는 것을 볼 수 있었다.

2. Bottle Neck 문제를 위한 혼잡제어 기술

우선 혼잡도의 측정은 싱크 주변 노드의 버퍼 사용량에 따라 결정된다. 혼잡도가 일정 수준을 넘어설 경우, 싱크는 주변 노드들의 혼잡 정도와 데이터들의 우선순위를 파악하여 적절한 가중치를 줌으로써 주변노드들이 데이터를 보낼 수 있는 슬롯 시간을 결정해 준다. 그와 동시에 주변 노드들은 받는 데이터의 urgency를 구분하여 일정 urgency 이상이 되는 데이터들은 먼저 전송되도록 하는 선점 기법을 제공함으로서 데이터의 의미 있는 전송을 가능케 한다.

따라서 데이터 선점 기법과 타임 퀀텀 스케줄 기법을 통해서

본 연구는 한국과학재단에서 지원하는 국가지정연구실사업으로 수행하였음 (과제번호 : 2005-01352)

fairness, 충돌 회피, 의미 있는 데이터 전송을 가능하게 하고 이로 인해 싱크 주변에서의 bottle neck 문제를 완화하여 시스템의 성능이 향상된다.

2.1 혼잡 탐지 및 데이터 선점 기법

혼잡 제어 이전에 혼잡을 탐지하는 것이 중요하다. 가장 간단한 방법으로서 싱크 주변 노드들의 버퍼에 저장되어 있는 데이터의 양이 일정 수준을 넘었을 경우 혼잡상황이 발생한 것으로 판단할 수 있다. 본 논문에서는 버퍼용량을 기준으로 데이터가 채워진 상대적인 비율을 *threshold*로 두고 데이터양이 이를 초과하였을 경우 혼잡상황이라고 결정하였다.

센서 네트워크에서 혼잡상황이 아닌 경우 싱크 주변 노드들은 데이터를 받는 즉시 바로 싱크에게 보낼 수 있다. 하지만 주변 노드들의 데이터 처리 속도보다 데이터 유입 속도가 더 클 경우, bottle neck 문제로 인해 버퍼에 저장되는 데이터의 양이 점차 증가하면서 혼잡상황을 발생시킨다. 이 때 데이터는 저장된 순서대로 싱크에게 전송되므로 데드라인을 맞추지 못하는 데이터가 발생한다. real-time 응용 관점에서 이러한 데이터의 손실은 일어나서는 안된다. 따라서 혼잡이 발생했을 경우 데이터의 데드라인에 따른 우선순위를 두고 이를 기준으로 우선순위가 높은 데이터를 먼저 전송할 수 있도록 해야 한다. 이것은 버퍼 내에 존재하는 데이터들을 우선순위에 따라 정렬하고 정렬된 순서대로 싱크에게 전송함으로서 해결할 수 있다. 그러나 센서네트워크에서 데이터를 받을 때마다 모든 데이터들을 새로 정렬한다는 것은 데이터의 개수가 크지 않다 하더라도 오버헤드가 매우 큰 작업이다. 따라서 모든 데이터를 소팅하지 않고 urgency의 기준을 두어 그 이상의 우선순위를 갖는 데이터들만 정렬하는 기법을 제안한다.

먼저 싱크 주변 노드들이 혼잡을 탐지한 시점에 자신의 버퍼에 저장된 데이터들을 우선순위를 기준으로 데드라인이 가장 급한 urgent와 normal priority로 나누고 urgent한 데이터만을 최우선순위에 두고 정렬한다. Urgent한 데이터만 선점하는 이유는 normal priority 데이터를 정렬하는 오버헤드로 인한 지연을 방지하기 위해서이다. 이 후부터는 새로운 데이터가 유입될 때마다 urgent한지를 판단하여 urgent하다면 정렬 영역에 순서에 맞게 삽입을 하고, 그렇지 않다면 맨 마지막 순서에 삽입을 한다. 물론 혼잡이 해소될 경우 원래의 방식대로 데이터를 받은 순서대로 버퍼에 저장하여 전송한다.

정렬 기법은 삽입정렬을 선택했다. 삽입정렬은 정렬해야 할 데이터의 양이 대략 30개 미만으로 그다지 많지 않을 때 유용하다. 따라서 센서네트워크에서 오버헤드 없이 비교적 빠른 속도로 정렬할 수 있는 방법이다.

2.2 타임 퀀텀 스케줄 기법

본 논문이 제안하는 혼잡제어기법은 타임퀀텀을 기반으로 한다. 노드들이 초기화되는 단계에서 싱크는 이미 주변 노드들에게 모두 동등한 타임퀀텀을 할당해준다. 만약 혼잡상황이 발생했을 경우 위에서 설명한 데이터 선점 기법이 작동함과 동시에 타임퀀텀을 새로 할당하게 된다. 혼잡상황이 발생한 후부터 싱크 주변 노드들은 데이터를 싱크에게 보낼 때 자신의 버퍼에 저장되어 있는 데이터들의 우선순위들에 대한 통계값을 함께 보낸다. 따라서 싱크는 자동으로 주변노드의 혼잡도를 모니터링 할 수 있고 이 통계값을 토대로 스케줄링을 할 수 있다.

혼잡 상황이 아니더라도 collision과 fairness 문제는 언제나 일어날 수 있기 때문에 주변 노드들은 자신의 할당된 타임 퀀텀 안에서만 데이터를 보내게 된다. 물론 이러한 경우 타임 퀀텀은 모든 주변 노드에 똑같은 양으로 주어지지만 혼잡상황이 발생되었을 경우는 혼잡도가 심한 노드에게 타임 퀀텀을 더 할당하고 반대로 혼잡도가 크지 않은 노드에게는 타임 퀀텀을 상대적으로 더 적게 할당하는 방법을 사용한다.

$$time\_slot = \frac{N \times buf\_size(bytes) \times threshold \times a}{bitrate(byte/\sec) \times transmission\_time(\sec)} \quad (1)$$

우선 모든 싱크 주변 노드들이 한 번씩 데이터를 보낼 수 있는 전체 타임 슬롯의 크기를 결정해야한다. 위 식은 타임 슬롯 한 주기를 결정하는 식으로써 n은 싱크 주변 노드의 수이고, *buf_size*는 싱크 주변 노드 하나의 버퍼용량이고, *threshold*는 버퍼에 저장된 데이터가 버퍼 용량을 얼마나 채웠는지에 대한 상대적 비율이다. *threshold*는 이미 혼잡탐지의 기준이 된다고 앞서 언급했다. 이 세가지 인자들의 곱은 싱크가 혼잡을 해소하기 위해 앞으로 처리해야 할 데이터의 양을 표현한 것이다. 즉 싱크 주변 노드의 수가 많고 버퍼 크기가 크고 혼잡을 판단하는 기준이 더 높을수록, 싱크가 혼잡을 해소하기 위해 처리해야 할 데이터의 양은 많아진다.

a는 싱크가 혼잡을 해소하기 위해 처리해야 할 데이터 중에서 타임퀀텀 한 주기 동안 처리하고자 하는 데이터 양의 상대적인 비율을 나타낸다. 즉 전체 타임 슬롯의 크기를 결정짓는 가장 중요한 변수이다. *threshold*와 마찬가지로 a값 역시 실험을 통해 가장 적합한 값을 찾았다. *transmission_time*과 *bitrate*은 한 개의 데이터 프레임을 보내기 위해 필요한 최소시간과 하드웨어가 지원하는 전송속도이다. 이 두 상수는 타임 퀀텀을 *transmission_time* 단위의 개수로 표현하기 위해 사용되었다. 본 논문에서는 실험에 사용하는 하드웨어 telosB를 기준으로 각각 30msec와 256(kbyte/sec)으로 정했다.

여기서 a는 본 논문에서 제안하는 혼잡제어 기법이 real-time 응용에 적용될 수 있는지를 결정해준다. 예를 들어 telosB에서 전송속도가 256(kbps)이고, n=10, buf_size=1320(byte), threshold=0.6라 하자. 싱크 노드가 혼잡을 해소하기 위해 처리해야 할 데이터의 최대양은 7920(byte)가 되고, a가 0.32일 때 슬롯 한 주기가 1초가 된다. 즉 싱크 주변 노드들이 싱크에게 데이터를 전송할 기회를 갖는데 1초의 시간이 걸린다. 따라서 a는 전체 타임 슬롯의 크기를 결정하고 또한 혼잡을 해소하기 위해 한 주기 동안 처리해야 할 데이터 양의 비율을 표현해준다.

전체 타임 슬롯의 크기가 결정되면 각 주변 노드의 버퍼에 저장된 데이터의 양과 우선순위에 따른 가중치를 이용하여 타임 퀀텀을 할당한다. 이 과정에서 weight을 수식 2와같이 계산한다.

$$weight_i = \sum_{j=1}^{N} (priority_j \times \eta_j) \quad (2)$$

$$N = number\ of\ buffer\ frame, \quad \eta_j = \begin{cases} 1.0, & priority_j < 45 \\ 1.5, & priority_j \geq 45 \end{cases}$$

위 수식은 priority가 70 이상일 경우 urgent한 데이터로 판단하고 1.5배의 가중치를 주는 방법을 사용했다. 결국 이렇게 구해진 weight은 각 데이터의 priority와 버퍼에 저장된 데이터의 양을 고려한 값이 된다.

$$time\_quantum_i = \frac{weight_i}{\sum_{j=1}^{B} weight_j} \times time\_slot \quad (3)$$

Weight이 결정되면 최종적으로 i번째 주변노드의 타임 퀀텀을 수식 3을 이용해서 계산하게 된다. 위 식은 주변 노드들의 전체 weight 합에 대한 i번째 노드의 weight의 상대적인 비율로써 i번째 노드에 할당될 타임 퀀텀을 계산하게 된다.

주변 노드 각각의 타임 퀀텀이 결정되면 전체 타임 퀀텀에 컨트롤 슬롯을 추가한다. 컨트롤 슬롯은 싱크가 주변 노드들에게 컨트롤 메시지를 보내는 데 사용된다. 이렇게 상황에 따라 동적으로 타임 퀀텀을 변화시킴으로서 fairness를 보장할 수 있고, 타임 슬롯을

사용하기 때문에 collision avoidance도 가능하다. 이와 같은 타임 퀀텀 스케줄 기법을 그림 1에서 예를 들고 있다.

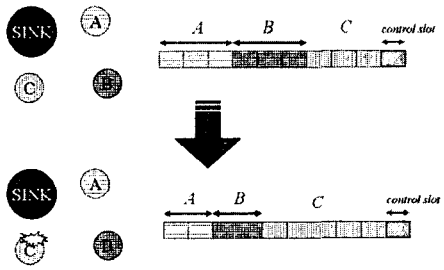
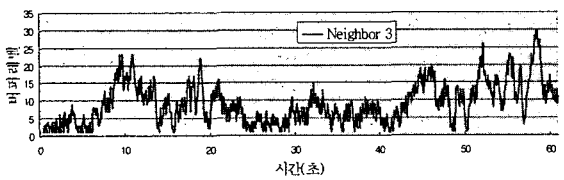


그림 1 타임 퀀텀 스케줄 기법

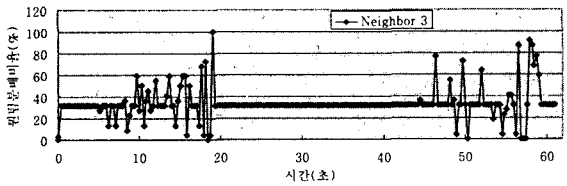
이러한 타임 퀀텀 스케줄 기법을 싱크 주변 노드에서만 쓰는 것이 아니라 전체 노드에서 사용한다면 혼잡상황을 더 쉽게 해소할 수 있을 것이다. 하지만 많은 수의 노드들에게 타임 퀀텀을 할당하는 작업의 오버헤드가 크고, 이 기법을 위해 선행되는 시간 동기화 문제 또한 센서노드의 여러 제약 조건 때문에 쉽지 않다. 따라서 본 논문이 제안하는 기법과 함께 전체 노드를 대상으로 기존의 혼잡제어 기술을 사용하도록 한다.

3. 성능 분석

제시한 기법의 성능 평가를 위해 TOSSIM[5] 시뮬레이터를 사용하였다. 타임 슬롯 한 구간은 30ms, threshold는 0.6으로 설정하였고, 3개의 싱크 주변 노드들이 1개의 싱크 노드에게 데이터를 전송하는 환경에서 진행되었다. 각 프레임은 임의의 데드라인을 갖고 30ms마다 0.3의 확률로 싱크 주변 노드들에게 전달된다.



(a) neighbors 3의 버퍼레벨 변화



(b) 3의 타임퀀텀 분배 비율 변화

그림 2 버퍼레벨에 따른 타임퀀텀 분배비율의 변화

그림 2 (b)는 제안된 혼잡 제어 기법이 싱크 주변 노드들의 버퍼에 따라 타임퀀텀을 다르게 분배하는 과정을 나타낸 것이다. 그림 2(a)에서 보듯이 Neighbor 3의 버퍼레벨이 증가하여 혼잡이 발생한 경우 weight에 비례하여 타임퀀텀이 차등하게 분배되고, 혼잡이 발생하지 않은 20~40초 구간에서는 타임퀀텀이 동등하게 분배되는 모습을 볼 수 있다.

그림 3은 제안된 혼잡 제어 기법을 사용했을 때와 사용하지 않을 때, 60초 동안 싱크 노드가 받은 프레임의 데드라인 만료 비율과 프레임 손실율을 나타내고 있다. 그림에서 알 수 있듯이, 데드라인 만료 비율은 평균 1% 정도 감소되었고, 프레임 손실율 역시 본 논문이 제안하는 기법에 의해서 약 28% 정도로 크게 감소하였다.

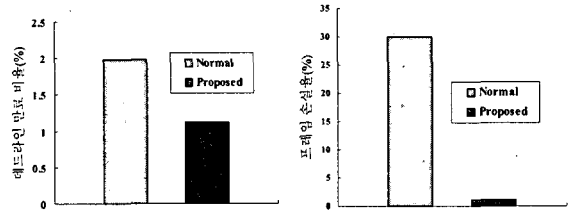


그림 3 데드라인 만료된 프레임 비율과 프레임 손실율

본 논문이 제안하는 혼잡 제어 기법을 사용할 경우, 타임 퀀텀 스케줄 기법에 따라 싱크 주변 노드들의 프레임 전송 시점이 중복되지 않기 때문에 충돌로 인한 프레임 손실은 거의 발생하지 않았고, urgent한 데이터를 먼저 보내는 데이터 선점 기법을 사용함으로서 데드라인이 경과되어 도착하는 프레임의 비율도 평균 1%에서 최대 2.5%까지 줄일 수 있었다.

4. 결론

본 논문이 제시하는 혼잡 제어 기법은 싱크 주변에서 일어나는 특수한 bottle neck 혼잡 문제를 해결하기 위해 데이터 선점 기법과 타임 퀀텀 스케줄 기법을 사용했다. 이 기법들은 fairness와 충돌 회피를 가능하게 하고, 데이터의 의미를 파악해서 데이터 손실을 최대한 줄였다. 이러한 싱크 주변 혼잡 제어 기법의 개발로, 기존하는 전체 토폴로지를 대상으로 하는 혼잡제어 기술을 함께 사용함으로서 혼잡에 의한 데이터 손실을 더욱 줄일 수 있음이 기대된다.

향후 연구로서 실제 센서 노드에 적용하여 실험하는 과정이 필요하고, 좀더 다양한 메트릭에 대해 성능평가를 할 예정이다. 또한 타임 퀀텀 스케줄 기법을 싱크 주변 뿐만 아니라 전체 노드에서 사용할 수 있도록 기법을 확장하는 연구도 이뤄질 것이다.

참고문헌

[1]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 2002.

[2]Fred Stann, John Heidemann. "RMST: Reliable Data Transport in Sensor Networks." In Proc. the 1st IEEE international Workshop on Sensor Net Protocols and Applications (SNPA) 2003.

[3]Y. Sankarasubramaniam, O. Akan, and I. Akyildiz. "Event-to-sink reliable transport in wireless sensor networks." In Proc. of the 4th ACM Symposium on Mobile Ad Hoc Networking & Computing 2003.

[4]C-Y. Wan, S. B. Eisenman, and A. T. Campbell. "CODA: Congestion detection and avoidance in sensor networks." In Proc. ACM SenSys 2003, 266 - 279.

[5]Levis, P., Lee, N., Welch, M, Culler, D.: TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications." In Proc. ACM SenSys 2003, 126-137

[6]CC2420 Data sheet. http://www.chipcon.com/files/CC2420_Data_Sheet_1_2.pdf.