센서네트워크에서 Link Quality를 고려한 효율적인 코드전파 기법

조성규° 차호정
연세대학교 컴퓨터과학과
{skcho°, hjcha}@cs.yonsei.ac.kr

Link Quality based Efficient Code Propagation in Wireless Sensor Networks

SungKywoo Cho° Hojung Cha
Dept. of Computer Science, Yonsei University

요 약

본 논문에서는 센서네트워크에서 Link Quality를 이용하여 코드와 같은 대용량의 데이터를 적은 에너지로 빠르게 전파하는 LQNP 기법을 제안한다. 센서 네트워크에서 코드의 전파 방법에 대한 기존 연구들은 Link Quality를 고려하지 않고 코드를 전파하는 정책을 쓰고 있다. 패킷 손실이 많은 센서네트워크에서 이 방법은 적합하지 않기 때문에 본 논문에서는 패킷 손실이 많은 상황에서도 효율적인 코드 전파를 수행할 수 있도록 Link Quality를 이용하는 방법을 살펴본다.

1. 서 론

센서네트워크란 저성능의 저가 센서 노드 여러 개가 무선 통신을 통해 서로 협력하여 특정 작업을 수행하는 새로운 컴퓨팅 분야이다. 센서네트워크는 주로 환경 모니터링이나 위치 추적과 같은 장기간의 정보를 필요로 하는 곳에 사용되며 이 때문에 여러 개의 센서 노드가 배치된 후에 환경의 변화나 원하는 정보의 변화 때문에 각 센서 노드에서 실행되는 프로그램을 수정하거나 바꿔야 할 경우가 있다. 이미 배치된 센서 노드의 프로그램을 일일이 하나씩 바꾸는 것은 어려우므로 이 때 *network programming* (네트워크를 통해 코드를 전파하여 노드를 프로그래밍 하는 것) 이 필요하다. *network programming*은 크게 Maté[1] 와 같이 virtual machine에 실행되는 소스코드를 전파하여 프로그래밍 하는 방법과 Deluge[2] 와 같이 바이너리 이미지를 전파하여 프로그래밍 하는 방법으로 나눌 수 있으며 본 논문은 바이너리 이미지와 같이 큰 크기의 데이터를 전파하는 방법을 다룬다.

센서 네트워크에서 코드전파 방법에 관한 대표적인 연구로는 XNP[3], MOAP[4], Deluge[2], GARUDA[5] 등이 있다. XNP[3]는 TinyOS에 포함되어 있으며 base station 이 자신과 통신할 수 있는 범위 안의 노드들에게 코드 캡슐을 주는 방법을 사용한다. base station 은 모든 이미지를 전파한 후에 각 노드로부터 수신 실패한 캡슐에 대한 정보를 받고 재전송을 해준다.(NACK 방식) 이 방법은 멀티 홉을 통한 코드 전파를 할 수가 없다는 단점이 있다. MOAP[4] 의 코드 전파는 새로운 코드를 가진 노드가 publish를 하면 새로운 코드를 원하는 노드가 subscribe를 하는 publish-subscribe 방식을 사용한다. 이를 이용하여 멀티 홉에서의 코드 전파가 가능하다. Deluge[2] 는 MOAP[4] 와 유사한 방법을 사용하지만 MOAP[4] 와 달리 이미지를 분할하여 전송하며 이로 인해 전체 코드의 일부만 가진 노드가 소스가 될 수 있다. (spatial multiplexing) 이를 통해 MOAP[4] 보다 코드 전파를 완료하는데 까지 걸리는 시간이 감소하게 된다. Deluge[2] 신뢰할 수 있는 코드전파를 위해서 주기적으로 계속 자신의 코드 상태를 방송하는 (adv) epidemic approach 를 사용하며 센서 노드들의 밀집 정도에 관계없이 잘 작동할 수 있도록 suppression 방법을 사용한다. 모든 노드는 메시지를 매 주기마다 보낼 수 있으며 만약 특정 주기에 메시지를 보내기 전에 자신이 보내려는 메시지와 비슷한 내용의 메시지를 엿들으면 자신이 보내려던 서 통신량을 줄일 수 있게 된다. GARUDA[5] 는 Deluge[2] 와는 달리 WFP 라는 강한 신호를 이용하여 신뢰할 수 있는 코드전파를 가능하게 한다. 또한 Hidden Terminal Problem 을 해결하기 위해서 코드 전파 전에 flooding 을 통해 자신에게 코드를 줄 소스를 정하게 된다.

지금까지의 코드전파에 관한 연구들은 소스를 선택하는 방법으로 link quality를 사용하지 않는다. 하지만 좋은 소스를 선택해야만 NACK 방식을 사용하여 큰 데이터를 전송할 경우 재전송을 최소한으로 줄일 수 있다. 본 논문에서는 link quality를 이용하여 소스를 정하는 방법을 통해 보다 효율적인 코드전파를 이루고자 한다.

2. Motivation

Deluge[2] 는 3 방향 프로토콜(adv-req-data)을 사용하며 노드들은 주기적으로 자신이 현재 갖고 있는 코드 정보를 나타내는 Advertisement 메시지를 방송한다. 이러한 ADV 메시지에는 현재 자신이 가진 코드의 정보를 담게 된다. Advertisement 메시지를 받은 노드는 만약 원하는 정보가 있다면 Request 메시지를 Advertisement 메시지의 소스 노드에게 보낸다. Request를 받은 노드는 Request 메시지에서 요구하는 데이터 수십 개를 방송하게 된다. 그림 1이 Deluge[2] 의 3 방향 통신 프로토콜을 잘 보여준다. 그림에서 볼 수 있듯이 A는 B와 C로부터 DATA를 받을 수 있지만 C가 B보다 먼저 advertisement 메시지를 보냈으므로 C에게 DATA를 요청한다. 그런데 C는 B보다 A로부터 멀리 있으므로 A는 C의 DATA 메시지를 B의 DATA 메시지보다는 잘 받지 못한다. 즉 C-A의 link quality 가 B-A의 link quality 보다 나쁠 것이다. 이러한 상황에서 C가 수십 개의 DATA 메시지를 방송하면 A는 이 중 많은 부분을 수신 실패할 것이고 B 혹은 C의 advertisement 메시지를 받을 때까지 기다렸다가 재전송을 요구할 것이다. 이 때 만약 A가 C의 advertisement 메시지를 또 먼저 받게 되면 이러한 작업을 반복하게 된다. 전체적으로 보면 코드 전파가 끝나기까지 시간이 오래 걸리게 되고 재전송 횟수가 증가하여 에너지도 많이 소비하게 된다. Deluge[2]의 이러한 문제점은 소스 노드의 선택 시 link quality 를 고려하지 않기 때문이다.

본 논문은 과학기술부 국가지정연구실(NRL) 사업의 결과임.

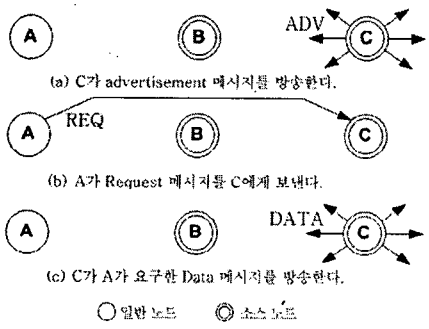

그림 1. Deluge[2] 3 방향 프로토콜

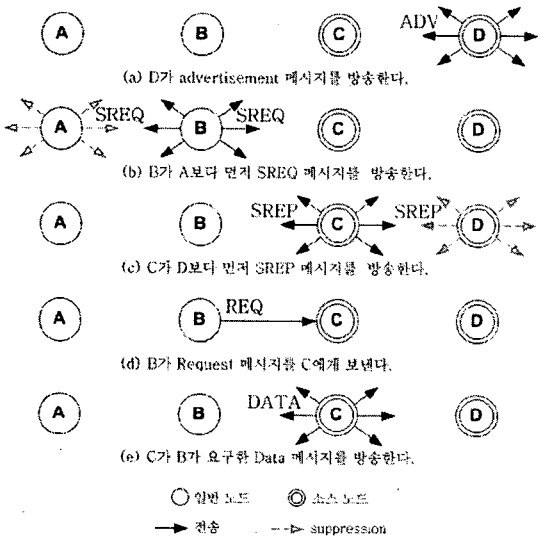

그림 2. LQNP 5 방향 프로토콜

3. Link Quality를 고려한 코드전파 기법

일반적으로 센서네트워크에서는 라우팅 알고리즘을 위해서 이웃 노드들과의 link quality를 유지하거나 하드웨어적으로 LQI (Link Quality Indicator)를 지원하여 노드간의 link quality를 쉽게 측정할 수 있다. 따라서 본 논문에서는 link quality 측정에 의한 overhead가 없거나 매우 적을 것이라고 가정한다.

앞에서 살펴보았듯이 3 방향 프로토콜(adv-req-data)로는 link quality 를 고려할 수 없다. 이러한 이유로 LQNP는 5 방향 프로토콜(adv-sreq-srep-req-data)을 사용한다. adv, req, data는 기존 3방향 프로토콜과 같은 역할을 하게 되고 link quality가 좋은 소스를 선택하기 위해서 req 앞에 sreq, srep를 추가하였다. sreq는 source request를, srep는 source replay를 의미한다.

그림 2가 LQNP의 5 방향 프로토콜을 나타낸다. 그림에서 A, B, C, D 노드는 모두 서로 통신가능하다고 가정한다. D가 C보다 먼저 advertisement를 하게 되면 Deluge의 3 방향 프로토콜에서는 만약 B가 D를 소스로 삼고 REQ 메시지를 보내지만 LQNP에서는 B가 link quality가 좋은 소스를 찾기 위해서 SREQ 메시지를 보낸다. SREQ에는 자신의 코드정보와 ID가 담겨있다. SREQ 메시지를 받은 노드들은 SREQ 에 포함된 코드

```
/* node i 는 다음과 같은 동작을 한다. */
Receiving ADV_Msg(CodeInfo, srcAddr):
  if(state=S_ADV and newCode(CodeInfo))
    state=S_SREQ; recvADVSrcAddr=srcAddr;
    //srcAddr 와의 link quality 에 반비례하도록 timer 설정
    SREQSendTimer_start(LQDelay(srcAddr));

SREQSendTimer_expired():
  if(state=S_SREQ)
    SREQSend(CodeInfo, i, recvADVSrcAddr);
    SREPWaitTimer_start();

Receiving SREQ_Msg(CodeInfo, srcAddr, ADVSrcAddr):
  if(state=S_SREQ and recvADVSrcAddr=ADVSrcAddr)
    state=S_ADV; SREQSendTimer_stop(); //suppression
    SREPWaitTimer_stop();
  if(state=S_ADV and oldCode(CodeInfo))
    state=S_SREP; recvSREQSrcAddr=srcAddr;
    //srcAddr 와의 link quality 에 반비례하도록 timer 설정
    SREPSendTimer_start(LQDelay(srcAddr));

SREPSendTimer_expired():
  if(state=S_SREP)
    SREPSend(i, recvSREQSrcAddr);

Receiving SREP_Msg(srcAddr, Destination):
  //SREP를 받았으므로 srcAddr 을 소스로 정하고 REQ 전송
  if(state=S_SREQ and Destination=i)
    SREPWaitTimer_stop(); state=S_REQ; REQSend(srcAddr);
  if(state=S_SREP and Destination=recvSREQSrcAddr)
    SREPSendTimer_stop(); state=S_ADV; //suppression

SREPWaitTimer_expired():
  if(state=S_SREQ)
    state=S_ADV;
```

그림 3. LQNP 알고리즘

정보보다 최신의 코드 내용을 자신이 갖고 있는지 확인해보고 만약 갖고 있다면 자신이 소스가 될 수 있으므로 SREP 를 보낼 수 있다. 그런데 link quality 가 좋은 노드가 먼저 SREP를 보낼 수 있도록 하기 위해서 SREQ의 송신노드와 수신노드의 link quality 에 반비례하도록 SREP 전송 지연시간을 설정한다. link quality 가 좋을수록 SREP 메시지를 빨리 전송하게 되고 이를 엿들은 다른 노드들은 지연 시간이 끝났어도 SREP 메시지를 전송하지 않는다. 그림 2를 보면 D 가 먼저 ADV 메시지를 보내면 B는 이를 받고 원하는 코드를 받기 위해 SREQ 메시지를 보낸다. 이를 받은 C, D는 자신들이 B의 소스가 될 수 있다고 판단하게 되고 SREP를 전송하기 위해서 지연시간을 B와의 link quality 에 반비례하도록 설정한다. B-C의 link quality가 B-D의 link quality 보다 좋기 때문에 C가 먼저 SREP 메시지를 전송하고 이를 엿들은 D는 SREP 전송을 포기한다. 이와 같이 LQNP 를 사용하면 B는 자신의 소스로 link quality 가 좋은 C를 선택하게 된다. 이제 (b)에서 A의 SREQ가 suppression 되는 이유를 설명하겠다. 만약 (b)에서 A가 B보다 먼저 SREQ를 전송하게 되면 SREP 메시지를 link quality 가 좋은 C가 먼저 보낸다고 하더라도 거리가 멀어서 코드 전파에 문제가 생긴다. 이를 해결하기 위해서 SREQ도 SREP 전송과 같은 방법을 사용한다. D로부터 ADV 메시지를 받으면 D가 자신의 소스가 될 수 있다고 판단하는 노드들 A, B는 자신

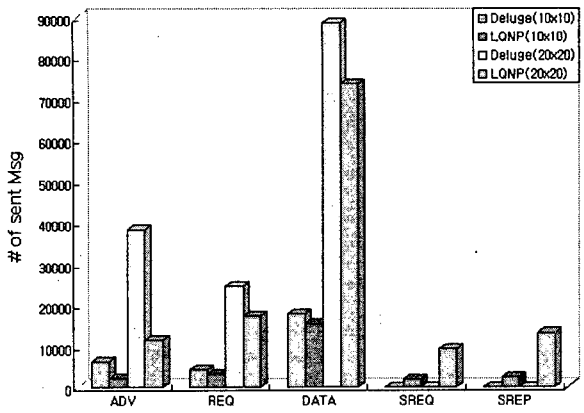

그림 4. 메시지 전송 개수 (10x10, 20x20 10feet 간격의 그리드 배치에서 5개의 page 전파 완료 시점의 메시지 종류 별 전송 개수)

과 D의 link quality에 반비례하도록 SREQ 메시지의 전송지연 시간을 설정한다. A-D보다 B-D가 link quality 가 더 좋으므로 B가 먼저 SREQ 메시지를 보내고 이를 엿들은 A는 SREQ 메시지 전송을 포기하게 된다. 구체적인 알고리즘은 그림 3과 같다.

앞에서 살펴보았듯이 SREQ, SREP 메시지를 보내기 전에 전송 지연시간이 필요하며 이 값은 link quality에 반비례해야 한다. 그런데 전송 지연 시간이 전체 코드 전파를 느리게 할 수 있다. 전송 지연시간에 따른 불이익을 해결하는 방법은 매번 data를 얻기 위해서 5way를 모두 수행하는 것이 아니라 그림 2의 (d)부터 수행하는 것이다. 그림 2의 (a), (b), (c) 단계를 이미 수행하였다면 자신의 소스를 알고 있고 그 소스가 link quality 좋다는 것을 알 수 있다. 그러므로 packet loss 에 의해서 DATA 메시지를 일부 수신 실패하더라도 그림 2의 (d) 단계부터 시작해도 관계없다.

4. 성능평가

Deluge와 LQNP의 성능을 비교하기 위하여 센서네트워크에서 많이 쓰이는 시뮬레이터인 TOSSIM을 사용하였다. 실험은 10x10, 10feet 간격의 그리드 배치와 20x20, 10feet 간격의 그리드 배치 두 가지 상황에서 진행하였다. 5.5KB의 코드(5개의 페이지)를 그리드 배치의 모서리(0, 0) 노드에서 전파하기 시작하여 모든 노드가 코드를 다 받을 때까지 실험하였다. LQNP는 앞에서 설명하였듯이 좋은 소스를 선택할 수 있으므로 소스 노드를 선택하는 원인이 되는 ADV 메시지 간격을 Deluge 보다 4배 길게 하였다. 그리고 위에서 살펴보았듯이 LQNP에서 한 번 정한 소스는 좋은 소스이므로 이를 계속 사용하기 위하여 REQ의 재시도 횟수를 8번으로 정하였다. 이것이 의미하는 것은 다음과 같다. 만약 소스를 결정한 후에 REQ 메시지를 보내면 DATA 메시지를 수신해야 하는데 만약 DATA 메시지 수신을 못할 경우 Deluge는 소스를 재설정하기 위해서 ADV 메시지를 기다린다. 하지만 LQNP 는 DATA 메시지를 못 받더라도 소스가 좋다는 것을 어느 정도 확신할 수 있으므로 계속 request를 시도할 수 있다. Deluge는 자신의 소스노드가 좋은 소스인지 알 수 없기 때문에 이렇게 값을 변경하는 것은 바람직하지 않다. 위의 두 실험에서 코드를 다 받은 시점은 10x10의 경우 Deluge와 LQNP 모두 900초 정도였고 20x20의 경우 Deluge와 LQNP 모두 1800초 정도였다. 이를 통하여 LQNP 가 Deluge 보다 코드 전파가 느리지 않음을 알 수 있었다.

위의 그림 4는 10x10, 20x20에서 코드를 다 받은 시점에서

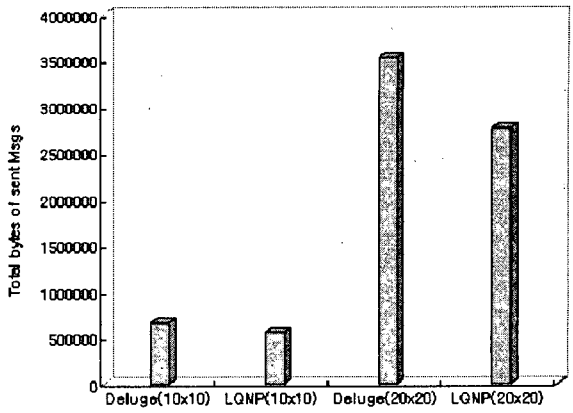

그림 5. 전송된 총 byte 크기

Deluge와 LQNP가 전송한 메시지 개수이다. 10x10, 20x20 모든 경우에서 LQNP가 ADV, REQ, DATA 메시지를 모두 조금 보낸다. 특히 20x20의 상황에서 DATA 메시지의 개수를 비교해 보면 LQNP가 Deluge 보다 상당히 작은 것을 볼 수 있다. 그러나 LQNP는 위의 그림 4와 같이 SREQ, SREP 메시지를 부가적으로 보내야 한다. 이것을 Deluge 와 비교하기 위하여 모든 메시지를 byte로 환산하여 총합을 구한 것이 그림 5이다. 이를 통해 LQNP가 SREQ, SREP 의 부가적인 메시지를 사용하지만 Deluge 보다는 전송되는 총 byte 수가 적다는 것을 알 수 있다. Deluge에 비해 LQNP의 성능이 10x10 보다 20x20에서 더 좋은 것을 볼 수 있는데 그 이유는 20x20에서 Deluge가 나쁜 소스를 선택할 확률이 높다는 것을 뜻한다.

5. 결론

본 논문에서는 Deluge 의 코드 전파 시 소스노드 선택에 있어서의 문제점을 지적하고 이를 해결하기 위해 link quality를 사용하여 좋은 소스를 선택하는 LQNP라는 프로토콜과 성능을 알아보았다. LQNP 는 시간 지연과 부가적인 메시지 교환에 따르는 오버헤드가 존재하지만 전체적인 성능 평가 결과 LQNP를 이용하는 편이 Deluge를 이용하는 것보다 더 나은 성을 보여준다는 것을 알 수 있다. 향후 연구과제로는 LQNP를 기반으로 코드 전파 시 발생하는 Hidden Terminal Problem 을 해결하는 것이다.

참고문헌

[1] Philip Levis, David Culler, "Maté : A Virtual Machine for Sensor Networks", *In the Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 85~95, publication 2002.

[2] Jonathan W. Hui, David Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale", *In the Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04), pages 81~94, publication 2004.*

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

[4] T. Stathopoulos, J. Heidemann, and D. Estrin, "A remote code update mechanism for wireless sensor networks", Technical report, UCLA, Los Angeles, CA, USA, 2003.

[5] Seung-Jong Park, Ramanuja Vedantham, Raghupathy Sivakumar and Ian F. Akyiliz, "A Scalable Approach for Reliable Downstream Data Delivery in WSN", *In Proceedings of the 5th ACM International Symposium on Mobile Ad hoc Networking and Computing (MOBIHOC)*, May 2004.