무선 센서 네트워크에서의 고신뢰성 지원 모바일 콜렉터 스케줄링 기법

조정민[0], 한주선, 하 란
홍익대학교 컴퓨터공학과
지인정보기술
jmcho35@gmail.com, jshan@zeen.snu.ac.kr, rhanha@cs.hongik.ac.kr

A Mobile Collector Scheduling Scheme for Supporting High Reliability in Wireless Sensor Networks

Jeong Min Cho[0], Joo Sun Hahn, Rhan Ha
Computer Engineering, Hongik University
Zeen Information Technologies, Inc.

무선 센서 네트워크에서, 센서 노드의 한정된 메모리 용량으로 인해 발생하는 데이터 오버플로는 데이터의 손실을 야기한다. 센서가 감지하는 데이터에 신뢰성을 보장하기 위해서는 데이터 손실을 막기 위한 효율적인 모바일 데이터 콜렉터의 스케줄링이 요구된다. 기존 연구 [1]과 [2]에서는 모바일 데이터 콜렉터의 이동거리를 최소화하여 데이터 전송 지연 시간을 최소화하고자 하였으나 센서 노드가 버퍼 오버플로가 발생하지 않을 만큼 충분한 메모리를 가지고 있다고 가정하고 있다. 현실적으로 센서 노드는 제한된 메모리 용량을 가지고 있으므로 버퍼 오버플로가 발생되기 전에 콜렉터에게 데이터를 전송하여 데이터 손실이 없도록 해야 한다. 기존 연구 [3-5]에서는 이종 센서 네트워크 환경에서 각 센서 노드의 데이터 생성 주기를 고려하여 버퍼 메모리 필요량과 모바일 데이터 콜렉터의 이동경로 및 이동속도를 동적으로 결정하는 방법을 제시하였다. 그러나 센서 노드의 메모리 용량 또는 모바일 데이터 콜렉터의 이동속도 등은 현실적으로 무한정 증가할 수 없으며, 이러한 현실적인 제약에 따른 데이터 손실은 여전히 발생한다. 또한, 네트워크 규모가 커질수록 모바일 데이터 콜렉터의 이동속도 제약에 따른 불가피한 데이터 손실 문제는 점점 심화된다. 따라서 모바일 데이터 콜렉터가 모든 센서 노드를 버퍼 오버플로가 발생하기 전 방문할 수 없는 네트워크 환경에서도 데이터 손실을 최소화하기 위한 방안이 필요하다.

본 논문에서는 모바일 데이터 콜렉터의 이동속도 제약에도 불구하고 높은 데이터 신뢰성을 제공하는 CCHR(Collector Control for High Reliability)기법을 제안한다. 전체 네트워크 시간을 일정 주기의 round로 나누고, 각 round의 시작 시점마다 해당 round 동안 모바일 데이터 콜렉터가 이동할 경로를 결정한다. CCHR 기법에서 모바일 데이터 콜렉터는 자신의 이동경로 결정을 위해 필요한 각 센서 노드의 위치 정보와 버퍼 오버플로 시간 정보를 알고 있다고 가정한다. CCHR 기법에서 모바일 데이터 콜렉터의 이동경로를 결정하는 알고리즘은 EDF(Earliest Deadline First) 기반의 3단계 알고리즘으로 구현되며, 콜렉터가 직접 데이터를 수집하는 센서 노드의 수를 최대화함으로써 센서 네트워크의 데이터 신뢰성을 제공한다. CCHR 기법의 첫 번째 단계에서는 버퍼 오버플로 시한이 가장 임박한 노드를 선택하여 모바일 데이터 콜렉터가 시한을 만족시키면서 방문할 수 있으면 v_node 집합에 포함시키고, 시한을 만족시키지 못하면 m_node 집합에 포함시킨다(그림 1). 이를 한 round 내에서 이동경로가 결정될 때까지 반복한다. CCHR 기법의 두 번째 단계에서는 v_node 집합에 포함된 노드들에 대해 모바일 데이터 콜렉터의 이동경로를 최적화한다. v_node 집합에 포함된 노드들의 버퍼 오버플로 시한을 만족시키는 범위 내에서 콜렉터의 이동거리를 단축함으로써, 3단계에서 m_node 집합에 포함된 노드들을 구제할 수 있는 여유시간을 마련한다. 그림 2에서 보듯이, 두 노드 a와 b 사이에 임의의 노드 c를 추가하려면 다음과 같은 조건을 만족해야 한다.

조건1) 노드 c가 추가되었을 때 이동경로의 총 길이가 변경 전 이동경로의 총 길이보다 짧아야 한다.

조건2) 노드 c가 추가되었을 때 v_node 집합의 나머지 노드들의 오버플로 시한이 모두 만족해야 한다.

CCHR 기법의 마지막 세 번째 단계에서는 m_node 집합에 포함된 노드들에 대해 v_node 집합에 포함시켜 가면서 모바일 데이터 콜렉터의 이동경로를 결정한다. 이때에도 역시 m_node 집합 노드의 추가는 v_node 집합에 포함된 노드들의 버퍼 오버플로 시한을 만족시키는 범위 내에서 행해진다.

```
Input : start_node, overflow_time[1..N],
        cost[1..N][1..N]

Initialize :
 current_time = 0;
 current_node = start_node;
 for each Node i { deadline[i] = overflow_time[i] }

Algorithm PHASE1 :
 m = n = 0;
 v_node[0] = current_node;

 while(next earliest deadline < current round's end
       time) {
   choose Node i (≠ current_node) with the next
     earliest deadline;
   if ( deadline[i] <
        current_time + cost[current_node][i]) {
     m_node[++m] = i;
     deadline[i] += overflow_time[i];
   }
   else {
     current_time += cost[current_node][i];
     current_node = i;
     v_node[++n] = i;
     v_time[n] = current_time;
     v_deadline[n] = deadline[i];
     deadline[i] = current_time + overflow_time[i];
   }
 }
END
```

그림 1 CCHR 1단계 알고리즘

```
Algorithm PHASE2 :
  for (i = 0; i < n-1; i++) {
   a = v_node[i];     b = v_node[i+1];
   for (j = i+2; j < n; j++) {
     c = v_node[j];
     compute path_dec[c] = (cost[a][b]+
         cost[v_node[j-1]][c]+cost[c][v_node[j+1]]) -
     (cost[a][c]+cost[c][b]+cost[v_node[j-1]][v_node[j+1]]);}

   // condition 1
   choose Node c with largest positive path_dec[c] value;

   // condition 2
   compute delay = cost[a][c]+cost[c][b]-cost[a][b];
   for (k = i+1; k < n; k++) {
     check if v_time[k] + delay ≤ v_deadline[k]; }
   if (there exists such Node c) {
     update v_node[1..n] and v_time[1..n] lists; } }
END
```

그림 2 CCHR 2단계 알고리즘

```
Algorithm PHASE3 :
  for (i = 1; i < m; i++) {
   c = m1_node[i];
   for (j = 0; j < n-1; j++) {
     a = v_node[j];       b = v_node[j+1];
     compute path_inc[a]=cost[a][c]+cost[c][b]-cost[a][b]; }
   choose Node a with the smallest path_inc[a] value;
   for (k = j+1; k < n; k++) {
     check if v_time[k] + path_inc[a] ≤ v_deadline[k];}
   if (there exists such Node a) {
     update v_node[1..n] and v_time[1..n] lists; } }
END
```

그림 3 CCHR 3단계 알고리즘

성능 평가로는 모바일 데이터 콜렉터의 속도가 충분한 환경에서의 누락 비율의 비교와 콜렉터의 이동 속도 감소에 따른 누락 비율의 변화 비교를 수행하였다. 성능 비교 대상으로는 기존 모바일 데이터 콜렉터 스케줄링 기법인 EDF with k-lookahead[5]와 MWSF[5]를 선정하였고, [5]의 실험 환경을 따르도록 하였다. 실험 결과, 기존 모바일 데이터 콜렉터 스케줄링 기법들에 비해 콜렉터의 이동속도 감소에도 불구하고 최대 43.3% 낮은 데이터 손실률을 유지함을 확인하였다. 향후 연구로는, 버퍼 오버플로 시한을 만족하지 못하는 노드들의 데이터 손실을 방지하기 위해, 이들 노드의 데이터를 모바일 데이터 콜렉터로 전송하는 프로토콜을 개발하고자 한다.

[1] M. Ma and Y. Yang, "SenCar: An energy-efficient data gathering mechanism for large-scale multihop sensor networks", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 18, No. 10, pp. 1476-1488, 2007.

[2] M. Ma and Y. Yang, "Data gathering in wireless sensor networks with mobile collectors", in *Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium*, pp. 1-9, 2008.

[3] Y. Gu, D. Bozdag, R. Brewer, and E. Ekici, "Data harvesting with mobile elements in wireless sensor networks", *Computer Networks Journal*, Vol. 50, No. 17, pp. 3449-3465, 2006.

[4] A. Somasundara, A. Ramamoorthy, and M. Srivastava, "Mobile element scheduling with dynamic deadlines", *IEEE Transactions on Mobile Computing*, Vol. 6, No. 4, pp. 395-410, 2007.

[5] A. Somasundara, A. Ramamoorthy, M. and Srivastava, "Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines", in *Proceedings of the 25th IEEE International Read-Time Systems Symposium*, pp. 296-305, 2004.