퍼지 및 지능시스템학회 논문지 2007, Vol. 17, No. 6, pp. 760-765

센서 네트워크 시스템에 적용 가능한 고장 검출 알고리즘 개발

Development of Fault Detection Algorithm Applicable to Sensor Network System

육의수* · 윤성웅* · 김성호*
Eui-Su Youk · Seong-Ung Yun · Sung-Ho Kim

* 군산대학교 전자정보공학부

요 약

센서 네트워크 시스템은 한정된 자원을 갖는 센서노드들을 광대한 영역에 설치하여 새로운 정보를 수집하고 모니터링하는 기능을 한다. 센서 노드와 센서의 고장(Sensor node faulty or Sensor faulty)은 열악한 설치 환경이나 제한된 리소스에 의해 종종 발생 되는데 이들 고장은 네트워크 내에서 요구되는 양질의 서비스 제공에 많은 문제를 가져온다. 본 논문에서는 센서 노드의 고장 검출 알고리즘으로 알려져 있는 Consensus 알고리즘과 센서노드에서 사용되는 센서의 고장을 검출할 수 있는 localized faulty sensor detection 알고리즘을 혼합하여 시스템에 안정된 서비스를 제공할 수 있는 방법을 제안하며 실제 시뮬레이션과 제작된 실험장치에 적용함으로써 그 유용성을 확인하고자 한다.

키워드 : 센서 네트워크, 센서와 노드의 고장검출, 컨센서스, LFSD

Abstract

The sensor network system which has limited resources is deployed in a wide area and plays an important role of gethering information and monitoring. Generally, fault of sensor nodes which was caused by limited resources and poor environment happens. Futhermore, this fault poses many problems related with required quality of whole network. In this paper, new fault detection algorithm which utilizes both consensus algorithm and localized faulty sensor detection scheme is proposed. To verify the feasibility of the proposed scheme, some simulation and experiment are carried out.

Key Words : Sensor network, fault detection of sensor and sensor node, consensus algorithm, localized faulty sensor detection

1. 서 론

최근 사람 중심의 정보화 사회에서 모든 사물에 컴퓨팅 및 통신 기능을 부여하여 언제 어디서나 사람과 사물 그리고 사물과 사물 간의 정보들이 유기적으로 결합될 수 있는 유비쿼터스 컴퓨팅 사회의 구축에 대한 관심이 급증하고 있다. 국내에서는 이러한 유비쿼터스 컴퓨팅 기술을 미래 도시 환경의 구축에 적용하기 위한 u-city 사업이 구체적으로 추진되고 있다. u-city는 USN(Ubiquitous Sensor Network)요소기술과 첨단 IT 인프라 기술을 행정, 의료, 교통, 물류, 정보가전, 환경, 재난방지 등의 현대 도시 환경에 필수적인 제반 분야에 적용함으로써 도시 생활의 편의 증대, 삶의 질 향상, 체계적인 도시관리, 복지 향상 및 안전보장 등을 추구할 수 있는 신개념의 도시를 의미한다. 이러한 u-City와 같은 유비쿼터스 환경을 구축하기 위해서는 USN에 대한 지속적인 연구가 요구되고 있다[1-2].

센서 네트워크 시스템은 다양한 기능을 가진 센서가 부착된 전력소비가 적고 저가인 소형의 센서노드로 구성되고, Single-hop 또는 Multi-hop을 이용한 네트워크 형태를 구성하여 베이스 노드 또는 싱크노드로 측정된 정보를 전송하게 된다. 위와 같은 시스템은 보통 사람이 접근하기 힘든 열악한 환경에 설치되거나 제한된 전원을 사용하기 때문에 종종 센서나 센서 노드의 고장을 가져오게 되고 이들 고장은 사용자에게 양질의 서비스 제공에 문제를 야기시킬 수 있으며 최악의 경우 전체 네트워크 시스템의 마비를 가져올 수도 있다. 때문에 센서 네트워크 시스템 설계시 위와 같은 문제를 최소화 할 수 있도록 시스템이 설계 되어야 한다.

센서 네크워크의 기본 구성요소인 센서 노드는 무선 통신과 관련된 각종 프로토콜 스택, 원칩 마이크로프로세서 및 각종 센서, 시스템 소프트웨어 및 미들웨어 등으로 구성되며

접수일자 : 2007년 10월 18일
완료일자 : 2007년 12월 3일
감사의 글 : 본 연구는 산업자원부의 지역혁신 인력양성사업의 연구결과로 수행되었음

이들 요소는 각각의 고장원인을 포함하게 된다[3]. 일반적으로 이러한 고장은 마이크로프로세서 및 전력 고갈 등과 같은 하드 고장(hard failure)과 센서 노드에 장착되는 각종 센서의 퀄리브레이션 에러, 랜덤 노이즈 에러 등과 같은 소프트 고장(soft failure)으로 분류될 수 있다.

센서 노드에서의 하드 고장과 관련된 연구로 전력고갈 및 하드웨어 고장 등의 검출을 위해 DRAM등과 같은 부품의 생산성 향상을 위해 널리 사용되고 있는 BIST(Built-In Self-Test)기법을 적용하고자 한 연구가 진행된바 있다[4]. BIST 기법은 어느 정도의 하드웨어 중복성(redundancy)이 보장되는 상호 연결된 다중 프로세서(Multiprocessor)시스템의 고장진단에 적용될 수 있으며 따라서 광범위한 영역에 분산 설치되는 센서 노드들 역시 하드웨어의 중복성을 갖기 때문에 효율적인 적용이 가능하게 된다. 또한 하드웨어 중복성이 없는 경우에 적용가능한 고장진단 기법으로 Consensus 알고리즘이 제안된 바 있다[5]. Consensus 알고리즘은 센서 네트워크상에서 고장이 없는 것으로 판정된 노드들에 의해 고장 노드를 검출하는 알고리즘으로 주변 노드로부터의 정보로 생성되는 Suspect matrix와 Fault vector를 이용하여 주변 노드들과의 지속적으로 정보 교류와 갱신을 통해 고장을 검출한다.

센서 노드상에서 발생할지도 모르는 센서 퀄리브레이션 에러, 랜덤 노이즈 에러 등과 같은 소프트 고장의 검출을 위한 다양한 기법들이 많은 연구자에 의해서 개발된 바 있다. 이 중 J. Chen 등은 주기적으로 얻어지는 주변 노드들의 센서값과 자신의 측정값간의 차이 및 이의 시간에 따른 변화를 기반으로 주변노드로부터 전송된 센서 데이터의 에러(유효성)를 검출할 수 있는 LFSD(Localized Faulty Sensor Detection) 알고리즘을 제안한 바 있다[6].

본 연구에서는 센서 네트워크 환경하의 센서노드에서 발생할 수 있는 하드 및 소프트 고장을 동시에 검출할 수 있는 알고리즘을 제안하고자 한다. 제안된 알고리즘은 Consensus 알고리즘에서 하드 고장의 검출을 위해 주기적으로 주변노드에 메시지를 전송한다는 점과 LFSD 알고리즘에서 센서 데이터에 포함되는 고장 검출을 위해 주기적으로 주변 노드로부터 측정 데이터를 받아들인다는 점에 착안하여 두 기법을 융합한 새로운 형태의 센서네트워크에 적용가능한 고장 검출 알고리즘을 제안하고 이의 유용성을 확인하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 Consensus 알고리즘에 대해 설명하며, 3장에서는 LFSD 알고리즘에 대해 설명하고, 4장에서는 본 연구에서 제안된 고장검출 알고리즘에 대해 설명하며 5장에서는 제안된 기법의 유용성 확인을 위해 시뮬레이션 및 실제 시스템에 대한 적용 실험을 수행하고 마지막으로 결론을 기술한다.

2. Consensus 알고리즘

Consensus 알고리즘은 고장이 없는 것으로 판정된 노드들로부터의 정보를 기반으로 고장이 발생된 센서 노드를 검출하는 알고리즘으로 각각의 노드는 다른 노드들의 상태에 대한 정보를 가지고 있어야 하며 이와 동시에 다른 고장이 없는 것으로 판정된 노드(fault-free node)들로부터의 정보를 지속적으로 관측하는 것이 요구된다. 이러한 기능을 수행하기 위해, n개의 노드로 구성된 센서 네트워크의 경우, 각각의 노드는 크기가 n인 초기에 0으로 설정된 beat-vector를 갖고 있어야 한다. 노드에 존재하는 beat-vector는 주변의 j 번째 노드로부터의 응답이 없을 경우 j 번째 노드에 이상이 발생한 것으로 판정하고 beat-vector의 j번째 값을 증가 시키고, 응답이 있을 경우 해당 영역의 값을 일정값 이상 감소시킨다. 또한 각각의 노드는 beat-vector외에도 Suspect matrix와 Fault vector를 갖는다. 2차원의 Suspect matrix는 설치 노드들에 대한 정보들로 구성된 Suspect vector들로 구성된다. Suspect matrix S[i,j]는 1 또는 0의 값을 가질 수 있으며 i 번째 노드가 j번째 노드의 고장이 의심되면 해당 비트를 1로 셋하게 된다. 즉 1로 표시된 값은 고장을 나타낸다. 정상동작되는 i노드들이 j노드를 고장으로 판정할 경우 Consensus가 이루어진다. 또한 각 노드는 Fault vector를 모니터링 하면서 고장노드의 유/무를 식별 할 수 있다. 만일 노드의 과반수 이상이 그 해당노드를 고장으로 판정하면 F의 해당요소 j는 1로 셋 된다.

```
1.  for Node ID=p where 0 ≤ p<n do
2.     Initialize F[i] and S[j,k] to 0 where 0 ≤ i,j,k<n
3.     On the receipt of S' from Node q do
4.        for(j=0, j<n, j++)
5.           if(j = q)then
6.              for(k=0, k<n, k++)
7.                 S[j,k]=S[j,k] ‖ S'[j,k]
8.              end for
9.           end if
10.          if(j=q)then
11.             for(k=0, k<n, k++)
12.                S[j,k]=S'[j,k]
13.             end for
14.          end if
15.       end for
16.    end do
17.    Every T_gossip seconds do
18.       for(k=0, k<n, k++)
19.          if(fail_check(k)=true) then
20.             S[p,k]←1 then S[p,k]←0
21.       end for
22.       for(k=0, k<n, k++)
23.          temp←0
24.          for(j=0, j<n, j++)
25.             if S[j,k]=1 then temp←temp+1
26.          end for
27.          if temp>n/2 then F[k]←1 else F[k]←0
28.       end for
29.       for(k=0, k<n, k++)
30.          temp←0
31.          for(j=0, j<n, j++)
32.             if S[j,k] ‖ F[j]=1 then temp←temp+1
33.          end for
34.          if temp=n then consensus_reached(j)
35.       end for
36.       Append S to message when gossip is sent
37.    end do
38. end for
```

그림 1. Consensus 알고리즘
Fig. 1. The consensus algorithm

그림 2는 노드상에서 동작되는 Consensus 알고리즘의 동작순서를 나타낸 것이며, 알고리즘의 기본 동작은 다음과 같다.

STEP 1: 노드 1의 Suspect matrix를 보면 노드 1이 이미 노드 0의 고장을 의심하고 있음을 알 수 있고, 노드 2의 Suspect matrix를 보면 노드 2 또한 0의 고장을 의심하고 있음을 알 수 있고, 이전에 노드 0의 고장을 의심하는 노드 3과 데이터 교환이 있었음을 알 수 있다.

STEP 2: 노드 2는 노드 1의 Suspect matrix를 수신하여 자신의 Suspect matrix와 수신된 노드 2의 Suspect matrix를 논리 연산 OR하여 업데이트한다.

STEP 3: 노드 2는 모든 Fault-free 노드들이 노드 0의 고장을 인지한 후 consensus가 완료되는 것을 보여준다. 고장노드 0과 관련된 값이 모두 1로 되면 고장벡터 F는 해당 열을 1로 갱신한다.

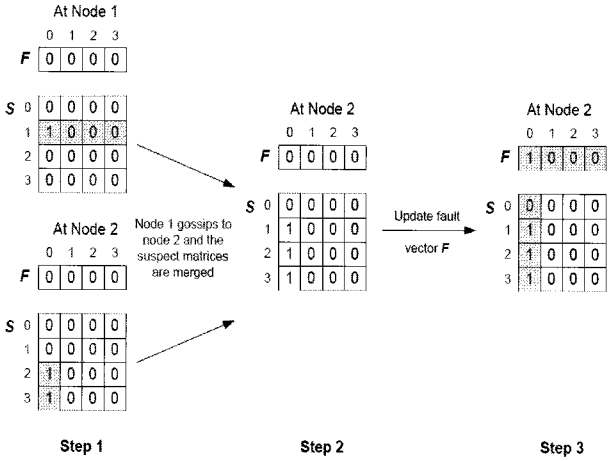

그림 2. 4개의 노드로 이루어진 시스템에서의 consensus 알고리즘

Fig. 2. Illustration of the consensus algorithm in the case of 4-nodes sensor network system

Step 1: Each sensor S_i tests every member of $S_j \in N(S_i)$ to generate test $c_{ij}\{0,1\}$ using the following method:
1: Each sensor S_i, set $c_{ij} = 0$ and compute d_{ij}^t;
2: IF $|d_{ij}^t| > \theta 1$ THEN
3: Calculate $\Delta d_{ij}^{\Delta t_l}$;
4: IF $|\Delta d_{ij}^{\Delta t_l}| > \theta 2$ THEN $c_{ij} = 1$;

Step 2: S_i generates a tendency value T_i based upon its neighboring sensors' test value:
1: IF $\sum_{S_j \in N(S_i)} c_{ij} < \lceil |N(S_i)|/2 \rceil$, where $|N(S_i)|$ is the number of the S_i's neighboring nodes THEN
2: T_i = LG;
3: ELSE T_i = LF;
4: Communicate T_i to neighbors;

Step 3: Compare the number of S_i's LG neighboring nodes with different test results to determine its status:
1: IF $(\sum_{S_j \in N(S_i) \text{and} T_j = LG}(1-2c_{ij}) \geq \lceil |N(S_i)|/2 \rceil$ THEN
2: T_j = GD;
3: Communicate T_i to neighbors;

Step 4: For the remaining undetermined sensors, do the following steps in parallel for $Maxd$ cycles:
1: FOR $i = 1$ to n
2: IF T_i = LG or T_i = LF THEN
3: IF T_j = GD $\forall S_j \in N(S_i)$, THEN
4: IF $c_{ji} = 0$ THEN
5: T_i = GD;
6: ELSE T_i = FT;
7: ELSE repeat
8: Communicate T_i to neighbors;

Step 5: If ambiguity occurs, then the sensor's own tendency value determine its status:
1: FOR each S_i, IF $T_j = T_k = GD$ $\forall S_j, S_k \in N(S_i)$, where $j \neq k$, and IF $c_{ji} \neq c_{ki}$ THEN
2: IF $T_i = LG$ (or LF) THEN
3: $T_i = GD$ (or FT)

그림 3. Localized Faulty Sensor Detection 알고리즘

Fig. 3. The Localized Faulty Sensor Detection algorithm

3. Localized Faulty Sensor Detection 알고리즘

Localized Faulty Sensor Detection(LFSD) 알고리즘은 인접한 노드간의 센서 값 비교를 통해 센서의 이상 유/무를 판단하는 알고리즘이다. 각각의 센서 노드는 규칙적으로 자신의 센서값을 인접한 노드에게 전송하며, 알고리즘의 동작 순서는 그림 3과 같이 5단계로 나뉘어 진다.

1번째 단계에서 각 노드들의 평가값 c_{ij}는 0으로 설정되며, 평가값은 같은 시간에 측정된 현재 노드의 센서값 S_i와 인접 노드로부터 측정된 센서값 S_j의 차와 미리 정의된 threshold 값과의 비교를 통해 결정되며, 평가값은 센서값의 차가 threashold 값보다 클 경우 1로 설정된다.

2번째 단계에서 각 노드의 상태값 T_i는 첫 번째 단계에서 결정된 인접 노드의 평가값을 토대로 변경되며, 상태값은 인접 노드의 상태에 따라 LG 또는 LF로 결정된다. 결정된 노드의 상태는 인접 노드에게 전송되며 수신된 인접 노드의 상태는 다음단계에서 상태를 결정하는데 사용된다.

3번째 단계에서는 인접 노드중 상태값 LG를 갖는 노드의 개수에 따라 노드의 GD 상태가 결정되며, 4번째 단계에서는 아직 결정되지 않은 노드의 상태값을 1~3번째 단계에서 결정된 인접 노드의 평가값과 상태값의 반복 비교를 통해 GD 또는 FT로 결정된다.

마지막으로 5번째 단계에서는 4단계까지 결정되지 못한 노드의 상태값이 인접 노드의 상태값을 통해 GD 혹은 FT로 결정된다.

4. 제안된 센서 노드 고장 검출 알고리즘

본 절에서는 제안된 센서 노드 고장검출 알고리즘의 개요에 대해 설명하겠다. 그림 4는 센서 노드의 내부 동작 시퀀스를 나타낸 것이며, 노드의 동작은 크게 타이머 이벤트 발생에 의해 실행되는 루틴과 메시지 수신시 실행되는 루틴으로 분리된다.

센서 노드는 이벤트에 따라 동작하게 되며, 이벤트 종류는 크게 그림 5에서와 같이 타이머 이벤트 발생시 수행되는 타이머 이벤트와 그림 6에서 볼 수 있듯이 메시지를 수신할 때 발생되는 메시지 수신 이벤트로 나눌 수 있다.

그림 5의 타이머 이벤트는 각 노드마다 설정된 일정 시간이 경과할 경우 발생되는 이벤트이며, 이 이벤트 내에서 수행되는 루틴은 크게 2가지로 구분되며 이들 각각은 하드 고장 검출을 위한 Consensus 알고리즘 루틴과 소프트 고장을 위한 LFSD 알고리즘 루틴이다. 이 루틴들의 수행 여부는 Tgossip 시간의 경과에 따라 결정된다. Tgossip은 하드 고장을 위한 노드 동작전 정해진 시간으로서, 노드들은 타이머 이벤트가 발생할 때 마다 이를 비교한다. Tgossip 시간 경과시 노드들은 주변노드와의 통신을 통해 응답을 확인하여 Suspect Matrix를 기반으로 Fault Vector를 갱신하며, 갱신된 Fault Vector를 기반으로 Consensus에 도달되었는지를 결정한다. 이때 응답여부에 따라 해당 노드의 beat-vector를 증가 혹은 감소시킨다.

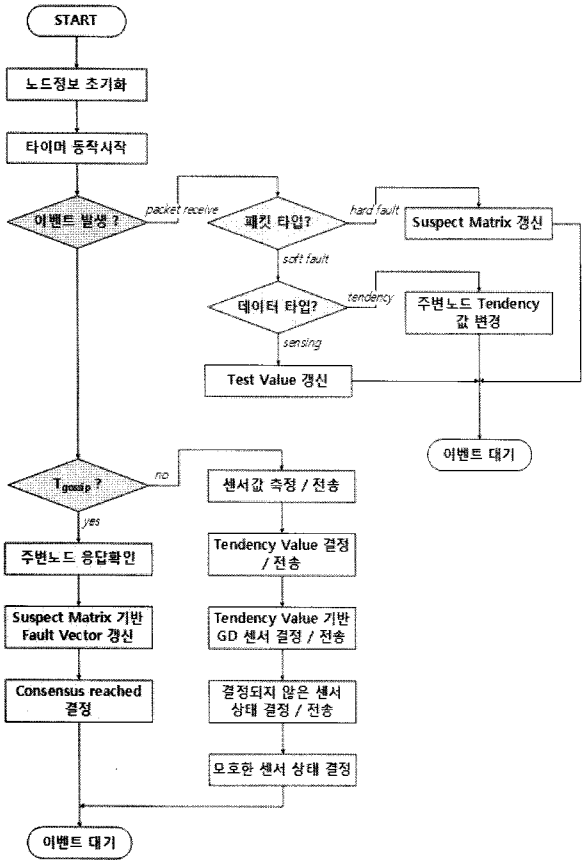

그림 4. 제안된 알고리즘에서의 센서 노드 동작 시퀀스
Fig. 4. Operation sequence of the sensor node

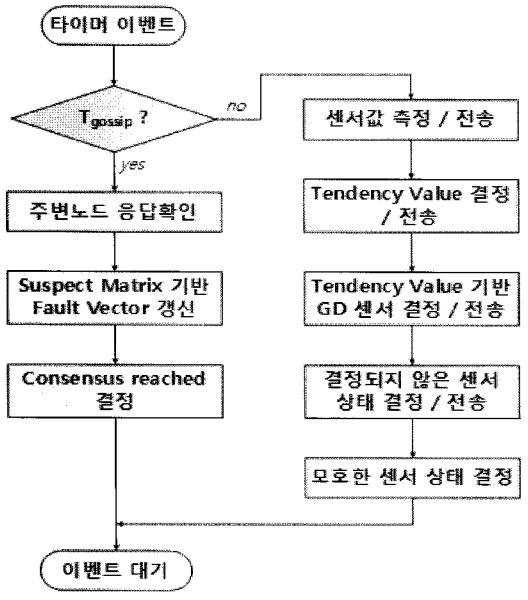

그림 5. 타이머 이벤트 처리과정
Fig. 5. The processing of timer event routine

Tgossip 시간이 경과되지 않았을 때 센서 노드는 자신의 센서로부터 값을 입력받아 이를 전송하며, 소프트 고장을 위한 상태값(Tendency Value)을 결정하게 된다. 최초 상태값은 주변 노드의 센서값에 따라 LG 또는 LF로 판별되며, 이후 주변 노드의 LG 상태값의 개수에 따라 GD 노드를 판별하게 된다. 이렇게 판별된 GD노드의 개수와 주변노드의 상태를 통해 아직 결정되지 않은 노드의 상태를 판별하고, 마지막으로 판별하기 모호한 노드의 상태를 결정한다.

그림 6은 메시지 수신시 발생되는 이벤트이며, 이 이벤트는 패킷과 데이터의 타입에 따라 구분된다. 패킷의 종류는 하드 고장과 소프트 고장을 위한 패킷으로 나뉘어지며, 하드 고장을 위한 패킷은 타이머 이벤트 발생시 다른 노드로부터의 응답확인을 위한 메시지로서, 이때 노드는 자신의 Suspect Matrix를 갱신한다. 소프트 고장을 위한 패킷은 데이터 타입에 따라 구분되며, 데이터의 타입에 따라 수신된 데이터를 기반으로 테스트값을 갱신하거나 상태값을 저장한다.

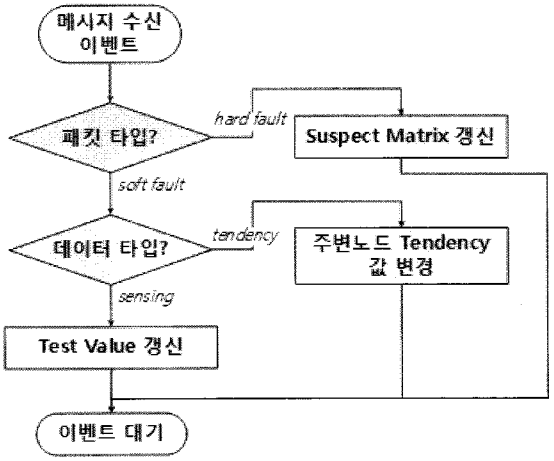

그림 6. 수신 메시지 처리과정
Fig. 6. The processing of receive message

5. 실 험

적용된 알고리즘의 유용성을 확인하기 위해 다음과 같은 실험을 진행하였으며, 실제 노드 및 임베디드 시스템에 적용하기 전 PC상에서 시뮬레이션을 진행하였다.

5.1 시뮬레이션 고찰

그림 7,8은 Visual C++를 이용하여 작성된 시뮬레이션을 위한 GUI화된 프로그램의 동작 화면이며 시뮬레이션은 450×550 픽셀내에 랜덤하게 뿌려진 33개의 가상노드를 대상으로 진행되었다. 또한 각각의 센서 노드는 그림 4에 같은 동작 시퀀스를 수행하도록 프로그램되었다.

GUI화된 프로그램은 원활한 시뮬레이션을 위해 GUI 화면에서 각 노드를 선택하여 선택된 노드에 인위적인 하드 및 소프트 고장을 발생시킬 수 있도록 설계되었다. 선택된 노드에 인위적인 하드 고장을 발생시킬 경우 해당노드는 적색으로 표시되며 소프트 고장은 청색으로 표시된다. 그림 7은 사용자가 임의의 센서 노드를 하드 고장 상태로 설정하였을 경우의 Consensus 알고리즘이 수행되는 화면을 나타내며 그림에서 적색으로 표시된 노드는 일정시간이 지나 Consensus 상태에 도달되면 흑색으로 표시된다. 각 노드의 타이머 이벤트는 1초마다 발생되도록 설정하였으며, Tgossip 시간은 10초로 설정하였다.

그림 8은 그림 7과 같은 환경하에서 임의의 노드에 발생

된 소프트 고장의 판별을 위한 시뮬레이션 동작화면이다. LFSD 알고리즘에서는 임의의 노드에서 주변노드의 확인이 요구되며 이를 위해 본 시뮬레이션에서는 반경을 해당 노드 주변의 100픽셀로 설정하였고, threshold 값은 $\Theta_1=40$, $\Theta_2=60$으로 설정하였다. 타이머 발생는 앞의 경우와 마찬가지로 1초로 설정하여 실험을 진행하였다. 발생된 하드 고장의 인식결과와의 구별을 위해 소프트 고장이 인식된 노드는 흰색으로 표시되도록 하였다.

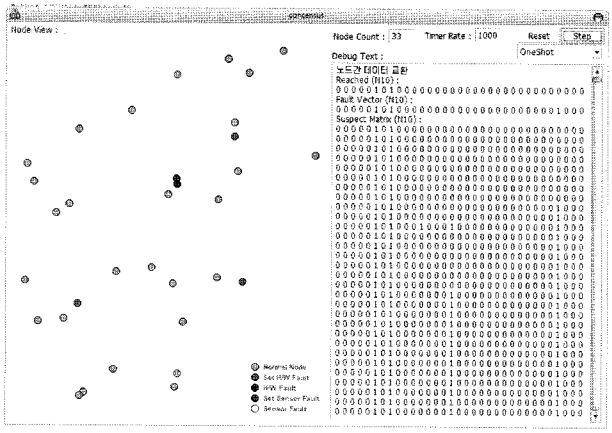

그림 7. 임의의 센서 노드에서 발생된 하드 고장 검출 화면
Fig. 7. Illustration of the hard fault detection

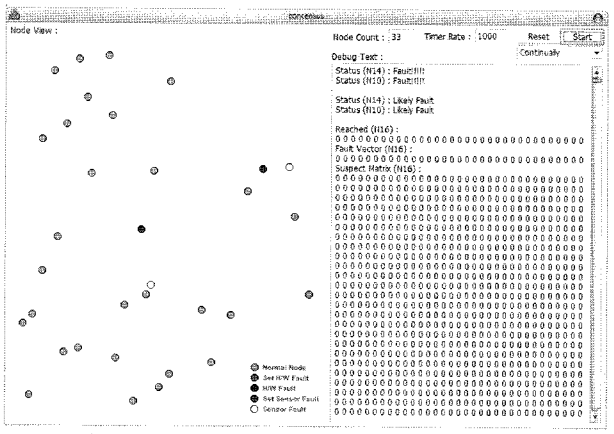

그림 8. 임의의 센서 노드에서 발생된 소프트 고장 검출 화면
Fig. 8. Illustration of the soft fault detection

상기의 시뮬레이션 수행을 통해 본 연구에서 제안된 알고리즘이 효과적으로 하드 및 소프트 고장을 판별할 수 있음을 확인할 수 있다. 그러나 본 시뮬레이션 결과는 실제 상황에서 발생할지도 모르는 전파간섭 등과 같은 불확실성에 기인하는 패킷 손실의 경우를 고려하지 못하였기 때문에 실제의 센서 네트워크 시스템에의 적용시에는 고장 판별율이 떨어질 수 있다.

5.2 실제 시스템에의 적용 실험

본 연구에서는 제안된 고장검출 알고리즘의 유용성 검증하기 위해 그림 9와 같은 시스템을 구축하였으며 10개의 센서 노드와 1개의 sink 노드 및 sink 노드에 연결되어 10개의 센서노드의 고장 상황을 모니터링하기 위한 임베디드 시스템으로 구성된다.

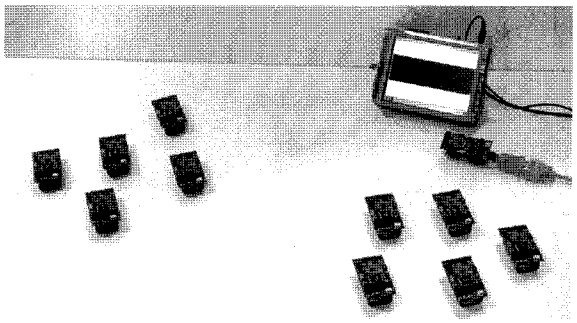

그림 9. 센서 노드 고장 검출을 위해 제작된 센서 노드 및 임베디드 시스템
Fig. 9. The sensor node and embedded system for fault detection of sensor node

센서 노드 및 Sink 노드상의 하드 및 소프트 고장검출 알고리즘은 TinyOS상에서 어플리케이션 개발시 사용되는 NesC 언어를 사용하여 구현되었다. 또한 임베디드 시스템에 연결된 sink 노드는 노드들 간에 전송되는 무선 데이터 패킷을 리슨하고 이를 RS-232 인터페이스를 통해 임베디드 시스템으로 전송하는 기능을 수행한다. 임베디드 시스템에는 임베디드 리눅스가 OS로 포팅되어 있으며 노드들간에 전송되는 무선 데이터 패킷의 수신및 고장 검출 상황을 실시간으로 모니터링하기 위한 어플리케이션 프로그램은 UI builder인 glade와 GIMP Toolkit인 GTK+를 사용하였다.

본 연구에서 제안된 알고리즘의 실제 시스템의 적용가능성을 확인하기 위해 다음과 같은 두 가지 경우에 대한 실험을 수행하였다.

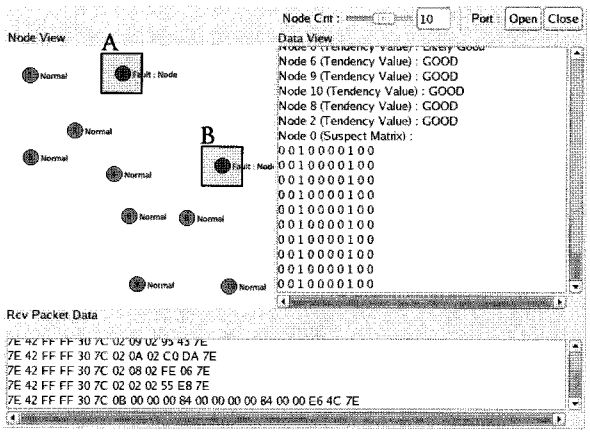

그림 10. 2, 7번째 센서 노드에서 발생된 하드 고장의 검출 알고리즘 동작 화면
Fig. 10. Illustration of the fault detection algorithm on the 2nd, 7th sensor node with hard fault

첫 번째 경우는 하드 고장의 검출 확인을 위해 7번째 센서 노드의 전원을 강제적으로 차단시켰으며 이 경우 임베디드 시스템상의 고장 검출 알고리즘 동작화면을 나타내면 그림 10과 같다.

두 번째의 경우는 소프트 고장의 검출 성능 테스트를 위해 5번째 센서 노드의 AD0번 포트에 연결된 센서를 제거하고 가변저항을 연결하여 센서값의 변화를 주었으며 이 경우의 고장 검출 알고리즘 동작화면은 그림 11과 같다.

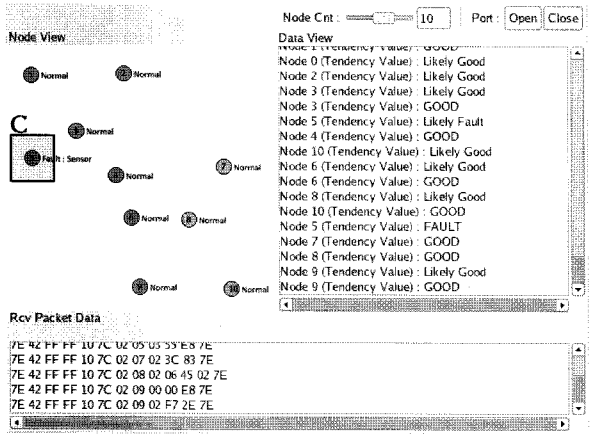

그림 11. 5번째 센서 노드에서 발생된 소프트 고장의 검출 알고리즘 동작 화면

Fig. 11. Illustration of the fault detection algorithm on the 5th sensor node with soft fault

6. 결 론

센서 네트워크의 기본 구성요소인 센서 노드 상에서 발생될 지도 모르는 하드 고장과 소프트 고장의 효율적인 검출을 위해 본 연구에서는 Consensus 알고리즘 및 Localized Sensor Fault Detection 알고리즘을 융합한 새로운 형태의 알고리즘을 제안하였으며 제안된 알고리즘의 유용성 확인을 위해 시뮬레이션 및 실제 제작된 시스템에의 적용 실험을 수행하였으며, 시뮬레이션 및 적용실험 결과 제안된 기법의 유용성을 확인할 수 있었다.

참 고 문 헌

[1] 김민수 외 2명,, "USN 미들웨어의 특징 및 기술개발동향," *주간기술동향* 통권 1284호, 2007,2

[2] 박승민, "센서 네트워크노드 플랫폼 및 운영체제 기술 동향," *전자통신동향분석* 제 21권 제 1호 ,2006, 1

[3] J. Chen, "Distributed Fault Detection of Wireless Sensor Networks", Iowa, 2006

[4] S. Ranganathan. A.D. George, R.W. Todd, Matthew C. Chidester, "Gossip-Style Failure Detection and Distributed Consensus for Scalable Heterogeneous Clusters", HCS Research Laboratory, 2000.

[5] M. Young, The Technical Writer's Handbook, Mill Valley, Seoul, 1989.

[6] 이문노, 문정호, 정명진, "광 디스크 드라이버의 트랙킹 서보 시스템을 위한 다목적 강인제어기의 설계," *제어.자동화.시스템공학 논문지*, vol. 4, no.5, pp. 592-599, October 1998.

[7] Z. Shiler and S. Dubowski, "Time optimal paths and acceleration lines of robotic manipulators," *Proc. of the 26th Conf. Decision and Control*, pp. 98-99, 1987.

[8] A. K. Somani and V. K. Agarwal. Distributed Diagnosis Algorithms for Regular Interconnected Structures. *IEEE Transaction of Computers*, Vol.41, No.7: 899-906, July 1992.

[9] A. Mahmood and E. J. McCluskey. Concurrent error detection using watchdog processors-a survey. *IEEE Transactions on Computers*, Vol.37, No.2: 160- 174, Feb.1988.

[10] Aspnes, J. and Herlihy, M. Fast Randomized Consensus Using Shared Memory. *Journal of Algorithms*. Vol 11, No 4, 441-461, 1990.

[11] Kuhl, J. and Reddy, S. Fault-Diagnosis in Fully -Distributed Systems. Proceedings of the 11th International IEEE Symposium on Fault-Tolerant Computing. 100-105, 1981.

저 자 소 개

육의수(Youk Eui-Su)
2004년 : 군산대학교 공과대학 학사
2006년 : 동 대학원 석사과정
2006년~현재 : 동 대학원 박사과정

관심분야 : 고장진단, 공장 자동화, Sensor Network
Phone : 010 - 2231 - 2849
E-mail : sixofnum@hotmail.com

윤성웅(Yun Seong-Ung)
2007년 : 군산대 공과대학 학사
2007년~현재 : 동 대학원 석사과정

관심분야 : 공장 자동화, 임베디드 시스템, 센서 네트워크
Phone : 016 - 512 - 0424
E-mail : songung@lycos.co.kr

김성호(Kim Sung-Ho)
1984년 : 고려대 공과대학 학사
1986년 : 고려대 대학원 석사
1991년 : 고려대 대학원 박사
1995~1996년 : JAPAN HIROSHIMA UNIVERSITY, POST-DOC.
1997~현재: 군산대 전자정보공학부 교수

관심분야 : 고장진단, 공장 자동화, 센서 네트워크
Phone : 063-469-4704
E-mail : shkim@kunsan.ac.kr