Proceedings of KFIS Autumn Conference 2007 Vol. 17, No. 2.

센서 네트워크 시스템에 적용 가능한 고장 검출 알고리즘 개발에 관한 연구

A study on the development of fault detection algorithm for sensor network system

윤성웅[1], 육의수[2], 김성호[3]

[1, 2, 3] 전북 군산시 군산대학교 전자정보공학부

E-Mail: [1] songung@lycos.co.kr, [2] sixofnum@hotmail.com, [3] shkim12244@hotmail.com

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.

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 설계 시 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 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 켈리브레이션 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 네트워크 상에서 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 캘리브레이션 에러, 랜덤 노이즈 에러 등과 같은 소프트 고장의 검출을 위한 다양한 기법들이 많은 연구자에 의해서 개발된 바 있다. 이중J. 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 Localized Faulty Sensor Detection 알고리즘을 제안한 바 있다[6].

본 연구에서는 센서 네트워크 환경하의 센서 노드에서 발생할 수 있는 하드 및 소프트 고장을 검출하는 알고리즘을 제안하고자 한다. Consensus 알고리즘에서 하드고장의 검출을 위해 주기적으로 주변노드에 메시지를 전송한다는 점과 Localized Faulty sensor Detection 알고리즘과 같은 소프트 고장검출 알고리즘이 센서 데이터에 포함되는 고장의 검출을 위해 주기적으로 주변 노드로부터 측정 데이터를 받아 들인다는 점에 착안하여 두 기법을 융합한 센서 노드 고장 검출 알고리즘을 제안한다.

본 논문의 구성은 다음과 같다. II장에서는 Consensus 알고리즘에 대해 설명하며, III장에서는 localized faulty sensor detection 알고리즘에 대한 내용을 설명하고, IV장에서는 본 연구에서 제안된 센서 노드 고장검출 알고리즘에 대해 설명하며 V장에서는 제안된 기법의 유용성 확인을 위해 시뮬레이션 및 실제 시스템에 대한 적용 실험을 수행하고 마지막으로 결론을 기술한다.

2. Consensus 알고리즘

Consensus 알고리즘은 고장이 없는 노드들에 의해 고장을 검출하는 알고리즘으로 각각의 노드가 다른 노드들의 상태에 대한 관측 정보를 가지고 있어야 하며 이와 동시에 다른 고장이 없는 노드(Fault-free node)들로부터의 의심정보를 지속적으로 관측하는 것이 요구된다. 이러한 기능을 수행하기 위해, n개의 노드로 구성된 센서 네트워크의 각 노드는 크기가 n인 bit-vector를 갖고 있으며 이 벡터는 0으로 초기화 된다. 이 벡터의 j번째 비트는 j번째 노드가 gossip list의 heartbeat 값이 종료된 경우를 고장으로 간주하고 이 경우 1셋 한다.

각 노드는 Suspect matrix와 Fault vector를 갖는다. 2차원의 Suspect matrix는 설치 노드들에 대한 정보들로 구성된 Suspect vector들로 이루어진다. 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. 정상 동작되는 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. 다음은 Consensus 알고리즘의 기본 동작은 다음과 같다.

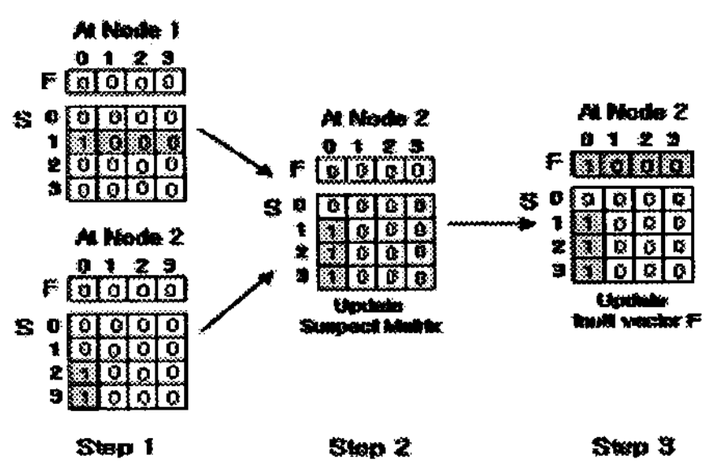

그림 1. Consensus 알고리즘의 기본동작

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 관련있는 값이 모두 1로 되면 고장벡터 F는 해당 열을 1로 갱신된다.

그림 2는 노드에서 동작되는 consensus 알고리즘을 나타낸다.

```
1.  for Node ID=p where 0 ≤ p<n do
2.     Initialize F[i] and S[j,k] to 0 where 0 ≤ j,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 2. Consensus 알고리즘

3. Localized Faulty Sensor Detection 알고리즘

Localized faulty sensor detection algorithm은 자신의 센서값과 자기와 이웃한 센서 노드의 센서 값을 비교하여 얻어진 결과를 이용하여 자신의 센서의 이상 유/무를 결정하는 알고리즘이다.

센서 값의 비교는 그들의 전송 영역 내에 있는 이웃 노드들의 센서값을 고려하여 얻을 수 있다. 각 노드들은 규칙적으로 그들이 측정한 센서값을 이웃 노드들에게 전송한다. 만약 자신이 측정한 값과 아웃노드의 값이 차이가 있다면 고장에 대한 의심을 갖게 되고 이전에 측정한 값과 비교하게 된다. Test value는 자신의 측정 센서값을 나타내는 S_i와 이웃의 모든 센서들을 나타내는 S_i의 차를 이용한 d 값과 이전에 측정한 d^t값과 현재 측정된 d^{t+1}값의 차인 $\Delta d^{\Delta t}$값, 그리고 사용자에 의해 미리 정해진 θ_1과 θ_2에 의해 결정된다.

만약 Test value c_{ij}가 0이면 센서 S_i와 센서 S_j는 둘다 Good 이거나 둘다 Faulty 일 것이다. 반면, 1 값을 가지면, S_i와 S_j는 다른 값을 가지며, 상태 또한 다를 것이다.

센서들은 여러가지 tendency(LG , LF , GD ,FT)을 가진다. 초기에 센서들의 tendency는 LG 또는 LF 일 수 있다. 이는 Test value에 의해서 결정되며, 결정 후에는 이웃노드들에게 자신의 tendency를 전송한다. 이후 Test value 값과 일치되는 LG 센서들은 GD인지 FT 인지를 결정할 수 있다. 만약 네트워크 내에서 GD 센서가 존재한다면, 그 노드의 Test value는 다른 센서들의 상태를 분석하는데 사용될 수 있다.

Localized faulty sensor detection algorithm은 다음과 같다.

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 $\Delta d_{ij}^{\Delta t}$;
4: IF $|\Delta d_{ij}^{\Delta t}| > \theta 2$ 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 nmmber 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 $(\sum_{S_j \in N(S_i) and T_j = LG}(1-2c_{ij}) \geq \lceil |N(S_i)|/2 \rceil$ 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

Step 1에서의 Test value c_{ij}는 사용자에 의해서 미리 정의된 threshold θ에 의해 결정되며, θ_1과 θ_2는 사용자에 의해 요구되는 값으로 설정 가능하다. STEP 5는 전체 네트워크에서 분석의 결과에 대한 유효성을 검증하는 단계이다.

4. 제안된 알고리즘

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

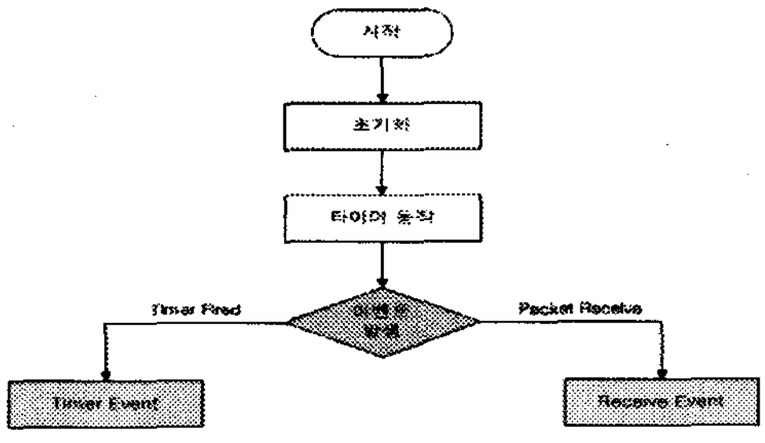

그림 4. 센서 노드 동작 시퀀스

그림 5와 그림 6은 타이머 이벤트와 메시지 수신 이벤트가 발생되었을 때 처리되는 과정을 나타낸 것이다. 타이머 이벤트가 발생되면 이벤트 처리 루틴 내에서 Consensus 관련 작업을 수행할 것인지 Localized faulty sensor detection 알고리즘 관련 작업을 할 것인지를 결정하게 되고 해당 함수들을 수행하게 된다. 메시지 수신 이벤트가 발생되게 되면 먼저 패킷의 종류가 무엇인지 체크하고 관련 함수를 수행하게 된다.

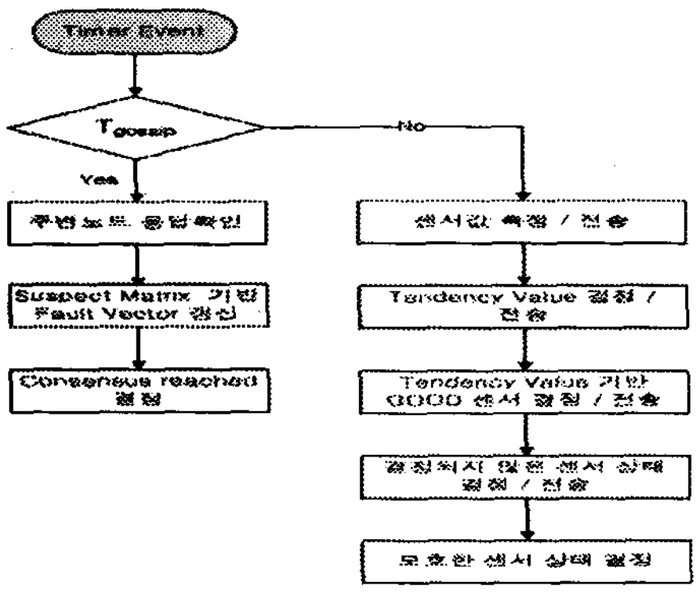

그림 5. 타이머 이벤트 처리과정

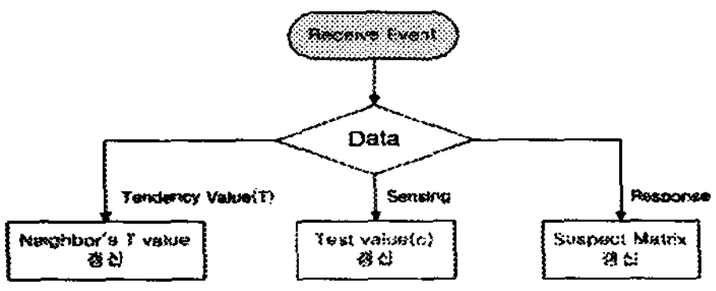

그림 6. 메시지 수신 처리과정

5. 실험

1. 시뮬레이션 고찰

다음 그림 7은 Visual C++를 이용하여 작성된 시뮬레이션 프로그램이다. 실험은 (450×550)px 공간안에 랜덤하게 뿌려진 33개의 가상노드를 대상으로 진행하였으며, 각각의 노드는 그림 4.의 순서도에 의해 이벤트를 처리하게 된다.

그림 7. 은 임의의 센서 노드를 H/W fault 상태로 설정하여 consensus 알고리즘이 제대로 수행되는지 여부를 파악한 실험 결과이다. 그림에서 검은색 노드는 consensus 알고리즘 처리 후 consensus reached 상태에 도달한 노드를 나타내며, 빨간색 노드는 임의로 설정한 H/W fault 노드를 나타낸다. 노드의 OnTimer 시간은 1s로 설정하였으며, Tgossip 시간은 10s로 설정하여 진행하였다.

그림 8. 은 상기 실험과 동일한 방법으로 임의의 노드를 Sensor Fault 상태로 설정하여 진행하였다. neighbor 결정을 위한 반경은 주변 100px로 설정하였고 threshold 값 θ_1 = 40, θ_2 = 60으로 설정, OnTimer시간은 위 실험과 동일하게 1s로 설정하여 실험을 진행하였다. 하얀색 노드는 알고리즘 수행후 Sensor Fault로 결정된 노드를 말하고, 파란색 노드는 임의로 설정한 Sensor Fault 노드를 말한다.

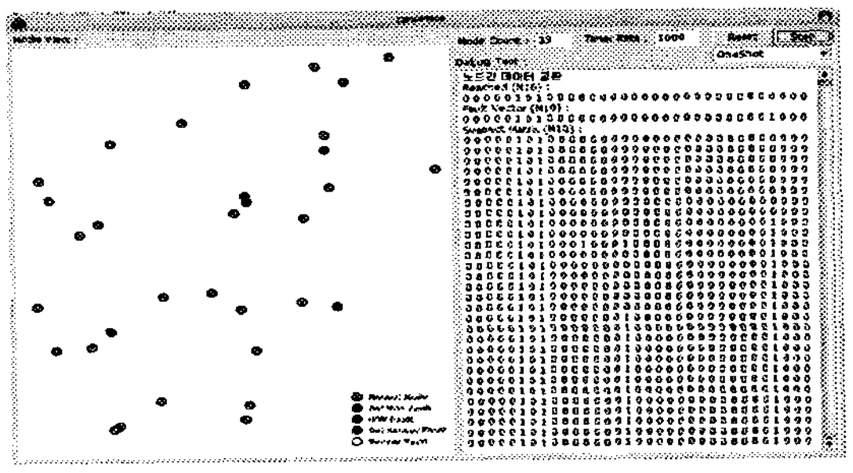

그림 7. 노드의 고장 검출

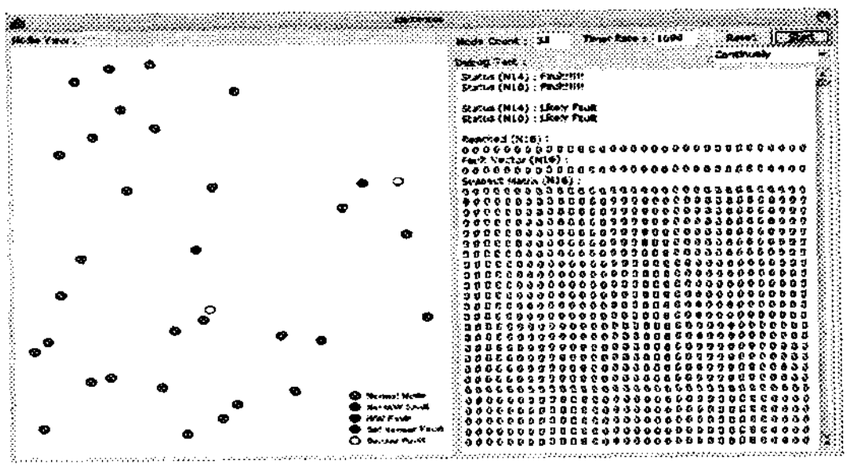

그림 8. 센서의 고장 검출

실제 실험과 달리 시뮬레이션에서는 실제 노드간 발생할 수 있는 패킷 손실 및 방해전파로 인한 거리측정 오류, 노드의 오동작 등 발생할 수 있는 예외를 제외한 실험이기 때문에 H/W 혹은 Sensor fault 로 설정한 노드가 알고리즘에 의하여 완벽하게 검출될 수 있다.

2. 실제 시스템에의 적용 실험

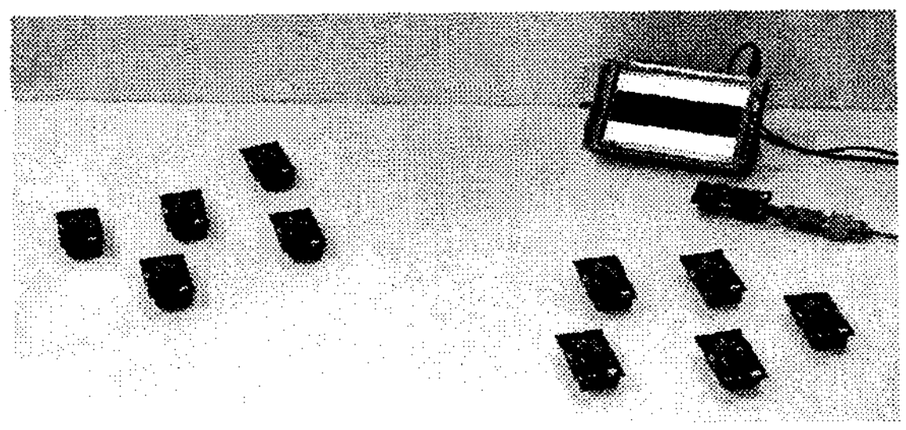

그림 9. 임베디드 기반모니터링 시스템

본 연구에서는 제안된 고장검출 알고리즘의 유용성 검증하기 위해 그림 9와 같이 10개의 센서 노드와 1개의 Sink 노드 그리고 임베디드 시스템으로 구성된 임베디드 기반 센서 네트워크 시스템을 구축하였다.

센서 노드 및 Sink 노드에는 TinyOS를 이용하여 프로그램 하였고, 10개의 센서 노드에는 제안된 알고리즘이 포팅되었고 Sink 노드는 노드들 간에 전송되는 무선데이터 패킷을 Listen 하여 임베디드 장비로 전송하는 프로그램이 포팅되어졌다. 또한 임베디드 장비에는 임베디드 리눅스가 포팅 되어졌고 GTK+를 이용하여 GUI 환경을 구축하였다

제안된 알고리즘의 유용성을 확인하기 위해 다음과 같은 두가지 실험을 하였다.

1) 노드의 고장 검출 여부를 확인하기 위해 7번 노드와 2번 노드의 전원을 Off 시켜 보았다.
2) 센서의 고장을 판별하기 위해 5번 노드의 AD0번 포트에 연결되는 센서를 제거하고 가변저항을 연결하여 센서값의 변화를 주었다.

그림 10은 실험 1)의 결과를 나타낸 것이다. 그림 10에서 A 영역과 B영역은 Consensus 알고리즘에 의해 고장으로 검출된 노드를 나타낸다.

그림 11은 실험 2)에 대한 결과를 나타낸 것이다. 그림에서 C 영역은 5번 노드가 이웃노드들의 정보를 이용하여 자신의 센서가 고장임을 감지한 것을 나타낸다.

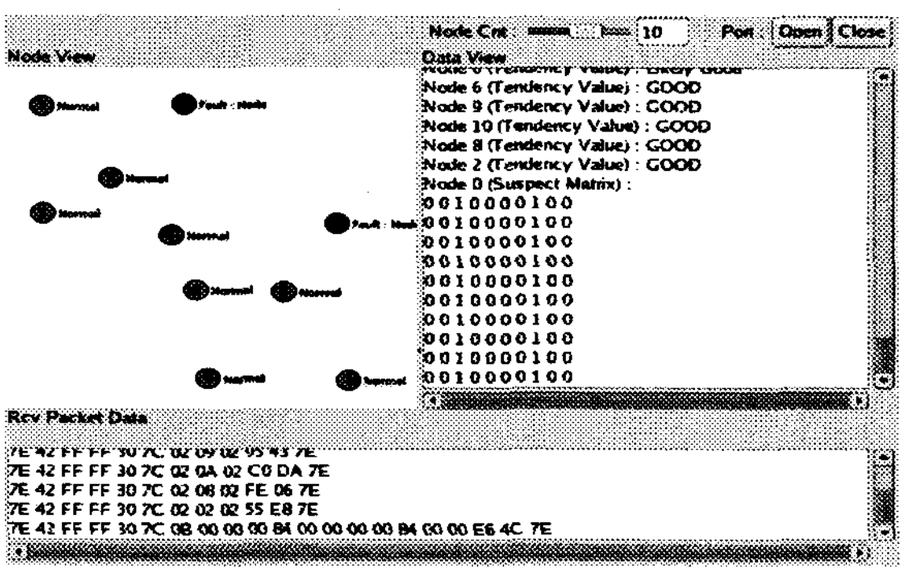

그림 10. 2번, 7번 노드 고장 검출 화면

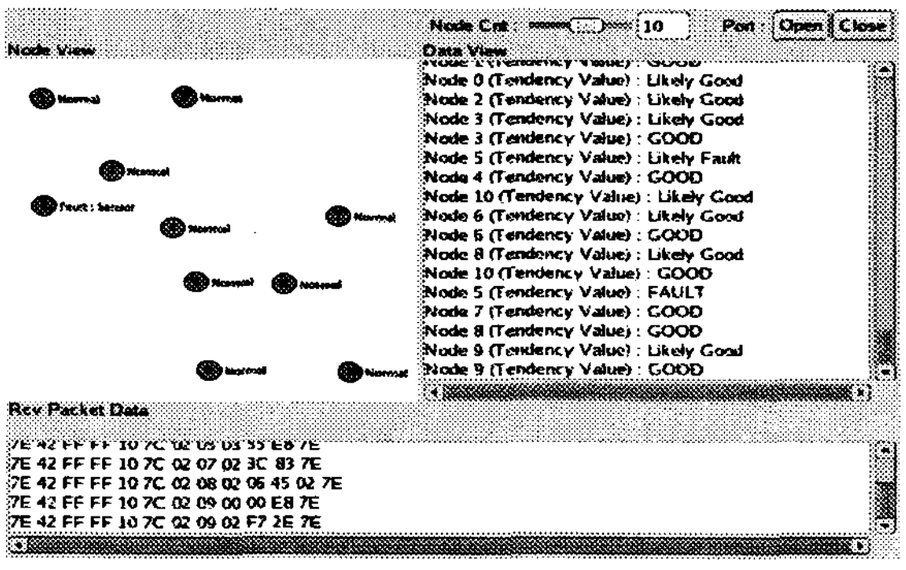

그림 11. 5번 센서 고장 검출 화면

6. 결론

센서 노드 상에서 고장은 크게 장치 고장과 센서 고장으로 나뉠 수 있다. 위 두가지 고장은 네트워크 마비나 데이터의 신뢰성 문제, 그 이외의 많은 문제들을 야기시키기 때문에 이를 해결하기 위한 연구가 다양하게 이루어지고 있다. 본 논문에서는 위 두 문제를 해결하기 위해 기존의 잘 알려진 Consensus 알고리즘과 Localized fault detection 알고리즘을 병합하여 장치에 고장 뿐만 아니라 센서의 고장까지 식별할 수 있는 알고리즘을 제안하고자 하였으며 Visual C++ 이용한 시뮬레이션과 실제 제작된 장치에서의 정상동작을 통해 그 유용성을 확인할 수 있었다.

본 연구는 산업자원부의 지역혁신 인력양성사업의 연구결과로 수행되었음

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.

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

[6] 이문노, 문정호, 정명진, "광 디스크 드라이버의 트랙킹 서보 시스템을 위한 다목적 강인 제어기의 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.