무선 센서 네트워크에서 비잔틴 오류를 허용하는 클럭 동기화 기법

(A Byzantine Fault-tolerant Clock Synchronization Scheme in Wireless Sensor Networks)

임 형 근 † 남 영 진 ††
(Hyung-Geun Lim) (Young Jin Nam)

백 장 운 † 고 석 영 †
(Jang Woon Baek) (Seok Young Ko)

서 대 화 †††
(Dae-Wha Seo)

요 약 본 논문에서는 무선 센서 네트워크에서 클럭 동기화 시 악의적인 노드의 클럭 동기화 방해 공격에 대처하기 위한 비잔틴 오류 감내 클럭 동기화 기법을 제안한다. 제안 기법은 클럭 동기화를 요구하는 노드가 m개의 악의적인 노드에 대처하기 위해 부모 노드뿐만 아니라 형제 노드로부터 $3m+1$개의 클럭 동기화 메시지를 수신하여 클럭 동기화를 진행한다. 시뮬레이터를 이용한 성능 평가를 통하여, 제안 기법은 기존 클럭 동기화 기법에 비하여 악의적인 노드의 클럭 동기화 방해 공격 시 동기 정확도 측면에서 최대 7배 향상된 성능을 보여주었다.

키워드 : 무선 센서 네트워크, 클럭 동기화, 비잔틴 오류

Abstract This paper proposes the Byzantine fault-tolerant clock synchronization scheme for wireless sensor networks to cope with the clock synchronization disturbance attack of malicious nodes. In the proposed scheme, a node which is requiring clock synchronization receives $3m+1$ clock synchronization messages not only from its parent nodes but also from its sibling nodes in order to tolerate malicious attacks even if up to m malicious nodes exist among them. The results show that the proposed scheme is 7 times more resilient to the clock synchronization disturbance attack of malicious nodes than existing schemes in terms of synchronization accuracy.

Key words : Wireless Sensor Networks, Clock Synchronization, Byzantine Fault

1. 서 론

센서 기술의 발달과 무선 통신 기술, 네트워크 기술, 초소형 운영체제 기술의 발달로 무선 센서 네트워크를 이용하는 응용분야가 확대되고 있다. 무선 센서 네트워크는 다수의 센서 노드와 하나 이상의 베이스 스테이션으로 구성된다. 센서 노드는 물리적 환경정보를 센싱하고 처리할 수 있으며, 멀티-홉(multi-hop) 통신을 통하여 베이스 스테이션으로 데이타를 전송할 수 있다. 사용자는 베이스 스테이션으로 전송된 센싱 데이타를 응용분야(군사, 환경, 의료 등)에 맞게 활용한다.

무선 센서 네트워크에서 동기 기반 통신 프로토콜 개발, 암호화 기술의 타임 스탬프, 이벤트 중복 감지 및 이벤트 발생 순서 구분 등 다양한 응용을 위해 센서 노드의 클럭 동기화 기술이 필수적이다[1]. 센서 노드는 센싱 데이타의 발생 시간을 결정하기 위해 로컬 클럭을 유지한다[2]. 로컬 클럭은 센서 노드의 수정 발진자 주파수에 의해 결정되며, 노드 주변의 환경적인 요인으로 인하여 변동될 수 있다[3]. 수정 발진자의 변동률에 의한 센서 노드의 클럭 차이를 보상해주기 위해 주기적인 클럭 동기화가 요구된다. 그리고 센서 네트워크의 활용 범위가 점점 넓어짐에 따라 악의적인 의도를 가진 공격자에 의한 데이타의 변질과 해킹이 증가될 것으로 예상된다[3]. 클럭 동기화 단계에서 악의적인 공격자에 의한 센서 노드의 클럭 정보 변질은 네트워크 전체의 클럭 동기화를 방해한다.

본 논문에서는 클럭 동기화 단계에서 수정발진자의 변동률을 고려한 주기적인 클럭 동기화를 수행하면서, 악의적인 노드(malicious node)에 의해 발생할 수 있는

· 본 연구는 2008년도 경북대학교 BK21사업과 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (IITA-2008-C1090-0801-0045)
· 이 논문은 2007 학술심포지움에서 '무선 센서 네트워크에서 비잔틴 오류를 허용하는 클럭 동기화 기법'의 제목으로 발표된 논문을 확장한 것임

† 비 회 원 : 경북대학교 전자전기컴퓨터학부
dreamjiny@ee.knu.ac.kr
jamti@ee.knu.ac.kr
kutc@ee.knu.ac.kr
†† 정 회 원 : 대구대학교 컴퓨터IT공학부 교수
yjnam@daegu.ac.kr
††† 종신회원 : 경북대학교 전자전기컴퓨터학부 교수
dwseo@ee.knu.ac.kr

논문접수 : 2008년 2월 4일
심사완료 : 2008년 3월 26일

정보과학회논문지 : 컴퓨팅의 실제 및 레터 제14권 제5호(2008.7)

비잔틴 오류(Byzantine fault)에 대처하는 클럭 동기화 기법을 제안한다. 본 논문의 구성은 다음과 같다. 2장에서는 센서 네트워크에서 클럭 동기화와 관련된 연구에 대해 논하고, 3장에서는 제안하는 BFCS(Byzantine Fault-tolerant Clock Synchronization) 기법에 대해 기술한다. 4장에서는 제안 기법의 성능을 분석하고, 5장에서 결론을 맺는다.

2. 관련 연구

기존의 센서 네트워크의 클럭 동기화에 관한 연구는 전파 지연 시간과 수정 발진자의 변동률에서 발생하는 로컬 클럭 에러률을 줄이는데 중점을 두었다. 센서 네트워크의 기존의 클럭 동기화 기법들을 동기 모델과 네트워크 토폴로지의 기준으로 분류할 수 있다[1]. 클럭 동기 모델에 따라 이벤트 순서화, 상대적 클럭, 보정된 클럭으로 나뉘어지며, 네트워크 토폴로지에 따라 트리, 클러스터, 메쉬로 다시 분류된다.

표 1 센서 네트워크 클럭 동기화 기법 분류

기법	클럭 동기 모델			네트워크 토폴로지			
	이벤트 순서화	상대적 클럭	보정된 클럭	평면	클러스터	트리	메쉬
Romer[4]	○			○			
RBS[5]		○		○	○		
LTS[6]		○				○	
TPSN[7]			○			○	
Tsync[8]			○			○	
FTSP[9]			○	○		○	
SRCS[3]			○				○

표 1에서 대부분의 클럭 동기화 기법들은 동기화 단계에서 발생할 수 있는 악의적인 노드의 동기화 방해 공격에 대한 고장 감내성을 고려하지 않았다.

센서 네트워크의 클럭 동기화를 진행하는데 있어 비잔틴 알고리즘을 활용하여 악의적인 노드에 대처하는 기법으로는 SRCS[3]가 있다. 비잔틴 알고리즘에 의하면 전체 네트워크에 연결된 노드 수를 N이라 하고, 그 중 오류를 가지는 노드의 수를 B이라고 했을 때, $N \geq 3B+1$을 만족하면 오류에 대처할 수 있다[10,11]. SRCS는 악의적인 노드 m개의 동기화 방해 공격에 대처하기 위해, 하나의 노드가 $3m+1$개의 클럭 동기화 메시지를 각기 다른 부모 노드들로부터 수신한다. $3m+1$개의 클럭 동기화 메시지를 받은 노드는 다수결의 원칙을 적용하여 고장을 일으킨 노드로부터 받은 동기화 메시지를 발견하고 제거한다. 그러나, 부모 노드에서 $3m+1$개의 클럭 동기화 메시지를 얻지 못 할 경우, 비잔틴 오류에 대처할 수 없고 올바른 클럭으로 동기화를 이룰 수 없다.

이러한 문제점을 해결하기 위하여 본 논문에서는 클럭 동기화를 요구하는 자식 노드가 상위에 존재하는 부모 노드들로부터 $3m+1$개의 클럭 동기화 메시지를 수신하지 못 할 경우, 자식 노드와 같은 레벨에 존재하는 형제 노드로부터 클럭 동기화 메시지를 수신하여, 비잔틴 오류에 대처하기 위한 나머지 클럭 동기화 메시지를 확보하여 동기화를 수행하는 기법을 제안한다.

3. 제안 클럭 동기화 기법

3.1 기본 동작과 가정

본 논문에서 제안하는 클럭 동기화 기법인 BFCS는 레벨이 할당된 메쉬 네트워크 구조에서 노드 간의 클럭 동기화는 그림 1와 같이 페어-와이즈 시각 동기화(pair-wised time synchronization) 기법[8]으로 진행한다. 노드 A는 노드 B에게 동기 시작 메시지(synchronization start message)를 자신의 아이디를 포함하여 전송한다. 이를 수신한 노드 B는 자신의 로컬 클럭 시간인 T1에 동기 요구 메시지(synchronization request message)를 노드 A로 전송한다. 노드 A는 자신의 로컬 클럭 시간인 T2에 노드 B에서 보내온 메시지를 받고 T3의 시간에 동기 응답 메시지(synchronization acknowledgment message)를 노드 B로 전송한다. 노드 B는 노드 A에서 보내온 동기 응답 메시지를 T4의 로컬 클럭 시간에 수신하게 된다.

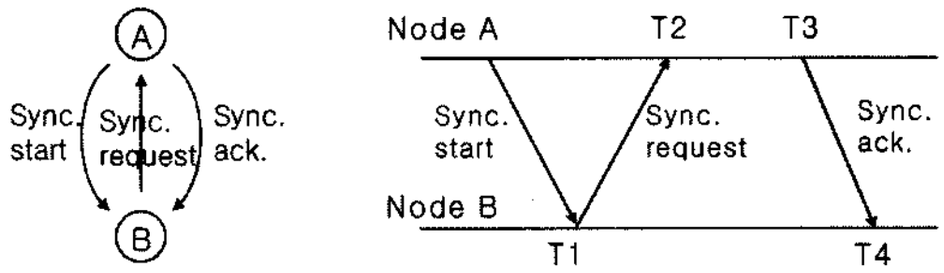

그림 1 페어-와이즈 시각 동기화 기법

이러한 과정을 통하여 오프셋 δ와 전파 지연 p를 얻는다. 식 (1)에 나타낸 δ는 노드 B가 노드 A에 대하여 시각 동기화를 이루기 위해 보정해야하는 로컬 클럭의 차이다. 식 (2)에 나타낸 p는 두 노드가 클럭 동기화 메시지를 교환할 때 네트워크 상에서 소모된 전파 지연 시간을 나타낸다.

$$\delta = \frac{(T2-T1)+(T3-T4)}{2} \quad (1)$$

$$p = \frac{(T4-T1)+(T3-T2)}{2} \quad (2)$$

단, 제안하는 BFCS는 루트 노드는 악의적인 노드가 아니며, 또한 정상적인 노드의 클럭 변화율이 ρ일 때에 악의적인 노드는 2ρ이상의 클럭 변화율을 갖는다고 가

정한다.

3.2 제안 알고리즘

각 노드에서 동작하는 BFCS 알고리즘은 그림 2와 같다. 레벨 0의 루트 노드가 동기 시작 메시지(SSM)를 레벨 1의 자식 노드들에게 송신하여 클럭 동기화가 시작되며, 레벨 i에 속한 노드와 레벨 $(i-1)$의 노드 간 페어-와즈 시각 동기화 기법을 수행한다. 기본적으로, 레벨 i의 클럭 동기화가 완료된 후에 레벨 $(i+1)$의 시각 동기화가 진행된다. 레벨 i에 속한 노드는 레벨 $(i-1)$에 속한 자신의 부모 노드(P_n)로부터 동기 시작 메시지(SSM)를 수신한다. 동기 시작 메시지를 수신한 후 이에 대한 응답으로 동기 요구 메시지(SRM)를 부모 노드에게 보낸다. 부모 노드는 동기 요구 메시지에 대한 응답 메시지(SAM)를 자식 노드에게 전송하고 자식 노드는 응답 메시지를 수신한다. $3m+1$개 이상의 부모로부터 응답 메시지를 수신한 경우, 응답 메시지에 포함된 오프셋들의 평균값을 구하여 평균값을 기준으로 가장 차이가 많이 나는 오프셋을 제거한 후에 나머지 오프셋 중 하나를 최종 오프셋으로 선택하여 자신의 로컬 클럭을 보정한다. 다음으로, 자신의 형제 노드(S_n)로 동기 시작 메시지(SSM)를 전송하며, 형제 노드로부터 동기 요구 메시지가 도착한 경우에 자신이 보정한 클록을 동기 응답 메시지에 담아서 보내준다. 끝으로, 자식 노드(C_n)에 동기 시작 메시지(SSM)를 송신한다. 만일 $3m+1$개 미만의 부모로부터 응답 메시지를 수신한 경우, 클럭 동기화를 정상적으로 마친 자신의 형제 노드(S_n)에서 보내온 동기 시작 메시지를 수신하여 $3m+1$개 이상의 응답 메시지를 확보할 수 있도록 한다. 부모 노드로부터 도착한 응답 메시지를 이용하여 클럭 동기화를 정상적으로 마친 경우에는 형제 노드로부터 도착하는 동기 시작 메시지는 무시한다.

```
// P_n = { P_i | i = 1, 2, ..., I }, parent list of node(n)
// C_n = { C_j | j = 1, 2, ..., J }, child list of node(n)
// S_n = { S_k | k = 1, 2, ..., K }, sibling list of node(n)
// m : the number of malicious nodes
// Synchronization Start Message(SSM)
// Synchronization Request Message(SRM)
// Synchronization Acknowledgement Message(SAM)
// level 0 : root node

procedure BFCS (node(n), level i)

    if i == 0 then // root node
        send SSM to C_n at level i + 1;
        receive SRM from C_n;
        send SAM to C_n;
    else // not root node
        receive SSM from P_n at level i - 1;
        send SRM to P_n;
        receive SAM from P_n;
        if (no. of received offsets from P_n < 3m + 1) then
            // try to receive more offsets from S_n
            do
                receive SSM from S_n;
                send SRM to S_n;
                receive SAM from S_n;
            while (no. of offsets < 3m + 1)
        endif

        calculate average offset;
        select final offset δ_n*;
        current clock = LC_n + δ_n*;

        // try to help its sibling node synchronize clock
        send SSM to S_n;
        receive SRM from S_n;
        send SAM to S_n;

        send SSM to C_n;

    endif

end BFCS
```

그림 2 제안 BFCS 알고리즘

3.3 알고리즘 적용 예

그림 3에서는 제안 알고리즘의 적용 예를 보여준다. 아래 예에서 m 값은 1로 설정 하였다. 노드 E는 동기화 단계에서 $3m+1$개의 부모 노드가 존재하지 않아, 부모 노드로부터 $3m+1$개의 오프셋을 확보하지 못한 경우이다. 이 경우, 일정 시간을 기다려 자신의 형제 노드에서 브로드케스트하는 동기 시작 메시지를 수신한다. 이 메시지를 수신하여 $3m+1$개의 오프셋을 계산한다. 그림 3에서 노드 E는 악의적인 노드 B를 포함하여 부모 노드로부터 $3m+1$개, 즉, 4개의 오프셋을 얻지 못한 경우에 해당한다. 노드 E는 부모 노드 A, B, C와 클럭 동기화 메시지 교환으로 각각의 오프셋 δ_{EA}, δ_{EB}, δ_{EC}을 구하였지만, 노드 D와 통신할 수 없어 레벨 1의 부모 노드들로부터 더 이상의 오프셋을 구할 수 없다. 그림 4은 $3m+1$개 미만의 오프셋을 부모 노드로부터 얻은 노드 E가 필요한 나머지 오프셋을 형제 노드를 통하여 얻는 과정을 나타낸 것이다. 그림 4의 (a)에서 노드 F는 부모 노드들로부터 $3m+1$개의 클럭 동기화 메시지를 교환하여 최종 오프셋(δ_F)을 선택하여 클럭 보정을 끝낸 상태이다.

레벨 2의 노드 F는 레벨 3으로 동기 시작 메시지를 보내기 전, 자신과 같은 레벨의 형제 노드들 중 $3m+1$개의 오프셋을 확보하지 못한 노드를 위해 동기 시작 메시지를 보낸다. 이 메시지를 형제 노드인 노드 E는 수신한다. 그림 4의 (b)에서 노드 E는 형제 노드 F에서 보낸 동기 시작 메시에 대한 응답으로 노드 F에게 동기 요구 메시지를 보낸다.

그림 5의 (a)는 노드 F가 노드 E에서 보낸 동기 요

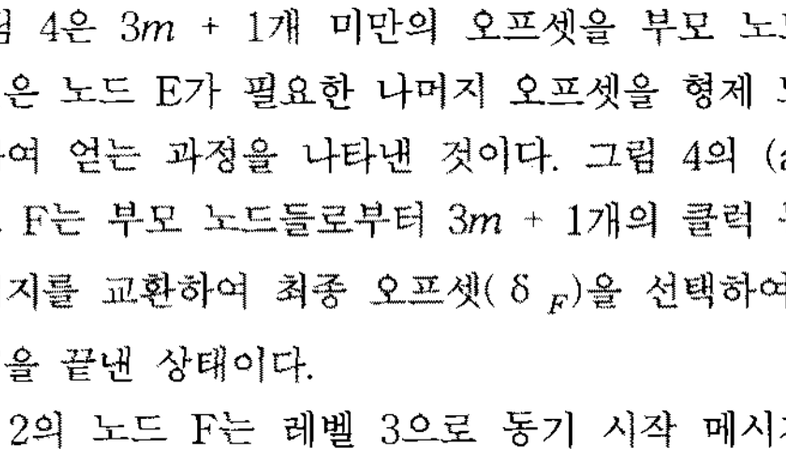

그림 3 오프셋 $3m+1$개를 확보하지 못한 경우

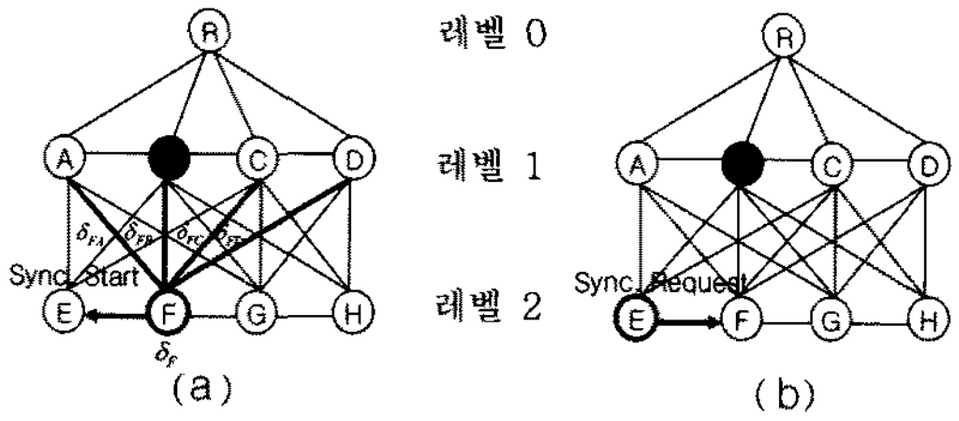

그림 4 노드 E의 동기 시작 메시지 청취와 동기 요구 메시지 송신

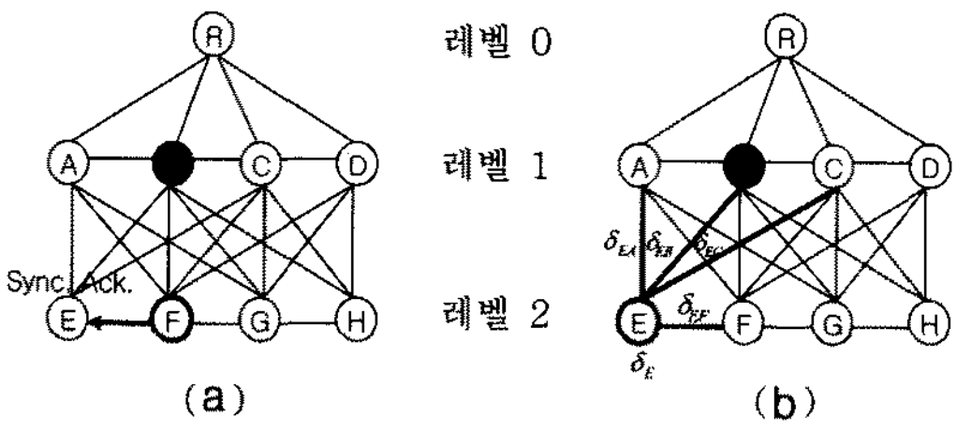

그림 5 노드 F의 응답과 노드 E의 $3m+1$ 오프셋 확보

구 메시지에 대한 응답 메시지를 노드 E에게 송신하는 것을 나타낸다. 노드 E는 노드 F에 대한 오프셋 δ_{EF}를 계산한다. 그림 5의 (b)에서 노드 E는 노드 F와 클럭 동기화 메시지 교환으로 얻은 오프셋 δ_{EF}과 이전에 부모 노드 A, B, C를 통해 얻은 오프셋 δ_{EA}, δ_{EB}, δ_{EC}를 포함하여, $3m+1$개의 오프셋으로 비잔틴 오류에 대처할 수 있다.

노드 E는 4개의 오프셋에 대하여 각각의 절대값을 구하고 오프셋의 합을 구한다. 이 합의 평균을 내어 평균값을 중심으로 가장 큰 오차를 가지는 하나의 오프셋를 제외하고 나머지 오프셋들 중 하나를 최종 오프셋 δ_E로 선택하고, 이를 노드 E의 로컬 클럭에 합하여 자신의 로컬 클럭을 보정한다.

4. 성능 평가

제안 클럭 동기화 기법의 성능을 평가하기 위해 TinyOS기반의 TOSSIM 시뮬레이터에서 시뮬레이션 환경을 구축하였다. 다음의 표 2에 시뮬레이션 파라미터들을 나타내었다.

표 2에서 악의적인 노드 존재율은 네트워크에 분포된 총 노드 중에 존재하는 악의적인 노드의 비율을 나타낸다. 로컬 클럭의 시간 단위인 mtick(minutes tick)과 stick(second tick)에서 stick은 0~65,535까지의 범위를 가지며, 1 mtick은 65,536 stick이다. 1 stick은 25μs에 해당한다. 허용 오차 범위는 수정 발진자에 의한 노드의 클럭 변화율을 고려하기 위한 값이다. 각 노드가 클럭동기화 메시지를 교환하여 오프셋을 구할 때, 이 범위에 속한 노드는 정상적인 노드로 간주한다.

표 2 시뮬레이션 파라미터

파라미터	설정값
총 노드 수	50~150 개
악의적인 노드 존재율	0~60%
주 기	10 mtick
정상적인 노드의 클럭 변화율	0~25 stick
악의적인 노드의 클럭 변화율	51~65,535 stick
허용 오차 범위	50 stick

그림 6은 네트워크에 분포된 총 노드 개수가 100개일 때, 악의적인 노드 개수의 변화에 따른 각 기법의 평균 로컬 클럭 에러와 메시지 발생량을 비교한 것이다. TPSN의 경우, 평균 로컬 클럭 에러가 악의적인 노드가 증가함에 따라 다른 두 기법에 비해 상대적으로 많이 발생하는 것을 알 수 있다. 그러나, SRCS와 제안한 BFCS는 악의적인 노드의 방해 공격에 대처하고 있음을 보인다. 특히, 제안한 BFCS는 SRCS에 비하여 악의적인 노드 수가 30%일 때, 최대 7배의 적은 수치로 정확도가 향상된 것을 보인다. 메시지 발생량 측면에서는 TPSN의 경우, 하나의 부모 노드로부터 클럭 동기화 메시지를 얻기 때문에 3,767 packet으로 일정한 수치를 보이며, 나머지 두 기법과 비교하여 상대적으로 적은 수치를 나타낸다. 제안한 BFCS와 SRCS를 비교하면, 처음 악의적인 노드가 없을 때는 메시지 발생량에 있어서 동일하지만, 악의적인 노드가 증가함에 따라 BFCS가 SRCS에 비해 약 1000개 이상의 증가된 수치를 나타낸다. 이는 SRCS에 비해 BFCS가 악의적인 노드가 증가할수록 높은 확률로 주위 형제 노드로부터 클럭 동기화 메시지를 얻어 비잔틴 오류에 대처하기 때문이다.

그림 7은 네트워크에 분포된 노드 중 악의적인 노드의 개수가 10개 존재할 때, 네트워크의 총 노드 개수의 변화에 따른 평균 로컬 클럭 에러와 메시지 발생량을 비교한 것이다. 총 노드 수가 증가함에 따라 비잔틴 오류에 대처하는 두 기법인 SRCS와 제안한 BFCS가 평균 로컬 클럭 에러의 감소률이 TPSN에 비해 크게 나타났다. TPSN의 경우 비잔틴 오류에 대처할 수 없음으로 악의적인 노드에 의한 클럭 오류가 전체 네트워크로 전파되기 때문에 다른 두 기법에 비해 평균 로컬 클럭 에러의 수치가 크게 나타났다. 메시지 발생량 측면에서 TPSN의 경우 노드 개수의 증가에 따라 메시지 발생량의 증가율이 SRCS와 BFCS에 비해 적게 나타났다. 이는 SRCS와 BFCS가 비잔틴 오류에 대처하기 위해 나타나는 증가된 수치이다. 제안한 BFCS와 SRCS를 비교하면 약 1,000개의 수치를 BFCS가 더 높게 나타내었다. 이는 오류에 대처하기 위해 형제 노드를 이용하는데서 오는 증가량이다.

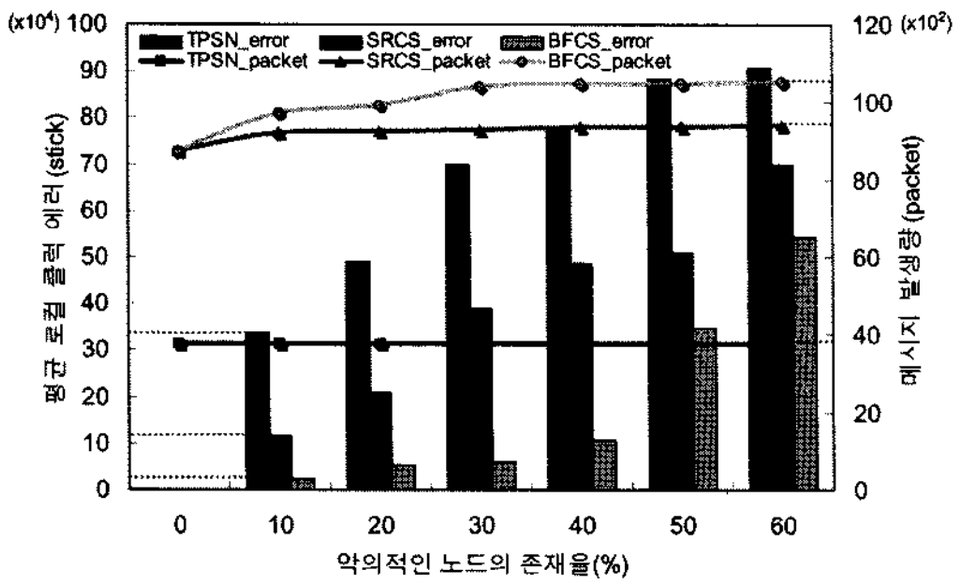

그림 6 악의적인 노드 개수 변화에 따른 평균 로컬 클럭 에러와 메시지 발생량 비교

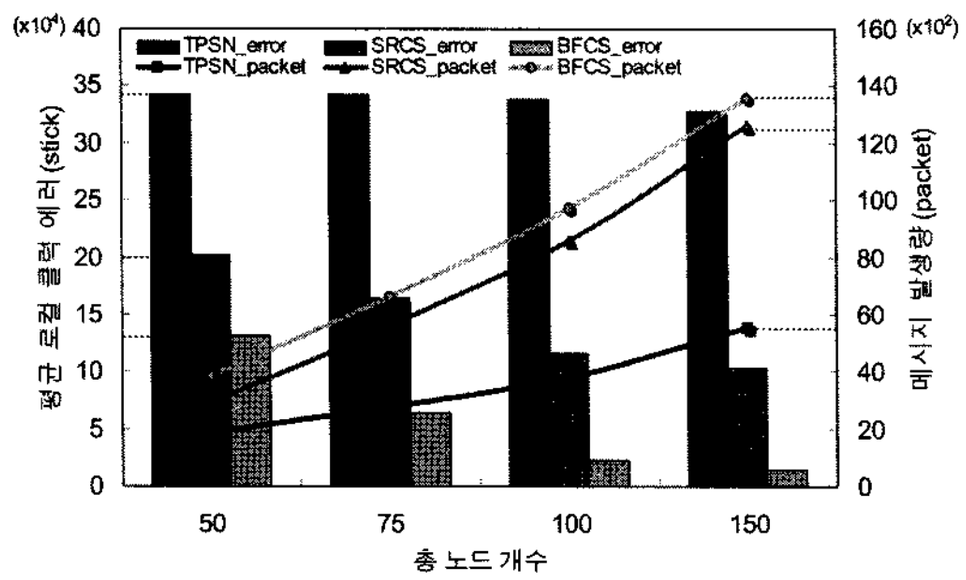

그림 7 총 노드 개수의 변화에 따른 평균 로컬 클럭 에러와 메시지 발생량 비교

5. 결 론

본 논문에서는 센서 네트워크에서 악의적인 노드에 의해 발생하는 비잔틴 오류에 대처할 수 있는 클럭 동기화 기법을 제시하였다. 제안한 클럭 동기화 기법에서는 동기화를 요구하는 임의의 노드가 부모 노드뿐만 아니라 형제 노드를 이용하여 $3m + 1$개의 클럭 동기화 메시지를 확보하여 비잔틴 오류에 대처한다. 시뮬레이터를 통한 성능 평가 결과, 제시한 클럭 동기화 기법인 BFCS는 기존의 클럭 동기화 기법인 SRCS와 비교하여 악의적인 노드의 동기화 방해 공격 시 동기 정확도 측면에서 최대 7배 향상된 결과를 보였다.

참 고 문 헌

[1] 황소영, 정연수, 백윤주, “센서 네트워크에서 신뢰성 있는 시각 동기 프로토콜”, 한국통신학회논문지, 제31권, 제3A호, pp. 274-281, 2006.

[2] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock Synchronization for Wireless Sensor Network : A Survey," Ad-Hoc Networks, Vol.3, No.3, pp. 281-323, May 2005.

[3] K. Sun, P. Ning, and C. Wang, "Secure and Resilient Clock Synchronization in Wireless Sensor Network," IEEE Journal on Selected Areas in Communications, Vol.24, pp. 395-408, Feb. 2006.

[4] K. Romer, "Time Synchronization in Ad Hoc Networks," Proc. of the 2nd ACM international Symposium on Mobile Ad Hoc Networking & Computing, pp. 173-182, Mar. 2005.

[5] J. Elson, L. Girod, and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts," Proc. of the 5th ACM Symposium on Operating Systems Design and Implementation, pp. 147-163, Dec. 2002.

[6] J. Greunen and J. Rabaey, "Lightweight Time Synchronization for Sensor Networks," Proc. of the 2nd ACM International Conference on Wireless Sensor Networks and Applications, pp. 11-19, Sep. 2003.

[7] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync Protocol for Sensor Networks," Proc. of the 1th ACM International Conference on Embedded Networked Sensor Systems, pp. 138-149, Nov. 2003.

[8] H. Dai and R. Han, "Tsync : A Lightweight Bidirectional Time Synchronization Service for Wireless Sensor Networks," ACM Mobile Computing and Communication Review, Vol.8, No.1, pp. 125-139, Nov. 2003.

[9] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The Flooding Time Synchronization Protocol," Proc. of the 2nd International Conference on Embedded Networked Sensor Systems, pp. 39-49, Nov. 2004.

[10] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," ACM Transaction on Programming Languages and Systems, Vol.4, pp. 382-401, Jul. 1982.

[11] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," Proc. of the 3rd Symposium on Operating System Design and Implementation, pp. 173-186, Feb. 1999.