적은 메모리 사용량을 가진 센서노드용 대용량 낸드 플래시 파일 시스템의 설계

한경훈° 이기혁 송준영 한형진 최원철 한지연 손기락
한국 외국어 대학교 컴퓨터 및 정보통신 공학부
{myway767°, knockin81, sjyking, nom96ny, wc.choi, hanjiyeon, ksohn}@hufs.ac.kr

Design of a High-capacity NAND Flash based File System for Sensor Node with very small Memory Footprint

Kyoung-hoon Han°, Ki-hyuk Lee, Jun-young Song, Hyung-jin Han, Won-chul Choi, Ji-yean Han
Kirack Sohn
Hankuk University of Foreign Studies

요 약

최근에 에너지의 효율성이 좋고 대용량화가 쉬운 낸드 플래시가 센서 노드를 위한 차세대 저장소로 각광을 받고 있다. 현재 대부분의 센서 노드용 파일 시스템은 노어 플래시 기반으로 개발되어 있으며 낸드 플래시에 적용할 수 있는 파일 시스템은 거의 존재하지 않는다. 대용량 낸드 플래시 메모리의 특성을 고려한 새로운 파일 시스템의 구축이 요구되지만, 센서 노드는 오직 4~10 KByte의 매우 작은 크기의 메모리를 지원하므로 효율성이 뛰어난 파일 시스템을 구축하는 것은 매우 어렵다. 본 논문은 1 Kbit의 매우 작은 크기의 EEPROM을 부착하여 이러한 메모리 한계를 극복하였으며 자원의 효율성, 대용량의 지원 및 신뢰성을 고려한 새로운 파일 시스템의 설계에 대하여 논한다. 위치를 유지해야 하는 데이터의 위치저장을 위하여 EEPROM을 사용하며 장기간 데이터를 수집할 때 페이지의 갱신을 최소화 할 수 있는 로그 리스트 기반의 페이지 처리 방법에 대해 제안한다. 이는 획기적으로 페이지 갱신 횟수를 줄임으로써 에너지를 절약하고 보다 긴 시간동안 데이터의 수집을 용이하게 만들며 센서 노드의 수명을 증가시킨다.

1. 서 론

최근 각광받고 있는 센서 네트워크 기술은 주변의 빛, 소리, 온도, 습도 등의 물리적 데이터의 수집을 바탕으로 환경, 의료, IT 등의 다양한 분야에 적용되고 있다. 점차 데이터의 수집 분야가 다양해지고 장기간 수집을 목적으로 사용되는 경우가 늘어나면서 대량의 데이터를 저장하고 관리하기 위한 효율적인 저장장치가 요구되고 있다.

현재 센서 노드의 저장소로는 플래시 메모리가 많이 사용되고 있다. 플래시 메모리는 최근에 주목받는 반도체 기반의 데이터 저장매체로서 비휘발성(non-volatile)이고 하드디스크와 같은 보조기억 장치보다 실행속도가 훨씬 빠르다는 장점이 있다. 또한 전력소모가 매우 적으며, 물리적 충격에 강하기 때문에 센서 노드용 저장소로 가장 적합하다. 하지만 기존 데이터의 덮어쓰기가 불가능 하며 쓰기 횟수의 제한이 있다. 이는 파일 시스템의 성능의 한계와 알고리즘 개발에 제약을 가지게 된다[1].

플래시 메모리는 낸드 플래시와 노어 플래시로 구분된다. 노어 플래시는 Read 속도가 빠르고 데이터의 안정성이 뛰어나다는 장점이 있지만 가격이 비싸고 회로가 복잡하여 대용량화하기가 힘들다. 낸드 플래시는 노어 플래시와 비교하여 Write 및 Erase속도가 월등히 빠르며 최근 단가가 낮아지고 대용량화되고 있는 추세이므로 앞으로의 센서 노드용 저장소로 적합할 것으로 예상된다. 하지만 현재 대부분의 센서 노드용 파일 시스템은 노어 플래시 기반으로 개발되었기 때문에 낸드 플래시에 적용할 수 있는 파일 시스템은 거의 없다.

대용량 낸드 플래시의 특성을 고려한 새로운 파일 시스템의 구축이 요구되지만 센서 노드는 오직 4~10 KByte의 매우 작은 크기의 메모리를 지원하므로 효율성이 뛰어난 파일 시스템을 구축하는 것은 매우 어렵다. 본 논문은 1 Kbit의 매우 작은 크기의 EEPROM을 부착하여 이러한 메모리 한계를 극복하였으며 자원의 효율성, 대용량 지원 및 신뢰성을 고려한 새로운 파일 시스템의 설계에 대하여 논한다. 그리고 본 파일 시스템은 시뮬레이터를 통해 구현하였다.

본 연구는 21 세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅및네트워크원천기술개발사업의 지원에 의한 것임

2. 관련 연구

기존 플래시 메모리 기반의 센서 노드용 파일 시스템은

센서 노드라는 장치적 제약 때문에 그 수가 많지 않다. 대표적인 파일 시스템으로는 Matchbox, ELF, Capsule이 있으며, 효율적인 파일 시스템의 설계를 위하여 기존 파일 시스템의 특징을 비교해 보았다.

표 1. 센서노드용 플래시 메모리 파일 시스템 특징 비교

구 분	Matchbox	ELF	Capsule
저장 장치	NOR	NOR	NOR, NAND
에너지 최적화	No	No	Yes
메모리 최적화	Yes	Yes	Yes
자원 평준화	No	Yes	Yes
Crash Recovery	No	Snapshot	Checkpointing
추상화	Filesystem	Filesystem	Object
EEPROM 사용		Metadata Index	

2.1 Matchbox

Matchbox[2]는 TinyOS에서 기본적으로 제공하는 파일 시스템이다. 노어 플래시 기반에서 설계되었으며 오직 메모리 최적화에 대한 측면만 고려되었다. 에너지 효율성과 자원 평준화는 고려되지 않았고 Crash Recovery에 대한 정책은 없다. 노어 플래시 기반이지만 Write-append, Seek 명령을 지원하지 않으며 Free 페이지 리스트를 메인 메모리에서 관리하므로 전원이 소실되면 부팅 시간에 재구성한다.

2.2 ELF

ELF[3]는 Matchbox와 같이 노어 플래시 기반에서 설계되었으며 노어 플래시의 특징을 살려 Write-modify 연산과 Seek 기능을 지원한다. 메모리 최적화와 자원 평준화를 고려했으며 Crash Recovery에 대한 정책으로 Snapshot을 사용한다. 유일하게 EEPROM을 사용하며 디렉토리와 파일의 메타데이터와 인덱스를 저장한다. 하지만 Atmel 데이터 플래시 장치에 특성화하여 구현되었기 때문에 다른 장치를 대상으로 사용할 수 없으며 버퍼를 순환 형태로 사용하여 센서 네트워크에서 사용하는 다양한 응용 프로그램들을 만족시키지 못한다.

2.3 Capsule

Capsule[4]은 노어 플래시에서 낸드 플래시의 특징을 살려 구현되었다. 에너지 최적화, 메모리 최적화, 자원 평준화를 모두 고려하였으며 Crash Recovery 정책으로 Checkpointing을 사용한다. 파일 기반이 아닌 객체 기반으로 스택, 큐, 인덱스 등의 다양한 자료 구조를 저장 시스템 단위에서 지원한다. 하지만 인덱스는 컴파일시간에 크기가 고정되므로 파일의 크기나 Fragment 수가 제한된다. 또한 Backward Chaining을 사용하므로 Write-modify와 Read연산이 매우 비효율 적이다. 이러한 이유 때문에 유일하게 낸드 플래시에서 동작되도록 설계되었지만 대용량 낸드 플래시 파일 시스템으로는 적합하지 않다.

3. 하드웨어 환경

고성능의 파일 시스템을 설계하기 위해서는 효율적인 알고리즘과 적절한 크기의 저장매체의 조합이 매우 중요하다. 파일 시스템의 요구사항에 대해 하드웨어를 적절히 배분하여 전체 시스템을 설계함으로써 보다 우수한 성능의 파일 시스템을 설계할 수 있다. 본 파일 시스템은 저 전력을 목적으로 제작된 TIP시리즈의 TIP810CM Mote를 기반으로 설계하였으며 그림 1은 TIP810CM Mote에 외부 플래시 메모리를 부착한 사진이다[5].

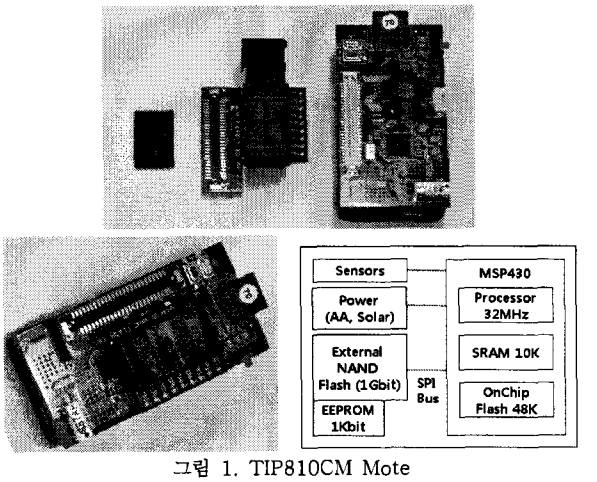

그림 1. TIP810CM Mote

3.1 SRAM 10 KByte

TIP810CM은 10 KByte의 SRAM을 제공한다. SRAM은 파일 시스템 뿐만 아니라 응용프로그램에서도 사용하므로 매우 작은 크기이다. 본 파일 시스템에서는 10 KByte 중에 2112 Byte를 4개의 버퍼로 사용한다. 버퍼 하나의 크기는 528 Byte이다. 첫 번째 버퍼는 파일 디스크립션 맵의 루트 페이지 전용으로 할당한다. 파일 디스크립션은 파일에 대한 메타 정보이고 파일 디스크립션 맵은 파일 디스크립션의 집합이다.

SRAM의 크기가 매우 작기 때문에 동시에 Open가능한 파일의 수를 2개로 제한시켰다. 파일이 Open될 경우 빈 버퍼를 점유하여 Close될 때 까지 사용한다.

3.2 낸드 플래시 1 Gbit

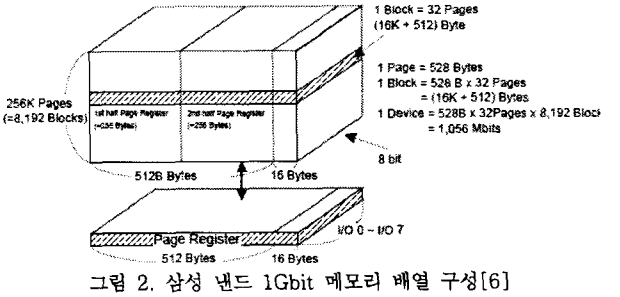

그림 2. 삼성 낸드 1Gbit 메모리 배열 구성[6]

저장소로는 삼성 1Gbit 낸드 플래시를 외부 메모리에 부착하였다. 삼성 1Gbit 낸드 플래시는 스몰 페이지 방식이며, 전체 블록수는 8192개 이고 블록당 페이지 수는 32개이다. 한 페이지는 528 Byte이고 512 Byte의 데이터 영역과 16 Byte의 스페어 영역으로 구성된다. 데이터 영역은 오직 한번만 기록할 수 있고, 스페어 영역은 두 번 기록할 수 있다[6]. 이곳에는 파일과 파일 디스크립션 맵이 저장된다.

3.3 EEPROM 1 Kbit

낸드 플래시는 동일한 페이지에 덮어쓰기가 불가능 하므로 특정 위치에 데이터를 보관할 수 없다. 이로 인해 데이터가 갱신될 때 마다 매번 그 위치가 변한다. 그래서 기존에 존재하는 대부분의 파일 시스템에서는 부팅타임에 전체 플래시 메모리를 스캔해서 필요한 정보들을 SRAM에 구성하여 사용한다. 하지만 플래시 메모리는 점점 대용량화되고 있기 때문에 매번 전체 메모리를 모두 스캔하기에는 시간과 전력 면에서 한계가 있다.

본 파일 시스템의 설계에서는 EEPROM을 사용하고, 1 Kbit를 외부 메모리로 부착했다. EEPROM은 동일한 위치에 덮어쓰기가 가능하고 비휘발성이므로 필요한 데이터의 위치를 기록하는데 적합하다. 즉, 부팅 시간에 플래시 메모리 전체를 스캔하지 않고 EEPROM의 특정 위치만 확인하면 필요한 정보를 얻을 수 있다.

표 3. EEPROM에 기록할 데이터 목록

목 적	할당 Byte
파일 디스크립션 맵의 루트 페이지 주소	3 Byte
마지막에 할당한 Free 페이지 번호	3 Byte
다음에 삭제할 Erase 블록 번호	2 Byte
마지막에 할당한 파일 ID	2 Byte
Open파일1 정보 (파일ID + 루트 주소)	5 Byte
Open파일2 정보 (파일ID + 루트 주소)	5 Byte
로그 리스트	6*12 = 72 Byte
로그 리스트 개수	1 Byte
유효 리스트	3*5 = 15 Byte
유효 리스트 개수	1 Byte
Garbage 리스트	3*5 = 15 Byte
Garbage 리스트 개수	1 Byte
총 계	125 Byte

표 3은 EEPROM에 기록하는 세부 목록이다. 총 125 Byte를 사용하는데 그중 3 Byte는 파일 디스크립션 맵의 루트 페이지의 주소를 저장하고, 3 Byte는 마지막에 할당한 Free 페이지의 번호를 기록해 놓으며 이 번호로부터 순차적으로 Free 페이지가 부여된다. 2 Byte는 다음에 삭제할 Erase 블록 번호를 기록해 놓으며 플래시 메모리가 70%이상 꽉 찼을 경우 이 위치로부터 블록 Erase를 실시한다. 10 Byte는 Open된 파일 2개의 정보를 기록하며 72 Byte는 발생한 파일 연산에 대한 로그 리스트를 기록한다. 15 Byte는 새롭게 할당된 페이지에 대한 유효 리스트를 기록하며, 나머지 15 Byte는 이전 페이지에 대한 Garbage 리스트를 기록한다.

4. 낸드 플래시 파일 시스템의 구조

4.1 페이지의 상태

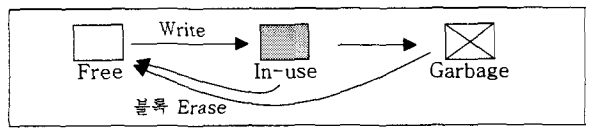

그림 3. 페이지의 상태변화

페이지는 Free, In-use, Garbage 세가지 상태가 존재한다. Free는 페이지가 비어있음을 나타내며, 이때 Write가 가능하다. In-use는 Write연산으로 값이 기록되어졌고, 현재 파일이나 파일 디스크립션 맵에서 사용중인 페이지를 나타낸다. Garbage는 사용 되었다가 지금은 사용되지 않는 페이지를 나타내며, 블록 Erase 후 Free상태가 된다.

4.2 페이지의 소속부여

파일에 속한 페이지는 헤더에 다음과 같은 소속을 기록한다.

- 속성 : 파일이라고 표시한다.
- 파일 ID : 어떤 파일에 속하는지를 기록한다.
- 레벨 : 트리의 몇 번째 레벨에 속하는지 기록한다.
- 형제번호 : 해당 레벨의 몇 번째 페이지인지 기록한다.

파일 디스크립션 맵에 속한 페이지는 다음과 같은 소속을 기록한다.

- 속성 : 파일 디스크립션 맵이라고 표시한다.
- 레벨 : 트리의 몇 번째 레벨에 속하는지 기록한다.
- 형제번호 : 해당 레벨의 몇 번째 페이지인지 기록한다.

4.3 파일의 자료구조

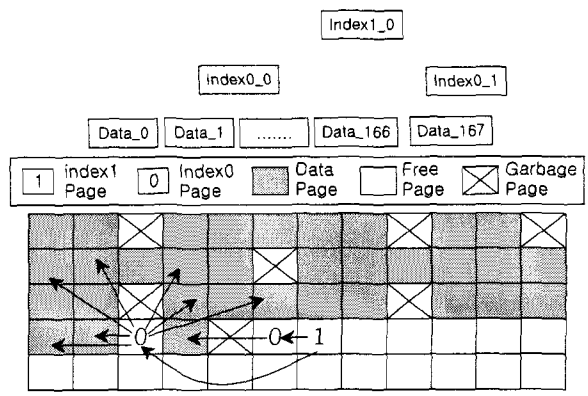

그림 4. 파일의 자료구조

파일은 100% Full의 B+ 트리를 응용하여 구성한다. 여기에는 실제 데이터를 저장하는 데이터 페이지와 자식의 주소를 갖는 인덱스 페이지가 존재한다. 데이터 페이지는 처음 9 Byte는 헤더를 저장하며 503 Byte는 데이터를 저장한다. 인덱스 페이지는 9 Byte는 헤더를 저장하며 501

Byte는 자식의 주소를 순차적으로 기록하는데, 낸드 플래시는 페이지의 주소를 3 Byte로 표현하므로 총 167개를 기록할 수 있다. 헤더에는 페이지의 소속과 자신으로부터 연결된 데이터 페이지의 크기의 합을 기록한다. 그림 4는 파일의 구조가 낸드 플래시에 맵핑된 그림이다.

4.4 파일 디스크립션 맵의 자료구조

파일 디스크립션은 파일의 메타 데이터이다. 20 Byte로 표현되며 파일의 이름, ID, Write 속성, 루트 페이지를 기록한다. 파일의 이름은 최대 14 Byte까지 지원한다. ID는 파일마다 고유하게 부여되며 1~65535 범위의 숫자가 Wrap-around방식으로 부여된다. Write 속성은 파일을 Write-append방식으로 사용할 것인지 Write-modify방식으로 사용할 것인지를 기록한다. 현재는 Write-append만 지원하지만 후에 Write-modify를 지원할 것이므로 미리 1 Byte를 할당했다. 루트 페이지는 파일의 최상위 레벨의 페이지 주소를 기록한다.

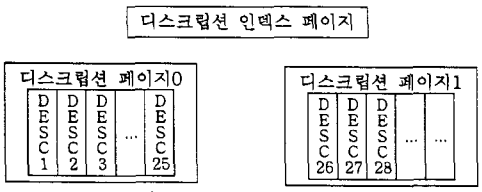

그림 5. 파일 디스크립션 맵의 자료구조

센서 네트워크의 특성상 파일의 개수가 많지 않기 때문에 파일 디스크립션 맵은 일반적인 트리로 구성한다. 디스크립션 페이지는 3 Byte는 헤더를 기록하며 500 Byte는 디스크립션을 기록한다. 디스크립션의 크기는 20 Byte이므로 한 페이지당 25개의 디스크립션을 기록할 수 있다. 디스크립션 인덱스 페이지는 4 Byte는 헤더를 기록하며 507 Byte는 디스크립션 페이지의 주소를 기록한다. 헤더에는 페이지의 소속과 자신으로부터 연결된 디스크립션의 개수의 합을 기록한다.

5. 낸드 플래시 메모리 특성을 고려한 파일 시스템 구성 알고리즘

5.1 에너지와 메모리 절약을 위한 로그 리스트의 사용

낸드 플래시는 동일한 페이지에 덮어쓰기가 불가능하므로 페이지의 추가 시에 추가된 페이지를 참조해야 하는 모든 페이지가 수정되어야 한다. 예를 들어 그림 6에서 168번째 데이터 페이지를 추가할 경우 파일의 인덱스 페이지 두 개가 갱신되어야 하며, 파일 디스크립션 맵의 두 개의 페이지가 갱신되어야 한다. 한 페이지를 추가할 때 마다 일어나는 4~5개의 불가피한 페이지 갱신으로 인해 많은 에너지와 메모리 소비가 발생한다. 본 논문에서는 로그 리스트를 사용하여 갱신되는 페이지의 수를 최소화 하는 방법에 대하여 제안한다.

페이지가 추가되면 EEPROM에 로그를 기록하고 관련있

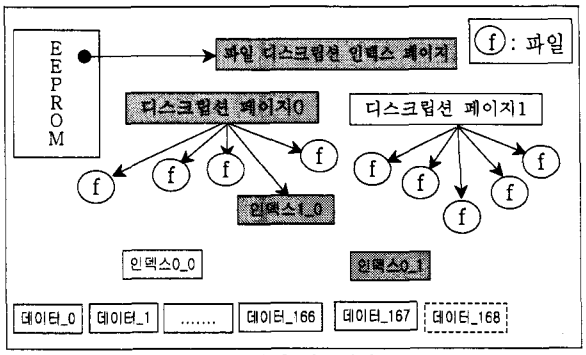

그림 6. 파일 시스템의 구조

는 페이지들은 바로 갱신하지 않는다. 로그 리스트가 꽉 차거나 연산이 종료되었을 때 비로소 파일의 연관된 페이지들을 갱신한다. 변경된 파일의 루트 페이지를 EEPROM에 기록하며, 파일 디스크립션 맵은 파일을 닫을 때 한번 갱신한다.

로그 리스트를 사용하면 로그 리스트를 사용하지 않을 때 보다 약 3~4배 이상의 메모리와 에너지를 절약할 수 있다. 이는 보다 긴 시간동안 데이터의 수집을 가능하게 하며 센서 노드의 수명을 10%이상 증가시킨다.

5.2 신뢰성 보장을 위한 유효 리스트와 Garbage 리스트 유지

파일 시스템은 작업 도중 전원이 소실될 수 있기 때문에 새 데이터 혹은 이전 데이터의 보존여부와 관계없이 항상 신뢰성을 보장해야 한다. 이로 인한 대책으로 유효 리스트와 Garbage 리스트가 사용된다. 페이지가 갱신되는 연산이 발생했을 때 새로운 페이지를 할당하면 유효 리스트에 기록한다. 이전 데이터를 갖고 있는 페이지는 Garbage 리스트에 기록한다. 정상적으로 연산이 종료되었을 경우 유효 리스트를 비우고 Garbage 리스트에 있는 모든 페이지를 Garbage시킨다. 비정상적으로 종료되었을 경우 유효 리스트를 모두 Garbage시키고 Garbage 리스트를 비운다.

5.3 메모리 관리를 위한 블록 Erase

낸드 플래시는 Erase된 블록에만 Write할 수 있으므로 파일 시스템의 사용량이 70%이상 되었을 경우 블록 Erase를 실시한다. Erase 대상인 블록에 In-use페이지가 존재하면 그 페이지를 다른 곳으로 옮겨야 하고 그것을 참조하고 있는 모든 페이지도 옮겨야 한다. 페이지가 어디에 소속되어 있는지 알 수 있는 정보가 없다면 파일 디스크립션 맵과 모든 파일을 검색해야한다. 소속된 정보로 파일 ID만 존재한다면 파일의 모든 인덱스 페이지를 검색하고 검색했던 경로를 기억해야 한다. 이는 시간이 오래 걸릴 뿐만 아니라 에너지 소비 또한 매우 크다.

본 파일 시스템에서는 모든 페이지에 파일 ID, 레벨번호, 형제번호를 부여함으로써 페이지 검색 시간을 획기적으로 개선하였다. 이것을 분석하면 자신의 위치뿐만 아니라 자신을 참조하고 있는 모든 페이지의 주소를 파악할 수 있다.

모든 In-use 페이지에 대해 소속을 추출하고 동일한 소속을 갖는 데이터 페이지들을 먼저 다른 페이지로 복사한다. 페이지 복사가 끝나면 그것들을 참조하고 있던 인덱스 페이지들을 갱신한다. 만약 소속된 곳이 존재하지 않는다면 Delete연산으로 인해 지워진 파일에 속한 페이지이므로 무시한다. In-use 페이지가 모두 다른 페이지로 옮겨지면 해당 블록을 Erase한다.

5.4 배드 블록 관리

낸드 플래시는 노어와는 달리 공장 출하 시에 배드 블록을 가지고 있을 수 있다. 이를 Factory 배드 블록이라고 하며 삼성 낸드 플래시의 경우 해당 블록의 0번 페이지의 스페어 영역 5번 Byte에 배드 블록 표시를 해놓는다. Factory 배드 블록은 데이터를 쓸 때 성공할 수 있지만 다음번 읽었을 경우 엉뚱한 데이터를 출력할 수 있으므로 절대 사용해서는 안된다. 실수로 배드 블록을 Erase하여 표시가 초기화 되었을 경우 다시 되돌릴 수 없으므로 블록을 Erase하기 전에 항상 확인해야 한다.

또한 실행시간에 배드 블록이 발생할 수 있는데 본 파일 시스템에서는 어떤 페이지에 대해 Read/Write연산을 수행하였는데 실패했을 경우 배드 블록이 발생했다고 간주한다. Factory 배드 블록을 발견했거나 실행시간에 배드 블록이 발생했을 경우 해당 블록의 In-use페이지를 다른 곳으로 옮기고 해당 블록의 31번 페이지의 14번 Byte에 배드 블록 표시를 한다. 이후 31번 페이지에 배드 블록 표시가 되어있거나 31번 페이지를 읽는데 실패했을 경우 그 블록은 배드 블록이므로 사용하지 않는다.

5.5 부팅 타임

부팅 타임에는 파일 시스템이 구동하는데 필요한 데이터를 버퍼에 올리고 이전에 센서가 비정상적으로 종료되어 미완료된 작업 내용을 확인한 후에 그것을 처리한다. 먼저 파일 디스크립션 맵의 루트 페이지를 버퍼에 올린다. EEPROM의 로그 리스트를 확인하고 미완료된 연산을 처리한다.

5.6 지원하는 명령어

• Create

Create는 새로운 파일을 생성하는 파일 시스템 명령어이다. 파일 디스크립션 맵을 탐색하며 파일이 이미 존재하는지 확인한다. 존재하지 않으면 새로운 파일 ID를 할당하고 EEPROM에 로그를 기록한다. 파일 디스크립션 맵을 갱신하고 EEPROM의 로그를 삭제한다.

• Delete

Delete는 파일을 삭제하는 파일 시스템 명령어이다. 파일 디스크립션 맵을 탐색하며 파일이 존재하는지 확인하고, 존재할 경우 EEPROM에 로그를 기록한다. 파일 디스크립션 맵에서 해당하는 파일 디스크립션을 삭제한 후 파일 디스크립션 맵을 갱신한다. 갱신 완료 후에는 EEPROM의 로그를 삭제한다.

• Open

Open은 Open되어 있는 동안 해당 파일에 대한 파일 연산을 허용하겠다는 명령어로 동시에 2개의 파일 Open이 가능하다. 먼저 EEPROM에서 Open된 파일의 개수를 확인하고, 파일 디스크립션 맵을 순환하며 Open할 파일의 메타정보를 찾는다. EEPROM에 파일의 메타정보를 기록하고 버퍼에 파일의 루트 페이지를 올린다.

• Close

Close는 Open되어 있는 파일을 닫음으로써 해당 파일에 대한 파일연산을 더 이상 허용하지 않는다. EEPROM의 로그 리스트에서 처리되지 않은 모든 로그를 처리한 후에 파일 디스크립션 맵을 갱신한다. 그리고 Open된 파일 정보를 삭제한다.

• Write-append

Write-append는 파일에 Write를 수행하는 연산으로 Open된 상태를 전제로 한다. 데이터가 Write될 때마다 버퍼에 기록하고 버퍼가 꽉 차면 Free페이지를 할당하여 그 내용을 기록한다. 기록이 완료되면 EEPROM에 로그를 기록한다.

• Read

Read는 파일의 데이터를 특정 Byte만큼 읽는 명령어로 Open된 상태를 전제로 한다. 첫 번째 데이터 페이지로 부터 요청된 Byte만큼 읽어서 반환한다.

• Seek

Seek은 파일 포인터를 특정 위치로 이동하는 명령어로 Open된 상태를 전제로 한다. 요청된 바이트가 포함된 데이터 페이지로 파일포인터를 이동한다.

• GetRemaining

GetRemaining은 파일이 현재 위치부터 얼마나 더 읽을 수 있는지 Byte의 크기를 반환해 주는 명령어로 Open된 상태를 전제로 한다. 파일의 전체 크기에서 현재 읽은 만큼의 Byte를 뺀 값을 반환한다.

• Sync

Sync는 버퍼의 내용을 플래시 메모리에 기록하는 명령어 로 Open된 상태를 전제로 한다. Free 페이지를 할당하여 그곳에 버퍼의 내용을 기록한 후에 EEPROM에 로그를 기록한다.

• Dir

Dir은 파일 시스템에 저장되어 있는 파일의 리스트를 반환해 주는 명령어이다. 파일 디스크립션 맵을 순환하며 파일의 리스트를 반환한다.

6. 결 론

현재 센서 네트워크에서는 점차 센서 노드에 대용량의

저장 시스템이 요구되는 추세이므로 기존의 저장소로 많이 사용되었던 노어 플래시 보다 대용량화가 쉬운 낸드 플래시 기반의 센서 노드가 늘어날 것으로 예상된다. 본 논문에서는 앞으로의 센서 노드용 저장소로 가장 적합한 대용량 낸드 플래시 기반 파일 시스템의 설계에 대해 논하였다.

본 파일 시스템은 자원과 에너지의 효율성, 대용량의 지원 및 신뢰성을 극대화하는 방향으로 설계되었으며, 매우 작은 크기의 EEPROM을 사용하여 센서 노드에서 제공하는 작은 크기의 메인 메모리의 한계를 극복하였다. 특히 본 논문에서는 로그 리스트 기반의 페이지 처리방식을 언급하였는데, 이는 플래시 메모리의 덮어쓰기가 불가능한 물리적인 단점을 극복하였으며, 로그 리스트를 사용하지 않는 방식 보다 약 3~4배 이상의 페이지를 절약한다. 그 효과로 보다 긴 시간동안 데이터의 수집이 가능하게 되고 쓰기 속도가 개선되었으며 센서 노드의 수명이 약 10%이상 향상된다. 우리는 이러한 파일 시스템을 시뮬레이터를 통해 구현하였다.

본 논문에서 제안한 효율성을 높이는 여러 가지 방법을 센서 노드용 파일 시스템에 적용한다면, 앞으로의 센서 노드용 파일 시스템의 성능 향상에 적지 않은 기여를 할 것으로 기대한다.

참고문헌

[1] Eran Gal and Sivan Toledo "Algorithms and Data Structures for Flash Memories", ACM Computing Surveys Vol 37, Issue 2, pp138-163, 2005.

[2] D.Gay. Design of Matchbox : The simple Filing system for Motes. In TinyOS 1.x distribution, http://www.tinyos.net, Aug2003.

[3] H.Dai, M.Neufeld, and R.Han. ELF: An efficient Log-structured Flash file system for micro sensor nodes. In SenSys,page 176-187,New York NY, 2004.

[4] Gaurav Mathur, Peter Desnoyers, Deepak Ganesan and Prashant Shenoy, "Capsule: An Energy-Optimized Object Storage System for Memory-Constrained Sensor Devices", Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys), Boulder CO, November 1-3, 2006.

[5] TIP7xx Series Manual : http://www.maxfor.co.kr.

[6] 128M * 8Bit NAND Flash Memory, SAMSUNG.