센서 데이터 수집을 위한 대용량 NAND 플래시 파일 시스템의 설계

(Design of High-capacity NAND Flash File System supporting Sensor Data Collection)

한 경 훈 † 이 기 혁 ††
(Kyoung-hoon Han) (Ki-hyeok Lee)
한 형 진 †† 한 지 연 ††
(Hyung-jin Han) (Ji-yean Han)
손 기 락 †††
(Kirack Sohn)

요 약 센서 노드의 활용 분야가 점차 다양화되는 추세이므로 활용 분야에 적합한 다양한 데이터 수집 방법이 요구된다. 데이터의 실시간 감시가 불필요한 경우 수집과 동시에 전송을 수행하는 현재의 데이터 수집 방법은 불필요한 전력 소모 및 데이터 손실을 발생시킬 수 있다. 데이터를 수집한 노드가 데이터를 저장하고 질의에 의해 필요한 데이터를 획득하는 새로운 방법이 요구된다. NAND 플래시는 에너지 효율성이 좋고 대용량화가 쉬워 앞으로의 센서 노드용 저장소로 적합하다. 센서 노드는 4~10 KByte의 적은 메모리를 지원하고 NAND 플래시는 덮어쓰기가 불가하고 쓰기 제한이 있어 효율성이 뛰어난 파일 시스템의 구축은 어렵다. 본 논문은 센서 노드 환경에서 대용량 NAND 플래시 파일시스템의 설계에 대해 논한다. 파일 시스템은 전송 비용을 줄여 보다 장시간 동안의 데이터 수집을 가능하게 한다. 앞으로 다양한 분야에 적용되어 센서 네트워크 환경에서 핵심 구실을 할 것으로 예상한다.

키워드 : 센서 노드, NAND 플래시, 파일 시스템

Abstract As the application fields of sensor nodes are getting diverse these days, it is required to have a way of collecting various data that is suitable for these application fields. In the case that the real-time surveillance over the data is unnecessary, present data collecting methods, which collect and transfer the data directly, can cause a waste of energy and data loss. A new method that store the collected data in a local storage and acquire them by query later on is required for nonreal-time applications. NAND flash has energy efficiency and large capacity so it is suitable for sensor nodes. Sensor nodes support 4-10 KBytes small sized memory and it is hard to build an effective file system since NAND Flash doesn't support overwriting NAND flash. This paper discusses an implementation of NAND Flash file system in sensor node environments. The file system makes long-term data collecting possible by reducing transmission cost. It is expected that this file system will play a central role in sensor network environments as it can be applied to various fields which call for long term data collecting.

Key words : Sensor node, NAND Flash, File system

1. 서 론

정부의 u-IT839 전략과 더불어 u-City, u-Health 등 다양한 사업의 대두로 센서 네트워크 기술은 군사, 의료, 기상, 환경, 가정 등 점차 활용분야가 다양화되고 있다.

현재의 센서 네트워크는 단말 노드에서 데이터를 수집한 후 RF 통신을 사용하여 목적지 노드로 전송한다. RF 통신이 소모하는 에너지양은 저장의 그것에 100배이다 [1]. 불필요한 전송을 방지한다면 더욱 오랜 시간 데이터를 수집할 수 있게 된다. 따라서 실시간 모니터링이 불필요한 센서 네트워크 응용에서는 대용량 저장소를 부착하여 수집한 센서 데이터를 저장하고 나중에 질의를 통해 필요한 데이터 또는 요약 데이터를 수집하는 것이 전력 소비량을 줄여서 센서 노드의 수명을 연장할 수 있다.

현재 센서 노드의 저장소로는 플래시메모리가 많이 사용되고 있다. 플래시메모리는 반도체 기반의 데이터 저장매체로서 비휘발성(non-volatile) 메모리이다. 하드디스크와 비교하여 빠른 속도와 저전력, 강한 내구성의 장점 때문에 센서 노드용 저장소로 주목받고 있다.

· 이 연구는 2009학년도 한국외국어대학교 교내학술연구비의 지원에 의하여 이루어진 것임
· 이 논문은 제35회 추계학술대회에서 '센서 데이터 수집을 위한 TinyOS 기반 대용량 NAND 플래시 파일 시스템의 설계'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 한국외국어대학교 컴퓨터 및 정보통신공학부
khhan@hufs.ac.kr
†† 비 회 원 : 한국외국어대학교 컴퓨터 및 정보통신공학부
knockin81@hanmail.net
nom96ny@hanmail.net
hanjiyean@empal.com
††† 비 회 원 : 한국외국어대학교 컴퓨터 및 정보통신공학부 교수
ksohn@hufs.ac.kr

논문접수 : 2008년 12월 19일
심사완료 : 2009년 4월 28일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제15권 제7호(2009.7)

플래시메모리는 NAND 플래시와 NOR 플래시로 구분된다. NAND 플래시는 NOR 플래시보다 쓰기 및 삭제 속도가 월등히 빠르며 최근 단가가 낮아지고 대용량화되는 추세이므로 앞으로의 센서 노드용 저장소로 적합하다.

센서 노드는 4~10 KByte의 적은 메모리를 지원하므로 YAFFS[2], JFFS[3] 등 메가바이트 단위의 많은 메모리를 요구하는 기존의 NAND 플래시 파일 시스템을 사용할 수 없다. 센서 노드의 장치적인 제약과 함께 덮어쓰기(in-place update)가 불가능하고 쓰기 횟수에 제한이 있는 NAND 플래시의 제약 때문에 효율성이 뛰어난 파일 시스템을 구축하는 것은 어렵다[4].

본 논문은 센서 노드 환경에서 표준으로 사용되는 대용량 NAND 플래시 파일시스템의 설계에 대해 논한다. 자원의 효율적인 활용, 에너지 절약 및 신뢰성을 보장하는 몇 가지 방법들을 제안한다.

2. 관련 연구

기존의 센서 노드용 파일 시스템은 센서 노드라는 장치적 제약 때문에 그 수가 적다. 공식적으로 발표된 센서 노드용 파일 시스템으로는 Matchbox, ELF, Capsule(표 1에 요약)이 있다.

표 1 센서 노드용 플래시메모리 파일 시스템 특징 비교

구 분	Matchbox	ELF	Capsule
저장 장치	NOR	NOR	NOR,NAND
에너지 최적화	X	X	O
메모리 최적화	O	O	O
자원 평준화	X	O	O
Crash 복원	X	Snapshot	Checkpoint
추상화	File System	File System	Object
연산	Create Delete Open Read Append	Create Delete Open Read Append Modify Seek Rename Mkdir Rmdir	Create Delete Open Read Append Move
EEPROM		Metadata Index	

2.1 Matchbox

Matchbox[5]는 NOR 플래시기반으로 TinyOS에서 기본으로 제공한다. 메모리 최적화 측면만 고려하였고 Crash 복원 정책은 없다. 자원 평준화를 고려하지 않아 플래시메모리의 한 곳을 과도하게 사용할 수 있으며 이는 장치의 수명 감소 원인이 된다. 부팅시간에 모든 페이지를 검사하여 메타데이터를 구성한다.

2.2 ELF

ELF[6]는 Matchbox와 같이 NOR 플래시 기반이며 NOR 플래시의 특징을 살려 Write-modify와 Seek 연산을 지원한다. 에너지 최적화는 고려되지 않았고 Crash 복원 정책으로 Snapshot을 사용한다. EEPROM에 디렉터리와 메타데이터, 인덱스를 저장한다.

2.3 Capsule

Capsule[1]은 NOR 플래시에서 NAND 플래시의 특징을 살려 구현되었다. 객체 기반 시스템으로 스택, 큐, 인덱스 등 다양한 자료구조를 저장 시스템 단위에서 지원한다. 하지만, 인덱스의 크기가 컴파일시간에 고정되어 파일의 크기와 그 수가 제한된다. Backward Chaining을 사용하므로 파일의 앞을 읽으려면 뒤에서부터 전체를 읽어야 한다. 유일하게 NAND 플래시에서 동작 가능하게 설계되었지만, 이러한 제약 때문에 대용량 NAND 플래시 파일 시스템으로는 적합하지 않다.

3. NAND 플래시 파일 시스템의 구조

디스크 I/O를 최소화하여 전력 소모를 줄이고 NAND 플래시의 덮어쓰기 불가, 쓰기 횟수 제한의 단점을 극복하기 위한 파일 시스템(그림 1)의 구조에 대해 논한다.

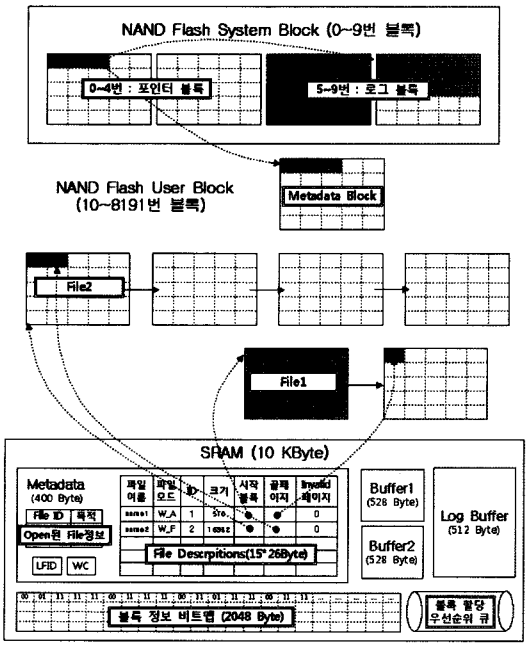

그림 1 파일 시스템의 전체 구조

3.1 NAND 플래시의 구조화

본 파일 시스템은 NAND 플래시를 시스템 블록과 사용자 블록으로 구분한다. 시스템 블록은 포인터 블록과 로그 블록으로 구분하고 메타데이터가 존재하는 블록의 위치정보와 로그 정보를 기록한다. 사용자 블록은 메타데이터와 사용자에 의해 생성된 파일이 저장된다.

3.2 블록의 4가지 상태

블록 단위로 삭제가 이루어지는 NAND 플래시의 특징에 맞춰 모든 공간의 할당은 블록 단위로 수행한다. 블록은 FREE_LOW, FREE_HIGH, IN_USE, GARBAGE 중 하나의 상태를 갖는다. 최초 모든 블록은 FREE_LOW 상태이고 할당되면 IN_USE 상태가 된다. 그림 2와 같이 DELETE 연산으로 삭제 예정인 블록은 GARBAGE 상태가 되고 가비지 컬렉터에 의해 블록 삭제가 수행되면 다시 FREE_LOW 혹은 FREE_HIGH 상태가 된다.

블록의 상태	비고
IN_USE	• 파일 혹은 메타데이터가 사용 중인 블록
GARBAGE	• 블록의 데이터가 더는 유효하지 않음 • 블록 삭제의 수행 대상
FREE_LOW	• 사용 가능한 블록 • 사용 횟수가 메타데이터의 Erase 횟수 값보다 작아 블록 할당 시 먼저 할당됨
FREE_HIGH	• 사용 가능한 블록 • 사용 횟수가 메타데이터의 Erase 횟수 값과 같아 FREE_LOW 블록이 모두 할당된 후 할당됨

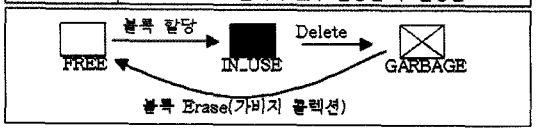

그림 2 블록의 상태 및 상태 변화

3.3 파일의 구조

파일은 블록의 Forward Chaining으로 구성한다. 데이터를 기록하려면 블록을 할당받고 페이지 단위로 차례대로 기록한다. 파일의 정보는 메타데이터에 기록하고 WRITE_APPEND 모드와 WRITE_FIXED 모드를 지원한다.

WRITE_APPEND 모드는 그림 3과 같이 데이터를 기록할 때 새로운 블록을 할당받고 할당받은 블록을 모두 기록한 후에 새로운 블록을 할당받는다. 이전 블록의 3번 페이지 스페어 영역에 다음 블록 번호를 기록한다.

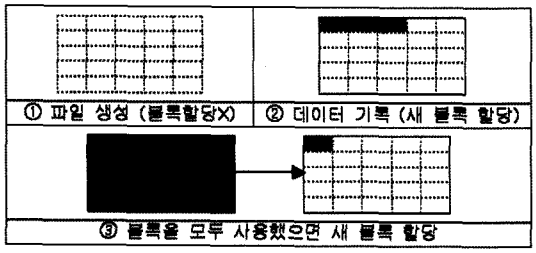

그림 3 WRITE_APPEND 모드의 파일

WRITE_FIXED 모드는 그림 4와 같이 파일을 생성할 때 크기를 입력받고 그 크기를 보장할 만큼 미리 블록을 할당한다. 원하는 크기의 공간을 미리 할당하기 때문에 저장장치의 용량이 부족하여 기록이 불가한 상황을 방지할 수 있다.

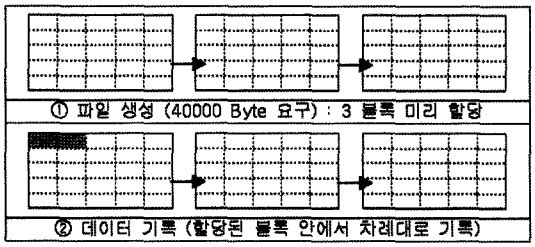

그림 4 WRITE_FIXED 모드의 파일

3.4 메타데이터

메타데이터는 파일 시스템을 유지하는 데에 필요한 정보이다. 메타데이터의 구성은 표 2와 같다.

표 2 메타데이터의 구성

목적	크기
파일 디스크립션 [MAX_FILES]	26*15 = 390 Byte
생성된 파일의 개수	1 Byte
Open 된 파일의 정보 [MAX_OPEN]	2*2 = 4 Byte
마지막에 할당한 파일 ID	1 Byte
블록 Erase 횟수	4 Byte
합계	400 Byte

• 파일 디스크립션

파일 디스크립션(표 3)은 파일의 메타정보이다. 최대 생성할 수 있는 파일의 개수는 15개로 제한한다. 비유효 페이지 개수는 Sync 연산에 의해 기록된 가득 차지 않은 페이지의 수를 나타낸다. Seek 연산으로 파일의 특정 위치를 찾을 때 사용된다. Sync 후에 Write 연산을 수행하면, 그림 5와 같이 이전 페이지의 내용에 새로운 내용을 추가한 후 다음 페이지에 기록한다. Sync 연산으로 기록한 페이지는 비유효 페이지가 된다. 비유효 페이지의 수는 기록하는 블록이 변경될 때 기록 중이던 블록의 4번 페이지 스페어 영역에 기록한다.

표 3 파일 디스크립션의 구성

목적	크기
파일 이름	14 Byte
파일 모드 (WRITE_FIXED/WRITE_APPEND)	1 Byte
파일 ID	1 Byte
파일 크기	4 Byte
시작 블록 번호	2 Byte
마지막 페이지 번호	3 Byte
비 유효 페이지 개수	1 Byte
합계	26 Byte

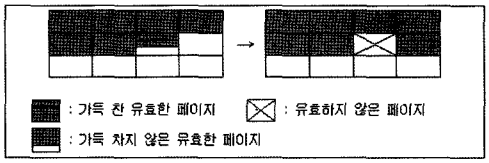

그림 5 비유효 페이지

• Open된 파일의 정보
Open한 파일의 정보이다. 최대 2개까지 동시 Open을 허용하며 READ_ONLY 모드와 WRITE_ONLY 모드를 지원한다.
• 블록 Erase 횟수
가장 많이 사용된 블록의 Erase횟수이다. NAND 플래시를 균등하게 사용하고자 활용되며 4.1절에서 설명한다.

4. NAND 플래시 파일 시스템 알고리즘

NAND 플래시와 제한된 하드웨어 환경에서 자원 평준화, 에너지 효율성, 신뢰성을 보장하기 위한 몇 가지 알고리즘을 제안한다.

4.1 NAND 플래시의 균등한 사용을 위한 정책

한 블록이 마모되면 전체 블록의 사용에 영향을 주는 플래시메모리의 특징 때문에 전체 블록을 균등하게 사용하는 것은 매우 중요하다. 본 파일 시스템은 블록마다 기록하는 Erase 횟수와 메타데이터에 기록한 최대 Erase 횟수, 블록 할당 우선순위 큐를 사용하여 Wear-leveling을 관리한다.

블록 할당 우선순위 큐에는 Erase 횟수가 가장 작은 5개의 블록 번호가 삽입된다. 사용 중이던 블록이 삭제되면 자신의 Erase 횟수에 1을 증가시키고 메타데이터에 기록된 최대 Erase 횟수와 비교한다. 최대 Erase 횟수와 같으면 FREE_HIGH 상태가 되고 작으면 FREE_LOW 상태가 된다. 우선순위 할당 큐에 삽입된 모든 원소들의 Erase 횟수와 자신의 횟수를 비교하여 특정 원소보다 작으면 적절한 위치에 삽입된다.

블록을 할당할 때에는 블록 할당 우선순위 큐의 블록이 우선한다. 큐가 비었으면 FREE_LOW 상태의 블록을 할당한다. FREE_LOW 블록이 존재하지 않으면 FREE_HIGH 블록을 모두 FREE_LOW 상태로 만들고 메타데이터의 Erase 횟수를 1 증가시킨다. 장시간 파일에 의해 점유되어 적게 사용된 블록이 우선 할당되므로 최대한 균등하게 블록이 사용되어진다.

4.2 메타데이터를 플래시메모리에 기록하는 시기

메타데이터는 사용자 블록에서 한 블록을 할당받아 한 페이지씩 차례대로 기록한다. 메타데이터 블록의 위치는 포인터 블록에 기록한다.

메타데이터는 *Create*, *Delete*, 블록 할당 연산에 의해 갱신되는 갱신 빈도가 낮은 정보들과 Write 연산에 의해 갱신되는 갱신 빈도가 높은 정보들로 구분된다. 메타데이터의 갱신 빈도는 표 4에 요약하였다.

Create와 Delete 연산은 센서 노드의 특성상 Write 연산과 비교하여 자주 발생하지 않는다. 갱신 빈도가 낮은 메타데이터가 변경될 경우 변경과 동시에 메타데이터 전체를 플래시메모리에 기록한다.

Write 연산이 발생하면 메타데이터에 있는 해당 파일의 크기, 마지막 페이지 번호, 비유효 페이지 개수 정보가 변경된다. 이 데이터들은 파일을 Close하는 시기와 기록할 블록이 변경되는 시기에 플래시 메모리에 기록한다. 갱신빈도가 높은 정보는 블록 정보 단위로 플래시메모리에 기록한다. 블록 정보 단위로 데이터를 갱신하면 페이지 정보 단위와 비교하여 약 30번의 디스크 I/O를 줄일 수 있다. 전원이 소실되면 부팅시간에 로그를 보고 원 상태로 복구한다. WRITE_ONLY 모드로 Open된 파일의 마지막 블록만 검사하면 된다.

표 4 메타데이터의 갱신 빈도

구분	갱신 빈도
파일 디스크립션	높음 (Write / Create / Delete)
생성된 파일의 개수	낮음 (Create / Delete)
Open 되어있는 파일의 정보	낮음 (WRITE_ONLY 모드 Open)
마지막에 할당한 파일 ID	낮음 (Create)
블록 Erase 횟수	낮음 (모든 블록 1회 사용 시)

4.3 신뢰성 보장을 위한 로그 블록

본 파일 시스템에서는 로그를 사용하여 신뢰성을 보장한다. 로그는 로그 블록을 한 블록씩 할당받아 페이지 단위로 차례대로 기록한다.

0번부터 3번 페이지는 이전 로그 블록의 모든 로그가 반영된 블록 정보 비트맵을 기록한다. 4번 페이지부터는 다양한 파일시스템 명령으로 발생하는 로그를 기록한다. 로그는 4 Byte이고 한 페이지에 최대 127개까지 기록할 수 있다.

WRITE_FIXED 모드로 *Create* 연산을 수행하면 블록을 미리 할당하는 데에 모두 성공하거나 모두 실패해야 한다. 생성하는 파일의 시작 블록 주소를 로그로 남기고 플래시메모리에 기록한 후 메타데이터를 갱신한다. 부팅 시 로그는 존재하지만 파일 디스크립션이 존재하지 않으면 전원이 소실된 경우이므로 파일의 시작 블록부터 연결된 모든 블록을 GARBAGE 상태로 만든다.

Delete 연산이 발생하면 삭제하고자 하는 파일의 링크를 따라가며 GARBAGE 로그를 남긴다. 파일의 끝에 도달하면 END 로그를 남기고 플래시메모리에 기록한 후 파일 디스크립션을 제거한다. 부팅 시 END 로그가 존재하지 않으면 해당 로그를 무시한다.

WRITE_ONLY 모드로 Open 연산을 수행하면 파일 ID와 마지막 블록 번호를 로그로 남기고 플래시메모리에 기록한다.

Write 연산은 기록할 블록이 추가될 때 추가된 블록 번호와 파일 ID를 로그로 남긴다. 로그 버퍼가 가득 차거나 Close 연산이 발생하면 로그를 플래시메모리에 기록한다. Close 전에 전원이 소실되면 Write 연산에 의해 기록된 모든 로그는 소실된다. 부팅 시 Open 로그는

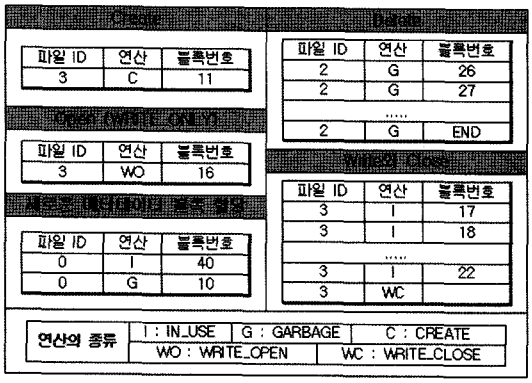

그림 6 로그가 플래시메모리에 기록되는 형태

발견했지만, Close 로그를 발견하지 못하면 Open 로그에 기록된 블록 번호부터 링크를 따라가 모든 상태를 복구한다.

메타데이터를 기록할 블록이 변경되면 이전 블록을 GARBAGE 상태로 만들고 새 블록을 사용한다는 의미의 로그를 남기고 플래시메모리에 기록한다. 그림 6은 로그가 플래쉬 메모리에 기록되는 형태를 보여준다.

4.4 위치정보 저장을 위한 포인터 블록

자주 변하는 메타데이터와 로그 블록의 위치정보를 기록하고자 포인터 블록을 사용한다. 위치정보는 메타데이터를 기록할 블록이 변경되거나, 로그를 기록할 블록이 변경되면 포인터 블록에 페이지 단위로 기록한다.

메타데이터를 기록할 블록이 변경되면 할당받은 블록과 이전 블록의 주소를 플래시메모리에 기록한다. 전원이 소실되어 새로운 메타데이터 블록에 메타데이터를 기록하지 못하면 이전 메타데이터 블록을 참고해 소실된 메타데이터를 복구한다.

로그를 기록할 블록이 변경되면 새로운 로그 블록의 위치정보를 플래시메모리에 기록한다. 로그 블록은 차례대로 할당되므로 새 블록의 주소만 기록한다.

5. 결 론

센서 노드의 활용 분야가 점차 다양화되는 추세이므로 활용 분야에 적합한 다양한 데이터 수집 방법이 요구된다. 본 논문에서는 센서 노드에 NAND 플래시를 부착하고 파일 단위의 데이터 관리를 위한 파일시스템의 설계에 대해 논하였다.

본 파일 시스템은 자원과 에너지 효율성, 대용량의 지원 및 신뢰성을 극대화하는 다양한 알고리즘을 제안한다. NAND 플래시와 매인메모리를 적절히 구조화하였고 블록 정보 비트맵과 블록 할당 우선순위 큐를 사용하여 자원 평준화를 고려하였다. 로그를 사용하여 신뢰성을 보장하며 포인터 블록을 사용하여 부팅시간에 전체 블록을 검사해야 하는 초과비용을 줄였다.

센서 노드용 파일 시스템은 전송 비용을 줄임으로써 보다 장시간 동안 정확한 데이터 수집을 가능하게 한다. 앞으로 다양한 분야에 적용되어 센서 네트워크 환경에서 핵심 구실을 할 것으로 예상한다.

참 고 문 헌

[1] 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.

[2] Aleph One. Yet Another Flash File System. http://www.aleph1.co.uk/yaffs.

[3] D. Woodhouse, Red Hat, Inc. "JFFS : The Journaling Flash File System," Journaling Flash File System (JFFS). http://sources.redhat.com/jffs2/jffs2-html/.

[4] Eran Gal and Sivan Toledo "Algorithms and Data Structures for Flash Memories," *ACM Computing Surveys*, vol.37, Issue.2, pp.138-163, 2005.

[5] D. Gay. Design of Matchbox : The simple Filing system for Motes. In TinyOS 1.x distribution, http://www.tinyos.net, Aug. 2003.

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