NTNU | Norwegian University of Science and Technology

QUANTIFYING DATA VOLATILITY FOR IOT FORENSICS WITH EXAMPLES FROM CONTIKI OS

Jens-Petter Sandvik Katrin Franke Habtamu Abie André Årnes

2022-03-29

Contents

Introduction

Contiki OS and Coffee File System

Volatility

Experiments

Conclusion and final thoughts

Background

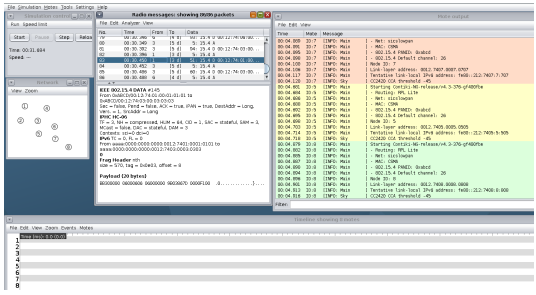
- IoT systems are found everywhere
- Network connected
- Spans from the most powerful cloud servers to small energy harvesting systems
- Huge variability to amount of storage and memory
- Non-standard interfaces for collecting data
- Increasing number of evidence locations
- Resource demanding to acquire data from devices
- Need a way of predicting which devices that still contain data

Research Questions

1. How can the data volatility in IoT devices be analytically calculated?
2. How can the data volatility in IoT devices be measured?
3. How well do the analytical results and measurements correspond?

Contiki

- Open source OS for resource constrained devices
- OS footprint: 100 kB
- Need 10 kB RAM for running
- Comes with an emulator and simulator: Cooja
- Emulates non-volatile storage
- Contains the IoT protocol stack as defined by IETF
  - IPv6, 6LoWPAN, RPL, CoAP, MQTT, SNMP, etc.



Coffee File System

- File system used in Contiki
- A simple flash based file system
- Contains very little metadata: 26 bytes, including file name
- Metadata only exist in file header
- File updates are written to an attached log file
- Every 4 write will copy contents to new file and create a new log file
- Garbage collection when FS write pointer reaches end of file system
- See my paper from last DFRWS-US conference on the Coffee file system

00032600	37 02 00 00 00 00 11 00 00 0f 66 69 6c 65 30 30	7.........file00
00032610	33 2e 74 78 74 00 00 00 00 00 00 46 69 6c 65 33 20	3.txt.....File3
00033700	00 00 00 00 00 00 05 00 00 17 66 69 6c 65 30 30	..........file00
00033710	33 2e 74 78 74 00 00 00 00 00 00 05 00 05 00 03 00	3.txt...........
00033720	03 00 46 69 6c 65 33 20 20 42 23 30 30 30 30 30	..File3 B#00000

Coffee vs YAFS vs ext4

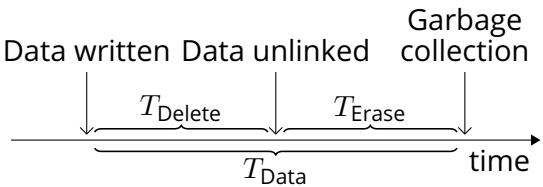
- More file systems are used in IoT devices and other computing systems
  - YAFFS2, ext4, Reliance Nitro, TSFS, exFAT, NTFS, +++
- YAFFS2
  - Similar to Coffee
  - More aggresive garbage collection, will move active data to new blocks
  - This lead to less deleted data surviving a GC
  - More metadata (mac times, owner, permissions, etc)
  - Used in many mobile phones
  - Less used today, as managed flash is more popular
- ext4
  - Known form Linux systems
  - Used for managed flash in many Linux-based systems
  - Centrally stored superblock and detached metadata from file contents
  - No garbage collection in file system

Data volatility

- Relative timescales: Order of Volatility
- Time from data is created until it is erased: *lifetime of data*
- … or the time before data is erased from now: *remaining lifetime of data*
- … or the probability of the existence of data at a time, t: *statistical lifetime of data*
- The probability for data to still be present in a device: More precise triage?

Data lifetime

- Adapt a model based on reliability analysis
- Total data volatility split into two periods:
  - Time between creation and deletion
  - Time between deletion and overwrite/ erase
- Data held by purpose — Data held by chance
- Acquisition method's importance



Volatility model

- A high-level perspective
- Model is a 6-tuple: $\mathcal{V}_{\mathcal{D}} = (L, E, A, M, D, S)$.
  - **L:** Storage abstraction layers
  - **E:** Events
  - **A:** Application activity functions
  - **M:** Storage management functions
  - **D:** Memory device reliability
  - **S:** Environment

Volatility in the Coffee File System

Time to first garbage collection

$$T_{\text{GC, first}} = r\left(\left\lfloor \frac{S_S S_n}{F_S} \right\rfloor L_r + 1\right) \quad (1)$$

$$r = \frac{1}{\sum_{i=1}^{F_n} \frac{1}{r_i}} \quad (2)$$

- S_S Sector size
- S_n Number of sectors in file system
- F_s File size (average)
- L_r Number of records in log file
- F_n Number of files
- r_i Writing rate of each file (given in seconds/write)

Volatility in the Coffee File System

- Number of writes is dependent on the number of files in the file system
- How they are spread across the sectors.
- Time between GC dependent on the remaining number of erased sectors

Number of writes before GC

$$N_{\text{min}} = \left\lceil \frac{\sum_{i=1}^{F_n} F_{S,i}}{S_S} \right\rceil \tag{3}$$

$$N_{\text{max}} = F_n + \sum_{i=1}^{F_n} \left\lceil \frac{F_{S,i} - 1}{S_S} \right\rceil \tag{4}$$

$$T_{\text{GC, min}} = r \left(\left\lfloor \frac{S_S(S_n - N_{\text{max}})}{F_S} \right\rfloor L_r + 1 \right) \tag{5}$$

$$T_{\text{GC, max}} = r \left(\left\lfloor \frac{S_S(S_n - N_{\text{min}})}{F_S} \right\rfloor L_r + 1 \right) \tag{6}$$

Survival time

Maximum survival time of a sector

$$T_{\text{sect,max}} = T_{\text{GC}} \left\lceil \frac{r_{\text{max}}}{T_{\text{GC}}} \right\rceil \tag{7}$$

T_{GC} Inter-GC time

r_{max} the slowest writing rate

Numerical example

3 files with a rate of 1 s/w, 50 s/w, and 1000 s/w, $S_S = 256$ pages, $S_n = 15$ sectors, $F_S = 22$ pages, $L_r = 4$ records:

$$r \approx 0.979 s/w, \qquad T_{\text{GC, min}} = 408 \qquad = 22$$

$$T_{\text{sect,max}}, = 1270 s \qquad T_{\text{GC, max}} = 635$$

Approximation of volatility

- Don’t need an exact model for every system
- Need to find a more generic term that can be tuned
- Scaling factor needs to be estimated experimentally

Approximation

$$T = \frac{rk}{2}\left(\left\lfloor \frac{S_n S_S}{F_n} \right\rfloor L_r + 1\right) \tag{8}$$

k Scaling factor

Experimental setup

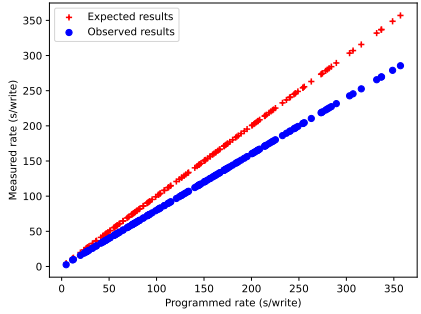
- Cooja simulator
- Standalone devices
- 3 applicatons per device
- Writing rate sampled at random from a uniform distribution between 1 and 1200
- Each simulation consisted of 16 devices
- Run for 10 hours simulation time
- Repeated 10 times with new writing rates for each repetition
- One device crashed consistently and was removed from the analysis

Canary tokens

- Writing is detected by writing and reading canary tokens
- Applications write canary tokens to file
- Patched two functions in `arch/platform/sky/dev/xmem.c` to record token operations:
  - `program_page`
  - `erase_sector`
- These would log any pages that contained one of the canary tokens
- T_{Delete} is time from written to time for next write of same token
- T_{Erase} is the time from T_{Delete} until sector is erased

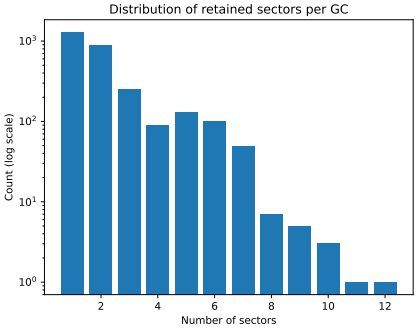
Results — Writing rate

- Discrepancy between programmed write rate and observed write rate
- Very consistent difference
- Wondered if this was because of write bursts after GC
- Drifting clock in simulator?
- 4 writing ops result in 5 writes



Garbage collection

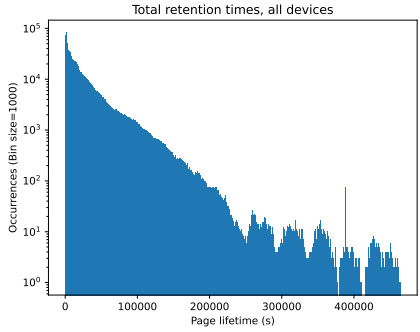
- Very consistent time used for GC
- Median time 56.8 s, quartiles 56.7 s and 61.1 s
- Some outliers: min = 13 s, max = 73 s
- Number of retained sectors n.e.d.(ish) trend
- Only one device with more than 5 retained sectors
- Minimum time between GC underestimated with median 3,6 hours
- Maximum time between GC underestimated with median 1,75 hours



Page lifetimes

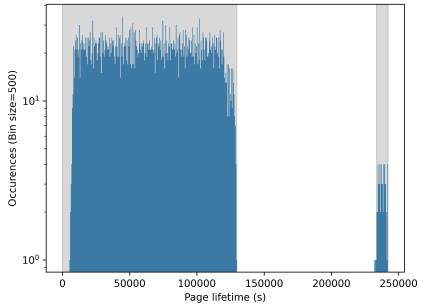
- Average retention time for all pages: 7:32:27
- Maximum retention time: 5 days, 8:58:41
- Approximation: scaling factor, $k = 1.065$

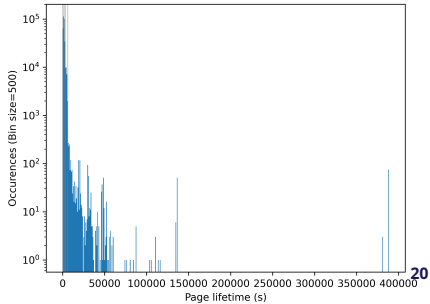
k	1.065
Mean	-51.150
Std.dev.	406.690
Min	-1627.143
25%	-217.455
Median	-3.147
75%	135.829
Max	961.772



Lifetime distribution

- Individual distribution of lifetimes
- Matched none of the distributions in SciPy
  - 104 distributions tested
  - Fitted to the closest matching distribution parameters
  - 3 distributions used too long time to finish
- Kolmogorov-Smirnov test
- p-value close to 0 for almost all tested distributions
- Not a simple distribution





Discussino and conclusion

- Other file systems?
- Generalization
- Analytical framework
- Patched simulation for measuring volatility
- Long retention time, even for resource constrained nodes
- Could not explain all outliers from analytical model

Thanks

?!

jens.p.sandvik@ntnu.no