Technische Universität Braunschweig

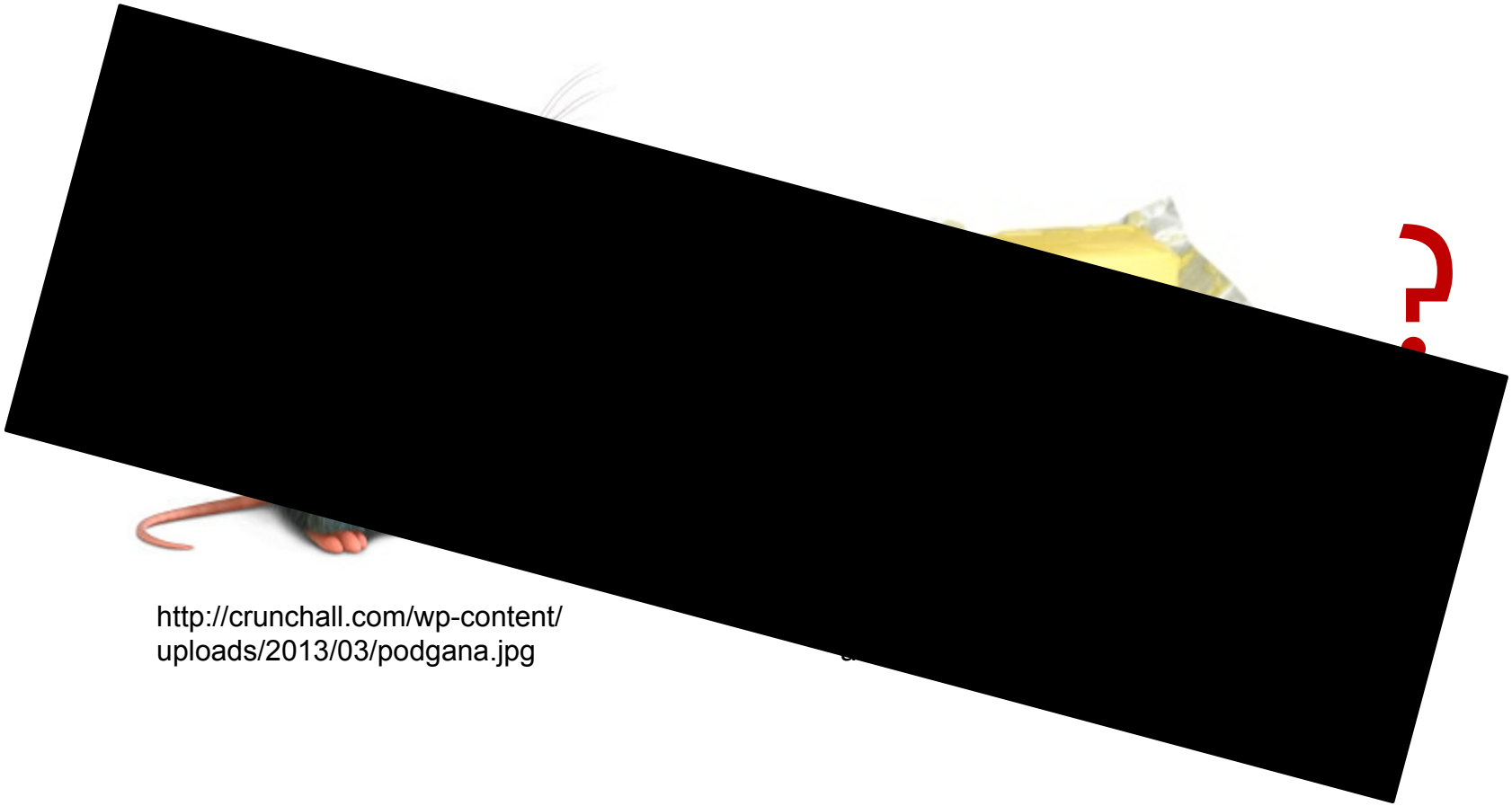
Institute of Operating Systems and Computer Networks

RATFAT: ReAl-Time FAT for Cooperative Multitasking Environments in WSNs

Sebastian Schildt, Wolf-Bastian Pöttner, Felix Büsching, Lars Wolf

PWSN, Boston - May 23, 2013

RAT + FAT?



http://crunchall.com/wp-content/
uploads/2013/03/podgana.jpg

Motivation

- Many WSN apps need persistent data storage
- Some WSN apps want large-scale data storage
- Often data is stored on nodes and processed on PC

} SD Cards

- Some WSN apps require real-time guarantees

We need a file system for WSN nodes that is “PC-compatible” and real-time capable!

State of the Art in WSN file systems

COFFEE in Contiki

- Unpredictable timing behavior (non real-time)
- Bad performance on large (> a few MB) volumes
- Not readable on standard PC Operating Systems
- But: high memory efficiency

FatFs + tinyFAT

- Unpredictable timing behavior (non real-time)

State of the Art in WSN Operating Systems

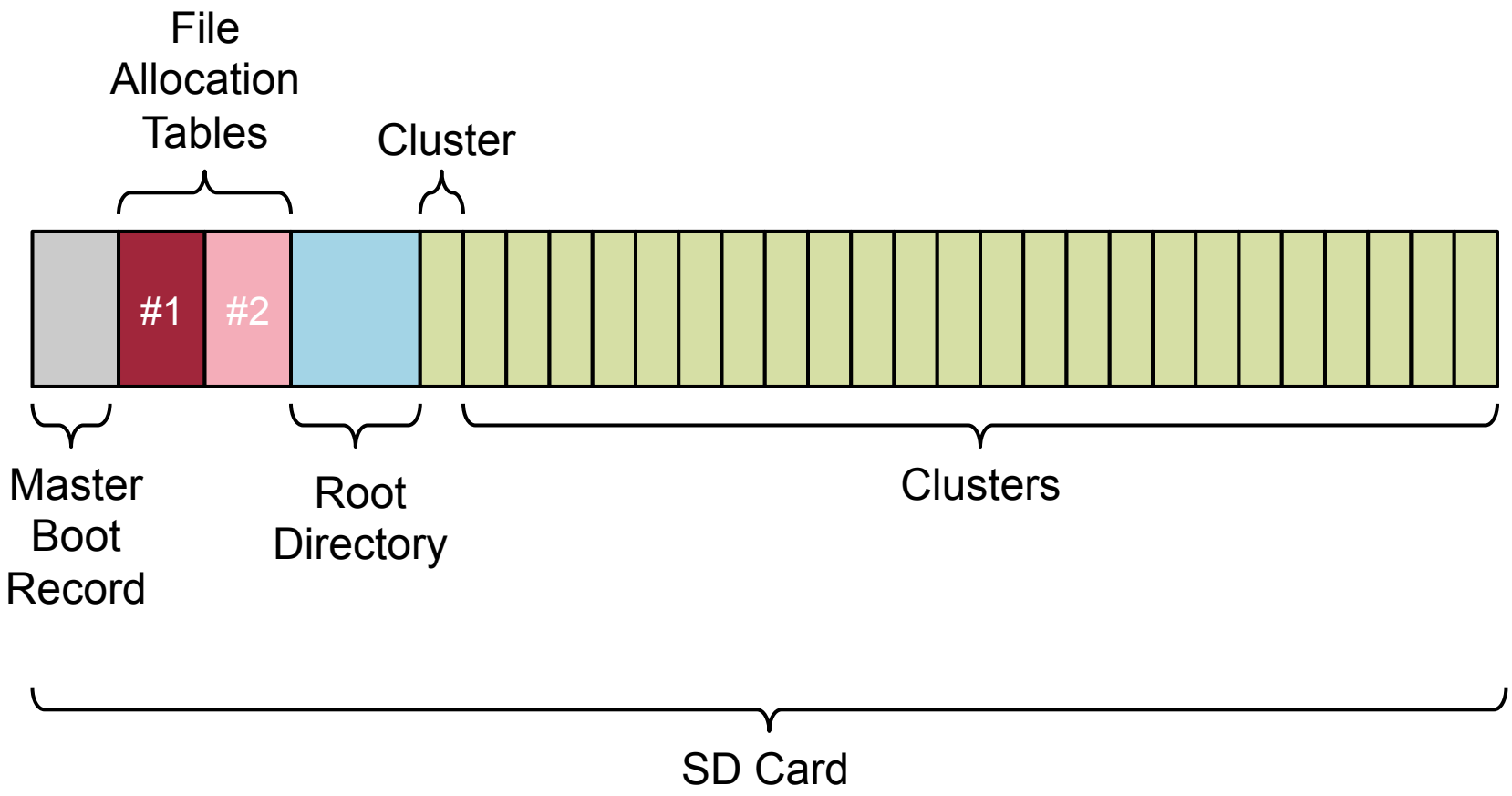
TinyOS 2

- Cooperative multitasking; no preemptions
- Round-robin scheduler

Contiki

- Cooperative multitasking; no preemptions
- Round-robin scheduler
- RTimers allow (certain) real-time tasks
  - Interrupting ongoing SPI operations may lead to undefined states
  - Concurrent access to (different) SPI devices is problematic

Organization of a FAT Volume



RATFAT Features

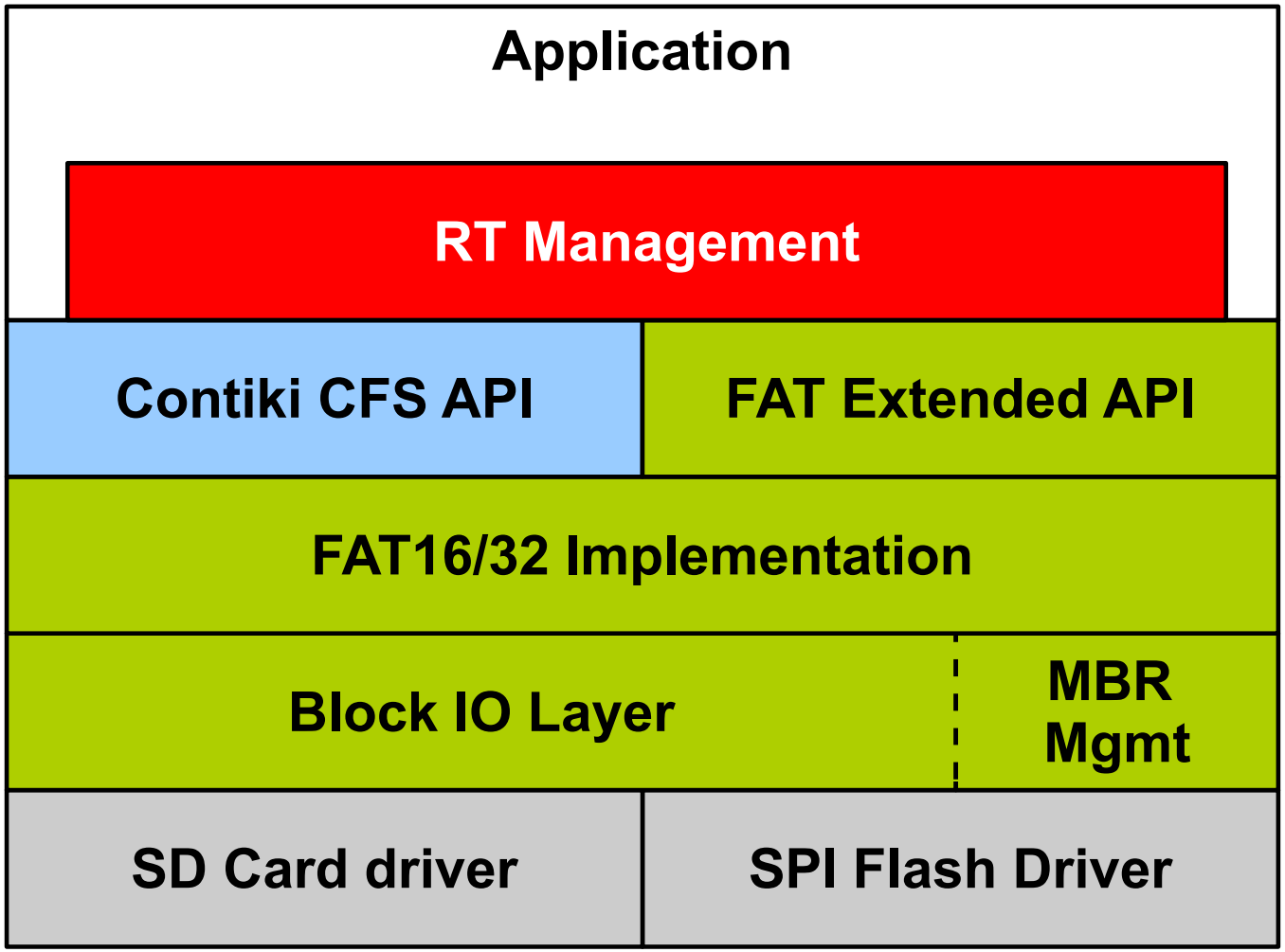
Highlights

- Fully compliant to FAT specification
- Support for FAT16 and FAT32 volumes and 8.3 naming scheme
- Runs on any* underlying flash memory (SD, SPI flash, etc.)
- Single-Buffer design (512 Byte)
- Compatible to Contiki File System API
- Extended API for additional functionality

Limitations

- No support for „long filenames“
- Keeping FATs in sync is “expensive”

RATFAT Architecture



RATFAT RT Overview

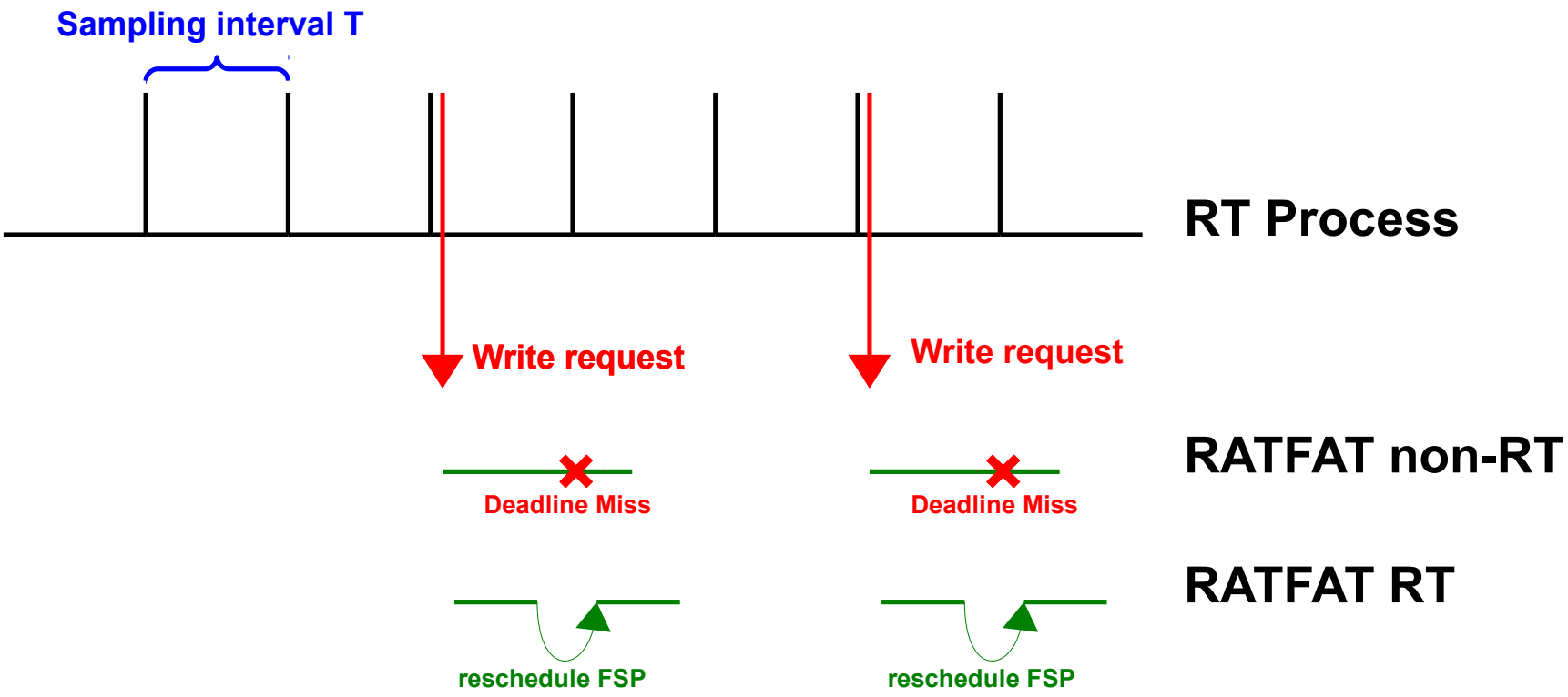
API

- Non-blocking API
- Enqueue file system requests in constant time

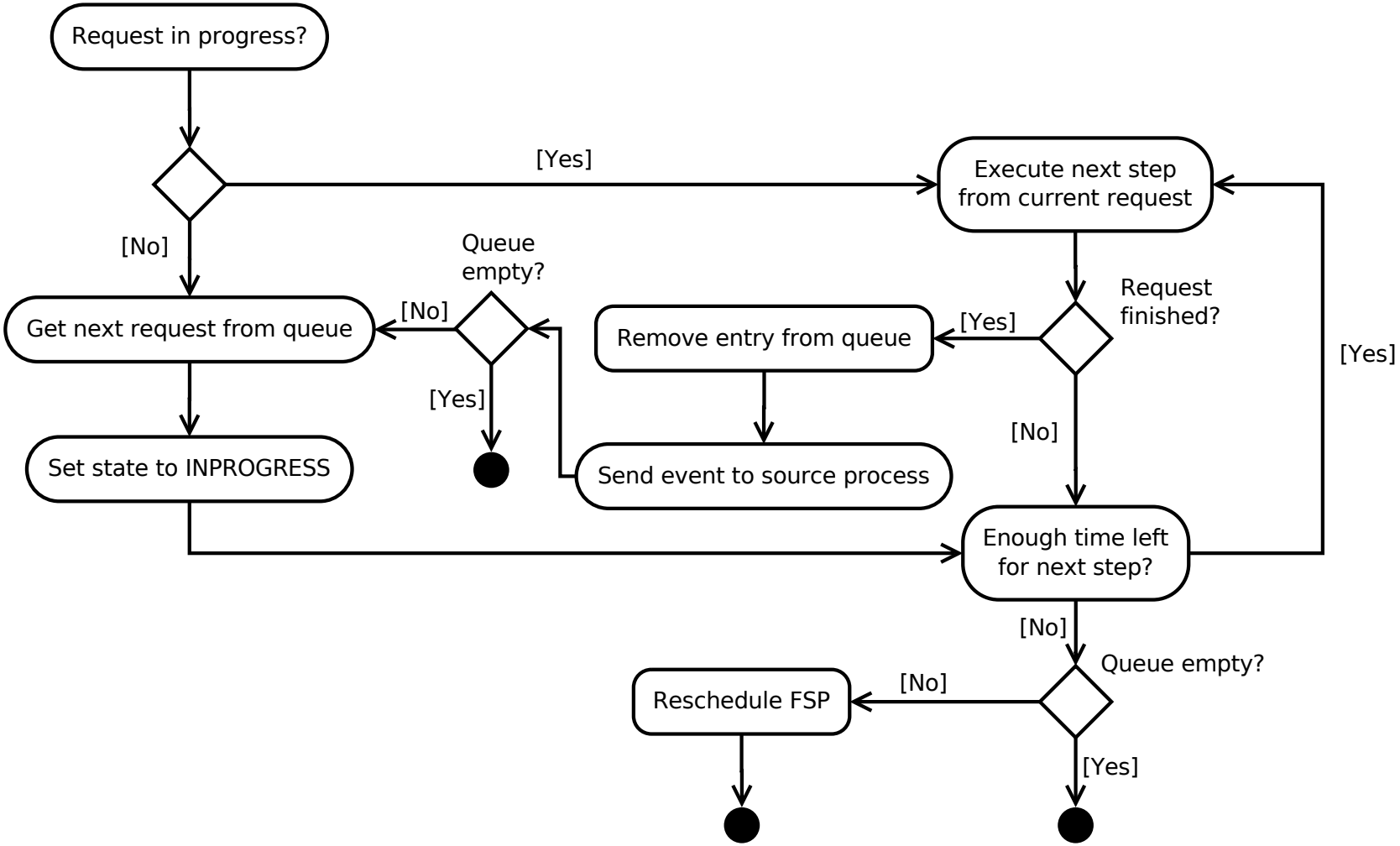
File System Process (FSP)

- Background process handling all file system requests
- Separate Stack, allows to hand control back to scheduler
- Splits up FS requests into multiple atomic operations
  - On INGA, atomic operations take up to 12 ms
- Maximum execution time per invocation
- Respects system timers, yields before

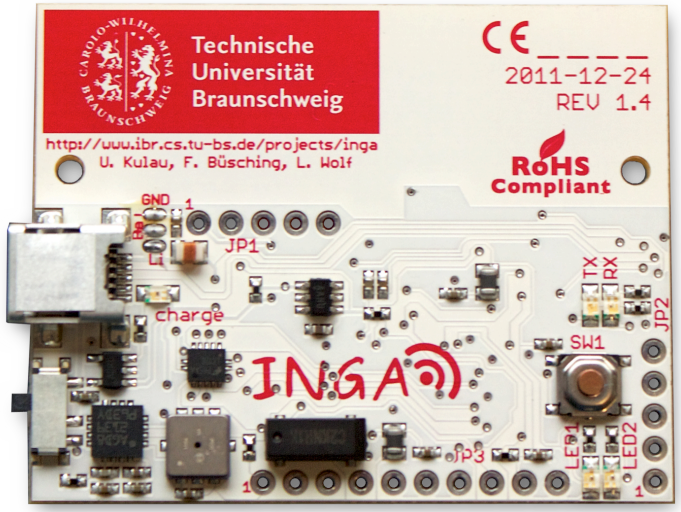
RATFAT RT Concept

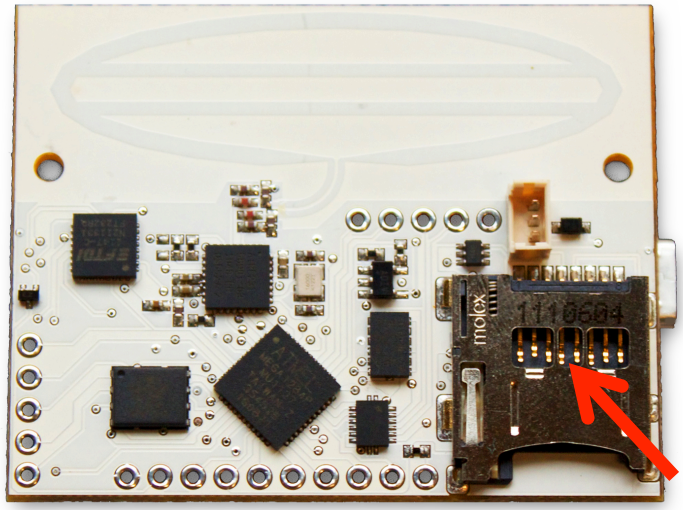


Behavior of the file system process (FSP)



Evaluation Setup





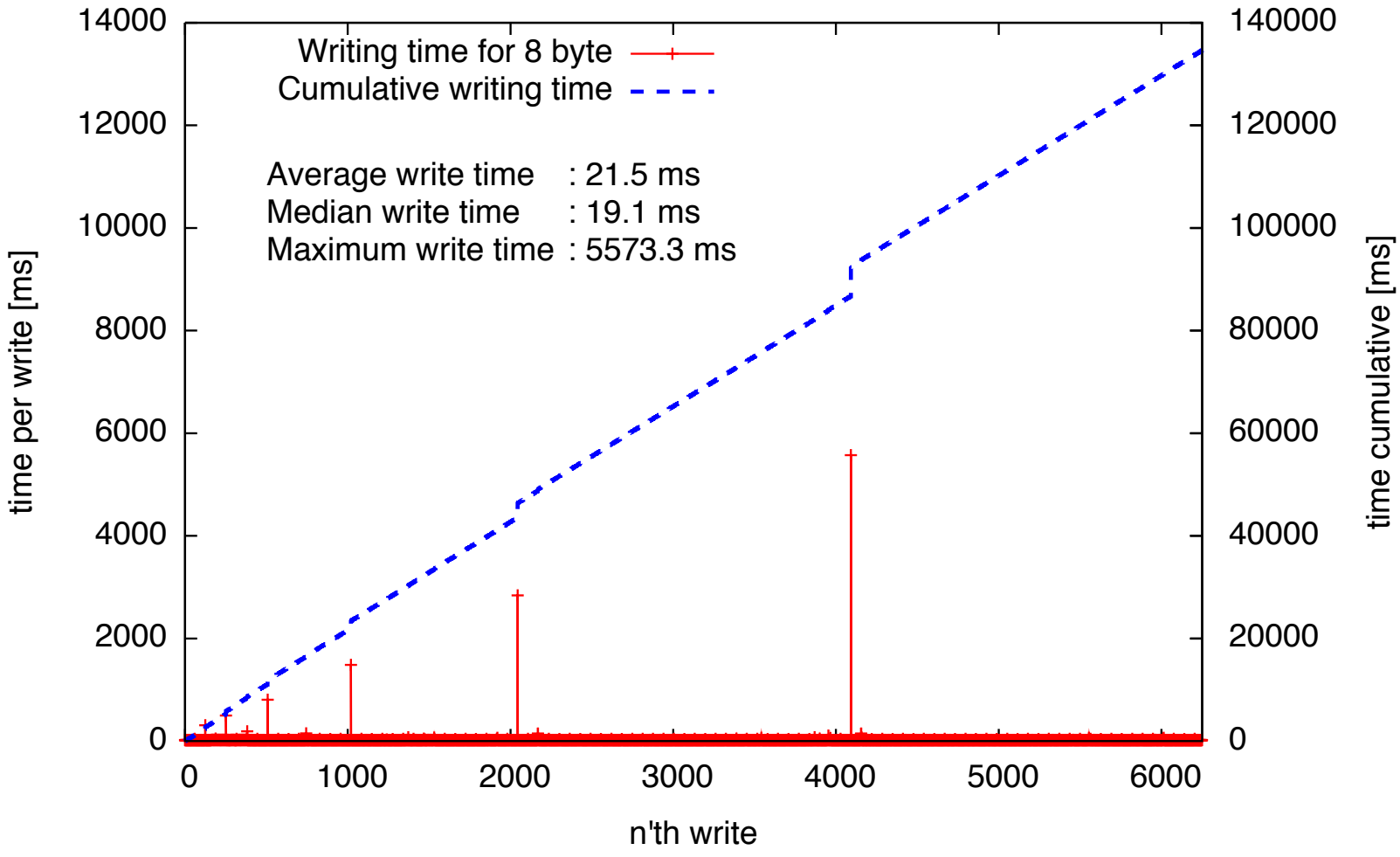
Micro SD
Card Slot

INGA Sensor Node

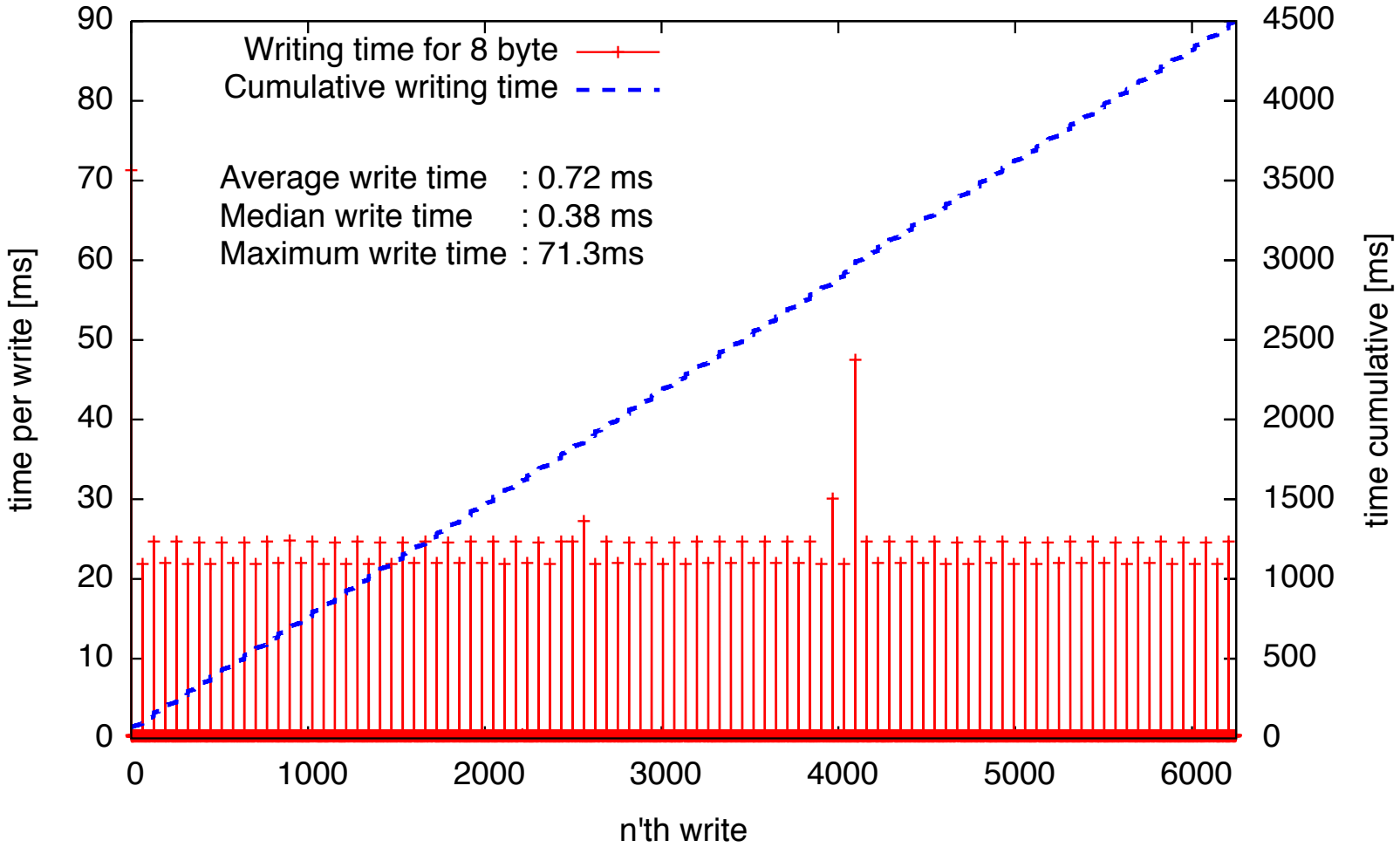
- Atmel ATmega 1284P: 128kb ROM, 16kb RAM

2GB Transcend Micro SD Card

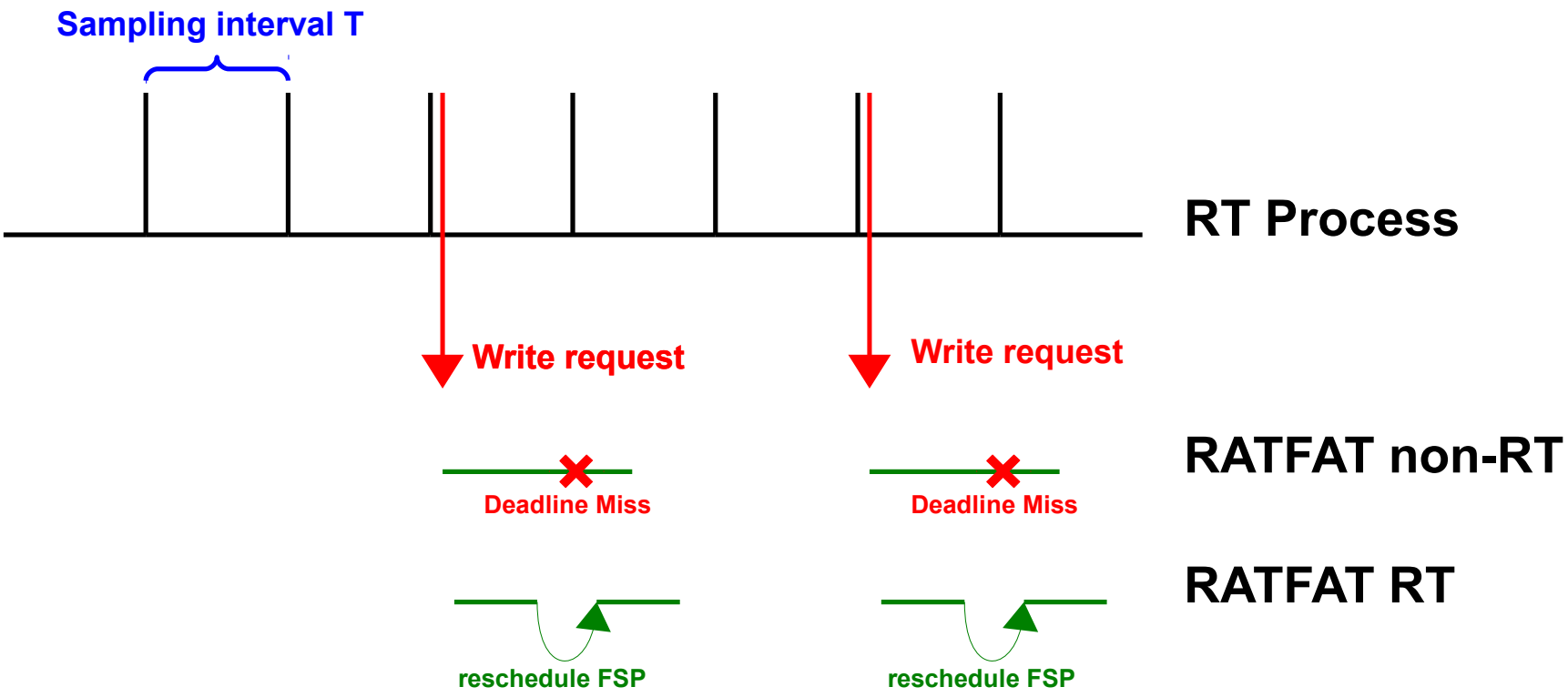
Baseline: Writing n times 8 byte with COFFEE



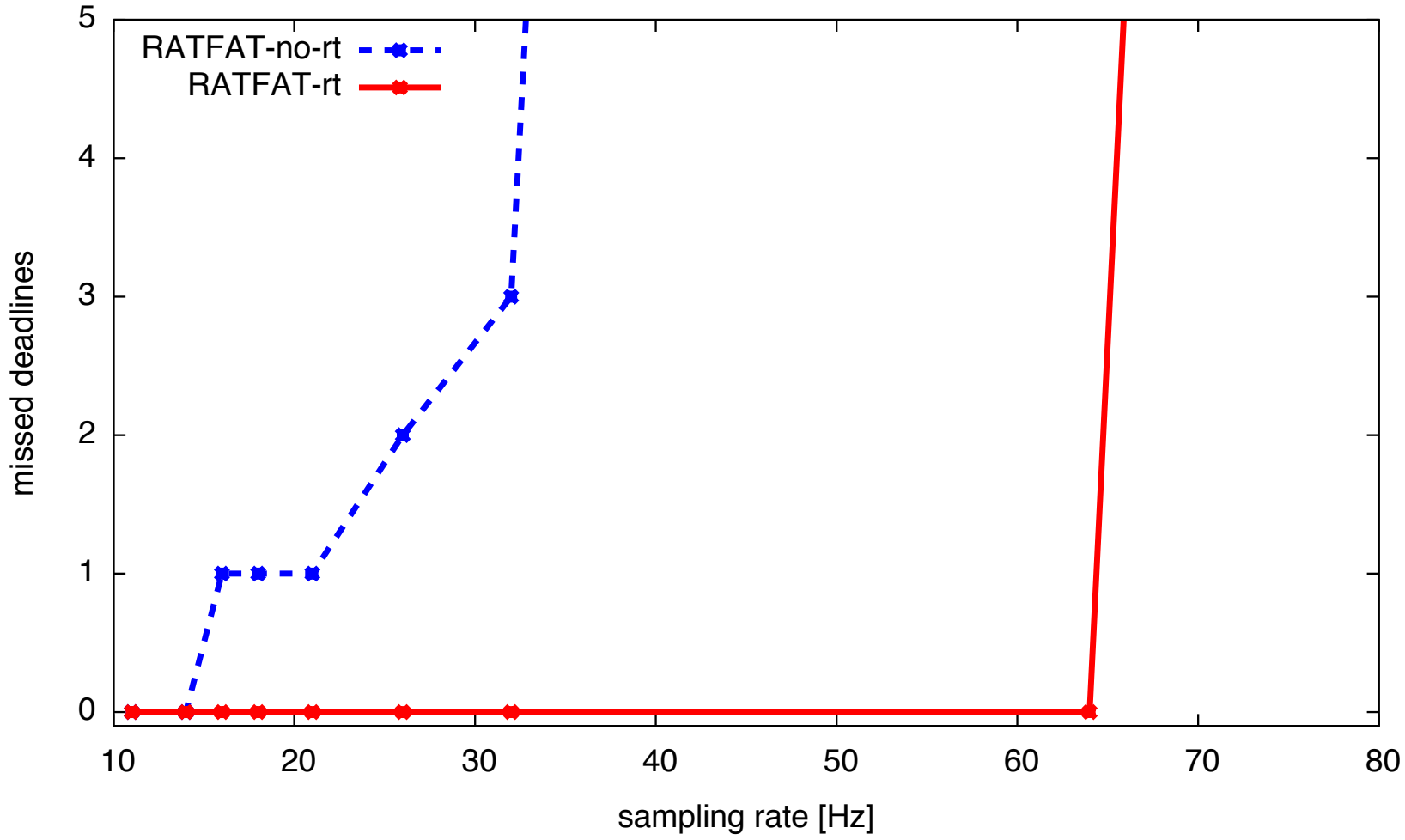
Writing n times 8 byte with RATFAT (non-RT)



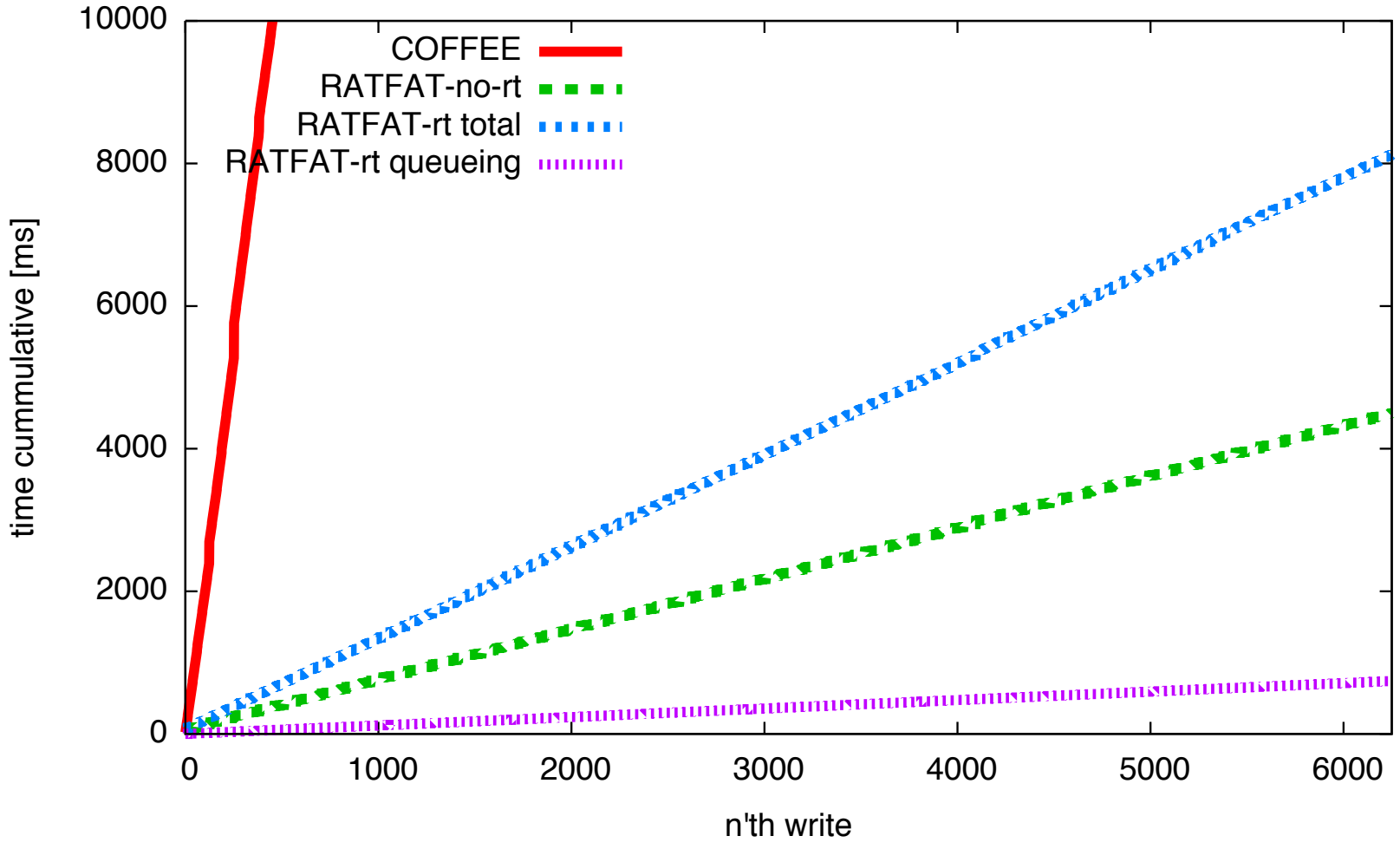
RATFAT RT Measurement Setup



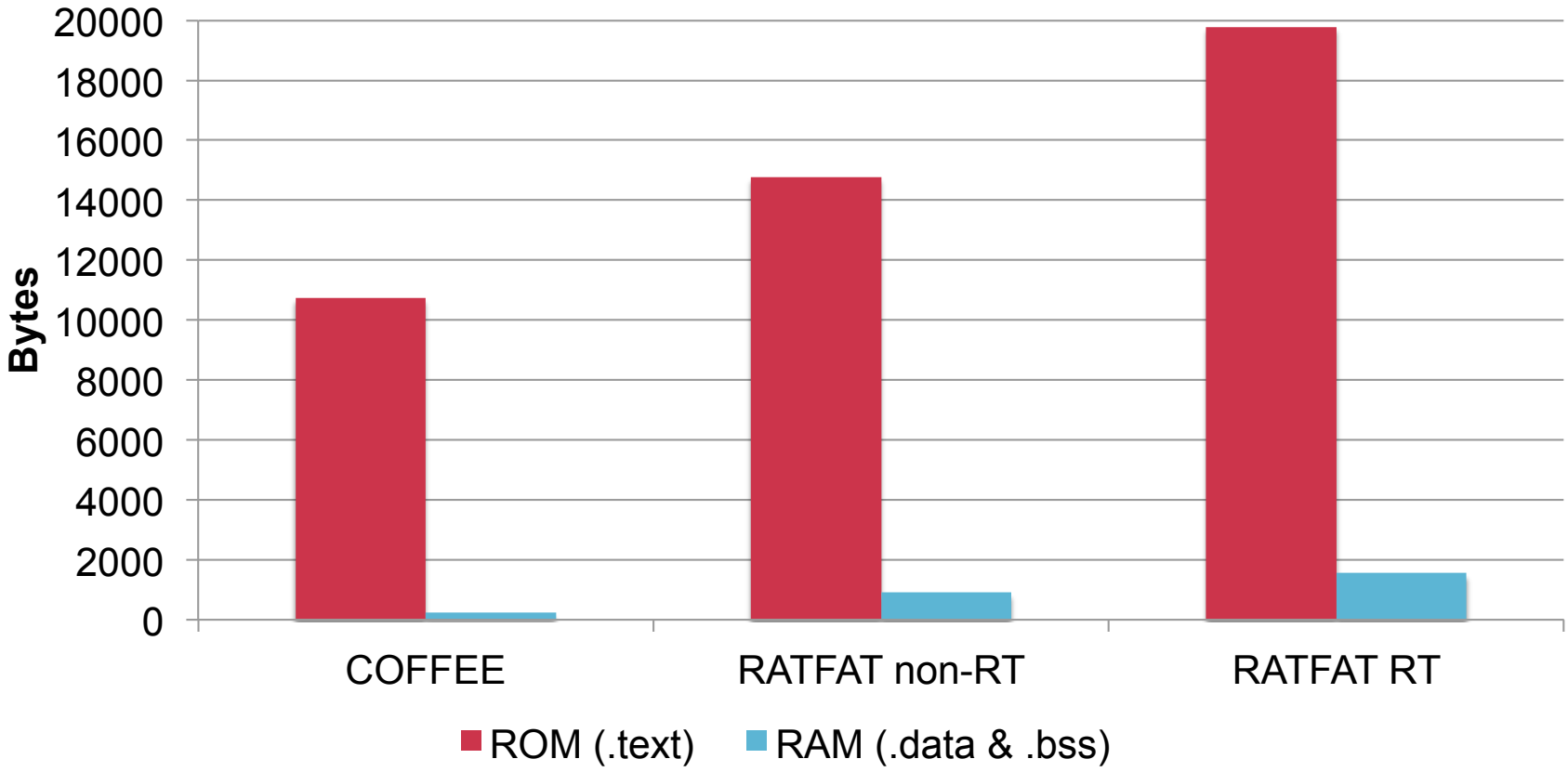
Maximum Sampling rate with RATFAT RT and non-RT



Comparison of COFFEE, RATFAT non-RT and RATFAT RT



Memory Footprint Comparision



INGA: 128kb ROM, 16kb RAM

Sebastian Schildt
schildt@ibr.cs.tu-bs.de
Wolf-Bastian Pöttner
poettner@ibr.cs.tu-bs.de

Conclusion

Motivation

- FAT on SD cards allows exchanging data between node and PC
- Some applications require real-time operations on WSN nodes

RATFAT

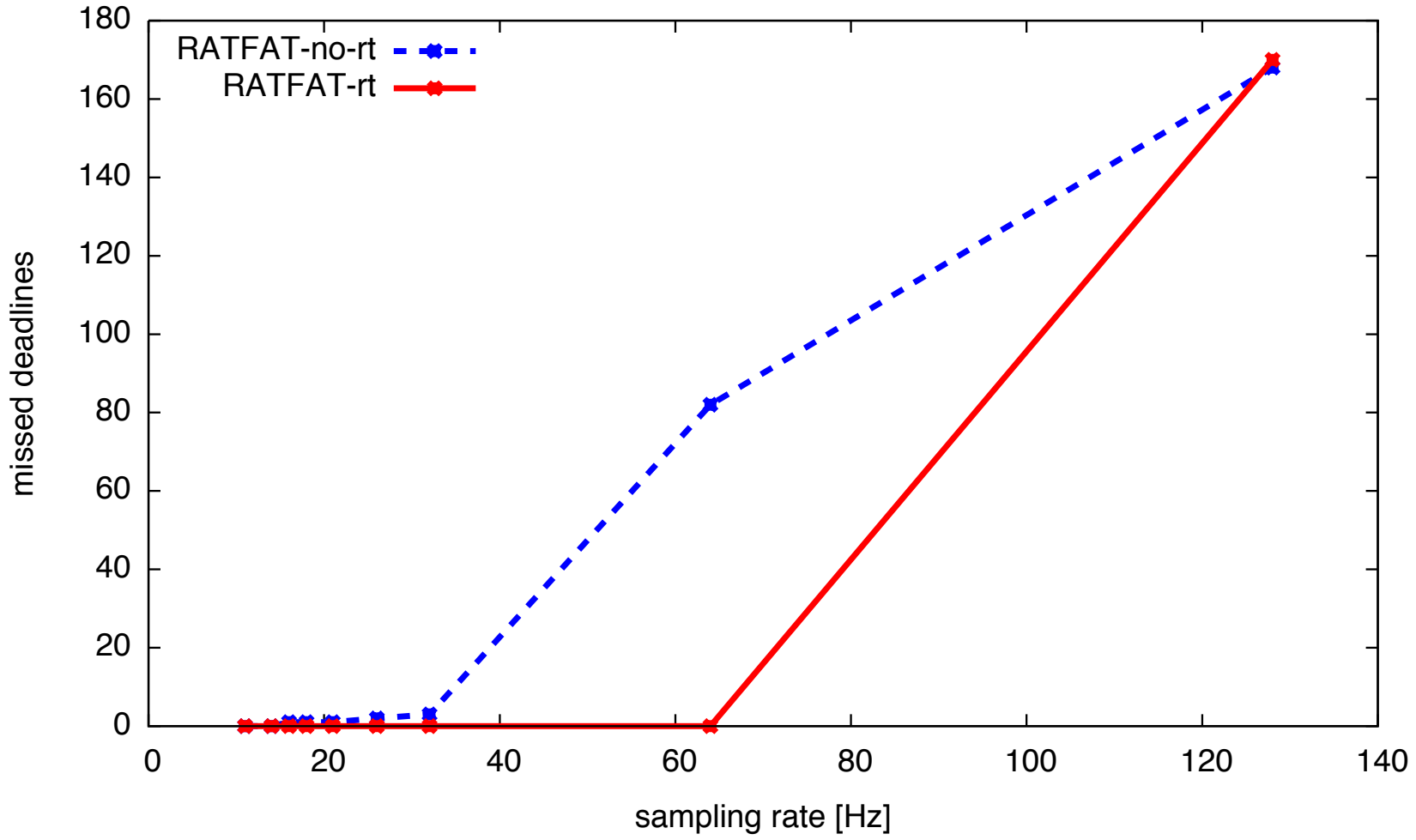
- FAT implementation for Contiki-based sensor nodes
- (Optional) real-time support for cooperative multitasking env.

Results

- RATFAT shows more predictable timing behavior than COFFEE
- Supports real-time tasks up to 64 Hz invocation frequency

Code: http://goo.gl/Ky1Ns

Maximum Sampling rate with RATFAT RT and non-RT



RATFAT Memory Footprint

	ROM (.text)	RAM (.data & .bss)
COFFEE	10,748 Bytes	233 Bytes
RATFAT non-RT	14,772 Bytes	907 Bytes
RATFAT RT	19,786 Bytes	1,556 Bytes
INGA (Atmega 1284p)	**128,000 Bytes**	**16,000 Bytes**

- Measured on INGA (8 bit MCU)