Wireless Sensor Network Operating System Abstraction

Master of Research Thesis Defense

Author: Andreea-Maria Picu
Supervisors: Eric Fleury and Antoine Fraboulet

Ecole Doctorale Informatique et Information pour la Société
Institut National de Sciences Appliquées de Lyon
69621 Villeurbanne Cedex, France

10 September 2007

Outline

1 Introduction

Outline

Outline

1 Introduction

2 Background
- Frequency Scaling
- Battery Discharge Model
- Timer Management

3 Connecting Hardware and Software
- Software Timer Allocation
- Full Platform Configuration
- Hardware Descriptions

Outline

1 Introduction

2 Background
- Frequency Scaling
- Battery Discharge Model
- Timer Management

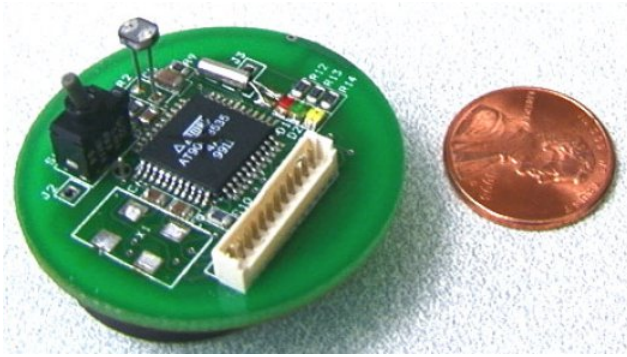
3 Connecting Hardware and Software
- Software Timer Allocation
- Full Platform Configuration
- Hardware Descriptions

4 Conclusion

What Is This About?

Specific mobile battery-powered computing devices, i.e. wireless sensors

- investigate advanced power saving techniques
- optimally exploit possibilities offered by hardware
- per-application power setting

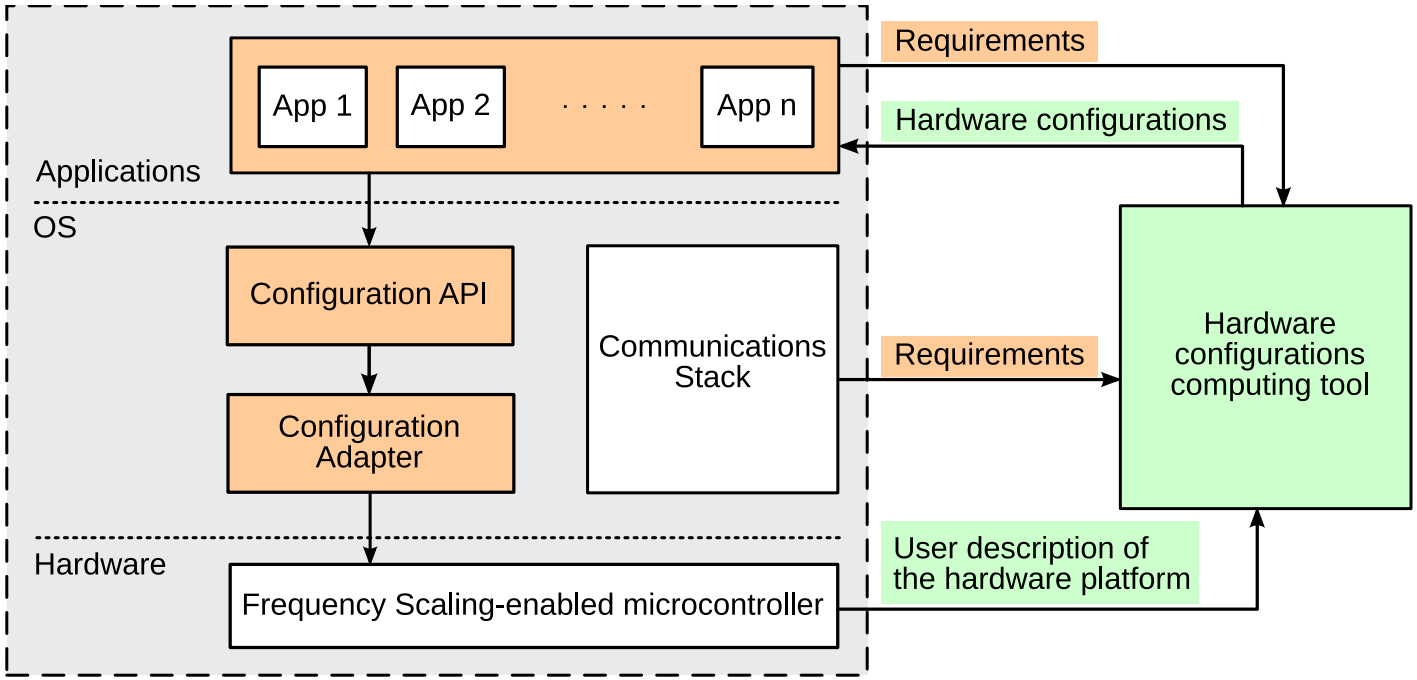


Motivations

1. Power management is a major challenge in WSNs
   - battery is not replaceable
   - research advances in battery size and capacity are very limited
   - performance constraints are constantly increasing
2. The bulk of the research on energy awareness concerns communication protocols
   - Computation, memory access or peripherals are also power-greedy

$\rightsquigarrow$ Need for a *holistic approach* that will give applications direct, fine-grain control over underlying hardware.

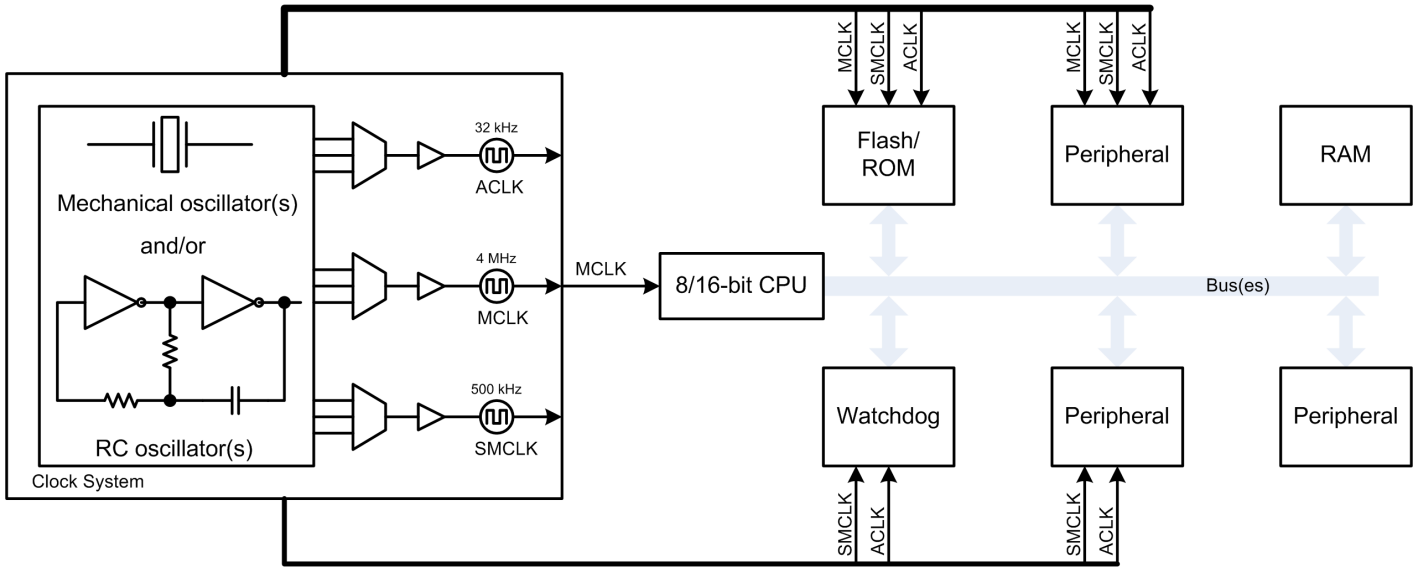
Objectives



Set up a novel power management infrastructure for sensor platforms, based upon:

- frequency scaling
- better use of hardware resources

Microcontroller Basics



- Peripherals choose among a set of several clocks
- The clocks themselves are the result of a multiplexing of several clock generators

$\rightsquigarrow$ Scaling the frequency will influence the entire clock system and peripherals using clocks

Frequency Scaling

- Processor clock is reduced by some multiple of the maximum
- Energy dissipation is minimised linearly

$$P = \alpha C V^2 F$$

P power consumed, α activity factor, C switched capacitance, V supply voltage, F frequency

- Frequency scaling is a necessary condition for voltage scaling
- Combination of frequency scaling and LPMs
  - saves almost as much energy as voltage scaling
  - much less expensive to implement
  - significant positive impact on battery capacity

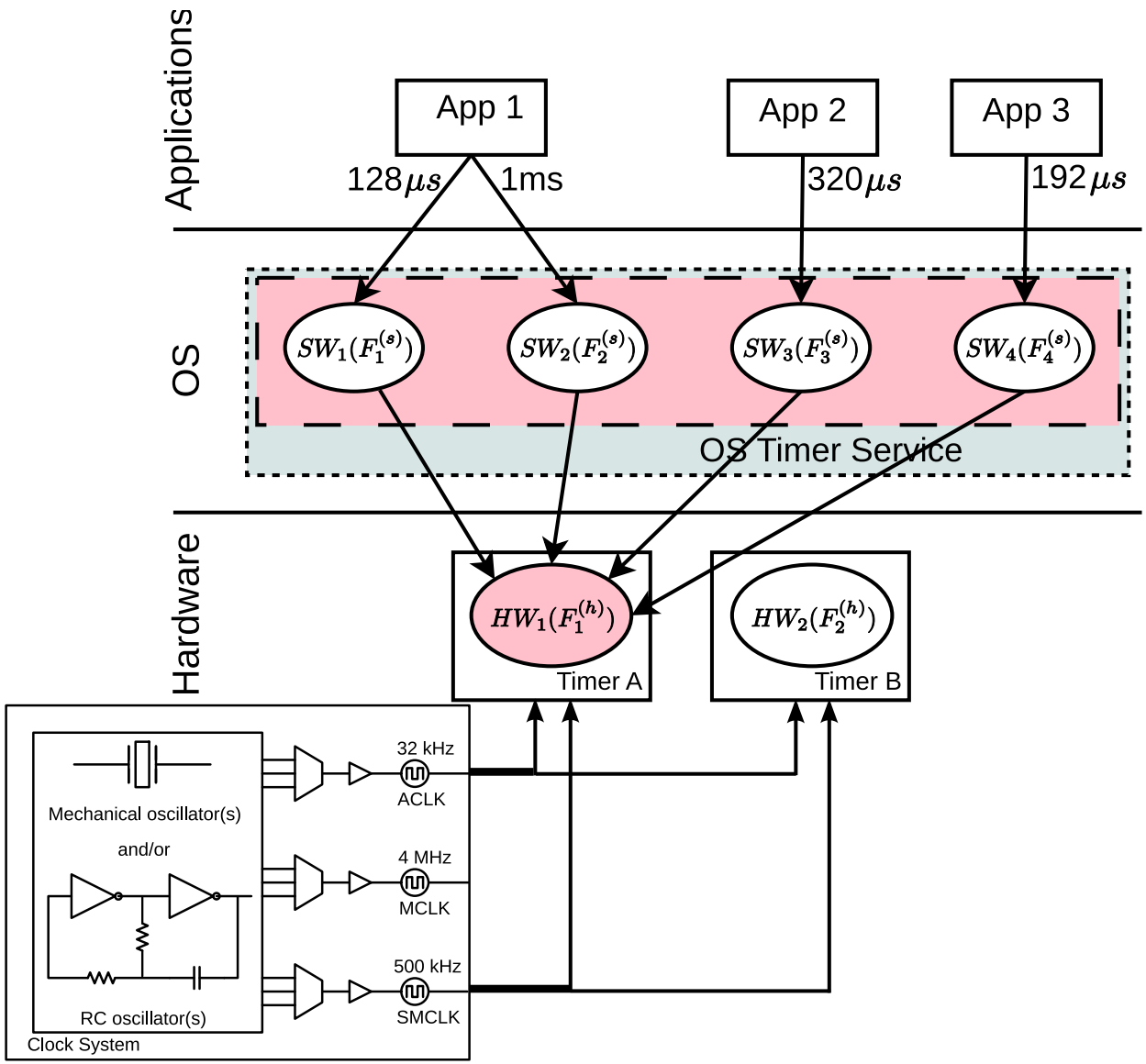
Battery Discharge Model

- Peak power drawn determines battery capacity (not average power)
- ($\nearrow$ battery life $\Leftarrow \searrow$ peak power) $>>>$ ($\nearrow$ battery life $\Leftarrow \searrow$ idle power)
- $\nearrow$ discharge rate $\Rightarrow \searrow$ battery capacity
- Battery $\approx$ ideal energy source (at low discharge rates)

The Importance of the Timer Service

- Processor is attached to various peripherals, e.g. timers
- Frequency scaling also has an affect on the energy consumed by these peripherals
- Timer service is a critical part of an OS
  - standard interface to an arbitrary number of timers
  - allow the device to be placed in LPM between timer events
- Wide variety of hardware timers on sensor platforms
  - **Atmel ATmega128**: two 8-bit timers, two expanded 16-bit timers, watchdog
  - **TI MSP430**: two cascadable 8-bit timers, two 16-bit timers, watchdog

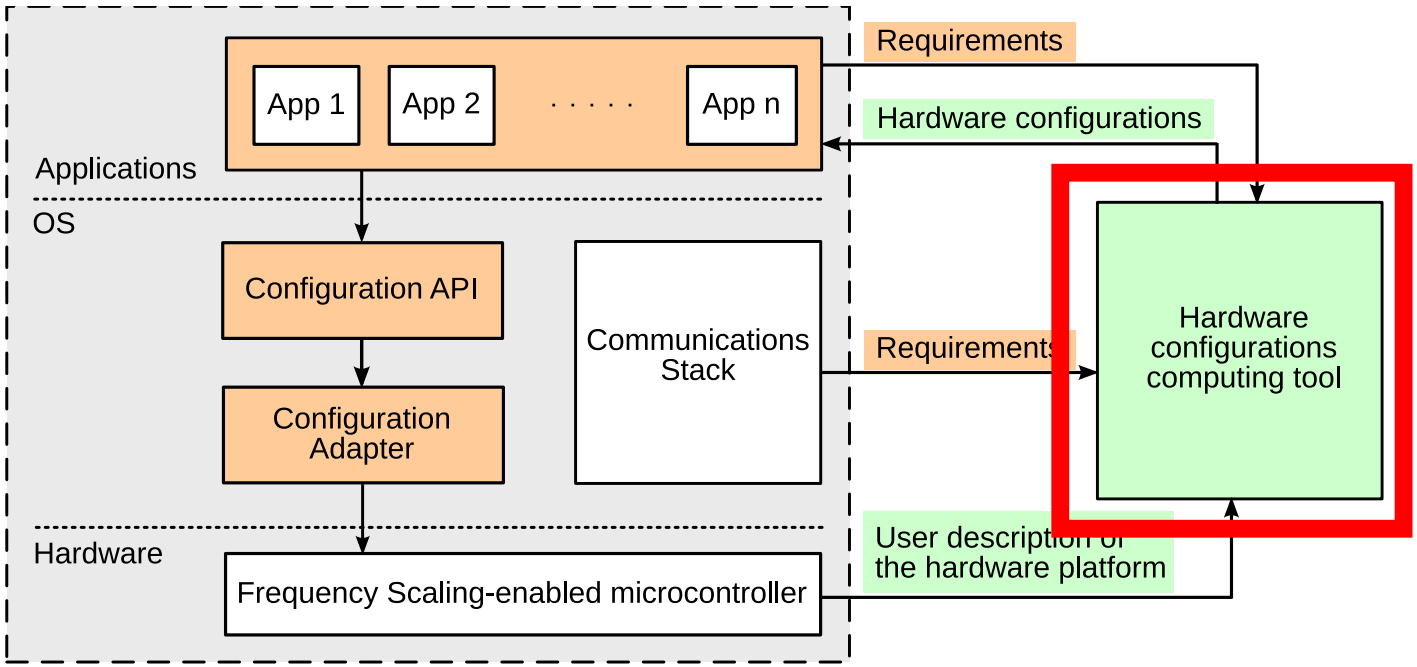
Timer Management in WSN OSs



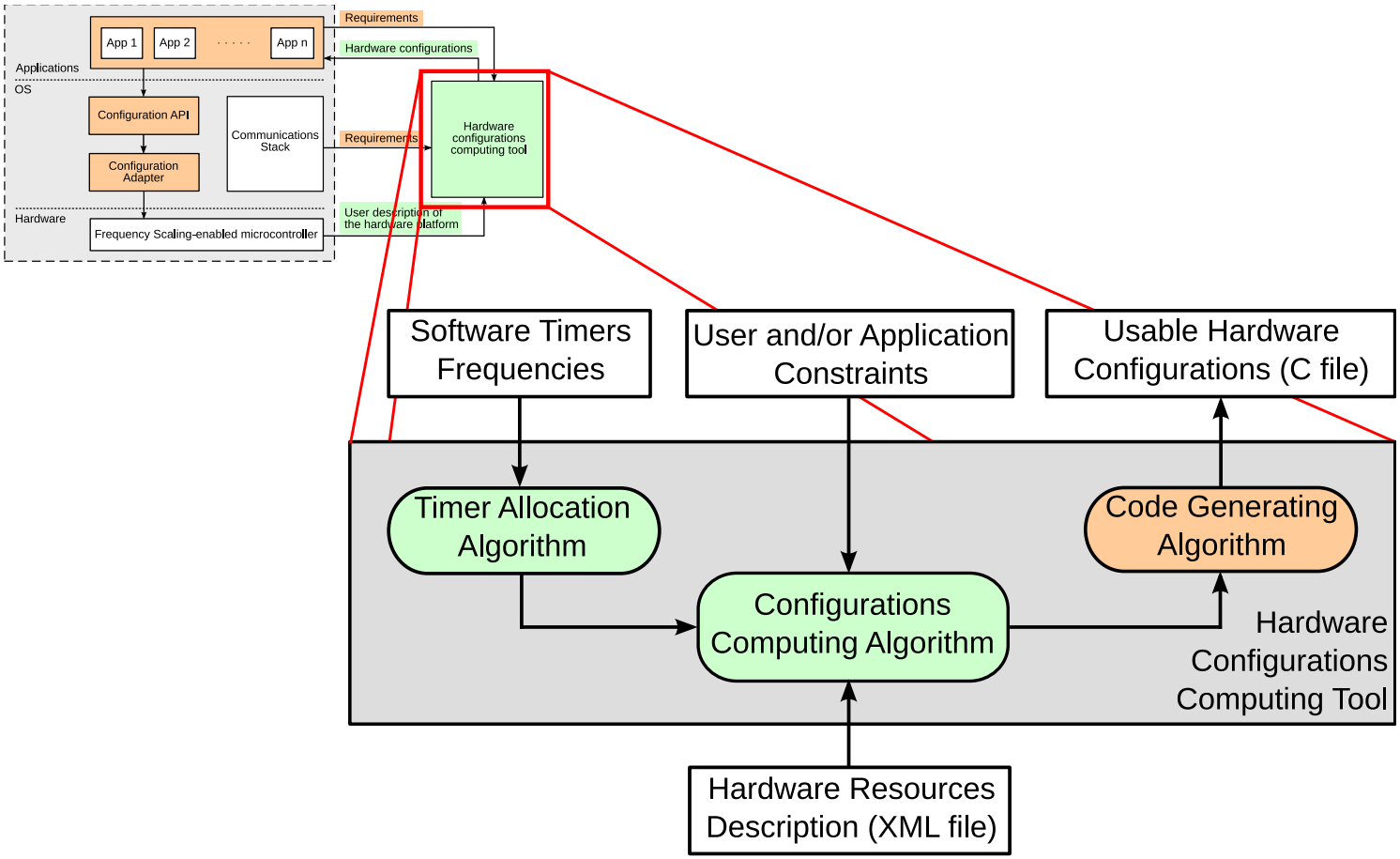
Simple Improvement

- WSN OSs:
  - use only one hardware timer
  - fix the hardware clock frequency at 4 MHz and virtualise software timers
  - hardware clock frequency must be relatively high for decent precision
- Possible improvement:
  - keep using only one hardware timer...
  - ... but, calculate the minimum hardware clock frequency that will satisfy all software timers
  - example for a set of software timer intervals in μs:
    - {1000 μs, 320 μs, 192 μs, 128 μs}
    - $\{2^3 \cdot 5^3, 2^6 \cdot 5, 2^6 \cdot 3, 2^7\}$
    - GCD = $2^3 \Rightarrow F^{(h)}$ = 125 kHz

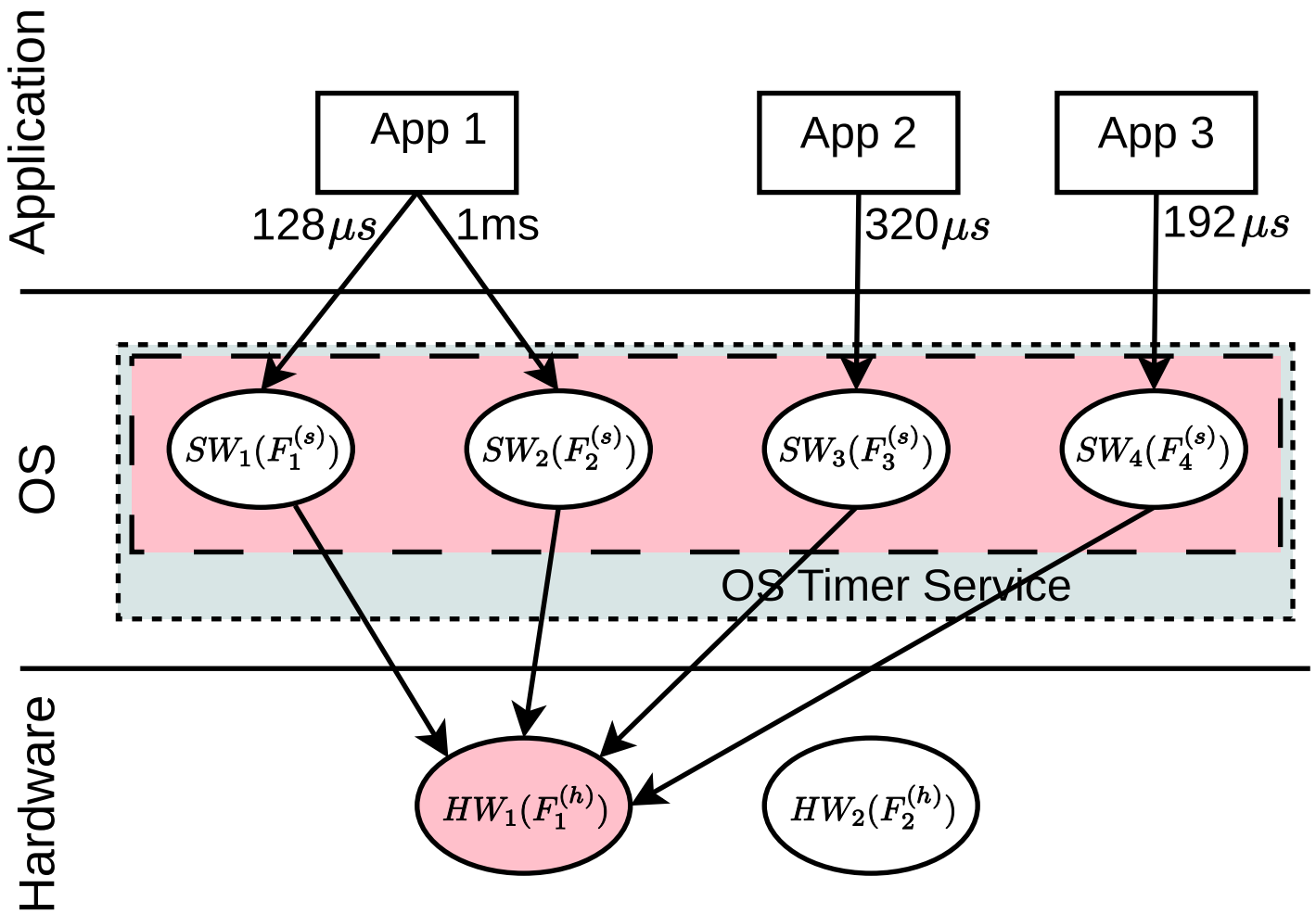
Position in the Infrastructure



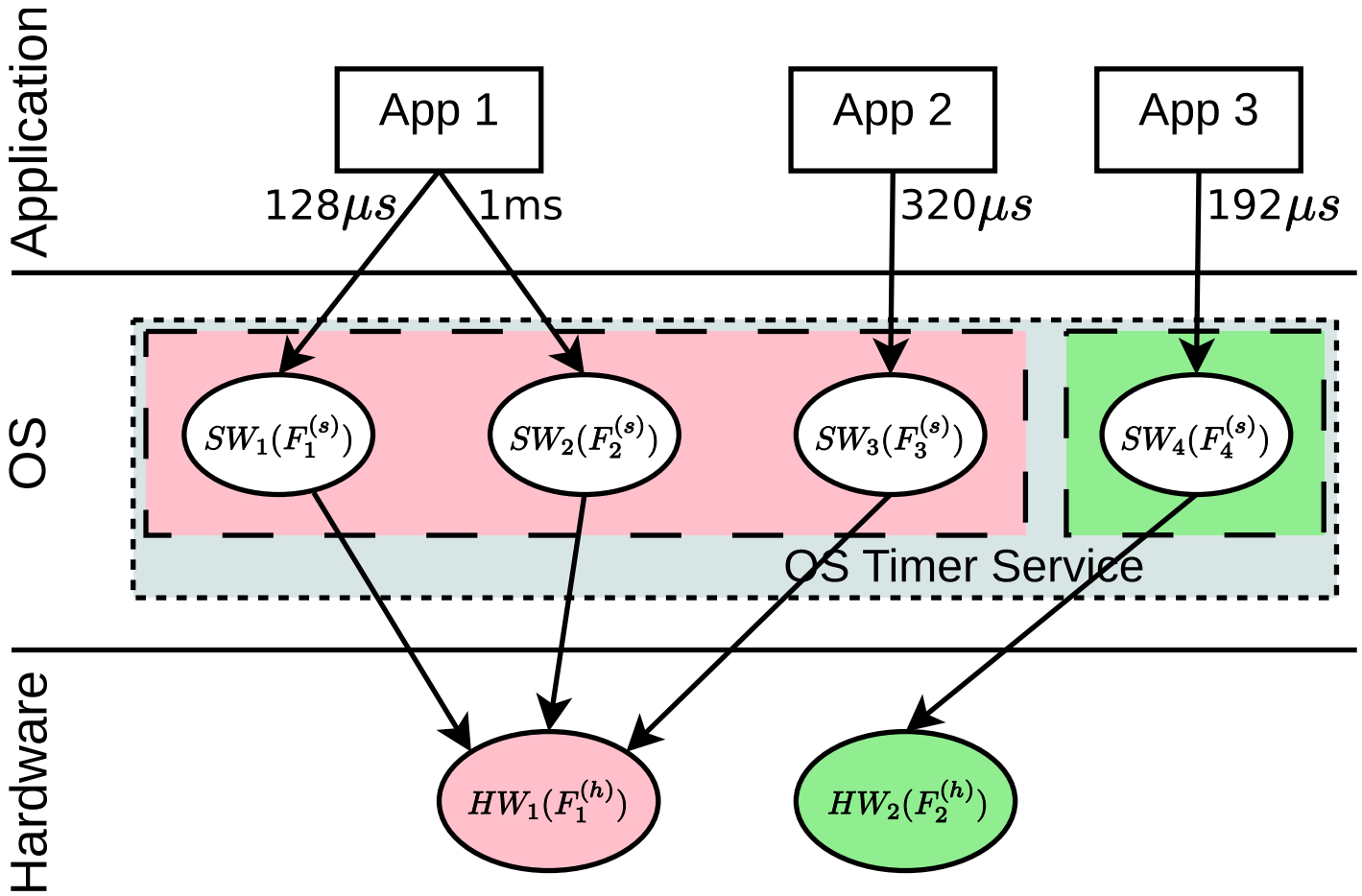
Connecting Hardware and Software



Timer Service in WSN OSs



Target Timer Service



Software Timer Allocation

- Important in determining the minimum operating hardware frequency
- Main goal: use all hardware timers as opposed to popular WSN OSs
- How: separate the single list of software timers from the OSs into smaller lists (one for each hardware timer)
- Two-step algorithm:
  - find proportional time intervals
  - find frequency minimising partition

Separation in Sets of Multiples

Initial set of software timers (μs): 128, 192, 320, 640, 768, 960, 1 000, 1 344, 2 240, 3 000, 9 000, 15 360, 30 720, 64 000, 122 880, 254 000, 491 520, 1 000 000, 1 280 000, 1 600 000, 10 000 000, 15 360 000

Minimum Timer (μs)	Other Timers (μs)

Separation in Sets of Multiples

Initial set of software timers (μs): 128, 192, 320, 640, 768, 960, 1 000, 1 344, 2 240, 3 000, 9 000, 15 360, 30 720, 64 000, 122 880, 254 000, 491 520, 1 000 000, 1 280 000, 1 600 000, 10 000 000, 15 360 000

Minimum Timer (μs)	Other Timers (μs)
128	

Separation in Sets of Multiples

Initial set of software timers (μs): 192, 320, 640, 768, 960, 1 000, 1 344, 2 240, 3 000, 9 000, 15 360, 30 720, 64 000, 122 880, 254 000, 491 520, 1 000 000, 1 280 000, 1 600 000, 10 000 000, 15 360 000

Minimum Timer (μs)	Other Timers (μs)
128	640, 768, 15 360, 30 720, 64 000, 122 880, 491 520, 1 280 000, 1 600 000, 10 000 000, 15 360 000

Separation in Sets of Multiples

Initial set of software timers (μs): 192, 320, 960, 1 000, 1 344, 2 240, 3 000, 9 000, 254 000, 1 000 000

Minimum Timer (μs)	Other Timers (μs)
128	640, 768, 15 360, 30 720, 64 000, 122 880, 491 520, 1 280 000, 1 600 000, 10 000 000, 15 360 000
192	

Separation in Sets of Multiples

Initial set of software timers (μs): 320, 960, 1 000, 1 344, 2 240, 3 000, 9 000, 254 000, 1 000 000

Minimum Timer (μs)	Other Timers (μs)
128	640, 768, 15 360, 30 720, 64 000, 122 880, 491 520, 1 280 000, 1 600 000, 10 000 000, 15 360 000
192	960, 1 344

Separation in Sets of Multiples

Initial set of software timers (μs): 320, 1 000, 2 240, 3 000, 9 000, 254 000, 1 000 000

Minimum Timer (μs)	Other Timers (μs)
128	640, 768, 15 360, 30 720, 64 000, 122 880, 491 520, 1 280 000, 1 600 000, 10 000 000, 15 360 000
192	960, 1 344
320	

Separation in Sets of Multiples

Initial set of software timers (μs): 1 000, 2 240, 3 000, 9 000, 254 000, 1 000 000

Minimum Timer (μs)	Other Timers (μs)
128	640, 768, 15 360, 30 720, 64 000, 122 880, 491 520, 1 280 000, 1 600 000, 10 000 000, 15 360 000
192	960, 1 344
320	2 240, 1 000 000

Separation in Sets of Multiples

Initial set of software timers (μs): 1 000, 3 000, 9 000, 254 000

Minimum Timer (μs)	Other Timers (μs)
128	640, 768, 15 360, 30 720, 64 000, 122 880, 491 520, 1 280 000, 1 600 000, 10 000 000, 15 360 000
192	960, 1 344
320	2 240, 1 000 000
1 000	

Separation in Sets of Multiples

Initial set of software timers (μs): 3 000, 9 000, 254 000

Minimum Timer (μs)	Other Timers (μs)
128	640, 768, 15 360, 30 720, 64 000, 122 880, 491 520, 1 280 000, 1 600 000, 10 000 000, 15 360 000
192	960, 1 344
320	2 240, 1 000 000
1 000	3 000, 9 000, 254 000

Which optimisation function?

- Optimal partition $\Leftrightarrow$ minimum hardware frequency for each cluster of software timers
- Hardware frequency of a cluster of software time intervals = multiplicative inverse of the GCD of the time intervals
- Minimise hardware frequencies of all clusters $\Leftrightarrow$ minimise the sum of all frequencies

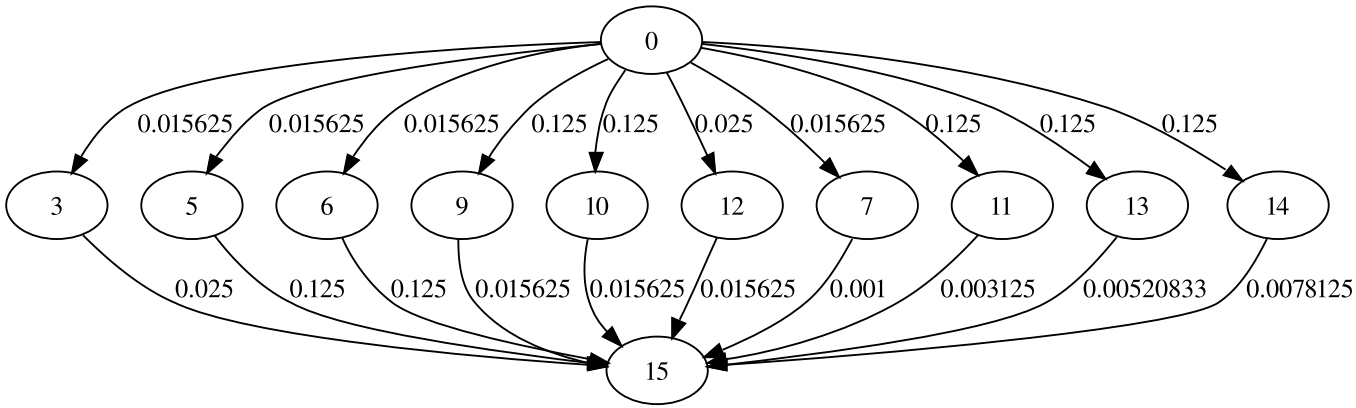
$$W(\mathcal{P}(sw)) = \sum_{i=1}^{|hw|} \frac{1}{GCD(C_i)}$$

- **NP-complete** optimisation problem

Jensen's Algorithm

New reduced set of software timers (μs): 1000, 320, 192, 128

- Jensen greatly improves total enumeration using distribution forms and other observations



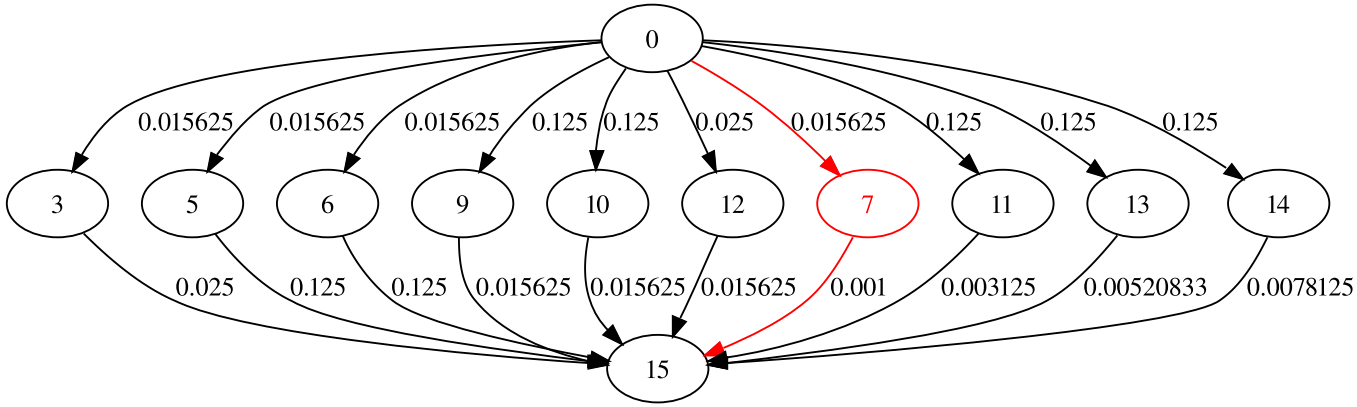
$$13 \equiv 1101 \Rightarrow \{1000, 320, 128\} - \{192\}$$
$$\{2^3 \cdot 5^3, 2^6 \cdot 5, 2^7\} - \{2^6 \cdot 3\}$$

$$GCD_1 = 2^3 \Rightarrow F_1^{(h)} = 125 \text{ kHz}$$
$$GCD_2 = 2^6 \cdot 3 \Rightarrow F_2^{(h)} = 5.208 \text{ kHz}$$

Jensen's Algorithm

New reduced set of software timers (μs): 1000, 320, 192, 128

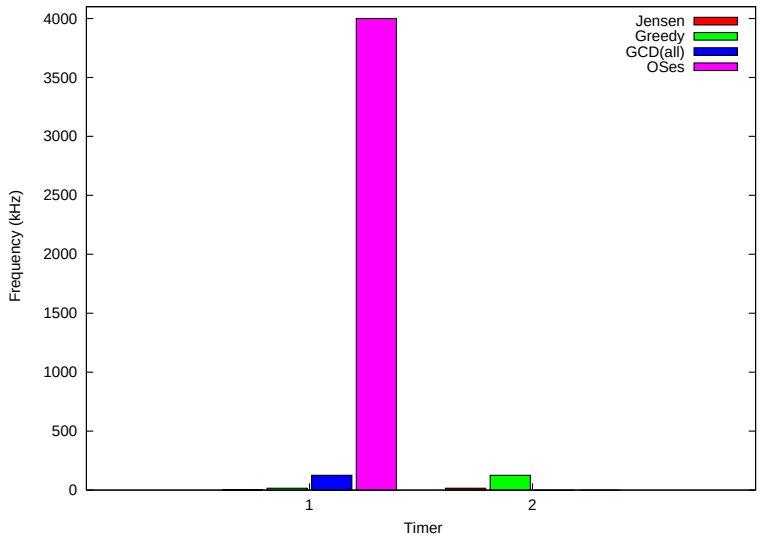
- Jensen greatly improves total enumeration using distribution forms and other observations

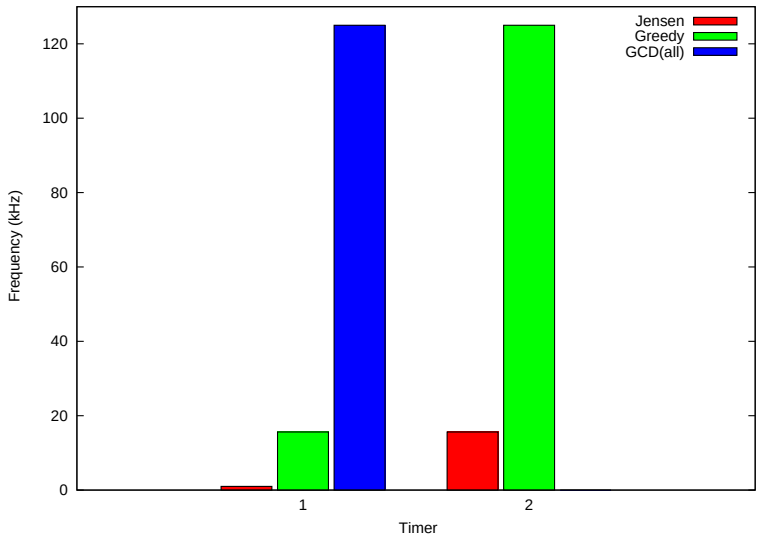


$$7 \equiv 0111 \Rightarrow \{320, 192, 128\} - \{1000\}$$
$$\{2^6 \cdot 5, 2^6 \cdot 3, 2^7\} - \{2^3 \cdot 5^3\}$$

$$GCD_1 = 2^6 \Rightarrow F_1^{(h)} = 15.625 \text{ kHz}$$
$$GCD_2 = 2^3 \cdot 5^3 \Rightarrow F_2^{(h)} = 1 \text{ kHz}$$

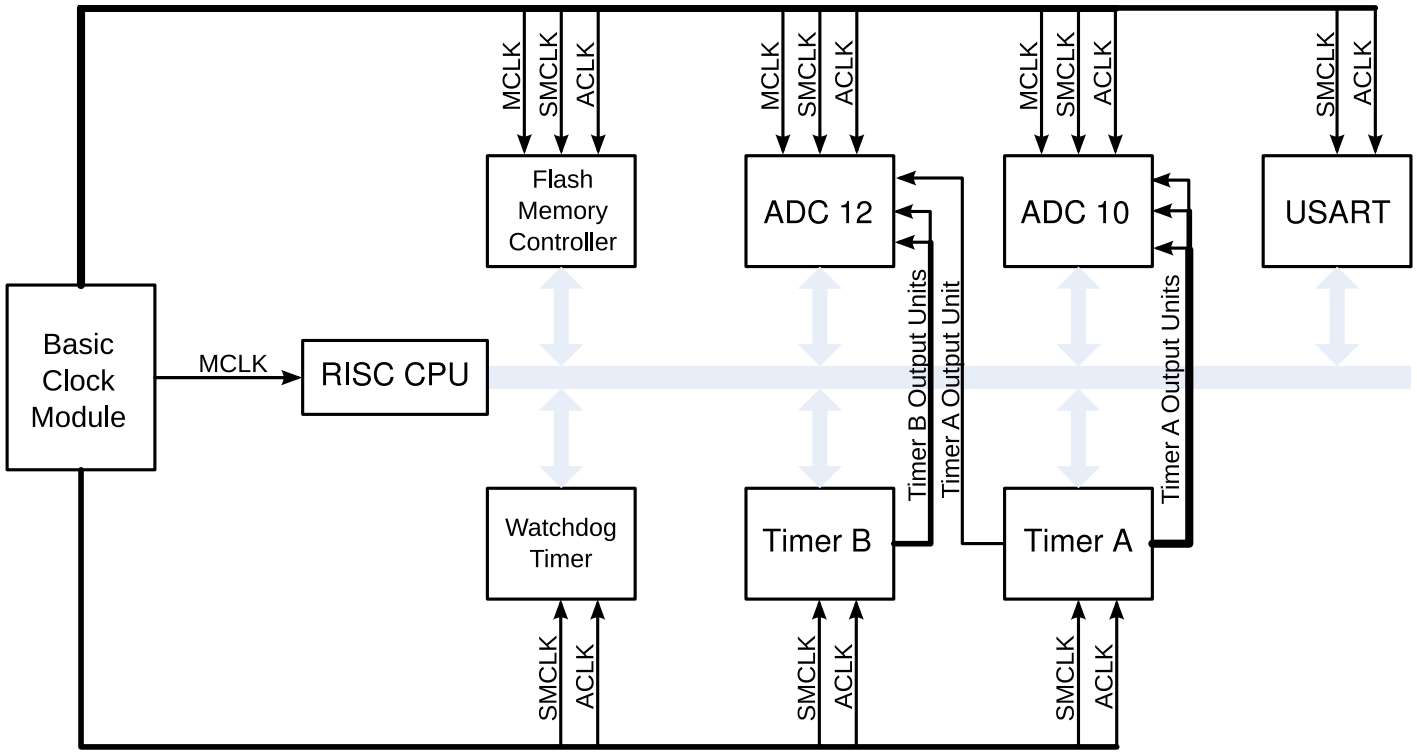
Results





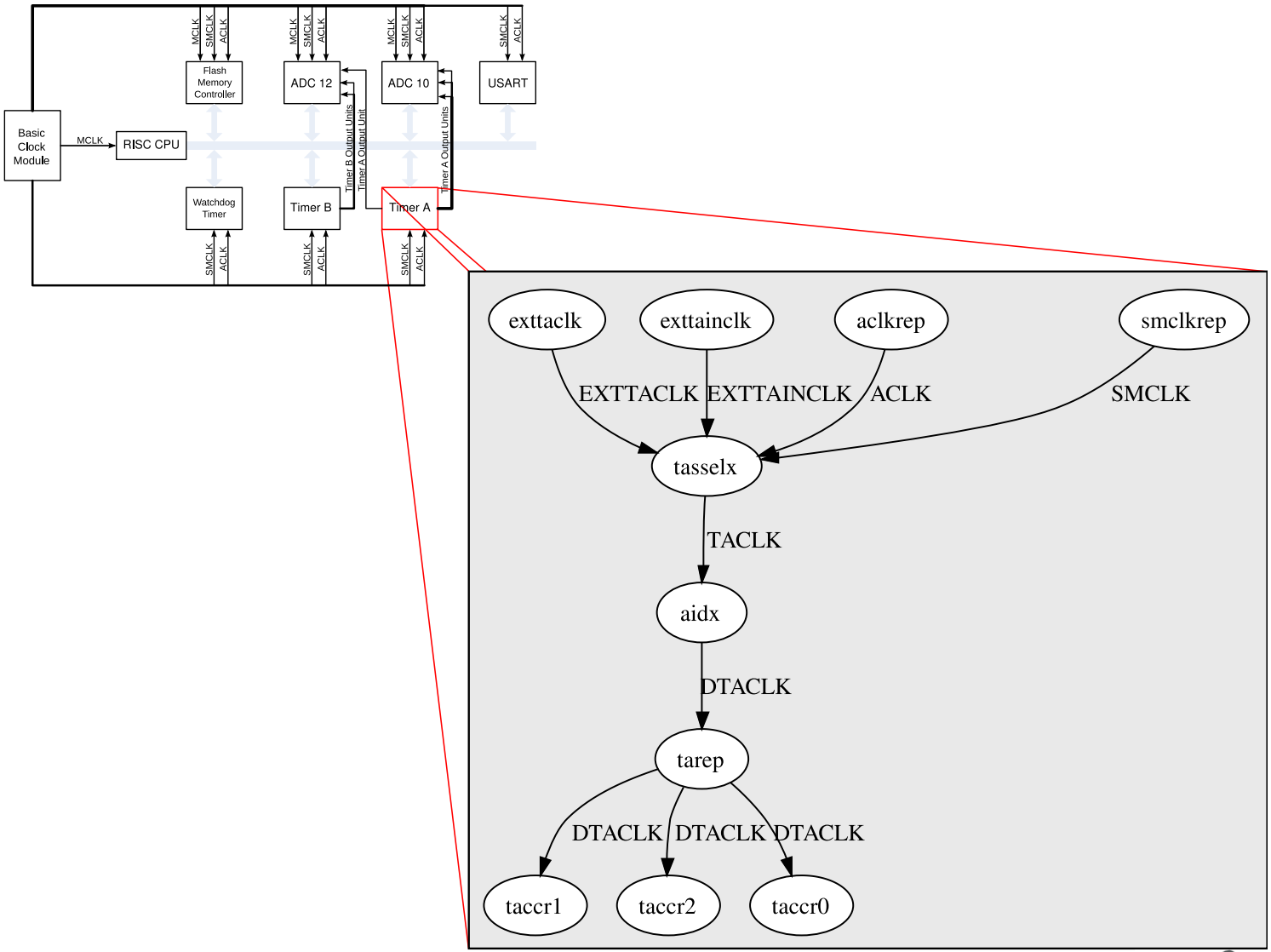
- The hardware timer allocation algorithm versus:
  - Greedy algorithm
  - An algorithm with a single list using the GCD of all software timers
  - The fixed frequency used in WSN OSs

The Platform: TI MSP430

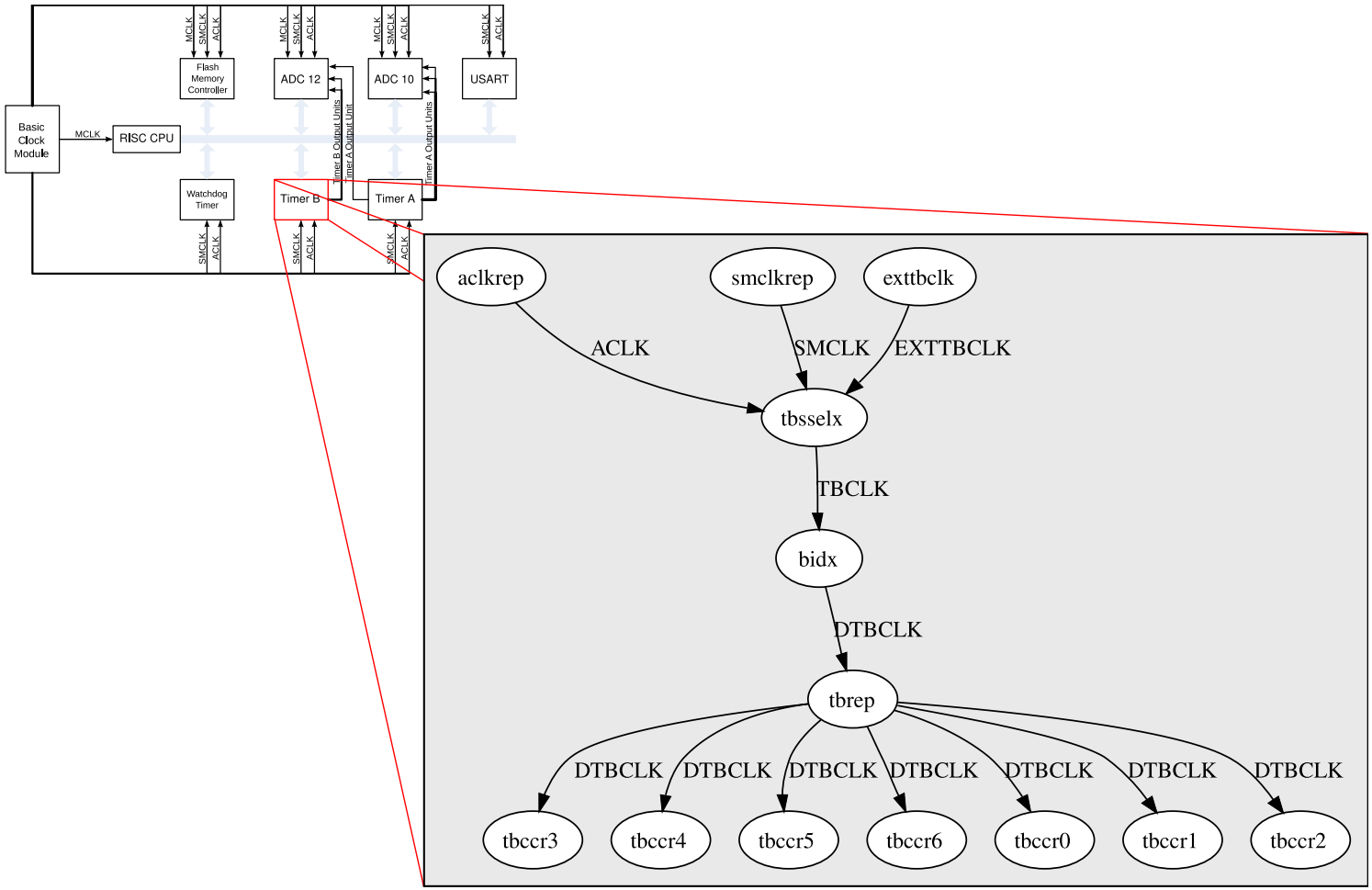


Now that we have the minimum hardware timers operating frequencies, we must compute the frequencies for the clock generating system and configure it

Calculating Configurations



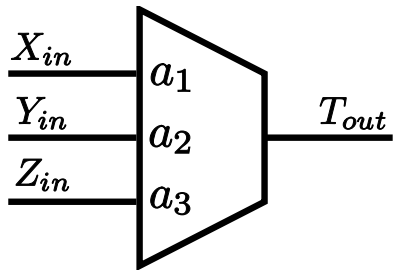
Calculating Configurations



Nonlinear System of Equations

- Registers are modeled as equations or systems of equations
- E.g., selection register or selector:

$$\begin{cases} a_1 X_{in} + a_2 Y_{in} + a_3 Z_{in} = T_{out} \\ a_1 + a_2 + a_3 = 1 \\ a_1, a_2, a_3 \in \{0, 1\} \end{cases}$$



- The whole system is a Mixed-Integer Nonlinear Programming or MINLP Problem

Frequency Optimisation Graph

- Directed connected acyclic annotated graph
- Clock sources are source vertices, dividers are sink vertices
- Allows to find all possible solutions, given a pre-imposed set of constraints
- Examples of constraints:
  - frequency range for clock sources
  - possible values for dividers

Contributions and Future Work

- Two complementary methods to reduce
  - power drawn from the sensor's battery
  - power dissipation within the microcontroller
- Future work
  - merge timers with near values
  - software timers description
  - development of a code generator
  - confirm efficiency experimentally

The End

Thank you for your attention.

Questions ???