Designing a Power Efficiency Framework for Battery Powered Systems

Dacian TUDOR and Marius MARCU

"Politehnica" University of Timisoara, Romania

Agenda

- Introduction & Motivation
- Power Efficiency Concepts
- System Profiling Experiences
- Framework Architecture
- Conclusions & next steps

Introduction

General Aspects

- System design implies a tradeoff between performance and power consumption.
- Software components and applications have a significant influence on overall power consumption.
- Power management has become one of the key challenges in the design of battery powered mobile devices.

Power Management Approaches

- **Dynamic Power Management** (DPM) algorithms: time-out based; behavior prediction; random (simulations);
- **Dynamic Voltage and Frequency Scaling** (DVFS): dynamically altering the processor voltage and frequency (e.g. IBM's PowerTune, AMD's PowerNow and Intel's Enhanced SpeedStep)

Motivation

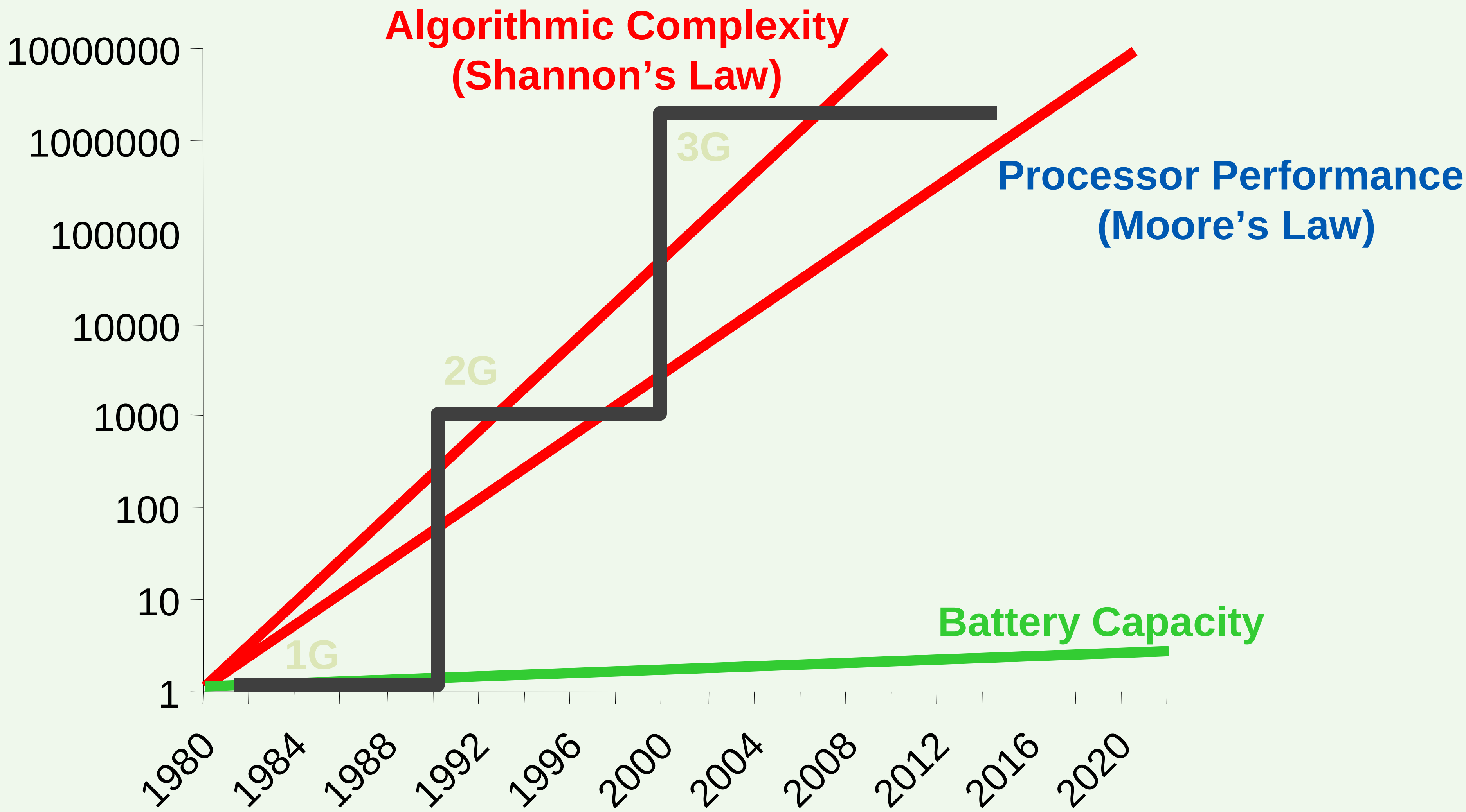
The problem: assign power consumption values to user-level applications which are making use of different power consuming components.

Example:

- Two applications: a music player and a web browser
- User’s desire: to keep both applications running for the next two hours.
- Supposing that audio decoding has negligible impact on power consumption (e.g. low power chip) → the player can decide to keep the audio quality unaltered.
- But the browser might decide to disable all video rendering

- Conclusion: without the ability to determine the application’s impact on power consumption, the music player application would have probably decided to lower the audio quality although it was not necessary.

Motivation – Algorithmic driving force *



* As presented in eMuCo Technical Report

Related Work

Related Work

- [1] → understanding applications characteristics is important for designing efficient power management (PM) systems.
- [2] → dynamic compilation technique for scheduling DVFS changes is proposed.
- [3] → a design framework (GreenRT), for developing power-aware soft real-time applications.
- [4] → a unified power management framework is proposed which presents a software architecture for unified hardware and software PM.
- Our work
  - Similar to [4] – share the same goals
  - Major difference: we propose on-the-fly power and system monitoring
  - Generic feedback loop to higher layers

References

[1] Karthick Rajamani, Heather Hanson, Juan Rubio, Soraya Ghiasi and Freeman Rawson "Application-Aware Power Management", IEEE International Symposium on Workload Characterization, IISWC, San Jose, (2006).

[2] Q. Wu, V. Reddi, J. Lee, D. Connors, D. Brooks, M. Martonosi and D. W. Clark "A Dynamic Compilation Framework for Controlling Microprocessor Energy and Performance", Proceedings of the 38th International Symposium on Micro-architecture, MICRO-38, (2005).

[3] Bo Chen, William Pak Tu Ma, Yan Tan, Alexandra Fedorova, Greg Mori "GreenRT: A Framework for the Design of Power-Aware Soft Real-Time Applications", Workshop on the Interaction between Operating Systems and Computer Architecture, WIOSCA 2008, Beijing, China (2008).

[4] Ashwin Iyenggar, Ambudhar Tripathi, Ajit Basarur and Indranil Roy "Unified Power Management Framework for Portable Media Devices", IEEE International Conference on Portable Information Devices, PORTABLE07, (2007).

Power Efficiency Concepts

- We address different power consumption sources, abstracted as “components”.
- Each power consumption source has its own profile or “fingerprint”.
- Profiles can be grouped in power consumption classes: **system (usage)** and **battery (consumption) parameters.**
- Each profile has an associated set of parameters
  - Power state
  - Usage pattern
  - Energy consumption
- Only key states are considered.

Parameters

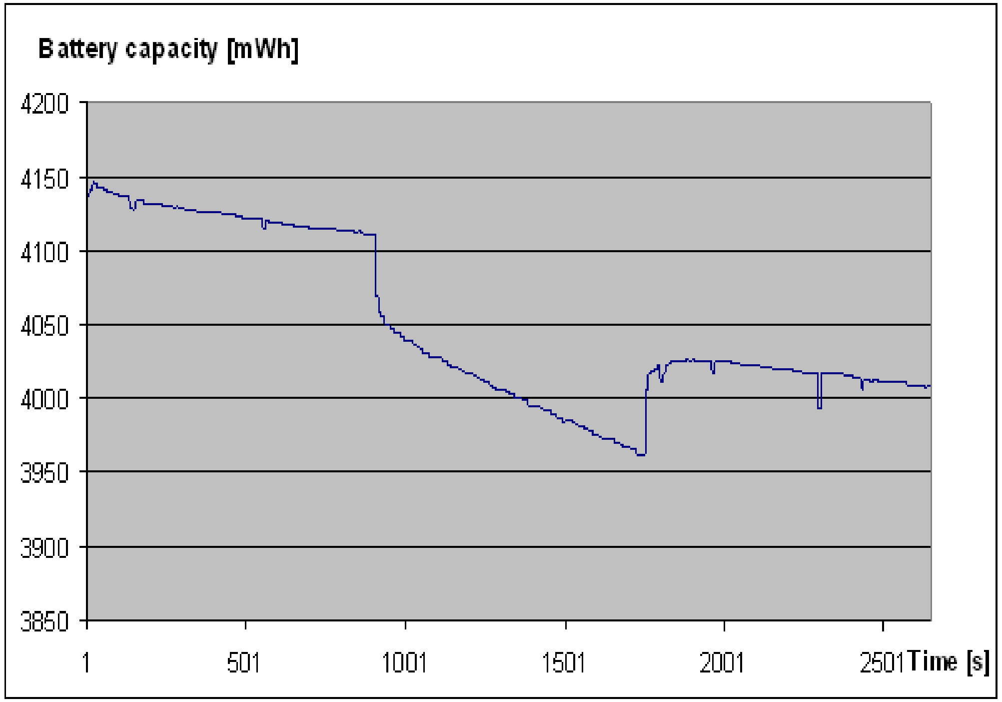
Physical	OS	Application
Battery:	**Battery:**	
I [mA] - current	Power states	
C[mWh]-battery capacity		
V [V] - battery voltage		Workload
P [mW] - discharge rate		type
T - temperature		
CPU:	**CPU:**	Workload size
T - temperature	CPU load	
F - fan speed	CPU counters	Usage
Ps - p-state	Memory	patterns
WLAN:	usage	
RSSI - power strength	Thread info	Threads
Security model	**WLAN:**	count
Display:	bandwidth	
Brightness level	error rate	
Audio:		
Volume level		

Power Efficiency Concepts – Measurements

Battery Parameters

- Higher level metrics:
  - battery discharge
  - battery lifetime
  - battery efficiency
  - battery discharge rate

battery capacity [mWh],
maximum battery capacity [mWh],
charge/discharge rate [mW],
current drawing [mA],
battery remaining life time [s],
battery temperature [oC]



Temperature measurements

- Some hardware components generate more heat as their power consumption increases.
- Temperature indication – hardware dependent - hard to link
- Informal measurement

Power Efficiency Concepts – Framework aspects

Basis	Prerequisites

The framework is based on:

- measurements
- metrics
- system states
- high level power efficiency metrics

The first step is system profiling.

- System decomposition – defining energy consuming components in order to analyze them in isolation.
- Component profiling – a profiling application runs specialized tests which aim to isolate component specific states in a step-like pattern.
- Configuration – power fingerprint knowledge base is stored on the device.
- Monitoring: the framework dynamically identifies the states, logs all monitored data and calculates the power fingerprint.

An optional step: application instrumentation.

- User-provided indications on power component usage (e.g. set power modes).
- Fine grained component split.

System Profiling Experiments

Performed experiments

CPU and memory

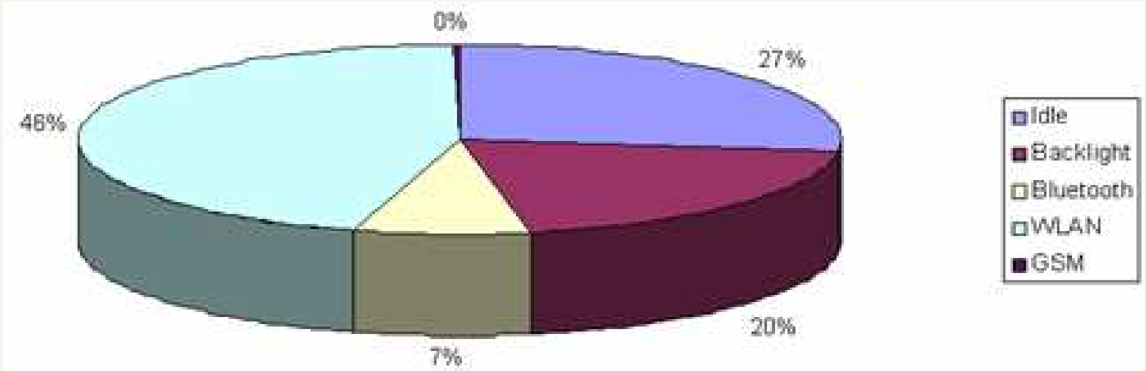
- CPU core temperature
- CPU load
- CPU performance or timestamp counters
- Memory usage
- CPU fan’s rotation speed
- CPU p-state and thread information.

WLAN

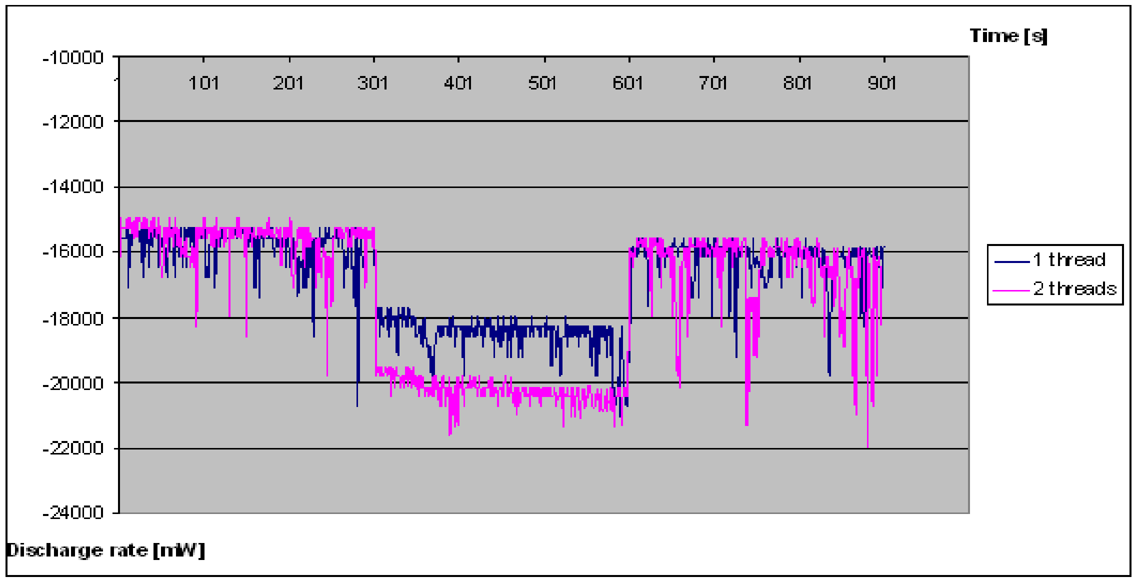
- RSSI (Radio Signal Strength Indicator) of the infrastructure access
- Security model used for wireless connection
- Connection bandwidth and error rates.

Power consumption distribution

- Power consumption distribution among components

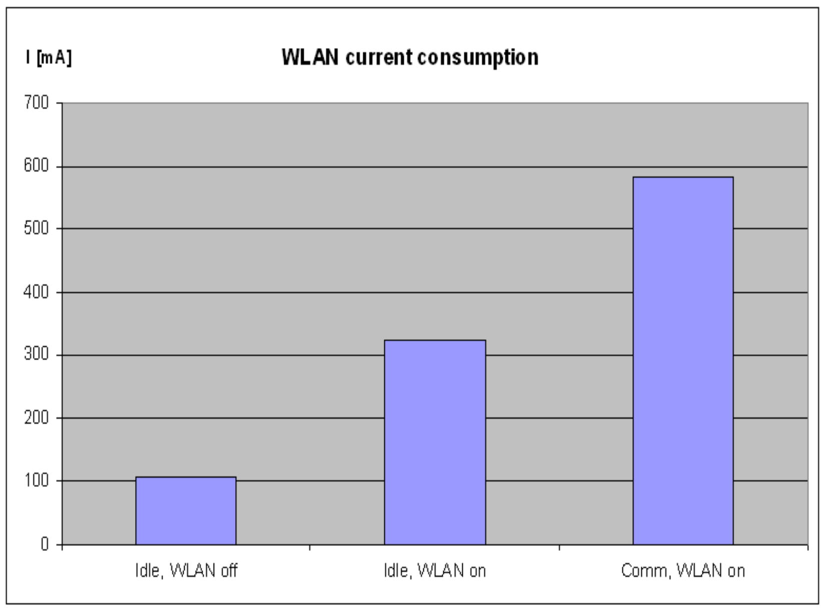


- Power profiles for CPU: integer and memory operations. **System**: Intel Pentium IV dual-core 2.0 GHz mobile processor

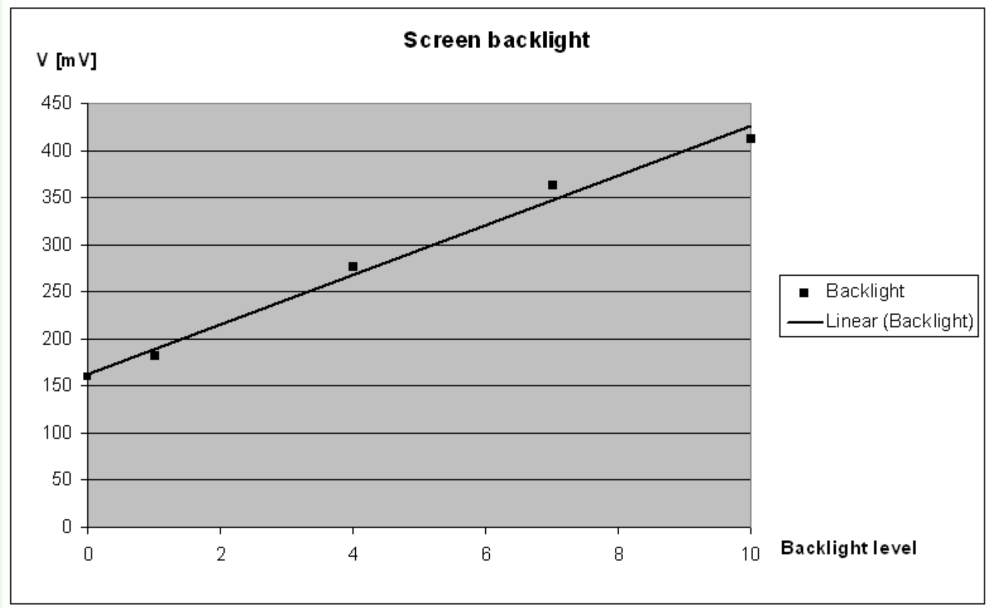


System Profiling Results

WLAN Current Consumption



Screen backlight



- Typically, mobile devices have different luminosity levels for screen backlight.
- The screen backlight power profile was obtained by running the power benchmark for a Fujitsu-Siemens LOOX T830 PocketPC device running Windows Mobile 5.0 OS.
- A linear dependency is observed (as expected).

Framework Architecture

Main goals and Constraints

- to provide
  - power efficiency monitoring services
  - a higher level representation and feedback loop
- to
  - applications
  - operating system or drivers
- **in order to improve the system's power efficiency.**

- **platform agnostic**
- **portable on different systems (or different layers of the system) → implementing a set of abstract interfaces**

Components

Specialized components accountable for power consumption:

- CPU
- Audio
- Display
- WLAN
- Bluetooth
- File System

Feedback loop

- High level domain specific application metrics.
- Predictions of remaining power depending on the history of the system usage.

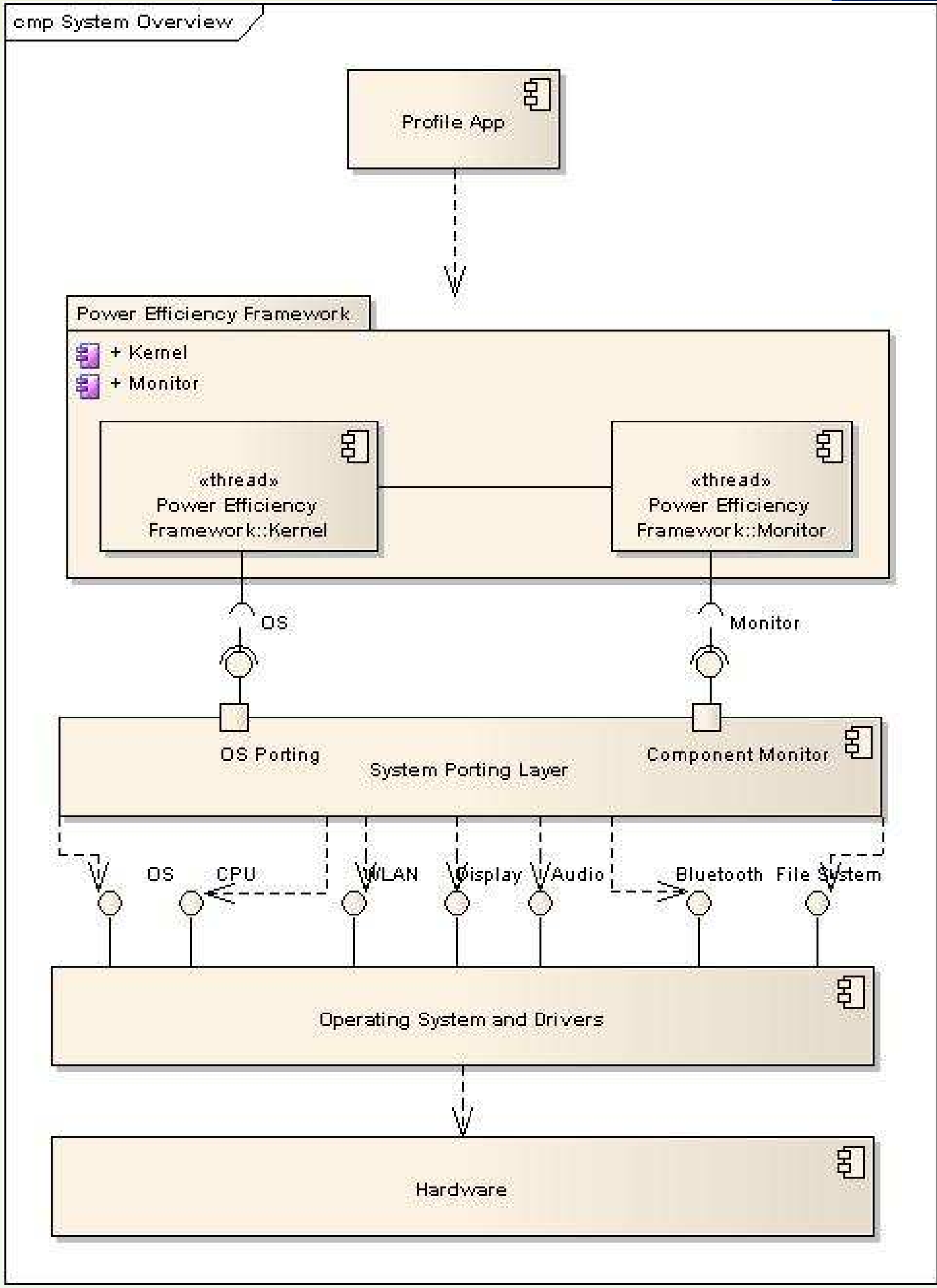
Framework Architecture

Layers

- Layered architecture
- Generic interfaces
- **System porting layer**
- Standard applications on top

Profiling Applications

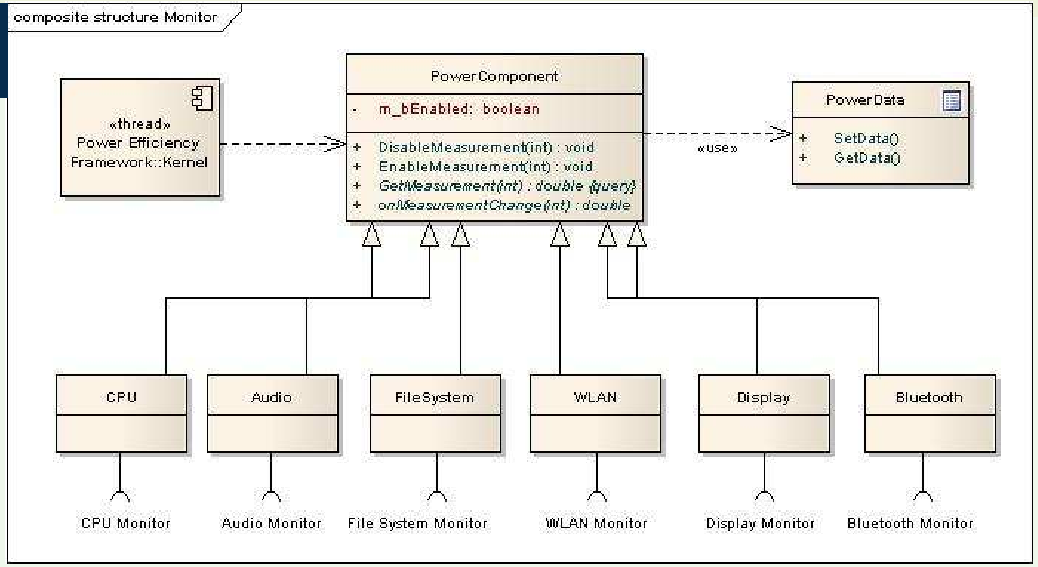
- They determine the system characteristics in terms of power efficiency,
- and a set of relevant system states in isolation.
- Profiling information is stored on the device.

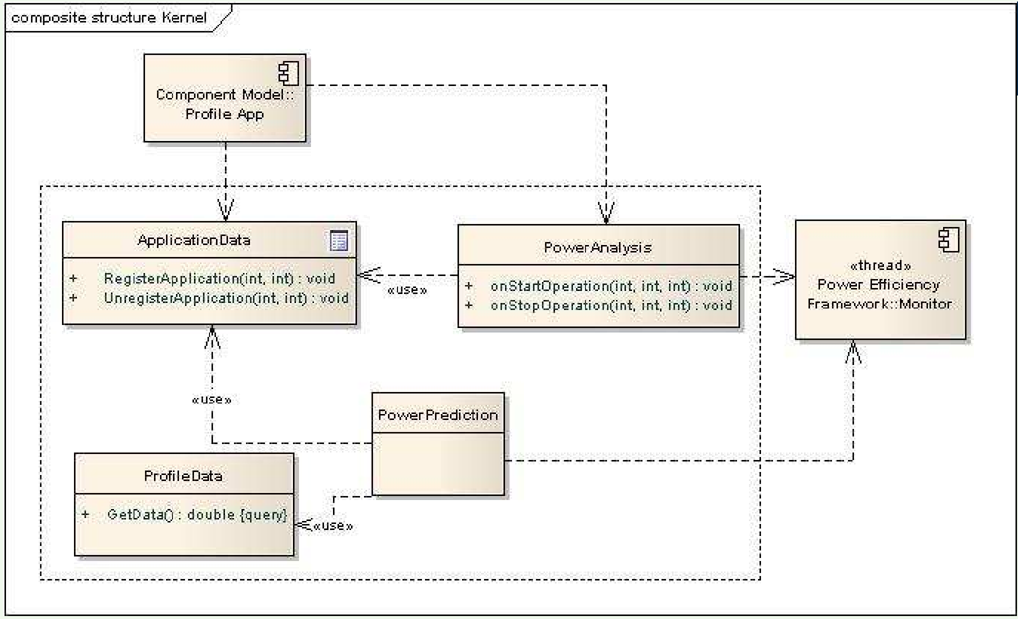


Framework Architecture – Monitor and Kernel

Monitor

- Runs in its own execution thread.
- Collects data from different components.
- Both polling and notification mechanisms.
- Translation between the framework's data representation and the one provided by the platform is realized in the porting layer.





Kernel

- Defines generic interfaces and metrics.
- Data processing and translation to higher level metrics
- Responsible for the metric split among execution threads and applications.
- Prediction unit based on history data.

Framework Architecture – Putting it all together

Basic Model

- A system is composed out of **n** components: $\mathbf{C_i}$.
- Every component $\mathbf{C_i}$ has a set of states and associated power consumption values ($\mathbf{S_{ij}}$, $\mathbf{V_{ij}}$).
- For every component, the states and values are determined by profiling means.
- States are particular to every component

Model Behavior

Application registration mode			Energy consumption distribution
Framework	Individual components	Profile Specified	
Yes	**No**	**No**	**Fair split across applications**
Yes	**Yes**	**No**	**Depending on the time spent executing calls to each component**
Yes	**Yes**	**Yes**	**The profiled state-value pairs are used**

Model Considerations - CPU

- Idle or asynchronous processing time is shared among components.
- Output:
  - Energy consumption per application/component
  - Global (system level) power metrics
  - Higher level metrics per application/components

Conclusions

Constructing power-aware applications is important

Most approaches do not provide an open and generic solution

We proposed basic concepts and architecture for an unified approach for both power consumption measurements and higher level power efficiency metrics

Next steps

** Model completion for all components*

** Integrate the framework in different system architectures*

This work has been carried out in the context of the eMuCo project.

eMuCo (www.emuco.eu)
is a European project supported by
the European Union under
the Seventh Framework Program (FP7)
for research and technological development.