Advanced Research
Intelligent
Embedded
Systems

Adaptation mechanisms in embedded systems

Cognitive Cyber-physical Systems: from theory to practice

Politecnico di Milano, DEIB, Italy

The need of adaptation in Cyber-physical Systems systems

- Many **cyber-physical systems require intelligent mechanisms** to deal with those situations where either evolution or time variance requests a reaction to grant a performance level
- Adaptation is the **basic form** of intelligence to be considered every time the **cyber-physical system has to react quickly**
  - minimize both time-to-reaction and energy consumption
  - adaptive mechanisms are the result of sophisticated techniques

Adaptation: form of intelligence associated with the execution of automatic cognitive processes

Adaptation mechanisms in cyber-physical systems

	Automatic Processes	Controlled Processes
Single Unit Level	• reducing the energy consumption of the device • maximizing the efficiency of the energy harvesting process	• the implementation of sophisticated mechanisms for adaptive sensing • reprogram the embedded device (when and how)
Group Level	• keeping the units clocks synchronized	• taking advantage of group information to improve accuracy as in distributed clock synchronization

Adaptation mechanisms in embedded systems

	Automatic Processes	Controlled Processes
Single Unit Level	• reducing the energy consumption of the device • maximizing the efficiency of the energy harvesting process	• the implementation of sophisticated mechanisms for adaptive sensing • reprogram the embedded device (when and how)
Group Level	• keeping the units clocks synchronized	• taking advantage of group information to improve accuracy as in distributed clock synchronization

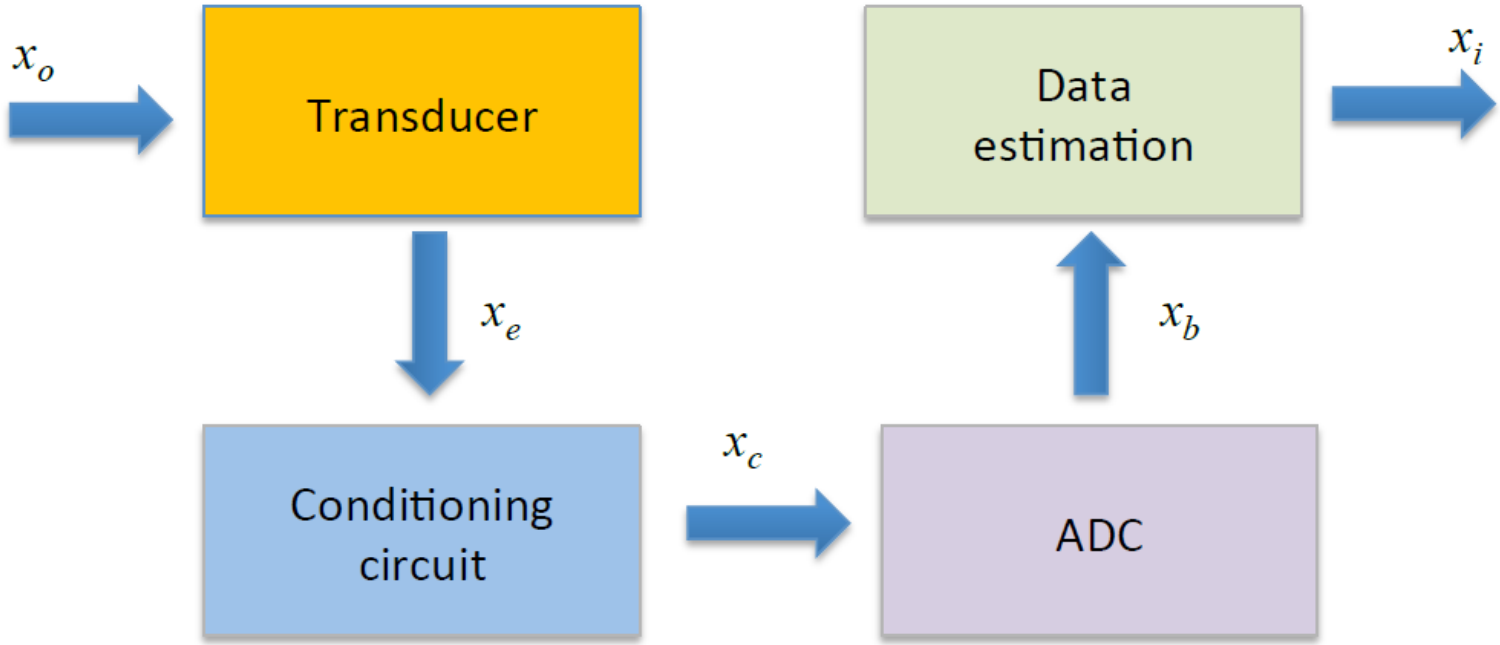
Reducing the energy consumption: energy management

- Even though we are good in energy harvesting, **energy consumption is an issue** and should be minimized
- The **Murphy's law** appears to be always there: once you need energy you cannot harvest it…
- **Some strategies to manage the available energy**
  - Throw away what is not needed (keep it simple)
  - Consider incremental applications
  - Duty cycling: The more you sleep the less energy you consume
  - Forecast and react methods
  - Sample what needed
  - Act directly on the Hw

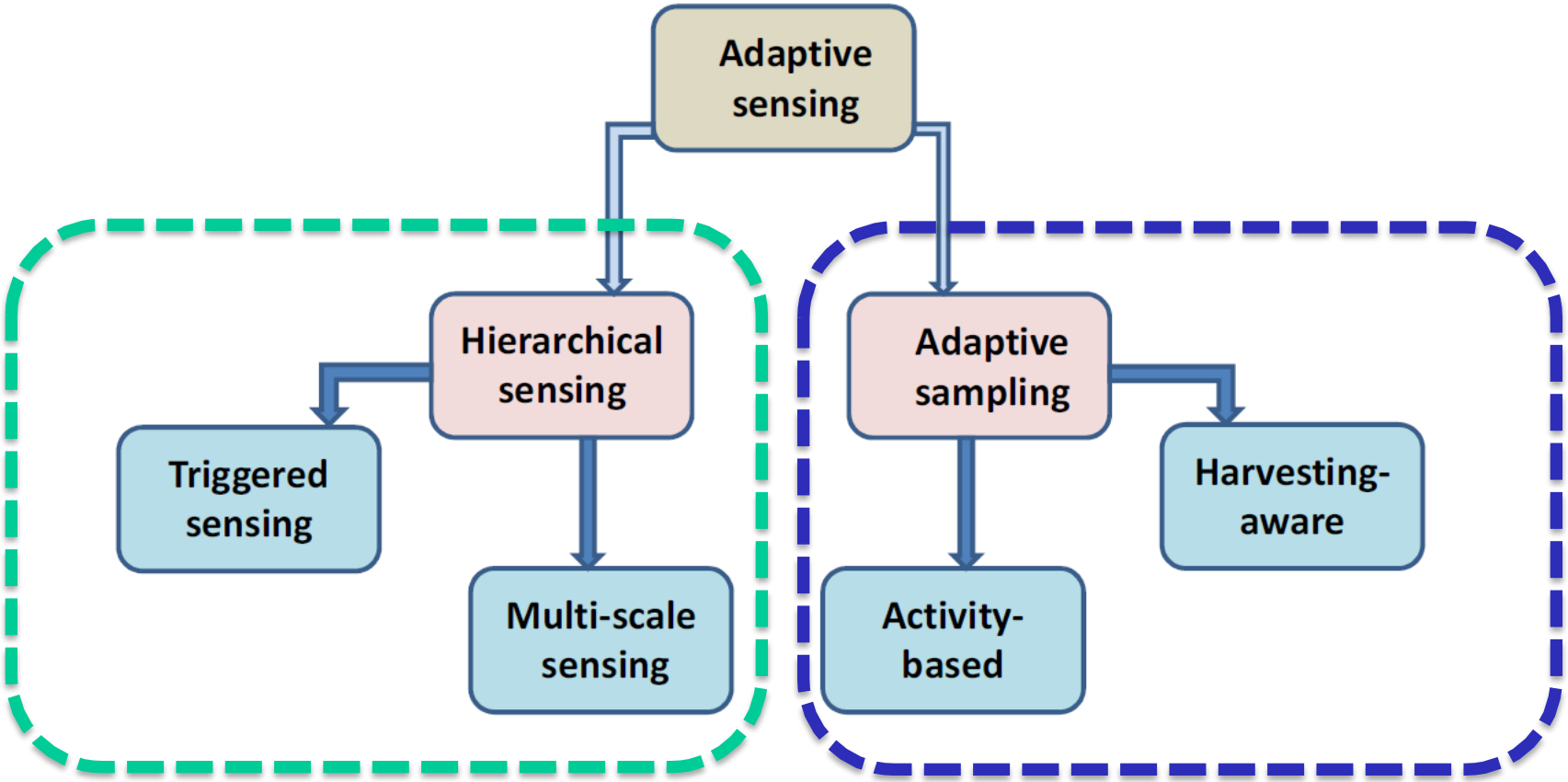
Adaptive Sensing

- **A possible data acquisition procedure:**
  1. The interrupt wakes up the microprocessor, possibly from a deep sleep modality;
  2. The interrupt routine sends the warm up directives to the sensors (if needed) and waits (or the task is inserted in the task wait list) for available data;
  3. If sensors are ready, the sampling procedure is activated, acquiring data and storing them in the memory;
  4. Once the acquisition task is completed the routine terminates.

The measurement chain (sensor level)

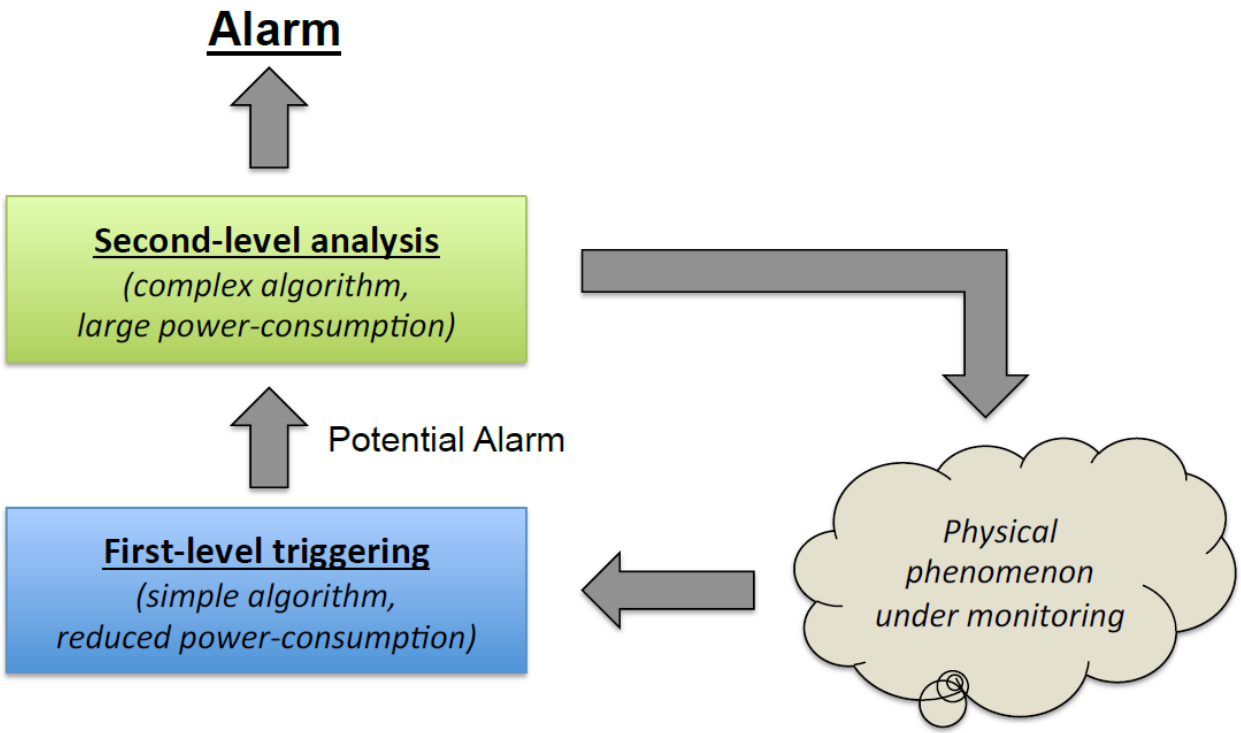


Adaptive Sensing



Hierarchical Sensing

- Data acquired with low resolution/low power consumption sensors are processed by simple algorithms. When a potential alarm is detected, high resolution sensors generally characterized by a higher power consumption are activated.



Hierarchical Sensing: Triggered Sensing

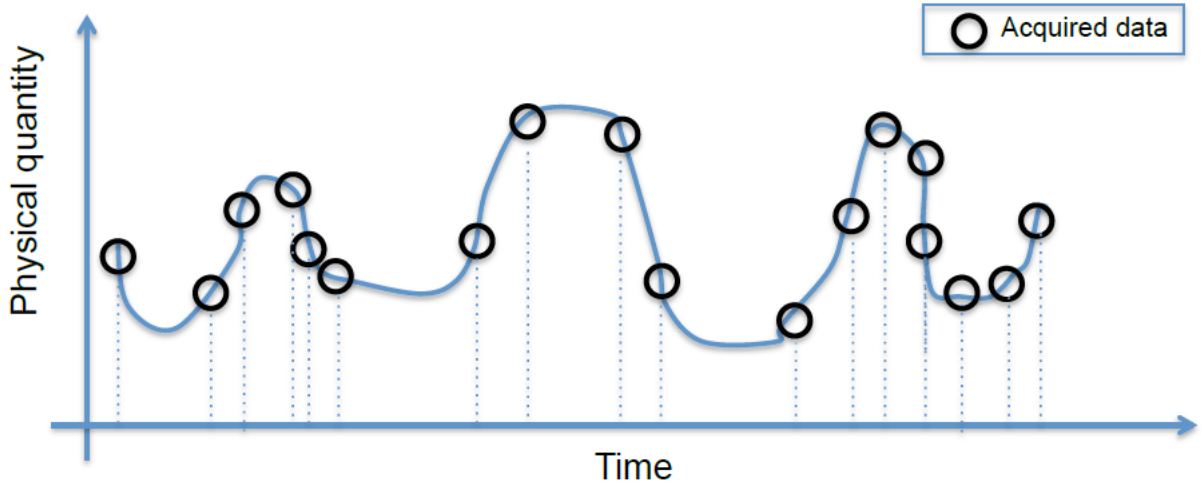
- In triggered sensing **features are used to decide whether to activate an alarm or not**.
- An example is based on **different resolution** (different energy consumption) **case**
- Another example is based on a **CMOS camera, reconfigurable in terms of spatial resolution**:
  - Low resolution scenes of the environment are quickly processed for target detection.
  - If targets are detected, some digital cameras undergo an adaptation phase that, after reconfiguration, provides higher quality images for target detection validation.

Hierarchical Sensing: Multi-scale Sensing

- In multi-scale sensing, we identify **areas** within the monitoring field that **require a more accurate inspection**
- *The method envisages a lower resolution when the receptive field is less relevant, and a higher resolution when high precision acquisitions are requested*.
- **Example**: An emergency management scenario.
  - The field to be monitored is instrumented with static, low resolution temperature sensors.
  - When an event is detected, a mobile sensor unit (mule) is sent to the area to collect additional high precision measurements.

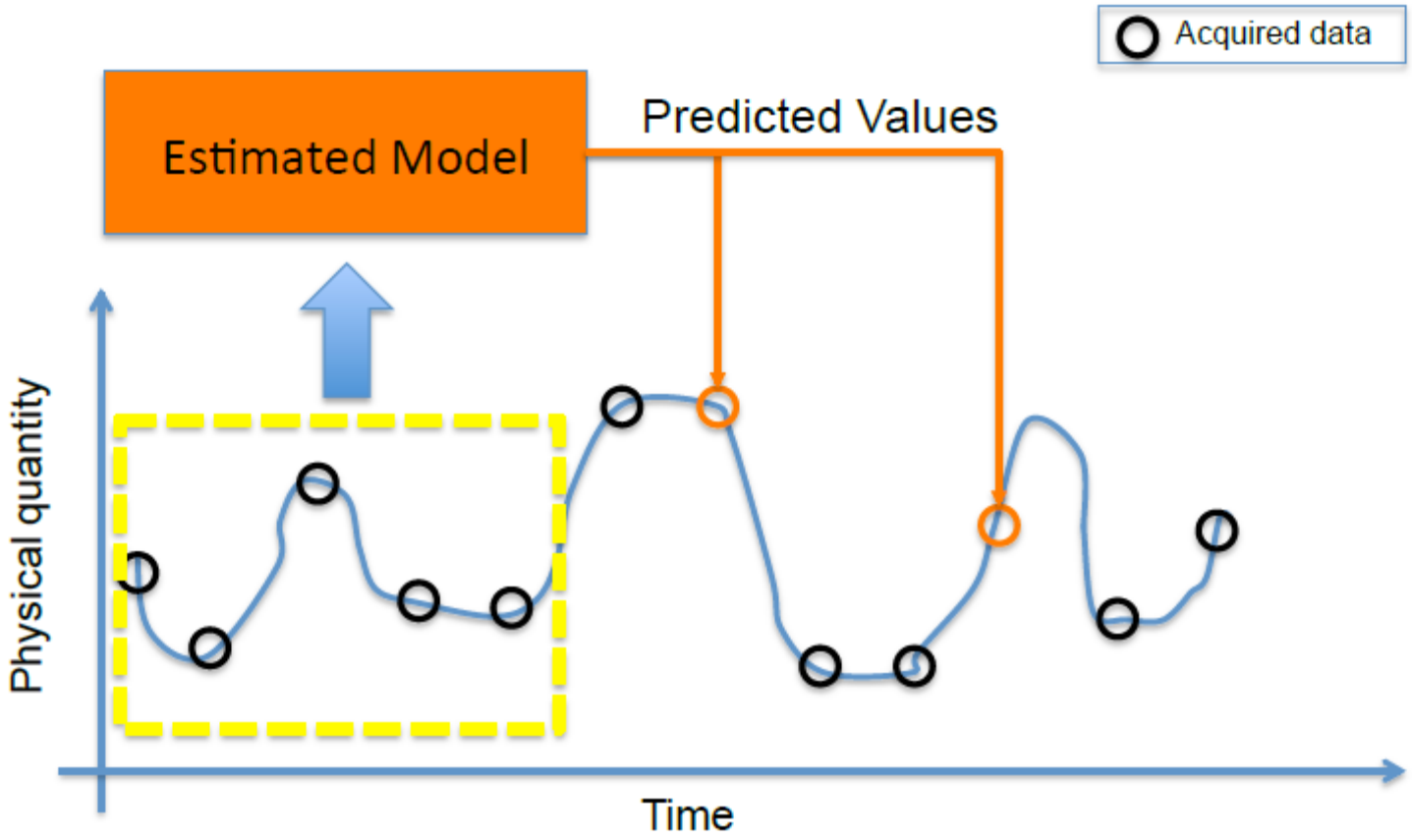
Adaptive Sampling

- **Adaptive sampling methods modify the sampling rate** based on the information content carried by sensed data, the available residual energy present in the batteries and the incoming or estimated harvested power.



Adaptive Sampling: Activity-based sampling

- In model-based sampling temporal and spatial locality redundancy are used to describe incoming data. If a discrepancy exists a new model must be generated



Adaptive Sampling: Harvesting-aware

- In harvesting aware policies, **the sampling frequency is increased or decreased based on information related to the residual energy**.
- **In extreme cases sensors can be switched off**, with strategies depending on
  - The information novelty content expected to be provided by the datastream in the near future (either predicted or estimated on past data)
  - The power consumption of the sensor, also balanced with its accuracy
  - The impact of the sensor information on the application (sensitivity analysis)

Code Level Adaptation

«Reprogram the code to track the change»

Code Adaptation

- **Partitioned the code** in atomic functional segments representing the smallest blocks that can undergo a change and **generate a functional dependency graph**
- **Feasible actions**
  - Substitute the code associated with a node.
  - Substitute a sub-graph (or the entire graph). It is possible that the new sub-graph has a different topology. However, the interface with the complement of the sub-graph is maintained.
- **Activate/remove arcs through parametric reprogrammability**. The execution flow is modified by enabling/disabling some parts of the code (the program code is kept into the memory)

Remote parametric-code reprogrammability

- The code is parameterized. By changing the parameters vector the program is reprogrammed

Algorithm 17: A parametric program. The code is parameterized in the parameter set $\theta = \{\theta_1, \theta_2, \theta_3, \cdots, \theta_n\}$. Depending on the values assigned to the set θ the code modifies its properties.

```
1-  i = 0;
2-  enable-sensor();
3-  while i < θ1 do
4-      data[i] = sample();
5-      if θ2=0 then
            data[i] =lowPass(data[i]);
        end
6-      i = i + 1;
    end
7-  disable-sensor();
8-  if θ3 = 1 then
        dataF =average(data);
    else
        dataF = weighted-average(data, θ4, ··· , θn);
    end
9-  output(dataF);
```

Remote code reprogrammability

- The program or subparts of it are changed at runtime

Algorithm 18: Remote code reprogrammability. The program is the code equivalent to the one in algorithm 17 with the configuration $\theta = \{4, 1, 0, \frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}, 0, 0\}$

```
1-  i = 0;
2-  enable-sensor();
3-  while i < 4 do
4-      data[i] = sample();
5-      i = i + 1;
    end
6-  disable-sensor();
7-  dataF = weighted-average(data, 1/8, 3/8, 3/8, 1/8);
8-  output(dataF);
```