PROGRAMMING BEYOND INDIVIDUAL NODES: STATE-CENTRIC PROGRAMMING

State centric Programming.

- Many sensor network applications, such as target tracking, are not simply generic distributed programs over an ad hoc network of energy-constrained nodes.
- Deeply rooted in these applications is the notion of states of physical phenomena and models of their evolution over space and time.
- Some of these states may be represented on a small number of nodes and evolve over time, while others may be represented over a large and spatially distributed number of nodes, as in tracking a temperature contour.
- A distinctive property of physical states, such as location, shape, and motion of objects, is their continuity in space and time.
- Their sensing and control is typically done through sequential state updates.
- System theories, the basis for most signal and information processing algorithms, provide abstractions for state update, such as:

$\mathbf{X^*_{+1} = /(X,,_{Mjt})}$

$\mathbf{y^* = s(Xjt,\ u_k)}$

- where
  - o **X** is the state of a system,
  - o *u* are the inputs,
  - o **y** are the outputs,
  - o *k* is an integer update index over space and/or time,
  - o *f*is the state update function, and
  - o *g* is the output or observation function.
- This formulation is broad enough to capture a wide variety of algorithms in sensor fusion, signal processing, and control.
- However, in distributed real-time embedded systems such as sensor networks, the formulation is not so clean as represented in those equations.
- The relationships among subsystems can be highly complex and dynamic over space and time.
- The following concerns, not explicitly raised inequations must be properly addressed during the design to ensure the correctness and efficiency of the resulting systems:
  - Where are the state variables stored?
  - Where do the inputs come from?
  - Where do the outputs go?

- Where are the functions *f* and *g* evaluated?
- How long does the acquisition of inputs take?
- Are the inputs in u_k collected synchronously?
- Do the inputs arrive in the correct order through communication?
- What is the time duration between indices *k* and *k* + 1? Is it a constant?

❖ These issues, addressing *where* and *when*, rather than *how*, to perform sensing, computation, and communication, play a central role in the overall system performance.

❖ However, these "nonfunctional" aspects of computation, related to concurrency, responsiveness, networking, and resource management, are not well supported by traditional programming models and languages.

❖ State-centric programming aims at providing design methodologies and frameworks that give meaningful abstractions for these issues, so that system designers can continue to write algorithms like Equations on top of an intuitive understanding of where and when the operations are performed.

❖ This section introduces one such abstraction, namely, collaboration groups.