5.2 SENSOR NETWORK SIMULATORS

- Node-level design methodologies are usually associated with simulators that simulate the behavior of a sensor network on a per-node basis.
- Using simulation, designers can quickly study the performance (in terms of timing, power, bandwidth, and scalability) of potential algorithms without implementing them on actual hardware and dealing with the vagaries of actual physical phenomena.

A node-level simulator typically has the following components:

Sensor node model:

- A node in a simulator acts as a software execution platform, a sensor host, as well as a communication terminal.
- In order for designers to focus on the application-level code, a node model typically provides or simulates a communication protocol stack, sensor behaviours (e.g., sensing noise), and operating system services.
- If the nodes are mobile then the positions and motion properties of the nodes need to be modelled.
- If energy characteristics are part of the design considerations, then the power consumption of the nodes needs to be modelled.

Communication model:

- Depending on the details of modelling, communication may be captured at different layers.
- The most elaborate simulators model the communication media at the physical layer, simulating the RF propagation delay and collision of simultaneous transmissions.
- Alternately, the communication may be simulated at the MAC layer or network layer, using, e.g., stochastic processes to represent low-level behaviours.

Physical environment model:

- A key element of the environment within which a sensor network operates is the physical phenomenon of interest.
- The environment can also be simulated at various levels of detail.
- For example, a moving object in the physical world may be abstracted into a point signal source.
- The motion of the point signal source may be modeled by differential equations or interpolated from a trajectory profile.
- If the sensor network is passive—i.e., it does not impact the behavior of the environment—then the environment can be simulated separately or can even be stored in data files for sensor nodes to read in.
- If, in addition to sensing, the network also performs actions that influence the behavior of the environment, then a more tightly integrated simulation mechanism is required.

Statistics and visualization:

- The simulation results need to be collected for analysis.
- Since the goal of a simulation is typically to derive global properties from the execution of individual nodes, visualizing global behaviors is extremely important.
- An ideal visualization tool should allow users to easily observe on demand the spatial distribution and mobility of the nodes, the connectivity among nodes, link qualities, end-to-end communication routes and delays, phenomena and their spatio-temporal dynamics, sensor readings on each node, sensor node states, and node lifetime parameters (e.g., battery power).
- A sensor network simulator simulates the behavior of a subset of the sensor nodes with respect to time.
- Depending on how the time is advanced in the simulation, there are two types of execution models: cycle-driven (CD) simulation and discrete-event (DE) simulation.

Cycle-driven (CD) simulation and discrete-event (DE) simulation.

- A CD simulation discretizes the continuous notion of real time into (typically regularly spaced) ticks and simulates the system behavior at these ticks.
- At each tick, the physical phenomena are first simulated, and then all nodes are checked to see if they have anything to sense, process, or communicate. Sensing and computation are assumed to be finished before the next tick.
- Sending a packet is also assumed to be completed by then. However, the packet will not be available for the destination node until the next tick.
- This split-phase communication is a key mechanism to reduce cyclic dependencies that may occur in CD simulations.
- That is, there should be no two components such that one of them computes yk = f(xk) and the other computes xk = g(yk) for the same tick index k.
- In fact, one of the most subtle issues in designing a CD simulator is how to detect and deal with cyclic dependencies among nodes or algorithm components.
- Most CD simulators do not allow interdependencies within a single tick.
  Synchronous languages, which are typically used in control system designs rather than sensor network designs, do allow cyclic dependencies.
- They use a fixed-point semantics to define the behavior of a system at each tick.
- Unlike CD simulators, DE simulators assume that the time is continuous and an event may occur at any time.
- An event is a 2-tuple with a value and a time stamp indicating when the event is supposed to be handled. Components in a DE simulation react to input events and produce output events.
- In node-level simulators, a component can be a sensor node and the events can be communication packets; or a component can be a software module within a node and the events can be message passings among these modules.
- Typically, components are causal, in the sense that if an output event is computed from an input event, then the time stamp of the output event should not be earlier than that of the input event.
- Noncausal components require the simulators to be able to roll back in time, and, worse, they may not define a deterministic behavior of a system .
- A DE simulator typically requires a global event queue. All events passing between nodes or modules are put in the event queue and sorted according to their chronological order.
- At each iteration of the simulation, the simulator removes the first event (the one with the earliest time stamp) from the queue and triggers the component that reacts to that event. ϖ In terms of timing behavior, a DE simulator is more accurate than a CD simulator, and, as a consequence, DE simulators run slower.
- The overhead of ordering all events and computation, in addition to the values and time stamps of events, usually dominates the computation time.
- At an early stage of a design when only the asymptotic behaviors rather than timing properties are of concern, CD simulations usually require less complex components and give faster simulations.
- Partly because of the approximate timing behaviors, which make simulation results less comparable from application to application, there is no general CD simulator that fits all sensor network simulation tasks.
- We have come across a number of homegrown simulators written in Matlab, Java, and C++. Many of them are developed for particular applications and exploit application-specific assumptions to gain efficiency.

- DE simulations are sometimes considered as good as actual implementations, because of their continuous notion of time and discrete notion of events.
- There are several open-source or commercial simulators available. One class of these simulators comprises extensions of classical network simulators, such as ns-2, J-Sim (previously known as JavaSim), and GloMoSim/QualNet.
- The focus of these simulators is on network modeling, protocols stacks, and simulation performance.
- Another class of simulators, sometimes called software-in- the-loop simulators, incorporate the actual node software into the simulation.
- For this reason, they are typically attached to particular hardware platforms and are less portable. Examples include TOSSIM for Berkeley motes and Em* (pronounced em star) for Linux-based nodes such as Sensoria WINS NG platforms.