5.3 THE SIMULATOR TOSSIM

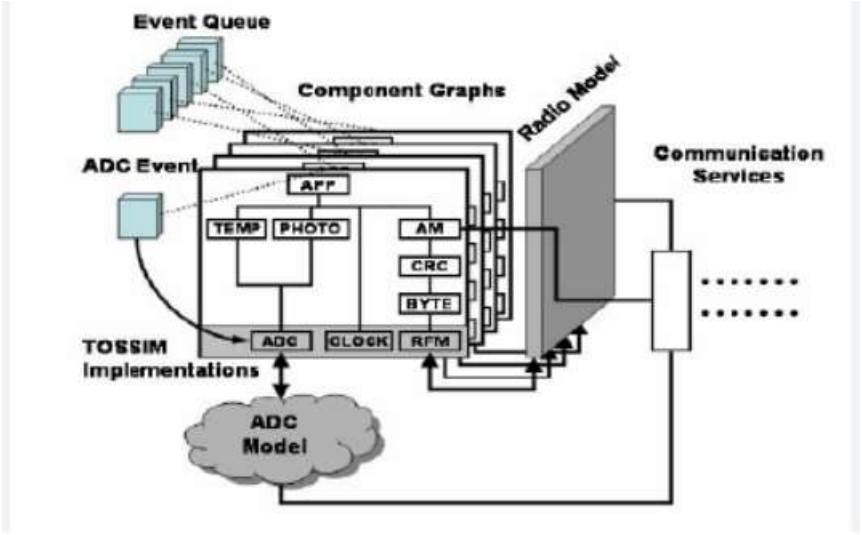

Fig: TOSSIM Architecture

- TOSSIM is a dedicated simulator for TinyOS applications running on one or more Berkeley motes.
- The key design decisions on building TOSSIM were to make it scalable to a network of potentially thousands of nodes, and to be able to use the actual software code in the simulation.
- To achieve these goals, TOSSIM takes a cross-compilation approach that compiles the nesC source code into components in the simulation.
- The event-driven execution model of TinyOS greatly simplifies the design of TOSSIM. By replacing a few low-level components, such as the ADC, the system clock, and the radio front end, TOSSIM translates hardware interrupts into discrete-event simulator events.
- The simulator event queue delivers the interrupts that drive the execution of a node.
- The upperLayer TinyOS code runs unchanged.
- TOSSIM uses a simple but powerful abstraction to model a wireless network.
- A network is a directed graph, where each vertex is a sensor node and each directed edge has a biterror rate.
- Each node has a private piece of state representing what it hears on the radio channel.
- By setting connections among the vertices in the graph and a bit-error rate on each connection, wireless channel characteristics, such as imperfect channels, hidden terminal problems, and asymmetric links, can be easily modeled. Wireless transmissions are simulated at the bit level.
- If a bit error occurs, the simulator flips the bit.
- TOSSIM has a visualization package called TinyViz, which is a Java application that can connect to TOSSIM simulations.
- TinyViz also provides mechanisms to control a running simulation by, e.g., modifying ADC readings, changing channel properties, and injecting packets.
- TinyViz is designed as a communication service that interacts with the TOSSIM event queue.
- The exact visual interface takes the form of plug-ins that can interpret TOSSIM events.
- Beside the default visual interfaces, users can add application-specific ones easily.

COOJA SIMULATOR

- Cooja is an emulator

- According to different sources, an emulator is:
  - ✓ a hardware or software system that enables one computer system (called the host) to behave like another computer system (called the guest): e.g. Cooja enabling your laptop to behave like a Z1 mote.
  - ✓ a system that typically enables the host system to run software or use peripheral devices designed for the guest system: e.g. Cooja enabling your laptop to run the RPL protocol, LIBP and/or other IoT protocols of interest.

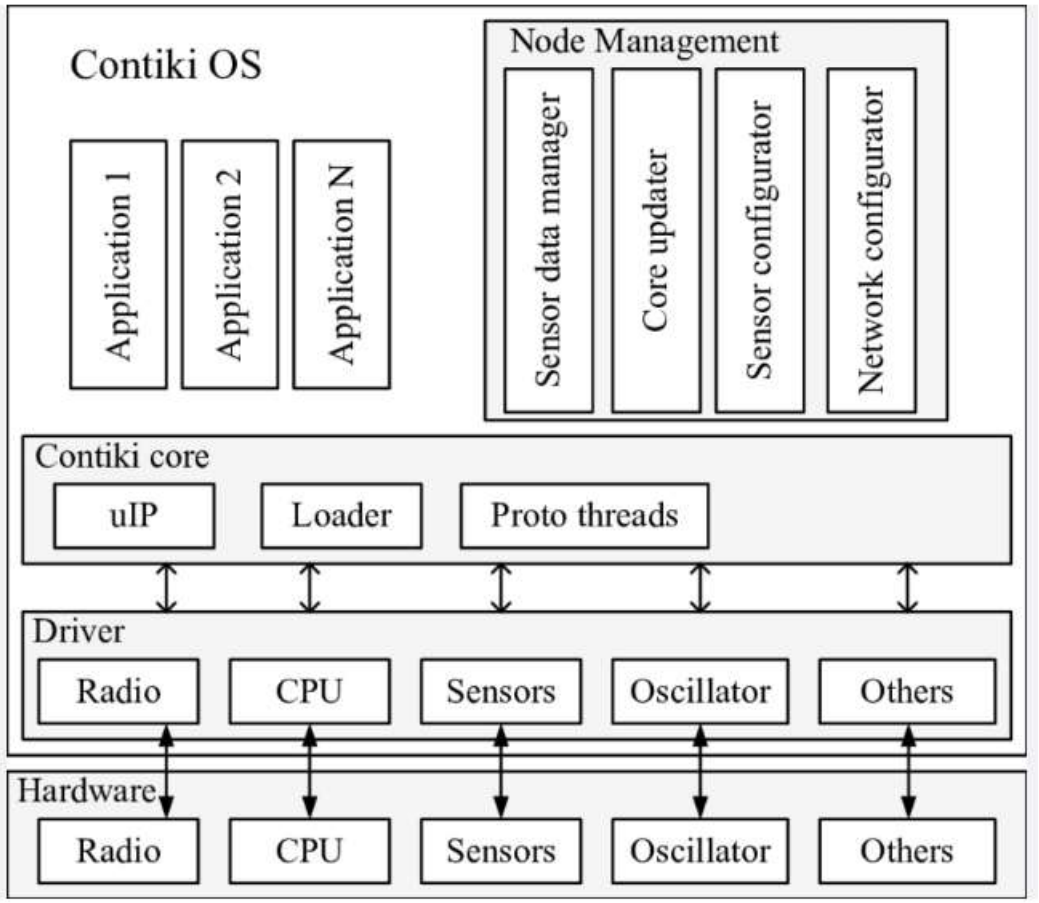

Fig: Architecture of Contiki

- Cooja is not a simulator
- According to different sources, a simulator is:
  - ✓ a hardware or software that that enables one computer system (called the host) to behave like another computer system (called the guest), but is implemented in an entirely different way :
  - ✓ e.g. A flight simulator gives you the feeling of flying an airplane, but you are completely disconnected from the reality of flying the plane, and you can bend or break those rules as you see fit.
  - ✓ e.g. Fly an Airbus A380 upside down between London and Sydney without breaking it.
- Cooja is a Contiki network emulator
  - ✓ An extensible Java-based simulator capable of emulating Tmote Sky (and other) nodes
- The code to be executed by the node is the exact same firmware you may upload to physical nodes
- Allows large and small networks of motes to be simulated ϒ Motes can be emulated at the hardware level
  - ✓ Slower but allows for precise inspection of system behavior ϒ Motes can also be emulated at a less detailed level
  - ✓ Faster and allows simulation of larger networks
- Cooja is a highly useful tool for Contiki development

- It allows developers to test their code and systems long before running it on the target hardware
- Developers regularly set up new simulations, to debug their software , to verify the behavior of their systems