OPERATING SYSTEMS FOR WIRELESS SENSOR NETWORKS

by

Mtech

February 25, 2016

OPERATING SYSTEM DESIGN ISSUES

1. Traditional operating systems are system software, including programs that manage computing resources, control peripheral devices, and provide software abstraction to the application software.
2. Traditional OS functions are therefore to manage processes, memory, CPU time, file system, and devices.
3. This is often implemented in a modular and layered fashion, including a lower layer of kernels and a higher layer of system libraries.
4. Traditional OSs are not suitable for wireless sensor networks because WSNs have constrained resources and diverse data-centric applications, in addition to a variable topology

contd..

1. WSNs need a new type 274 OPERATING SYSTEMS FOR WIRELESS SENSOR NETWORKS of operating system, considering their special characteristics.
2. There are several issues to consider when designing operating systems for wireless sensor networks.
    - Process management and scheduling
    - Memory management
    - Kernel model
    - Application program interface (API)
    - Code upgrade and reprogramming

1. Process management and scheduling
    - The traditional OS provides process protection by allocating a separate memory space (stack) for each process. But this approach usually causes multiple data copying and context switching between processes
2. Memory management
    - Memory is often allocated exclusively for each process/task in traditional operating systems, which is helpful for protection and security of the tasks
    - Since sensor nodes have small memory, another approach, sharing, can reduce memory requirements
3. Kernel model
    - The event-driven and finite state machine (FSM) models have been used to design microkernels for WSNs
    - The event-driven model may serve WSNs well because they look like event-driven systems

contd..

1. Application program interface (API)
    - Sensor nodes need to provide modular and general APIs for their applications. The APIs should enable applications access the underlying hardware
2. Code upgrade and reprogramming
    - Since the behavior of sensor nodes and their algorithms may need to be adjusted either for their functionality or for energy conservation, the operating system should be able to reprogram and upgrade

Sensor operating systems (SOS) should embody the following functions, bearing in mind the limited resource of sensor nodes

1. Should be compact and small in size since the sensor nodes have very small memory.The sensor nodes often have memories of only tens or hundreds of kilobytes.
2. Should provide real-time support, since there are real-time applications, especially when actuators are involved.
3. Should provide efficient resource management mechanisms in order to allocate microprocessor time and limited memory.
4. Should support reliable and efficient code distribution since the functionality performed by the sensor nodes may need to be changed after deployment.
5. Should support power management, which helps to extend the system lifetime and improve its performance.