UNIT-IV

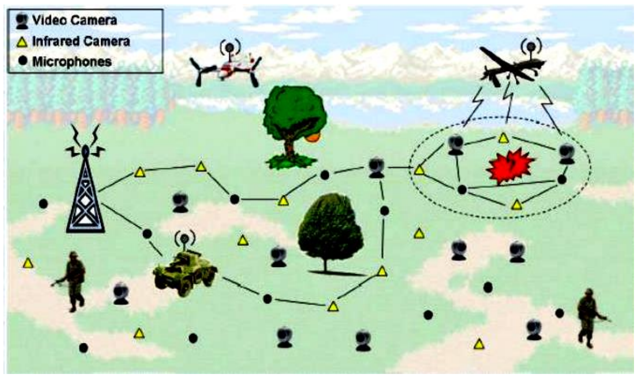
1. Infrastructure Establishment for WSN

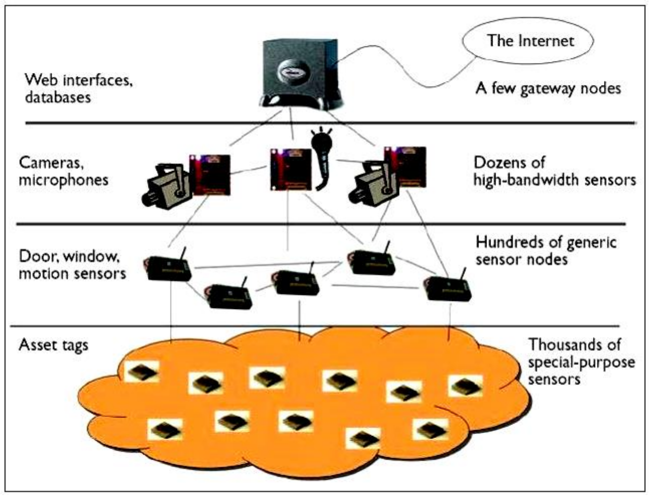
➢ **Localization and Positioning, tracking:**

Properties of positioning, Possible approaches, Task driven Sensing, Rolls of Sensor nodes and utilities, Information based sensor tracking, joint routing and information aggregation, Sensor Network Databases-BIGDATA, Sensor network platforms and tools, Single-hop localization, Positioning in multi-hop environments, Impact of anchor placement,

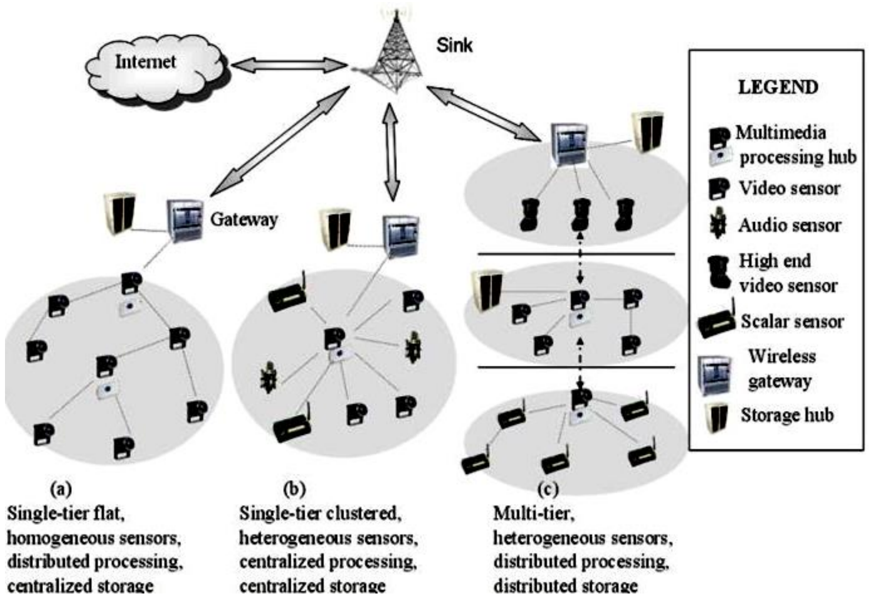
➢ **Operating Systems for WSN:**

OS Design Issues, Examples of OS(Architecture, Design Issues, Functions): Tiny OS, Mate, Magnet OS, MANTIS, Nano-RK OS Architecture Block Diagram, LiteOS Architectural Block Diagram, LiteFS Architectural Block Diagram, Content delivery networks. Introduction to Internet of Things(IoT).




The Internet
Web interfaces, databases
A few gateway nodes
Cameras, microphones
Dozens of high-bandwidth sensors
Door, window, motion sensors
Hundreds of generic sensor nodes
Asset tags
Thousands of special-purpose sensors

Architecture:


Internet
Sink
Gateway
LEGEND
Multimedia processing hub
Video sensor
Audio sensor
High end video sensor
Scalar sensor
Wireless gateway
Storage hub
(a)
Single-tier flat, homogeneous sensors, distributed processing, centralized storage
(b)
Single-tier clustered, heterogeneous sensors, centralized processing, centralized storage
(c)
Multi-tier, heterogeneous sensors, distributed processing, distributed storage

Definition:

The task of initiating collaborative environment for sensor network when that network is activated is called infrastructure establishment.

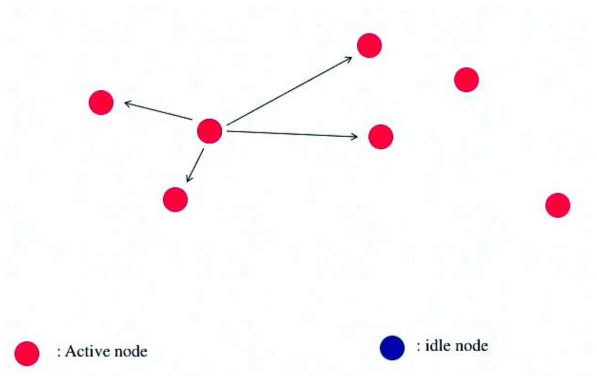
When sensor network is activated various task must be performed to establish necessary infrastructure that will allow useful collaborative work to be performed:

1) Discovering other nodes
2) Radio power adjustment to ensure adequate connectivity.
3) Cluster formation.
4) Node placement in a common temporal and special framework.

Some common techniques used to establish the network are:

1) Topology control
2) Clustering
3) Time synchronization
4) Localization

2. Topology Control



- ☞ A sensor node that wakes up execute a protocol to discover which other nodes it can communicate with. (bidirectional).
- ☞ At initial state each node try to connect with neighbors according to the radio link capacity of its own.
- ☞ The neighbor is determined by the radio power of the node as well as local topology and other conditions that may degrade performance of the radio link.
- ☞ Sensor node are capable of broadcasting less that their maximum possible radio power. (for energy saving and network lifetime)
- ☞ Example : Homogeneous topology : all nodes with same transmission range.

<u>Critical Transmission Range Problem</u>

Computing minimum common transmitting range "r" such that the network is connected.

Solution :

1) Depends on physical placement of the node.
2) If node location is known CTR problem has a simple solution.

CTR is defined as longest edge of minimum spanning tree

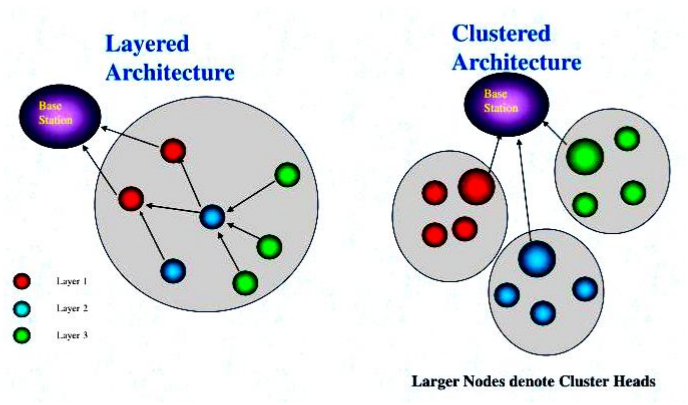
<u>Solution to CTR problem</u>

<u>Example :</u>

GRG (Geometric Random Graph):

N points are distributed into a region according to some distribution and then some aspect of the node placement is investigated with high probability

3. Clustering,



Hierarchical architecture enables more efficient use of sensor resources such as:

- Frequency spectrum
- Bandwidth
- Power

Advantages:

1) Health monitoring of network is easy.
2) Identifying misbehaving node is easy.
3) Some nodes can act as watchdogs for other nodes.
4) Maintenance of network is easy.

Cluster formation:

1) Initially unique ids (UIDs) are assigned to each node
2) Node with higher ID than its uncovered neighbors declares itself as cluster head.
3) Cluster head nominated nodes then communicate with each other.
4) Node that can communicate with two or more cluster heads may become gateway.

Gateway: node that aid in passing traffic from one cluster to other.

Uncovered neighbors: node that have not been already claimed by another cluster head.

4. Time synchronization,

- Every node is operating independently so their clocks may not be synchronized with each other.
- It is important to run network efficiently

- To detect events
- For localization
- Estimating internodes distances.
- To arrange TDMA schedule

- In wired network NTP is used to achieve coordinated universal time (UTC).
- In NTP highly accurate clock is mounted on one of the machine of the network. This is not applicable for WSN:
    - No master clocks are available.
    - Inconsistent common delay.
    - Connections are variable/dynamic and unpredictable.
- Time difference caused by the lack of common time origin is called as clock phase difference or clock bias.
- Methods for clock synchronization in WSN :
    1) Clock phase diff estimation using three message exchanges.
    2) Interval method.
    3) Reference broadcast.

5. Localization and Positioning,

What?

— To determine the physical coordinates of a group of sensor nodes in a wireless sensor network (WSN)

— Due to application context, use of GPS is unrealistic, therefore, sensors need to self-organize a coordinate system

Why?

— To report data that is geographically meaningful

— Services such as routing rely on location information; geographic routing protocols; context-based routing protocols, location-aware services

Localization in Wireless Sensor Networks

In general, almost all the sensor network localization algorithms share three main phases

- DISTANCE ESTIMATION
- POSITION COMPUTATION
- LOCALIZATION ALGHORITHM

- ❖ The distance estimation phase involves measurement techniques to estimate the relative distance between the nodes.
- ❖ The Position computation consists of algorithms to calculate the coordinates of the unknown node with respect to the known anchor nodes or other neighboring nodes.
- ❖ The localization algorithm, in general, determines how the information concerning distances and positions, is manipulated in order to allow most or all of the nodes of a WSN to estimate their position. Optimally the localization algorithm may involve algorithms to reduce the errors and refine the node positions.

Distance Estimation

There are four common methods for measuring in distance estimation technique:

- ANGLE OF ARRIVAL (AOA)
- TIME OF ARRIVAL (TOA)
- TIME DIFFERENT OF ARRIVAL (TDOA)
- THE RECEIVED SIGNAL STRENGH INDICATOR (RSSI)

- ☞ ANGLE OF ARRIVAL method allows each sensor to evaluate the relative angles between received radio signals
- ☞ TIME OF ARRIVAL method tries to estimate distances between two nodes using time based measures
- ☞ TIME DIFFERENT OF ARRIVAL is a method for determining the distance between a mobile station and nearby synchronized base station
- ☞ THE RECEIVED SIGNAL STRENGTH INDICATOR techniques are used to translate signal strength into distance

Position Computation

The common methods for position computation techniques are:

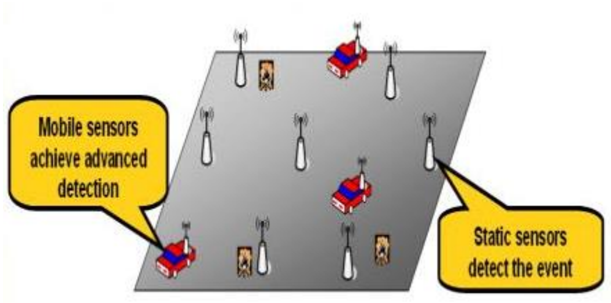
- LATERATION
- ANGULATION

☞ LATERATION techniques based on the precise measurements to three non collinear anchors. Lateration with more than three anchors called multilateration.

☞ ANGULATION or triangulation is based on information about angles instead of distance

Classifications of Localization Methods

According to the ways of Sensors implementation, we classify the current wireless sensor network localization algorithms into several categories such as:


Mobile sensors achieve advanced detection
Static sensors detect the event

- Centralized vs Distributed
- Anchor-free vs Anchor-based
- Range-free vs Range-based
- Mobile vs Stationary

6. Sensor Tasking and Control.

➢ Because of Limited battery power and Limited bandwidth careful tasking and the control id needed.

➢ Information collected from the sensors.

— All information aggregation is needed.

— Selective information aggregation is needed.

➢ Which sensor nodes to activate and what information to transmit is a critical issue.

➢ Classical algorithms are not suitable for WSN :

— Sense values are not known.

— Cost of sensing may vary with the data.

Designing strategy for Sensor Tasking and Ctrl:

1) What are the important object in the environment to be sensed?
2) What parameters of these object are relevant?
3) What relations among these objects are critical to whatever high level information we need to know?
4) Which is the best sensor to acquire a particular parameter?
5) How many sensing and the communication operations will be needed to accomplish the task?
6) How coordinated do the world models of the different sensor need to be?
7) At what level do we communicate information in a spectrum from a signal to symbol?

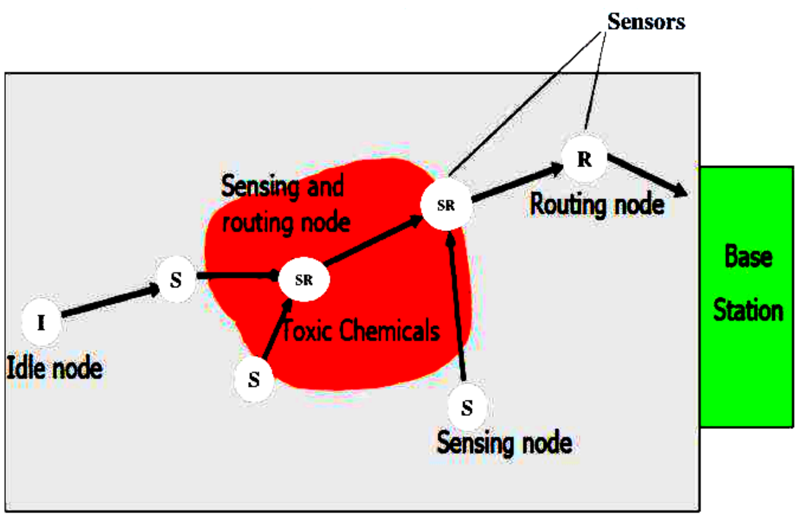
Roles of Sensor nodes and utilities

- A sensor may take on a particular role depending on the application task requirement and resource availability such as node power levels.

Example:

- Nodes, denoted by SR, may participate in both sensing and routing.
- Nodes, denoted by S, may perform sensing only and transmit their data to other nodes.
- Nodes, denoted by R, may decide to act only as routing nodes, especially if their energy reserved is limited. Nodes, denoted by I, may be in idle or sleep mode, to preserve energy.

Roles of Sensor nodes



UNIT-V

1. Sensor Network Platforms and Tools

Commercially available sensor nodes:

1. Specialized sensing platform such as Spec node designed at University of California-Berkeley.
2. Generic Sensor platform – Berkeley Mote.
3. High bandwidth sensing platform such as iMote.
4. Gateway platform such as Stargate (Sink node).

- A real-world sensor network application most likely has to incorporate all these elements, subject to energy, bandwidth, computation, storage, and real-time constraints
- There are two types of programming for sensor networks, those carried out by end users and those performed by application developers.

End users

- An end user may view a sensor network as a pool of data and *interact with the network via queries.*
- Just as with query languages for database systems like SQL
- good sensor network programming language should be expressive enough to encode application logic at a high level of abstraction
- At the same time be structured enough to allow efficient execution on the distributed platform.

Application developer

- An application developer must provide end users of a sensor network with the capabilities of data acquisition, processing, and storage.
- Unlike general distributed or database
- Systems, collaborative signal and information processing (CSIP) software comprises reactive, concurrent, distributed programs running on

ad hoc, resource- constrained, unreliable computation and communication platforms.

2. Sensor Node Hardware – Berkeley Motes

- Sensor node hardware can be grouped into three categories.
  - Augmented general-purpose computers
  - Dedicated embedded sensor nodes
  - System-on-chip

- Berkeley motes due to their small form factor, open source software development, and commercial availability, have gained wide popularity in the sensor network research community.

❖ __Augmented general-purpose computers__

+ Off-the-shelf operating systems such as WinCE, Linux and

- Off-the-shelf operating systems such as WinCE, Linux and
- With standard wireless communication protocols such as 802..11 or Bluetooth. Relatively higher processing capability
- More power hungry
- Fully supported
- Popular programming languages
- Ex: PDAs

❖ __Dedicated embedded sensor nodes__

- In order to keep the program footprint small to accommodate their small memory size, programmers of these platforms are given full access to hardware but barely any operating system support.
- Typically support at least one programming language, such as C.
- Ex: mica, liny0S, nesC

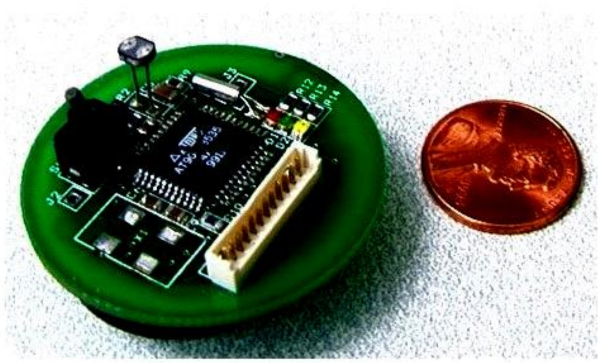
❖ __System-on-chip (SOC)__

- Build extremely low power and small footprint sensor nodes that still provide certain sensing, computation, and communication capabilities.

- Currently in the research pipeline with no predefined instruction set, there is no software platform support available.

<u>Berkeley Motes</u>

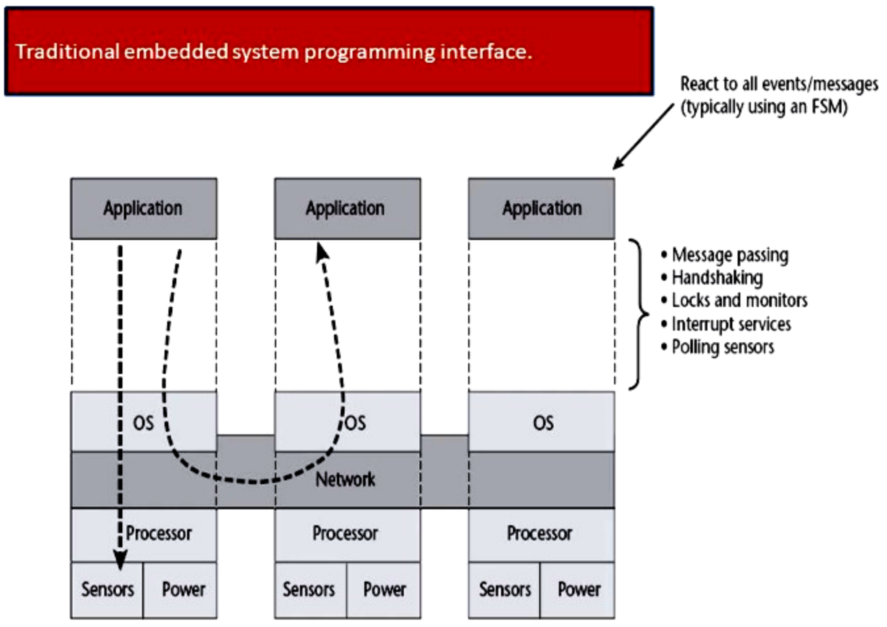
Berkeley Motes



- Berkeley Mote with processing board, sensing board and AA battery pack.
- The mote was essentially a small form factor computer with self-contained processing, sensing and power resources.
- TinyOS provides a set of software components that allows applications to interact with the processor, network transceiver and the sensors.

3. Programming Challenges

- Event-driven execution allows the system to fall into low-power sleep mode when no interesting events need to be processed.
- At the extreme, embedded operating systems tend to expose more hardware controls to the programmers, who now have to directly face device drivers and scheduling algorithms, and optimize code at the assembly level.



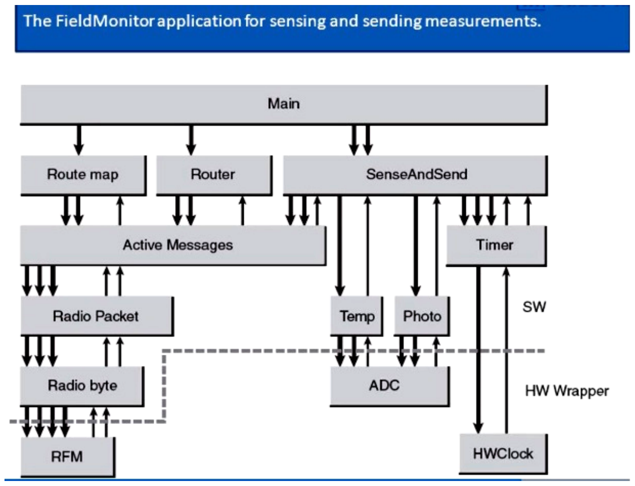
- ➢ Traditional programming technologies rely on operating systems to provide abstraction for processing, I/O, networking, and user interaction hardware.
- ➢ When applying such a model to programming networked embedded systems, such as sensor networks, the application programmers need to explicitly deal with message passing, event synchronization, interrupt handing, and sensor reading.

4. Node level software platforms

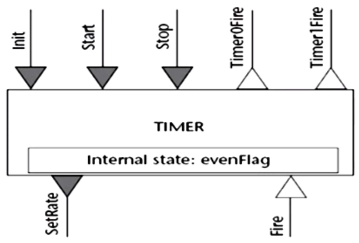
- ➢ Node-centric design methodologies: Programmers think in terms of how a node should behave in the environment.
- ➢ A node-level platform can be a node centric operating system, which provides hardware and networking abstractions of a sensor node to programmers.
- ➢ A typical operating system abstracts the hardware platform by providing a set of services for applications, including file management.
- ➢ Memory allocation, task scheduling, peripheral device drivers, and networking.

Operating System: TinyOS

- ➢ Let us consider a TinyOS application example FieldMonitor
- ➢ Where all nodes in a sensor field periodically send their temperature and photo sensor readings to a base station via an ad hoc routing Mechanism

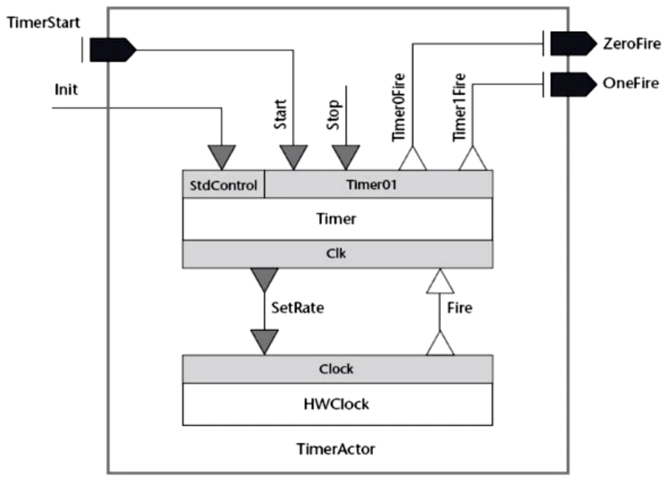


The Timer component and its interfaces



- In nesC, code can be classified into two types:
- *Asynchronous code (AC): Code that is reachable from at least one* interrupt handler.
- *Synchronous code (SC): Code that is only reachable from tasks.*

TinyGALS



Node-level 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.

5. State-centric programming.

- Applications more than simple distributed programs
  - Applications depend on state of physical environment
- Collaboration Groups
  - Set of entities that contribute to state updates
  - Abstracts network topology and communication protocols
- Multi-target tracking problem
  - Global state decoupled into separate pieces
    - Each piece managed by different principle
    - State updated by looking at inputs from other principles
    - Collaboration groups define communication and roles of each principle

Definition:

X: State of a system

U: Inputs

Y: Outputs

K: Update index

F: State update function

G: Output observation function