TinyOs

NAME : Omran Salem Buozarebah

ID : 163110466

OUTLINE

- WHAT IS A TINYOS?
- WIRELESS SENSOR NETWORKS
- TINYOS GOALS
- TINYOS COMPONENTS
- TASK
- COMMAND
- EVENTS
- ABSTRACTIONS FOR RESOURCES
- SHARED RESOURCE CONFIGURATION

WHAT IS A TINYOS?

TinyOS is a *free* and *open source component-based operating system* and *platform targeting wireless sensor networks.*

TinyOS is an embedded operating system written in the *nesC programming language* as a set of cooperating tasks and processes.

WIRELESS SENSOR NETWORKS

TINYOS was developed primarily for use with networks of small wireless sensor.

A number of trends have enabled the development of extremely compact, low-power sensors.

Smaller size in turn reduces power consumption. Low power and small size treads are also clear in wireless communication hardware, micro-electrical sensors (MEMS) and transducers.

NETWORK TOPOLOGY

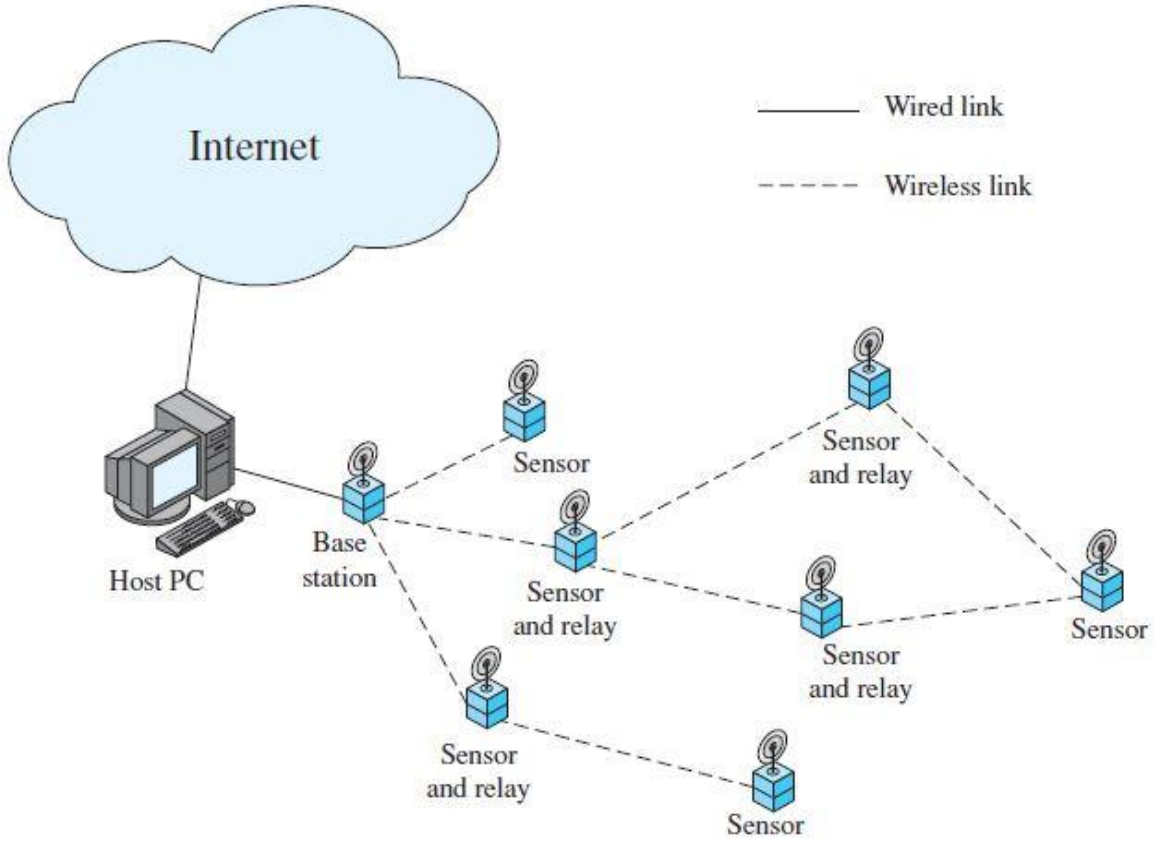

Figure 13.10 Typical Wireless Sensor Network Topology

TINYOS GOALS CON…

- **Allow high Synchronization**: Several different flows of data must be kept moving Synchronization, In addition external controls from remote sensors or base stations must be managed.

- **Operate with limited resources**: A single platform may offer only kilobytes of program memory and hundreds of bytes of RAM.

- **Be robust**: Once deployed, a sensor network must run unattended for months or years Ideally, there should be redundancy both within a single system and across the network of sensors.

TINYOS GOALS

- **Adapt to hardware evolution**: it should be possible to upgrade the hardware with little or no software change, if the functionality is the same.

- **Support a wide range of applications**: Applications show a wide range of requirements in terms of lifetime, communication, sensing, and so on.

- **Support a diverse set of platforms**: As with the past point, a general-purpose embedded OS is desirable.

TINYOS COMPONENTS

An embedded software system built using TinyOS consists of a set of small modules called components, each of which performs a simple task or set of tasks and which interface with each other and with hardware in limited and well-defined ways. The only other software module is the scheduler.

To meet the demanding software requirements of this application, a rigid, simplified software architecture is dictated, consisting of components. The TinyOS development community has implemented a number of open-source components that provide the basic functions needed for the WSN application.

Within a component, tasks are atomic:

1- Once a task has started it runs to completion, It cannot be preempted.

2- another task in the same component, and there is no time slicing.

However, a task can be preempted by an event. A task cannot block or spin wait.

COMMAND

- It is a non-blocking requests.
- A command is typically a request for the lower-level component to perform some service, such as initiating a sensor reading.

EVENTS

In TinyOS may be tied either directly or indirectly to hardware events, The lowest level software components interface directly to hardware interrupts, which may be external interrupts, timer events, or counter events.

An event handler in a lowest level component may handle the interrupt itself or may propagate event messages up through the component hierarchy.

ABSTRACTIONS FOR RESOURCES

- **Dedicated**

  A resource that a subsystem needs exclusive access at all times.

- **Virtualized**

  The virtualized abstraction may be used when the underlying resource need not be protected by mutual exclusion.

- **Shared**

  The shared resource abstraction provides access to a dedicated resource through an arbiter component. The arbiter enforces mutual conclusion allowing only the user at a time to have access to a resource and enabling the client to lock the resource.

SHARED RESOURCE CONFIGURATION CON…

- **Resource**: The client issues a request at this interface requesting access to the resource.
- **Resource Requested**: This is similar to the Resource interface.

  In this case, the client is able to hold onto a resource until the client is notified that someone else needs the resource.

SHARED RESOURCE CONFIGURATION

- **Resource Configure**: This interface allows a resource to be automatically configured just before a client is granted access to it.

- **Resource-specific interfaces**: Once a client has access to a resource, it uses resource-specific interfaces to exchange data and control information with the resource.

SHARED RESOURCE CONFIGURATION

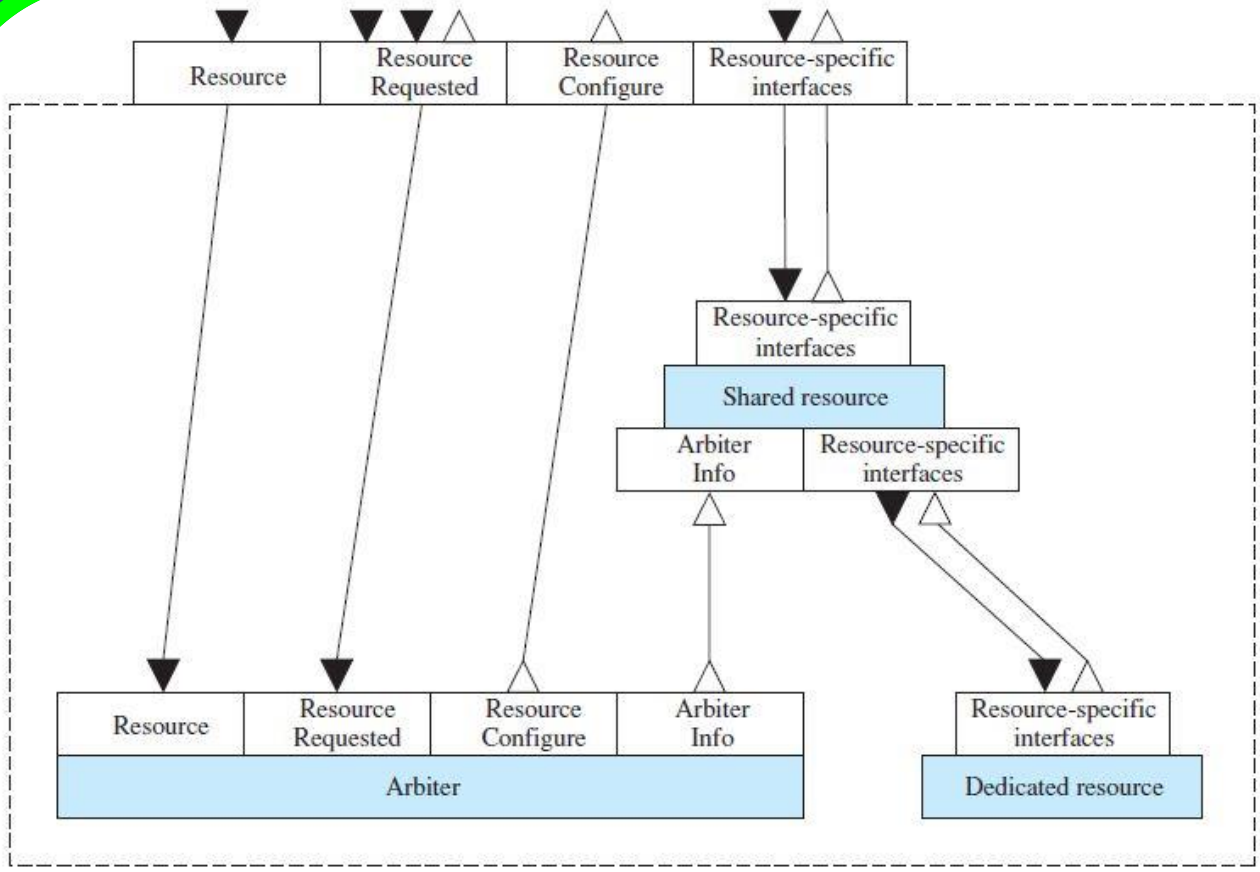

Figure 13.13 **Shared Resource Configuration**

THANK YOU