Ασύρματα Δίκτυα Αισθητήρων

Πέππας Κωνσταντίνος

Ερωτήσεις επανάληψης

4.10 How is concurrency supported in TinyOS?

In TinyOS tasks are executed up to completion, which means only one task accesses a resource at a time.

Ερωτήσεις επανάληψης

- **4.11 What is a split phase programming and how is it useful in wireless sensor networks?**
- A split phase program divides a function call into a call (which will be immediately
- acknowledged) and return (which will be notified as an asynchronous event when the
- called function completes execution).

Ερωτήσεις επανάληψης

- **4.12 Explain the difference between configuration components and modules in TinyOS**.
- A configuration component describes how different modules are interconnected to build an executable service or application, whereas a module is an implementation of an interface.

Ερωτήσεις επανάληψης

- **4.13 Why do threads require their own separate stacks and what is the problem with this approach in wireless sensor networks?**
- Threads require separate stacks in order to store their own context.
- This requires a memory space larger than a single stack based execution.
- Since memory is a scarce resource in wireless sensor networks, multithreading is expensive.

Ερωτήσεις επανάληψης

- **4.14 Give three reasons for supporting dynamic reprogramming in wireless sensor networks.**
- (a) The sensing task may change overtime;
- (b) The application code may need debugging and correction; and
- (c) Policies related to the environment in which the wireless sensor network operates may change and therefore, the network may need to adapt to this change.

Ερωτήσεις επανάληψης

- **4.15 Explain the difference between event based and thread based operating systems. Discuss some of the advantages and disadvantages of the two approaches in the context of wireless sensor networks.**
- In event based programming, interaction between processes is based on events and event handlers.
- Tasks are executed to completion, unless they are interrupted by events.
- This way concurrency is supported and execution is efficient.
- Since only one task is executed at a time, long duration tasks may block short duration tasks, but this problem can be overcome by using a sorted queue scheduling.
- In multithreaded programming, multiple threads run concurrently.
- Threads can be suspended ensuring non blocking operation.
- However, thread management introduces resource overhead on the operating system.

Ερωτήσεις επανάληψης

- **4.16 Explain the difference between static and dynamic memory allocation.**
- In a static memory allocation, memory is allocated to a piece of program at compilation time.
- If the memory requirement of the program is known at compilation time and this requirement remains unchanged, static memory allocation is efficient.
- But sometimes it is difficult to foresee the memory requirements of a piece of program, in which case static memory allocation is inflexible.
- In dynamic memory allocation, the memory requirement of a piece of program is decided at runtime, and memory is allocated accordingly.
- While it is flexible, programs are usually allocated memory a little more than they require, which can be inefficient in resource constrained devices.

Ερωτήσεις επανάληψης

- **4.17 How is separation of concern supported in the following operating systems:**
- **(a) Contiki**
- Contiki distinguishes between core services (which are essential to theOS and remain unchanged once the system is running) and dynamic reloadable services which can be reprogrammed at runtime**.**
- **(b) SOS**
- SOS provides a small monolithic kernel and reloadable (reprogrammable) modules.
- SOS saves the context of a module outside of the module so that context transfer during dynamic reprogramming is possible.
- **(c) LiteOS**
- In LiteOS, applications are not a part of the OS, so each can be independently developed.

Ερωτήσεις επανάληψης

- **4.18 Explain the following concepts in TinyOS:**
- **(a) Commands**
- Commands are nonblocking requests for service.
- **(b) Tasks**
- Tasks are monolithic processes that should be executed to completion
- **(c) Events**
- An event is an occurrence of interest outside of a process and prompts the process to act (or handle the event).

Ερωτήσεις επανάληψης

- **4.19 What is the difference between a TinyOS command and a SOS message?**
- The two are the same in that both are executed asynchronously and both are scheduled before they are processed.
- Whereas tasks are executable processes, messages require message handlers to process

Ερωτήσεις επανάληψης

- **4.20 Why is the state of a module stored in a separate memory space (outside of the module) in SOS?**
- So that dynamic module update or reprogramming is possible.
- If a module's state is stored outside of it, it can easily be replaced or modified.

Ερωτήσεις επανάληψης

- **4.21 Explain how SOS supports dynamic reprogramming.**
- When a new module is available, a code distribution protocol advertises it in the network.
- The local distribution protocol evaluates the advertisement in terms of relevance and resource requirements and if all is fine, proceeds with downloading.
- Once the downloading is successfully completed, module insertion takes place.
- During module insertion, the kernel creates metadata to store the absolute address of the handler, a pointer to the dynamic memory holding the module state and the identity of the module.
- Then the SOS kernel invokes the handler of the module by scheduling an init message for the module

Ερωτήσεις επανάληψης

- **4.22 How is multithreading supported in a Contiki environment?**
- Contiki is by default an event based operating system, but offers multithreading an alternative library service, which can be dynamically loaded and integrated as a part of the application code.

Ερωτήσεις επανάληψης

- **4.23 What is the function of a program loader in Contiki and why is it important?**
- The program loader is responsible for dynamically loading a module.
- If the module is available locally, it loads it from the program memory into the active memory, but if the module is not available locally, then it employs the communication module to fetch the binary image

Ερωτήσεις επανάληψης

- **4.24 How is module replacement supported in Contiki?**
- Contiki supports dynamic reprogramming by separating replaceable modules from the core services (which make up the program loader, the communication service and the kernel).
- The former can be replaced by employing the core services

Ερωτήσεις επανάληψης

- **4.25 What is the advantage of considering a wireless sensor network as distributed file system in LiteOS?**
- Users can easily navigate through and programthe sensor nodes.
- For both aspects LiteOS provides intuitive interfaces, particularly, for Linux users.

Ερωτήσεις επανάληψης

- **4.26 What is differential patching in LiteOS?**
- A differential patching estimates the location of a program in the active memory and carry out module update based on this knowledge.

Ερωτήσεις επανάληψης

- **4.27 Explain the functions of the following message handlers in SOS:**
- **(a) *inithandler***
- The *inithandler* is called by the scheduler when first a module is initialized.
- During dynamic module replacement, it is useful to pass over the context of the previous module.
- **(b) *finalhandler***
- The *finalhandler* is called before a module is removed from the active memory so that it can gracefully release all the resources it owns.

Ερωτήσεις επανάληψης

- **4.28 Which type of scheduling strategy do the following operating systems employ:**
- (a) **TinyOS**: FIFO
- (b) **SOS**: FIFO
- (c) **Contiki**: FIFO, Priority scheduling (for poll handlers)
- (d) **LiteOS**: Priority scheduling with an optional round robin

Ερωτήσεις επανάληψης

- **4.29 How does TinyOS deal with dynamic reprogramming?**
- TinyOS requires a separate module (outside of the OS) to support dynamic programming
- **4.30 Why is separation of concern in TinyOS not a priority?**
- Because of code efficiency

Ερωτήσεις επανάληψης

- **Describe the difference between nodecentric and application centric programming.**
- Node centric programming focus on the development of sensor applications and software for each sensor device, whereas application centric programming considers and develops software for the networks as a whole.

Ερωτήσεις επανάληψης

- **Explain the difference between *provides* and *uses* interfaces in nesC.**
- The provides interface is a set of method calls that are exposed to higher layers.
- The uses interface describes the use of some kind of service.

Ερωτήσεις επανάληψης

- **What options does nesC provide developers to prevent race conditions?**
- Race conditions can be prevented by using synchronous code, which is always atomic to other synchronous codes.
- If asynchronous code is used, one option is to convert code that modifies shared state into synchronous code.
- Another option is to use atomic sections, i.e. brief code sequences that nesC will always run atomically.

Ερωτήσεις επανάληψης

- **A common strategy to ensure atomicity is to disable interrupts in an operating system as long as atomic operations are being executed. What is the danger of disabling interrupts?**
- Important events that trigger interrupts, may not be reported (either at all or only after a delay, i.e. after interrupts have been reenabled), which can have severe consequences.

Ερωτήσεις επανάληψης

- **What are the main advantages and disadvantages of thread based programming models?**
- The thread based programming model allows multiple tasks to make progress in their execution without the concern that a task may block other tasks indefinitely.
- However, thread based approaches increase the operating system complexity and may also require more complex synchronization support in the operating system.

Ερωτήσεις επανάληψης

- **We introduced several macro programming models. Contrast how these different models are able to address multiple (or all) sensor nodes simultaneous**
- Abstract regions groups sensors together using certain neighborhood relationships, e.g. “the set of nodes within distance d”.
- Discovery of region members can be achieved using periodic advertisements.
- In EnviroTrack, groups are formed by sensors which detect certain user defined entities in the physical world, with one group formed around each entity.
- Groups are then identified by context labels, which are logical addresses that follow the external tracked entity around in the physical environment
- Database approaches treat a wireless sensor network like a distributed database that can be queried to obtain sensor data.
- That is, the network can be represented logically as a database table that has (as an example) one row per node per instant in time and each column corresponds to a type of sensor reading.

Ερωτήσεις επανάληψης

- **Why is it necessary to provide the opportunity to dynamically reprogram a sensor network? What is challenging in distributing a new program to all sensor nodes in the network?**
- Reasons for reprogramming a sensor network are that details of certain applications and application characteristics may not be known until after deployment, sensor applications may need upgrades or bug fixes, and usage scenarios of sensor networks may change during their lifetimes.
- Challenges in reprogramming include the need for reliable code distributions (all nodes must receive all pieces of the code), quick code dissemination (to limit downtimes), and energy efficient dissemination.
- Reprogramming should interfere with the goals of the sensor network as little as possible.
- Another challenge is that during reprogramming, different sensors may run different applications or versions of applications. In such scenarios, it is important to prevent failures and miscommunications due to version mismatches.