SAPIENZA
UNIVERSITÀ DI ROMA

Laboratorio di Sistemi Wireless

14 Maggio 2012

A. Cammarano, A.Capossele, D. Spenza

SAPIENZA
UNIVERSITÀ DI ROMA

Contacts

Cammarano: cammarano@di.uniroma1.it
Capossele: capossele@di.uniroma1.it
Spenza: spenza@di.uniroma1.it

Tel: 06-49918430
Room: 333

Slides: *www.dsi.uniroma1.it/~spenza/*

Outline

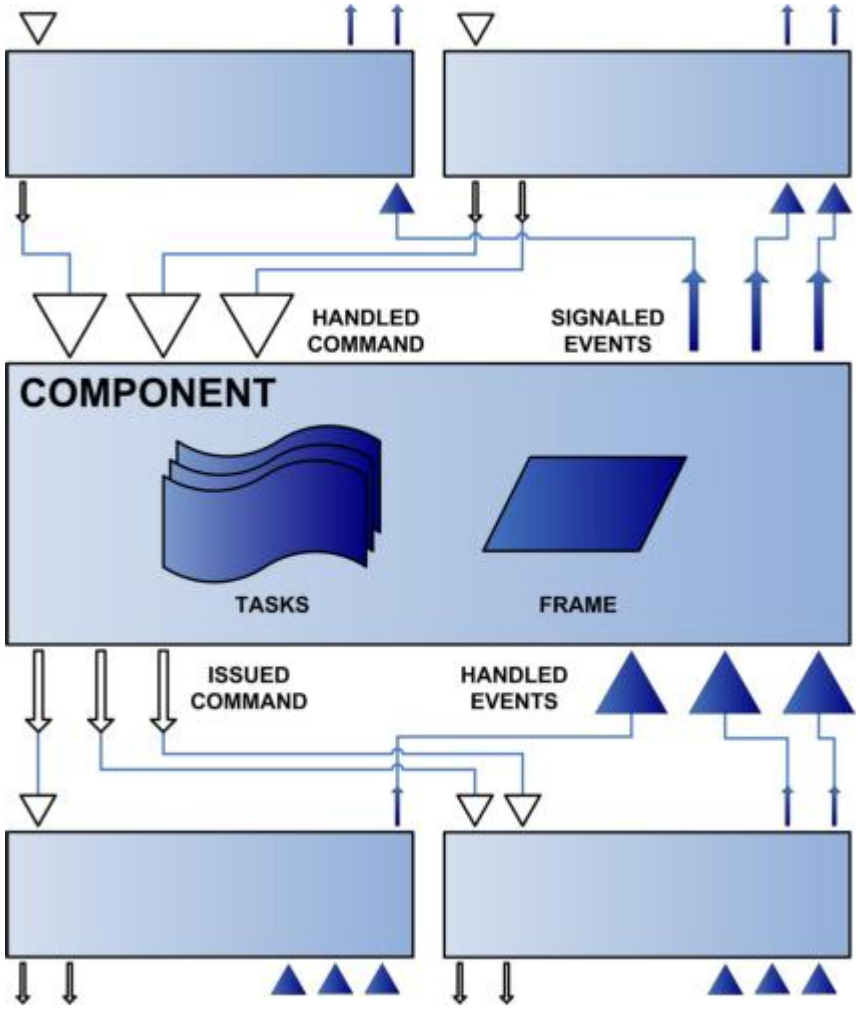
- Tasks and split-phase operations
- Mote-mote radio communication
- The BlinkToRadio Application

In the last episode

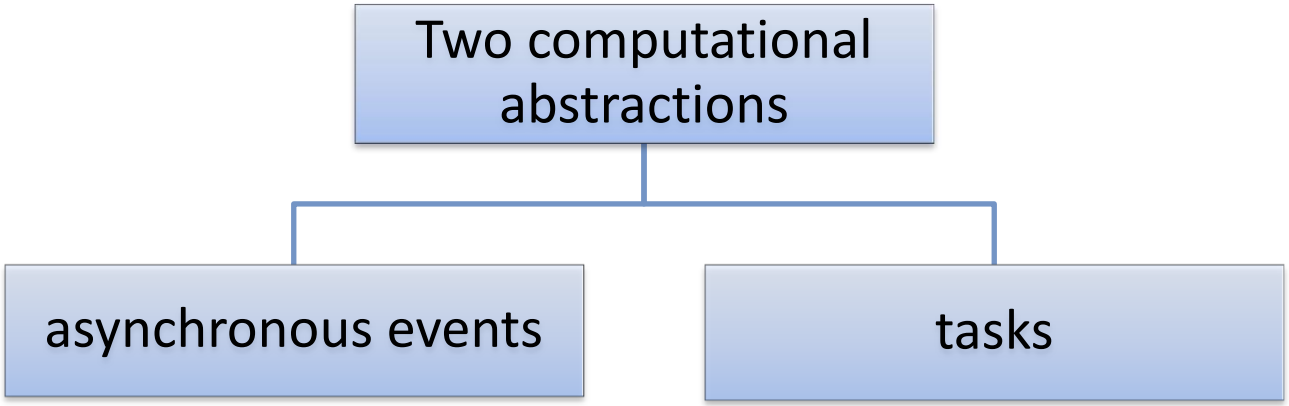
- **NesC**: C dialect
- **TinyOS**: event driven OS
- **split-phase**: call → callback (event)
- **kernel** with two hierachical levels: task and event
- **single stack system**: local variable in the stack, binary code are stored contiguously in memory

In the last episode

1. Application consists of *components* ***wired***
2. Two scopes:
   - Specification (name of interfaces)
   - Implementation
3. App ***provides*** and ***uses*** interfaces
4. Interfaces ⟷ functionalities
5. Interfaces are bidirectional
   - Commands implemented by interface's provider
   - Events implemented by the interface's user



TinyOS Execution Model: Tasks



- can run preemptively
- interrupt handlers
- race conditions!

- schedule a function to be called later
- run in a single execution context
- no preemption!
- FIFO

Sync code should be kept short

Blink application: event handler for Timer0.fired()

```
event void Timer0.fired() {
    uint32_t i;

    call Leds.led0Toggle();
}
```

Let's introduce some latency..

```
event void Timer0.fired() {
    uint32_t i;

    for (i = 0; i < 10001; i++) { }

    call Leds.led0Toggle();
}
```

The `Timer` interface is synchronous → long computations interfere with timers operations

Tasks

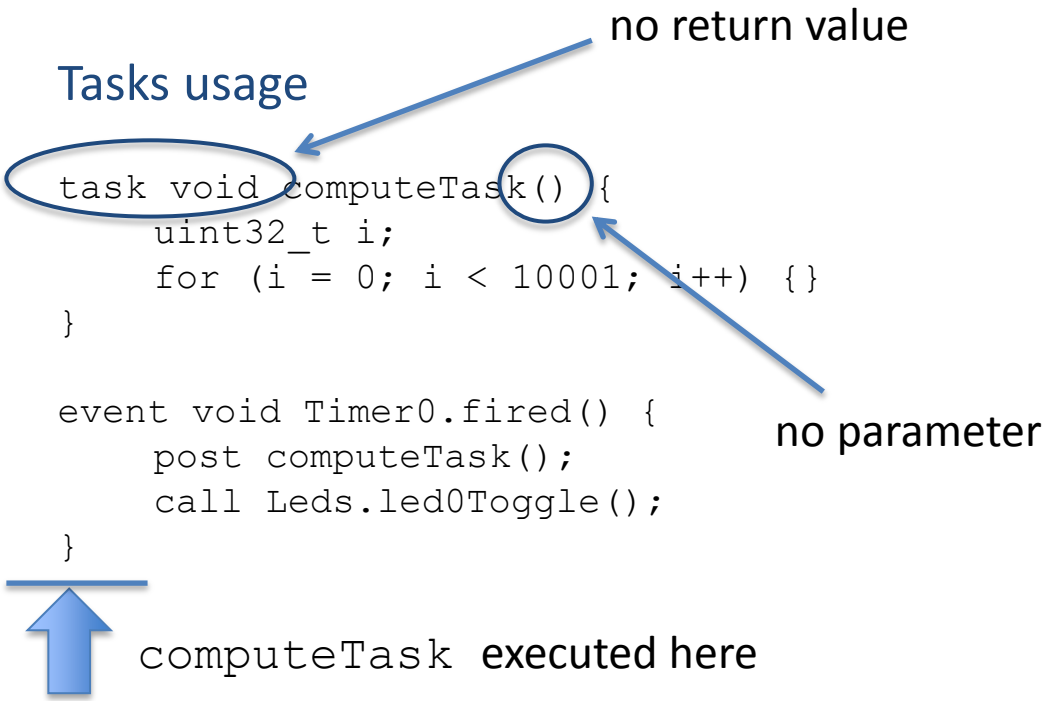
Tasks usage

```
task void computeTask() {
    uint32_t i;
    for (i = 0; i < 10001; i++) {}
}

event void Timer0.fired() {
    post computeTask();
    call Leds.led0Toggle();
}
```

Tasks

Tasks usage



Split-Phase

Interfaces are wired at compile time → optimized callback system.
No blocking operations are allowed.

Blocking	Split-Phase

`state = WAITING;`
`op1();`
 	
`sleep(500);`
 	
`op2();`
`state = RUNNING;`
	`state = WAITING;`
	`op1();`
	
	`call Timer.startOneShot(500);`
	
	`event void Timer.fired() {`
	`  op2();`
	`  state = RUNNING;`
	`}`

Communication Interface

TinyOS provides

i. *Communication services by means of interfaces*
ii. *Components implementing such interfaces*
iii. *Abstract data type* `message_t`

```
typedef nx_struct message_t {
     nx_uint8_t header[sizeof(message_header_t)];
     nx_uint8_t data[TOSH_DATA_LENGTH];
     nx_uint8_t footer[sizeof(message_footer_t)];
     nx_uint8_t metadata[sizeof(message_metadata_t)];
} message_t;
```

Read and Write through accessors and mutuator functions

Communication Interface

In /tos/interfaces:

Basic Communication Interfaces

- Packet, Send, Receive, PacketAcknowledgments, RadioTimeStamping

Active Message Interfaces

- AMPacket, AMSend

In /tos/system:

Basic Components

1. AMReceiverC
2. AMSenderC
3. AMSnooperC
4. AMSnoopingReceiverC
5. ActiveMessageAddressC

ActiveMessageC for the telosb are all implemented by CC2420ActiveMessageC

Today Application: BlinkToRadio

A one-timer version of Blink application that sends the counter value over the radio channel.

First Step:

Implement a version of Blink using a single timer and the `set` function (look at Leds component implementation).

Tutorial Link

docs.tinyos.net/tinywiki/index.php/Mote-mote_radio_communication