Sistemas Distribuídos com Redes de Sensores

Noemi Rodriguez

2012

TinyOS – SO para motes com recursos limitados

- aplicação única
- bateria deve ter vida longa
- memória muito limitada
- Exemplo: micaz
  - Atmel ATmega128: microcontrolador de 8 bits
  - 4K memória RAM
  - 128K memória flash
  - chip de rádio TI CC2420: IEEE 802.15.4 a 250 kbps

TinyOS - Características

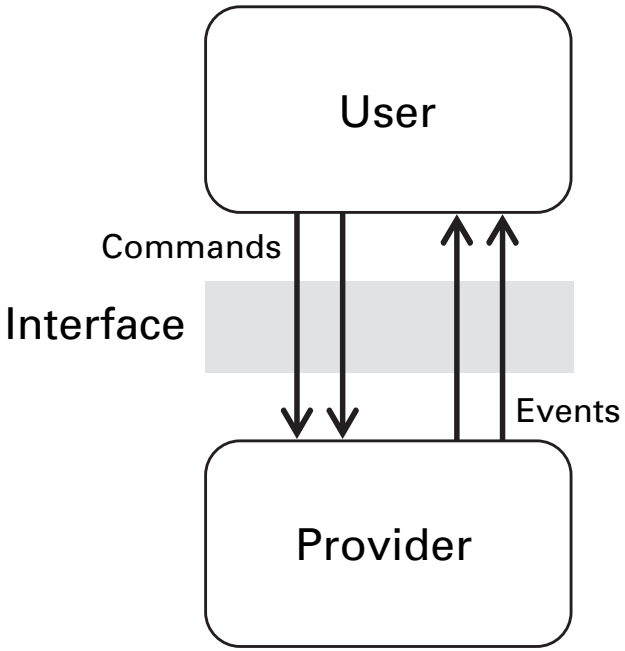
- conjunto de serviços:
  - sensoreamento
  - comunicação
  - armazenamento
  - timers
- modelo de execução concorrente
- linguagem NesC

Programação no TinyOS

- modelo de componentes de NesC
  - modularização e reuso
- modelo de concorrência
  - interação entre componentes: *split phase*
  - não há threads com pilhas independentes
  - interação com interrupções
- API: conjunto de componentes prontos para uso

Programação em NesC

- aplicação composta por componentes chamados de *módulos* e *configurações*
- interação através de *interfaces* nem definidas
- módulos implementam código e configurações fazem *wiring*



Programação em NesC – interfaces

- *interfaces* declaram conjunto de *comandos* e eventos
  - podem ser interpretados como serviços e *callbacks*
- interfaces podem ser implementadas por módulos ou por configurações

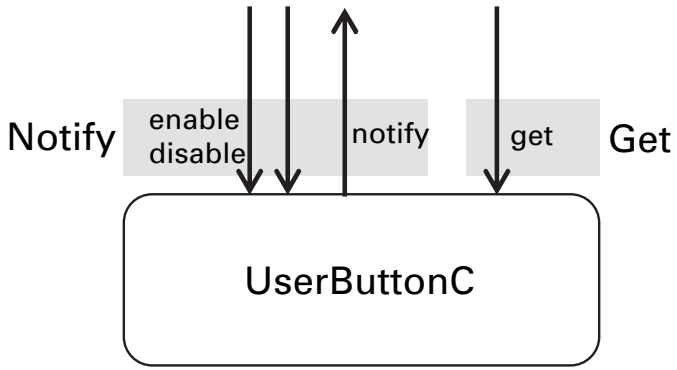
```
interface Boot {
  event void booted();
}

interface Leds {
  command void led0On();
  command void led0Off();
  command void led0Toggle();
  ...
}
```

Programação em NesC – componentes

- componentes compostos por 2 blocos: assinatura e implementação
  - assinatura contém descrição de interfaces fornecidas (servidores) e usadas (clientes)

```
interface Notify<val_t> {
  command error_t enable();
  command error_t disable();
  event void notify(val_t val);
}
```



Programação em NesC – componentes

- sintaxe C estendida
- deve implementar comandos e pode gerar eventos de interfaces fornecidas
- pode chamar comandos e deve tratar eventos de interfaces utilizadas

```
module PowerupC {
  uses interface Boot;
  uses interface Leds;
}
implementation {
  event void Boot.booted() {
    call Leds.led0On();
  }
}
```

Programação em NesC – configurações

- amarração entre utilizadores (clientes) e fornecedores (servidores)
- amarração entre interfaces utilizadas e componentes que as oferecem

```
configuration PowerupAppC {
}
implementation {
  components MainC, LedsC, PowerupC;
  MainC.Boot -> PowerupC.Boot;
  PowerupC.Leds -> LedsC.Leds;
}
```

Programação em NesC – interfaces e componentes genéricos

- interfaces *genéricas* permitem a parametrização por tipos

```
interface Timer<precision_tag> {
  command void startPeriodic(uint32_t dt);
  command void stop();
  event void fired();
  ...
}
```

- componentes *genéricos* permitem instâncias múltiplas

```
configuration BlinkAppC {}
implementation {
  components MainC, BlinkC, LedsC;
  components new TimerMilliC() as Timer0;
  components new TimerMilliC() as Timer1;
  components new TimerMilliC() as Timer2; /* Wirings below */
  ...
```

Programação em NesC – alocação de memória

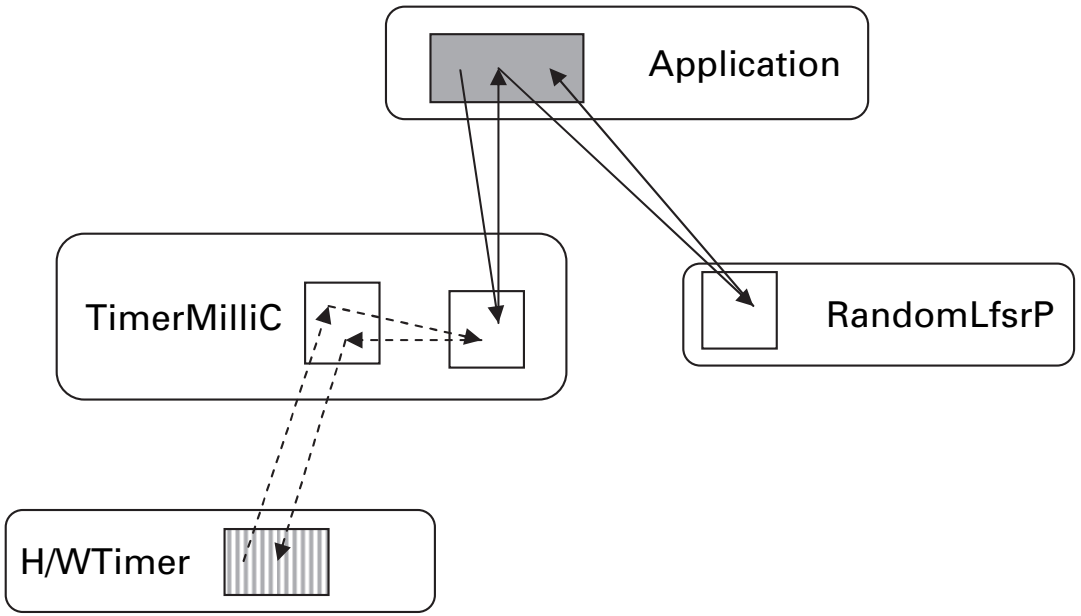
- memória é recurso precioso...
- restrições com alocação dinâmica e recursão

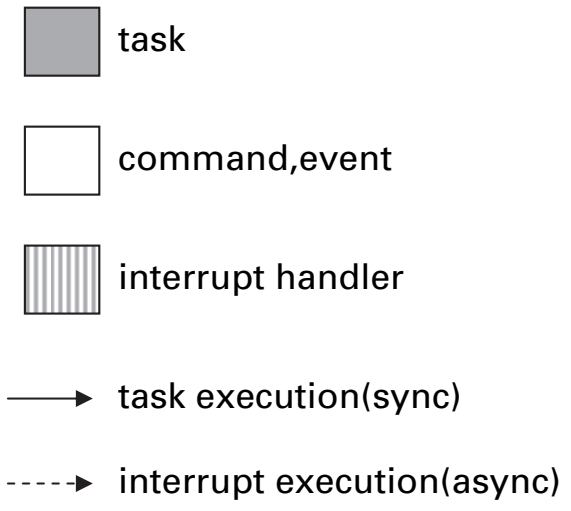
Programação em NesC – *split phase*

```
interface Send {
  command error_t send(message_t* msg, uint8_t len);
  event void sendDone(message_t* msg, error_t error);

  command error_t cancel(message_t* msg);
  command void* getPayload(message_t* msg);
  command uint8_t maxPayloadLength(message_t* msg);
}
```

TinyOS – modelo de execução





Bibliografia

- Philip Levis and David Gay. *TinyOS Programming*. Cambridge University Press. 2009.
- http://docs.tinyos.net