RSSF + Testbed + Terra

Adriano Branco
abranco@inf.puc-rio.br

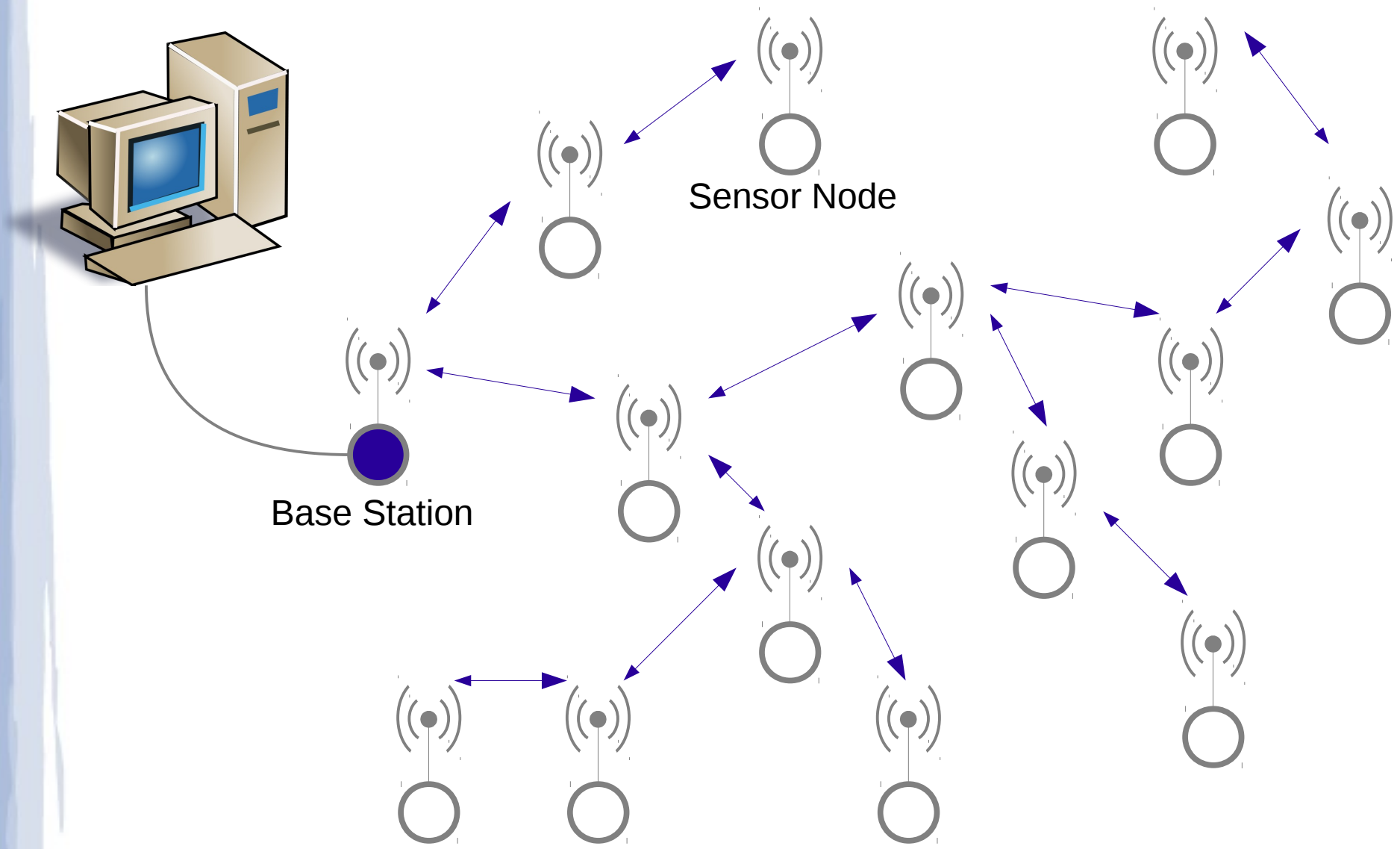
Outubro,2014

Agenda

- Parte 1
  - Introdução a RSSF
  - Programação em RSSF - TinyOS
- Parte 2
  - Testbed
- Parte 3
  - Terra – Simplificando a programação
  - Tarefas práticas usando Terra

Introdução RSSF

Rede de Sensores sem Fio (RSSF)



RSSF - Mote

TMote Sky
(Sentilla)

MicaZ (Crossbow)

WeC (Berkeley)

- Pequeno, baixa potência e sem fio
- Mínimo de CPU, memória e rádio
  - Tipicamente 8Mhz, 4~10KB RAM
  - Alcance do rádio de 100 m
- Baixo consumo
  - Um par de baterias AA pode operar por meses ou anos.

Rene (Berkeley)

iMote2 (Intel)

Redes de Sensores sem Fio

- Aplicações
  - Internet das coisas
  - Espaços inteligentes
  - Cidades inteligentes
- Exemplos
  - Automação predial e residêncial
  - Monitoração ambiental
  - Monitoração médica
  - Etc...

Alguns desafios em RSSF

- Hardware x Software x Energia
- Aplicações específicas x Frameworks
- Comunicação
  - Intercomunicação/Roteamento x Economia de energia
- Programação
  - Facilitar a já complicada programação em Sistemas Distribuidos
  - Linguagens e Sistemas Operacionais
  - Middlewares e Máquinas Virtuais
  - Etc..

Programação em RSSF

Desafios na construção de aplicações

- Limitação de recursos
  - Tempo de vida da bateria – o rádio é o maior consumidor de energia
  - Microcontrolador – RAM:4K~10K ROM:132K~48K
- Comunicação
  - Rede Ad-hoc, instabilidade dos nós, ruído, colisão de rádio, etc...
- Programação
  - Sistema distribuido com um modelo de programação orientado a eventos
  - Programação remota

Suporte a programação

- Principais sistemas
  - TinyOS
    - Orientação a eventos baseado num modelo de componentes como uma extensão de C.
  - Contiki
    - Cooperativo e preemptivo programado via macros em C.
- Exemplos de outros sistemas
  - MANTIS, Nano-RK, LiteOS, SOS
- Simuladores
  - TinyOS:TOSSIM, Micaz HW:Avrora, Contiki:Cooja

Operações “Split-phase”

- Operações em duas etapas
  - Primeiro dispara a operação (comando)
  - Final da operação interrompe o sistema (evento)
- Exemplos
  - Conversor A/D dos sensores
    - Comando: read()
    - Evento: readDone(valor)
  - Envio de msg via rádio
    - Comando: send(msg)
    - Evento: sendDone(status)
  - Recebe msg do rádio
    - Evento: receive(msg)

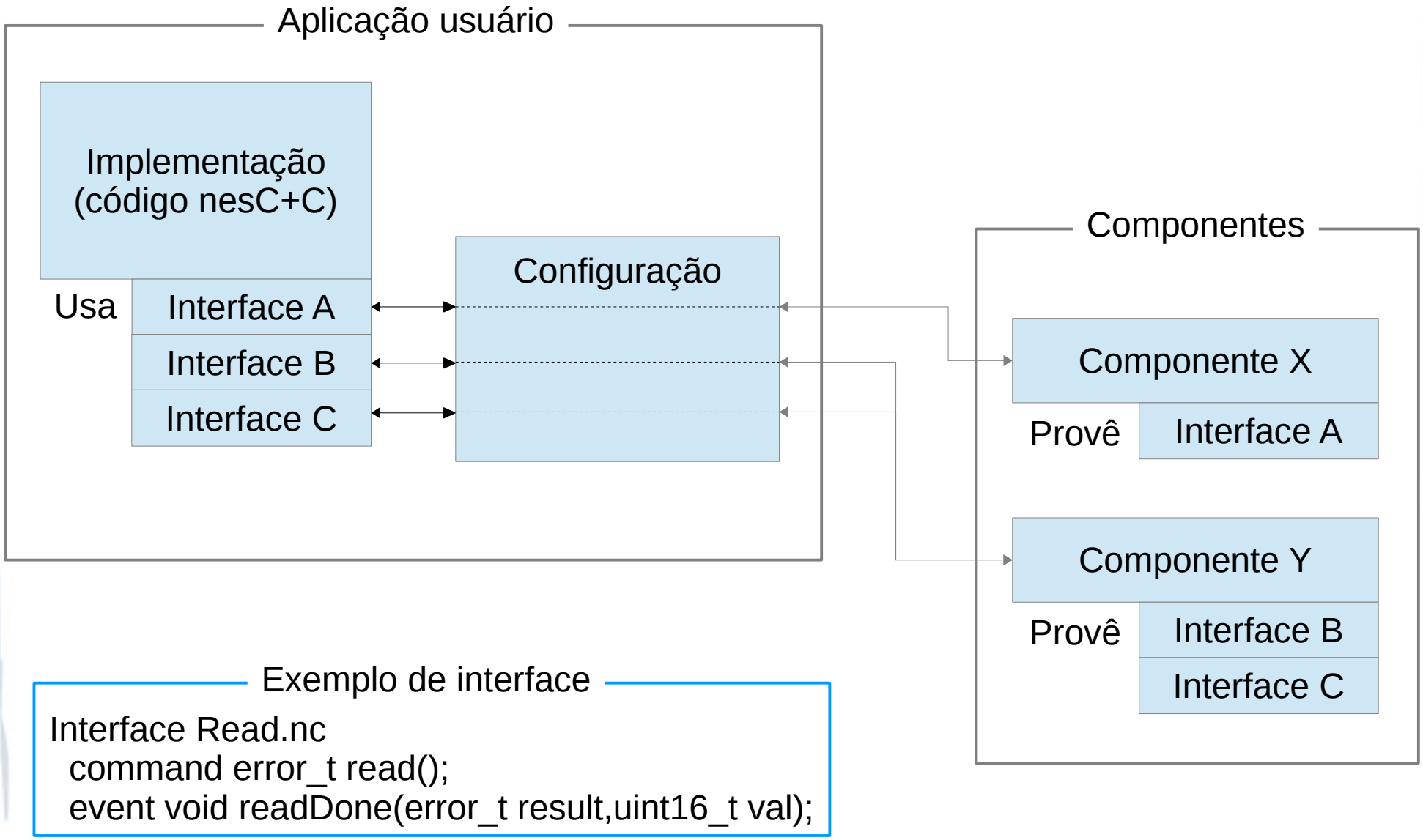
Tipos de variáveis

- Normalmente utiliza-se tipos inteiros com tamanho explicito
  - int8_t, int16_t, int32_t
  - uint8_t, uint16_t, uint32_t
- Tipo network – big-endian
  - nx_int8_t, nx_int16_t, nx_int32_t
  - nx_uint8_t, nx_uint16_t, nx_uint32_t
- Estruturas de C – struct e nx_struct

TinyOS: Tasks x Scheduler

- O Scheduler executa uma tarefa até o final por vez. (FIFO)
- Não aceita tarefas duplicadas, mas uma tarefa em execução não ocupa a fila.
- Tarefas não tem parâmetros e não retornam valores.
- Suporte nesC para definição de tarefa e “postagem”.
  - task void xxx(){ …}
  - ….; post xxx(); …..;
- Praticamente a execução do usuário são derivados de tarefas.

NesC – Modelo de Componentes



Exemplo de interface

```
Interface Read.nc
  command error_t read();
  event void readDone(error_t result,uint16_t val);
```

NesC: Comandos x Eventos

Definição de uma interface

```
Interface xxx;
   command void start();
   event void started();
```

Código A

```
uses interface Boot;
uses interface xxx;

event Boot.booted(){
  ….;
  ….;
  call xxx.start();
}

event xxx.started(){
  ….;
  ….;
}
```

Código A

```
provides interface xxx;

command void start(){
   ….;
   ….;
   ….;
   signal xxx.started();
   ….;
   }
```

NesC: Comandos x Eventos

Definição de uma interface

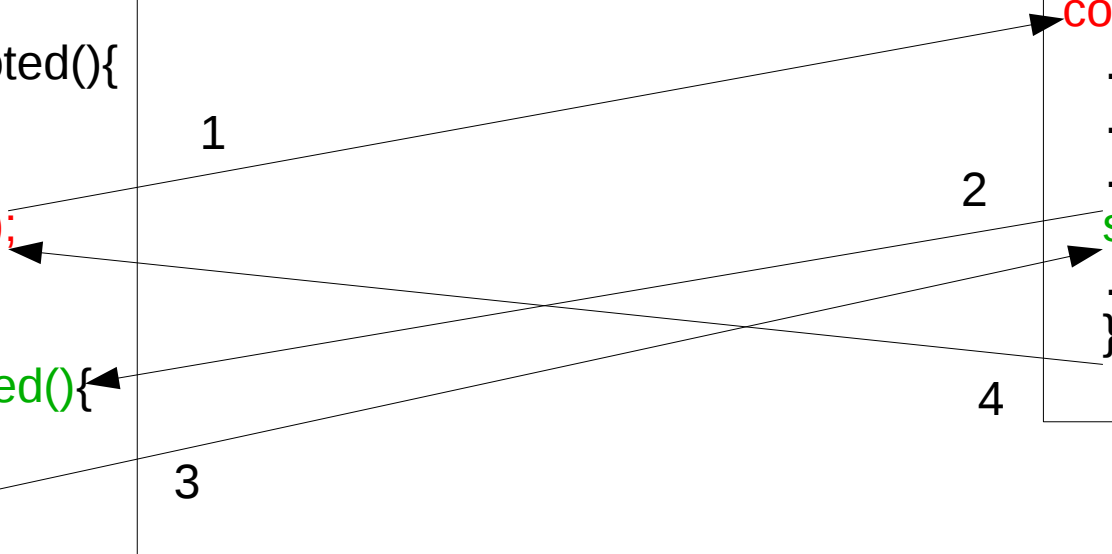
```
Interface xxx;
   command void start();
   event void started();
```

Código A

```
uses interface Boot;
uses interface xxx;

event Boot.booted(){
  ….;
  ….;
  call xxx.start();
}

event xxx.started(){
  ….;
  ….;
}
```

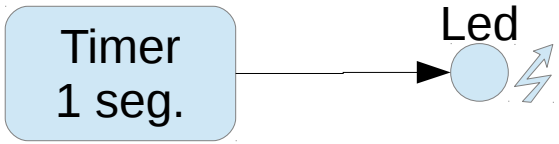
Código B

```
provides interface xxx;

command void start(){
   ….;
   ….;
   ….;
   signal xxx.started();
   ….;
   }
```



Exemplo Simples

- Blink
  - Três LEDs
  - Três temporizadores (Timer)
  - A cada disparo do temporizador, o respectivo Led troca de estado. (Aceso/Apagado)
  - Na inicialização, configuramos cada temporizador com disparo periódico para um determinado tempo.



Blink - Implementação

```
module BlinkC
{
  uses interface Timer<TMilli> as Timer0;
  uses interface Timer<TMilli> as Timer1;
  uses interface Timer<TMilli> as Timer2;
  uses interface Leds;
  uses interface Boot;
}
implementation
{
  event void Boot.booted()
  {
    call Timer0.startPeriodic( 250 );
    call Timer1.startPeriodic( 500 );
    call Timer2.startPeriodic( 1000 );
  }
...
```

```
...
  event void Timer0.fired()
  {
    dbg("BlinkC", "Timer 0 fired @ %s.\n", sim_time_string());
    call Leds.led0Toggle();
  }

  event void Timer1.fired()
  {
    dbg("BlinkC", "Timer 1 fired @ %s \n", sim_time_string());
    call Leds.led1Toggle();
  }

  event void Timer2.fired()
  {
    dbg("BlinkC", "Timer 2 fired @ %s.\n", sim_time_string());
    call Leds.led2Toggle();
  }
}
```

Blink - Configuração

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


  BlinkC -> MainC.Boot;

  BlinkC.Timer0 -> Timer0;
  BlinkC.Timer1 -> Timer1;
  BlinkC.Timer2 -> Timer2;
  BlinkC.Leds -> LedsC;
}
```

Ambiente de desenvolvimento

(senha: terra)

- Máquina virtual Ubuntu pré-instalada com:
  - TinyOS
  - TinyOS Tools
  - TOSSIM/Avrora
  - Customização Terra – TerraNet
    - Editor + Compilador
    - Simulador Visual
    - Ferramenta de carga remota e monitoração
  - SerialForwarder – comunicação PC ↔ Mote
  - TOSSAM – Suporte Lua para PC ↔ Mote

Tarefa rápida (1/2)

- Ir para o diretório tos/Blink
  - BlinkC.nc → Implementação
  - BlinkAppC.nc → Configuração
  - Makefile → Compilação
  - ex1.py → Script de simulação

Tarefa rápida (2/2)

- Compilar o Blink para micaz e instalar no mote
  - make micaz
  - make micaz install.1 mib520,/dev/ttyUSB0
- Simulador TOSSIM
  - Compilar: make micaz sim
  - Executar: python ex1.py
  - Edite o BlinkC.nc e observe as linhas “dbg()”
- Altere os tempos dos timers e verifique o log da execução.

Final da Parte 1