Internet of Things (IoTs)
Development Software Platforms

By

Quan LE-TRUNG, Dr.techn.

https://sites.google.com/site/quanletrung/

Jul.2015, UIT

Outlines

- Review on different open-source development software platforms
- TinyOS Introduction
- TinyOS Applications
- Running TinyOS Applicaition with WSIM/WSNET/AVRORA
  - Provided Tutorials, Do at home
- Inside IoTs: in-network processing & routing

IoTs Open-Source Development Software Platforms (1/2): Platforms

- ContikiOS, http://www.contiki-os.org/
- TinyOS, http://www.tinyos.net/
- Adruino, http://www.arduino.cc/
- BeagleBoard, http://beagleboard.org/
- Raspbian, http://raspbian.org/
- OpenIoT, http://openiot.eu/
- Cloud-based IoTs [Emulation/Scalability/Availability]
- Simulation/Emulation: ns-2, InstantContiki, Wsim/Wsnet/Avrora
- …

IoTs Open-Source Development Software Platforms (2/2): Standards

- Open InterConnect Consortium (OIC), http://openinterconnect.org/
- AllSeen Alliance, https://allseenalliance.org/
- ZigBee Alliance, http://www.zigbee.org
- **IPSO Alliance**, http://www.ipso-alliance.org
- IETF. **6LowApp Charter**, http://trac.tools.ietf.org/area/app/trac/wiki/BarBofs/IETF75/6LowApp
- IETF. **ROLL Charter**, http://www.ietf.org/dyn/wg/charter/roll-charter.html
- IEEE 802.15 WPAN™ Task Group 4 (TG4), http://www.ieee802.org/15/pub/TG4.html
- IEEE 802.15 WPAN™ Task Group 1 (TG1), http://www.ieee802.org/15/pub/TG1.html
- IEEE 802.11TM WIRELESS LOCAL AREA NETWORKS, http://www.ieee802.org/11/
- HART Communication Foundation. **Wireless HART Technology.** http://www.hartcomm.org/protocol/wihart/wireless_technology.html
- http://www.isa.org/isa100
- http://www.iai-tech.org
- http://www.buildingsmart.com
- *Open Geospatial Consortium* (OGC). **CITYGML**: http://www.opengeospatial.org/standards/citygml
- …

Outlines

- Review on different open-source development software platforms
- TinyOS Introduction
- TinyOS Applications
- Running TinyOS Applicaition with WSIM/WSNET/AVRORA
  - Provided Tutorials, Do at home
- Inside IoTs: in-network processing & routing

TinyOS Introduction [1/x]: TinyOS

- Open-source operating system designed for wireless embedded sensor networks
- Component-based architecture
- Enables rapid implementation with minimal code size
- Event-driven execution model enables fine-grained power management, power/battery saving capabilities, etc.
- http://www.tinyos.net/

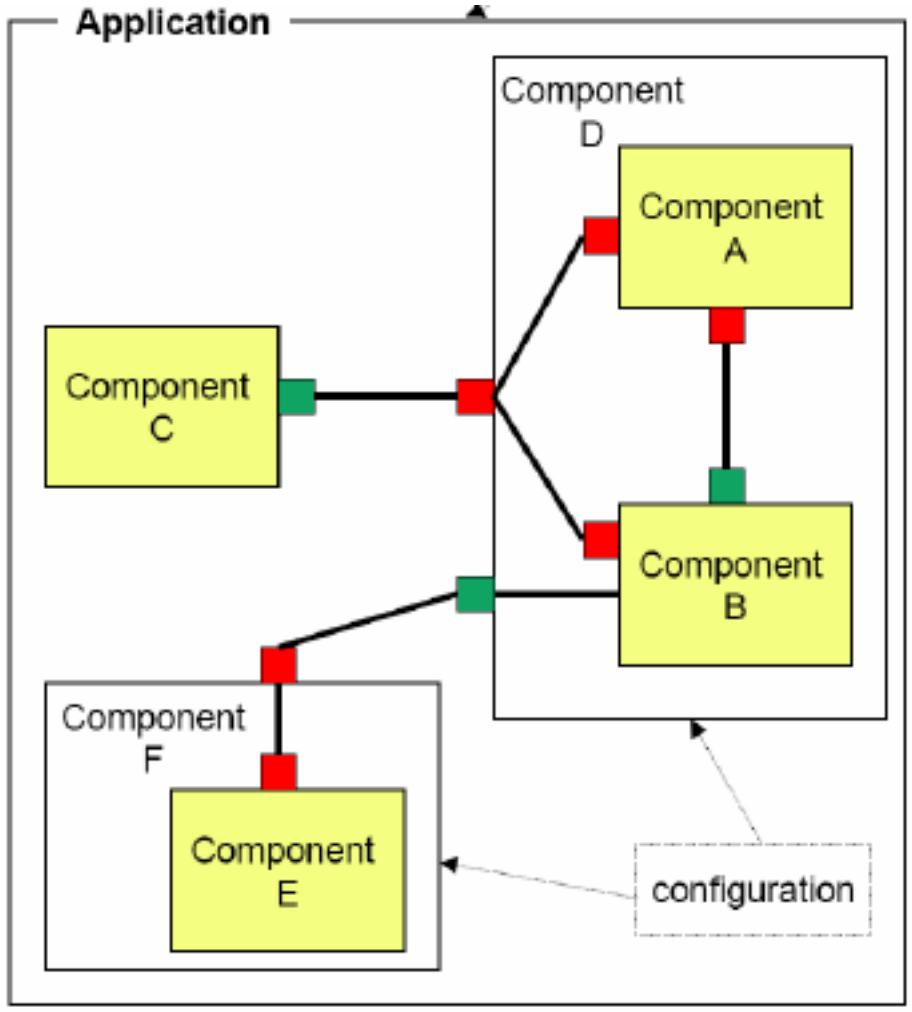
TinyOS Introduction [2/x]: nesC

- (New) structured component-based language
- C-like syntax (extension of C-language)
- TinyOS operating system, libraries and applications are written in nesC
- Supports the TinyOS concurrency model
- Goal: build components that can be easily composed into complete, concurrent systems
- http://www.tinyos.net/

  - No Dynamic Memory (no malloc)
  - No Function Pointers
  - No Heap

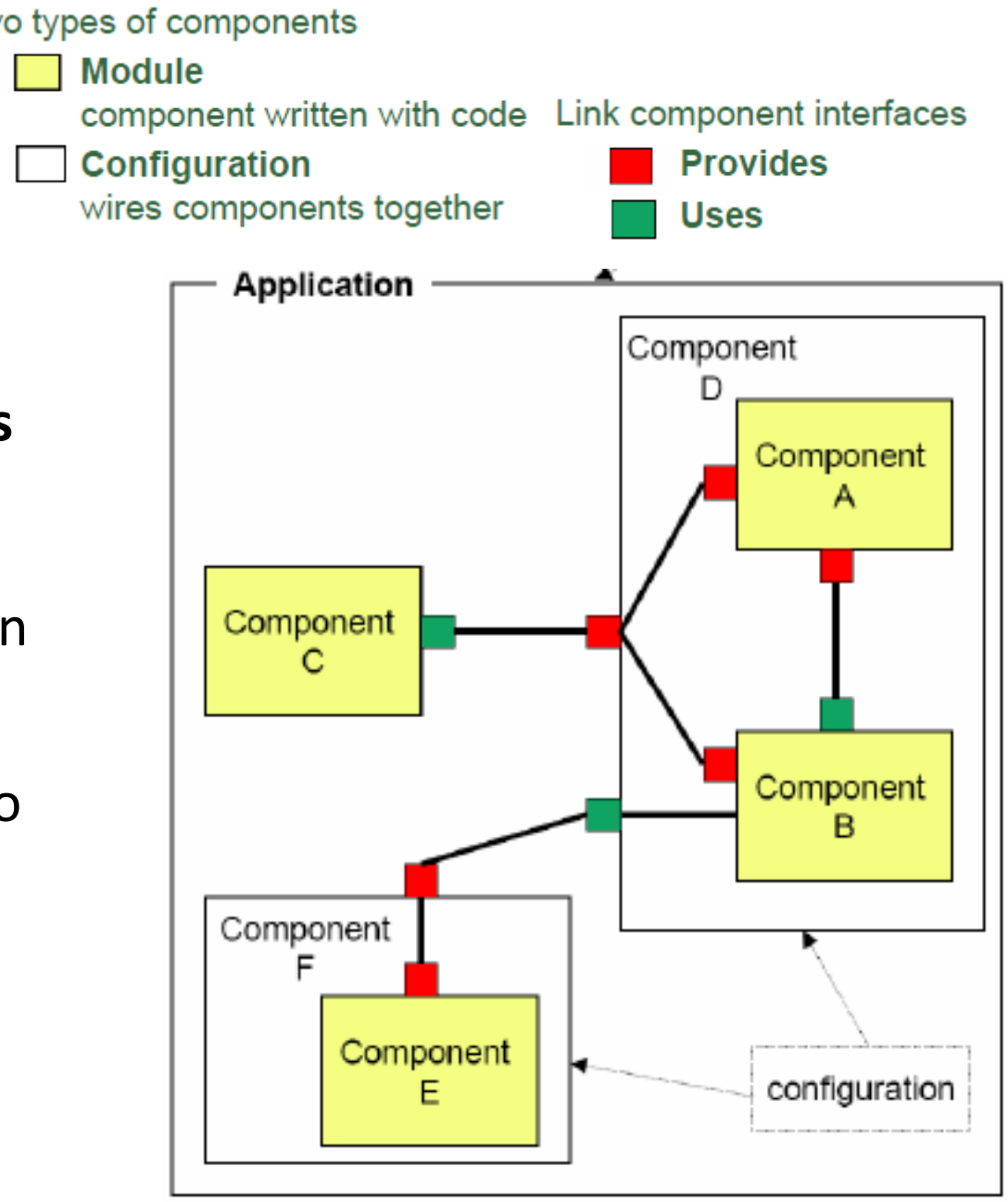
TinyOS Introduction [3/x]: Programming concepts

- Application
  - A TinyOS/nesC **application** consists of one or more **components**
  - Are linked/wired together via **configurations** to get a **run-time executable**
- Component
  - Basic building blocks for nesC applications
  - Two types: Modules & Configurations
  - Can provide and use interfaces (bidirectional in contrast to unidirectional in common programming languages like JAVA)
- Module
  - A component that implements one or more interface
  - Contains application code...



TinyOS Introduction [3/x]: Programming concepts

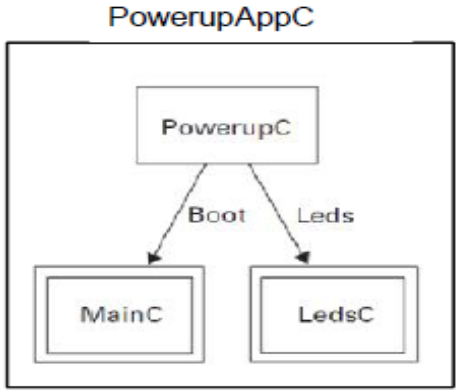
- Configuration
  - A **component** that wires **other components** together
  - Idea: build **applications as set of modules, wiring together** by providing a configuration
- Interface
  - **Abstract definition** of the interaction of two components
  - Concept is similar to JAVA interfaces
  - NesC-interfaces are **bidirectional**



Components

A configuration is a component built out of other components.
It *wires* "used" to "provided" interfaces.
It can instantiate *generic* components
It can itself provide and use interfaces

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




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

Points to the implementer

Wiring: Pick implementations for used interfaces

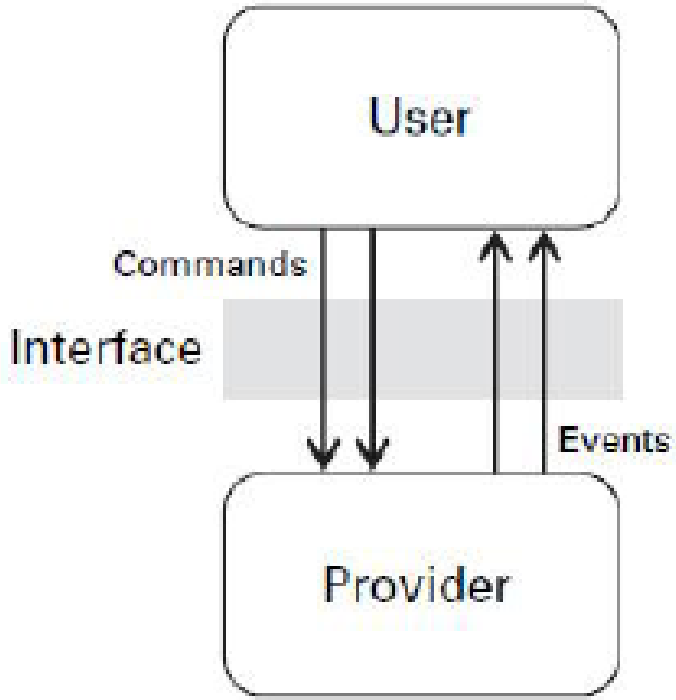
- Basic unit of *nesC code is compon*
  - *Configurations*
    - Wire other components together
    - Configurations compose modules into larger abstractions
  - *Modules*
    - Variables and executable codes
    - Implement program logic

Components start with a *signature* specifying
- the interfaces *provided* by the component
- the interfaces *used* by the component

A module is a component implemented in C
- with functions implementing commands and events
- and extensions to call commands, events

Interfaces

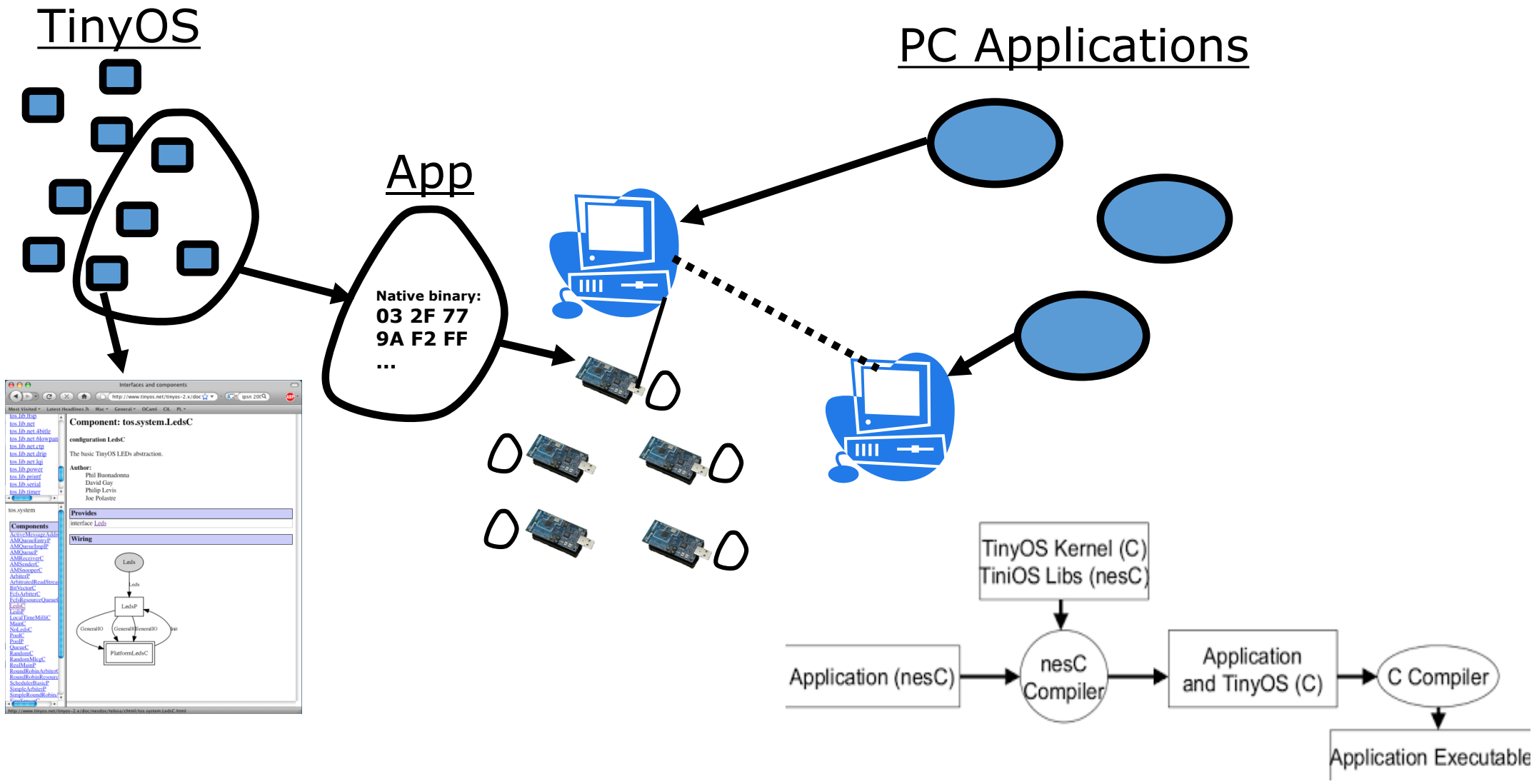
- Collections of related functions
- Define interactions between components
- Interfaces are bidirectional
  - **Commands**
    - Implemented by provider
    - Called by user
  - **Events**
    - Called (signaled) by provider
    - Implemented (captured) by user



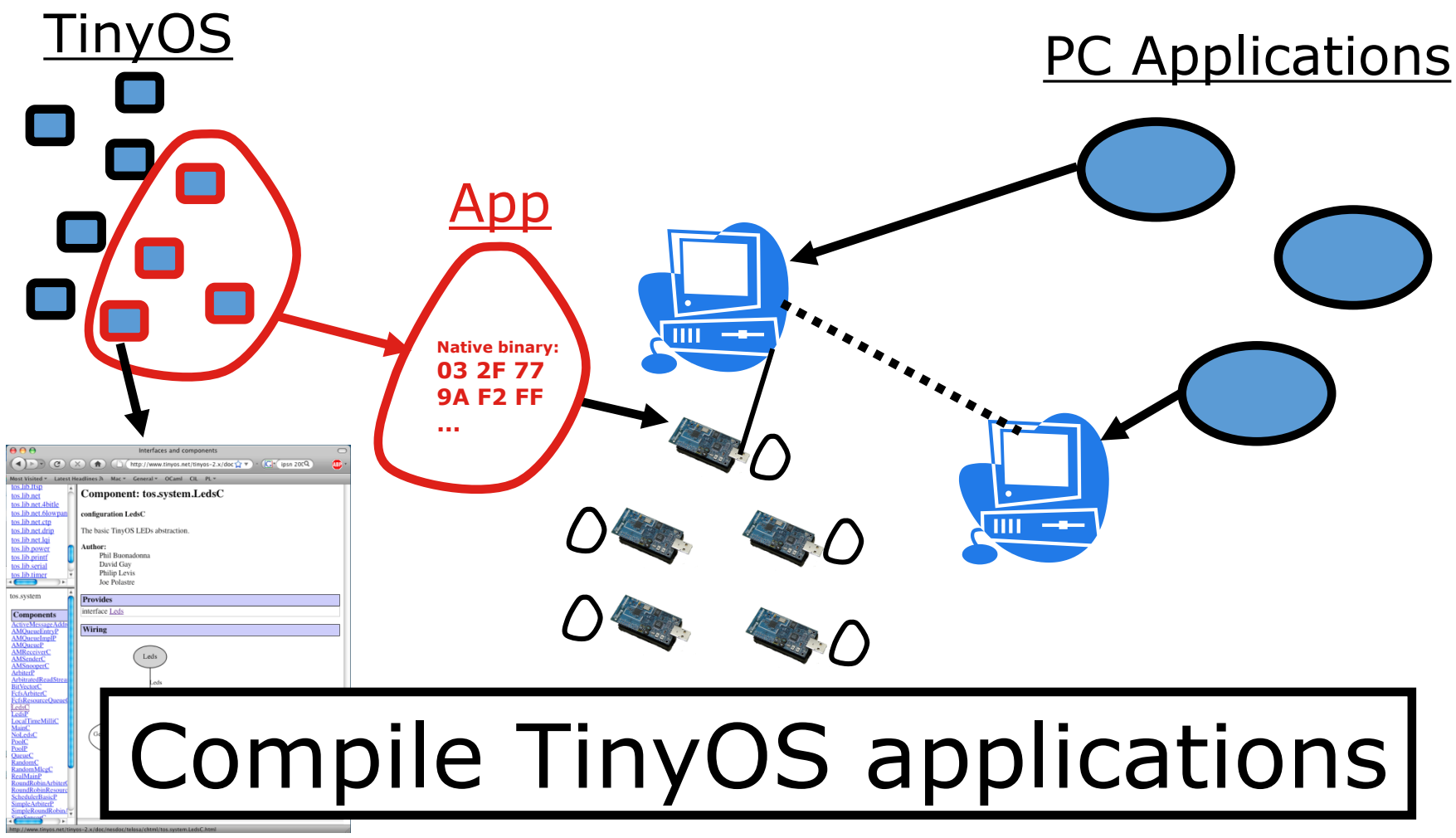
In TinyOS, all long-running operations are split-phase:
• A command starts the op: read
• An event signals op completion: readDone
Errors are signalled using the error_t type, typically
• Commands only allow one outstanding request
• Events report any problems occurring in the op

```
interface Timer<tag> {
  command void startOneShot(uint32_t period);
  command void startPeriodic(uint32_t period);
  event void fired();
}
```

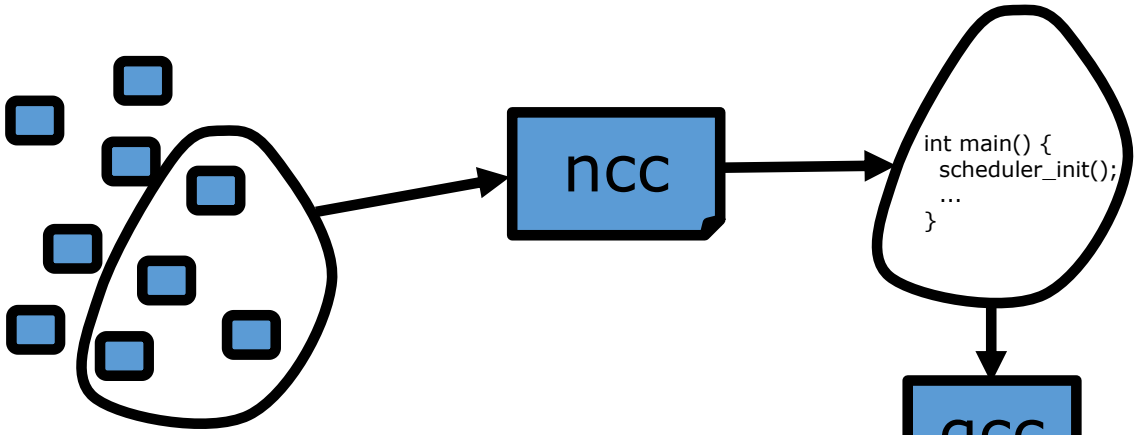
The Toolchain

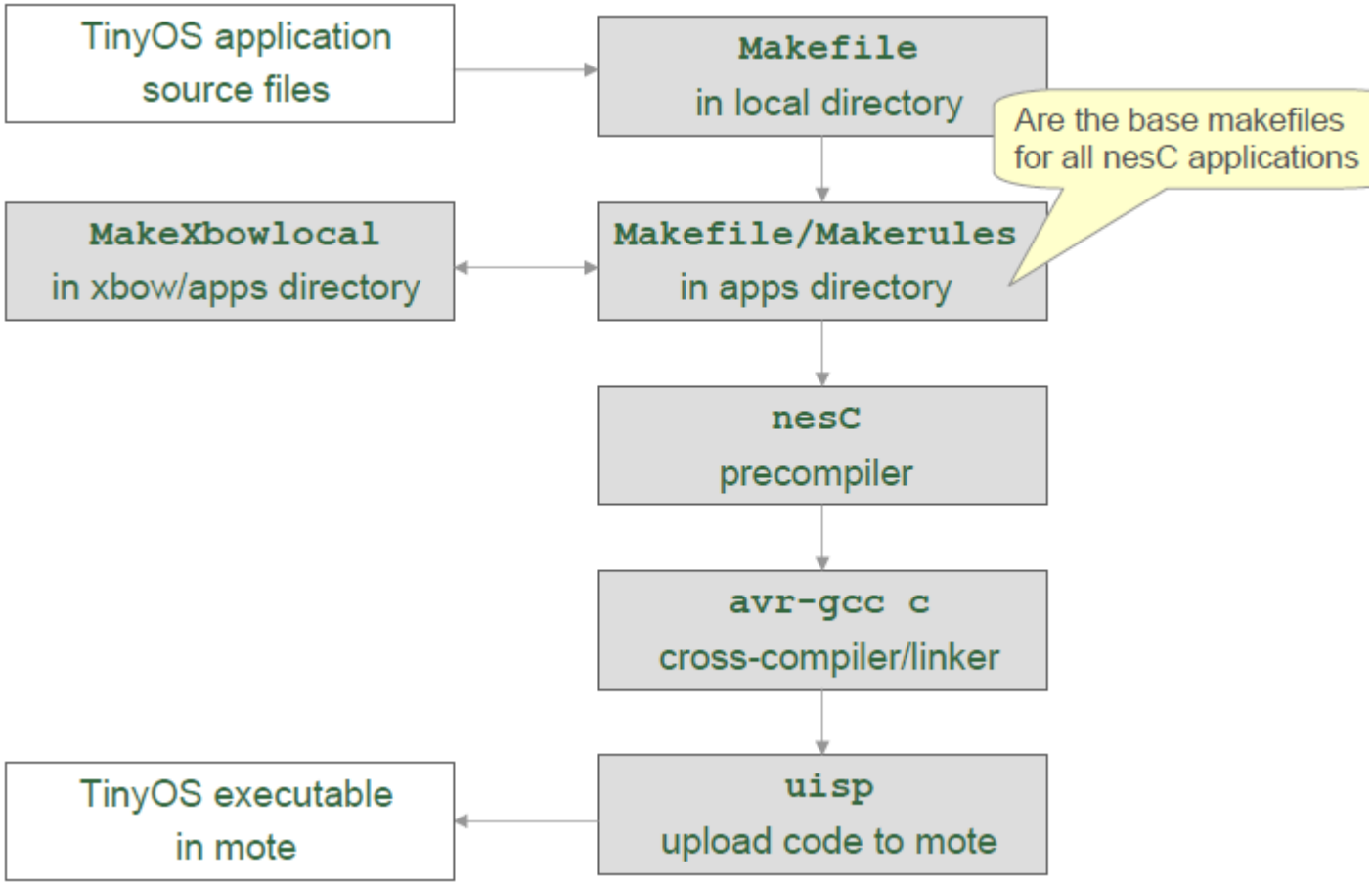


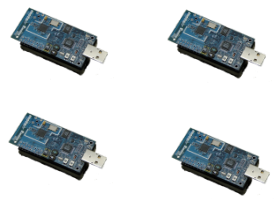
The Toolchain



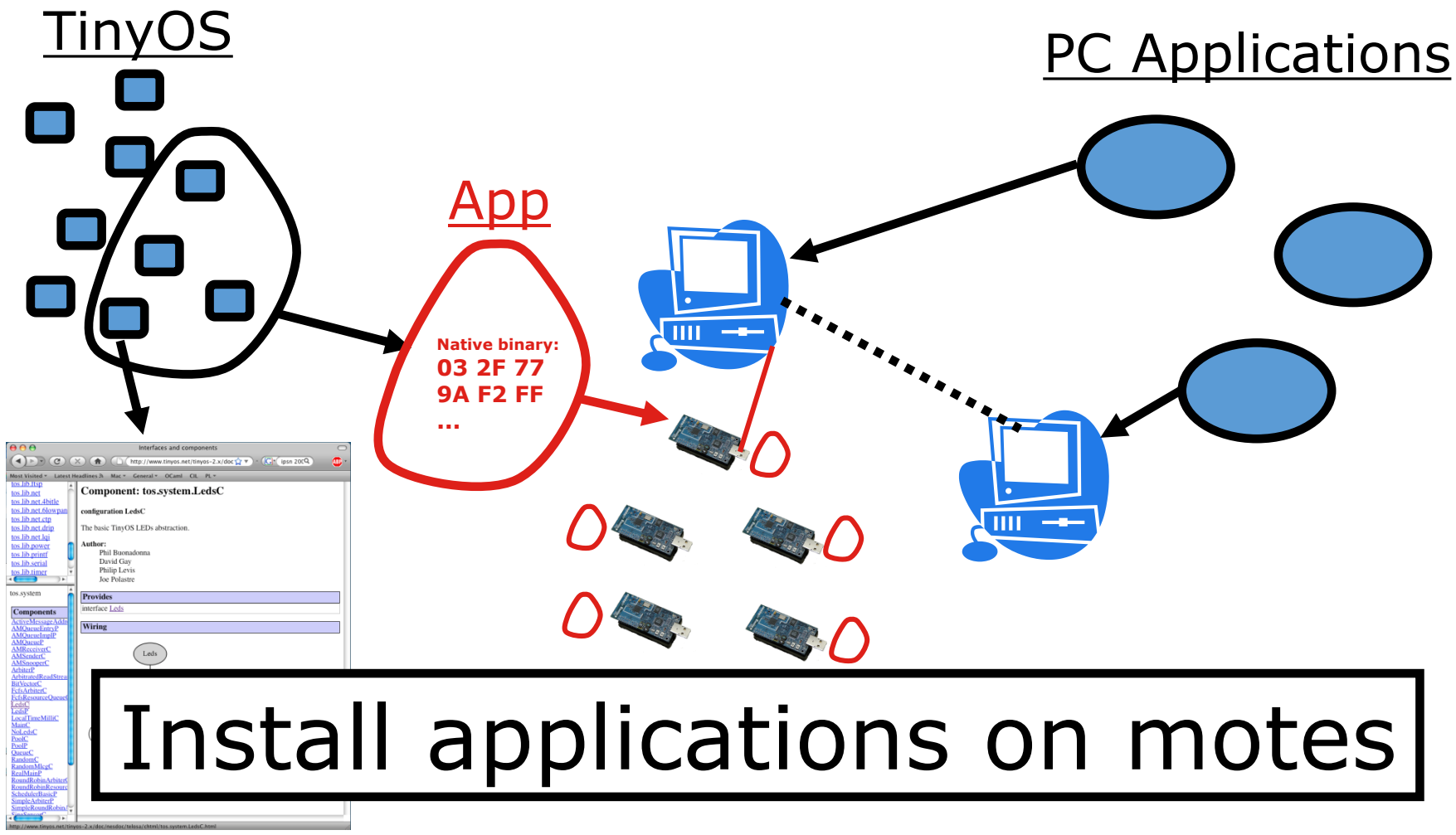
“Make”: Compile Applications





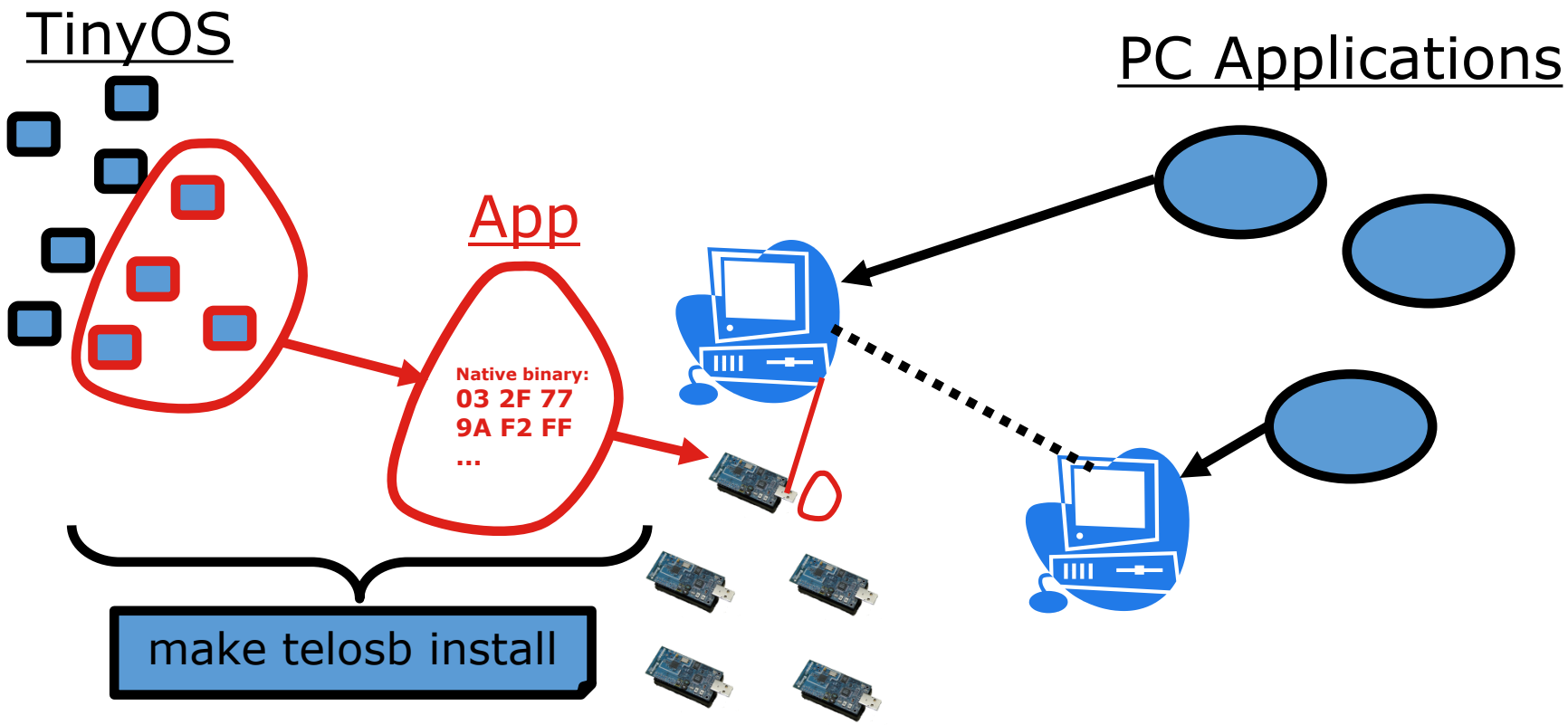


The Toolchain



Install applications on motes

The “Make” System



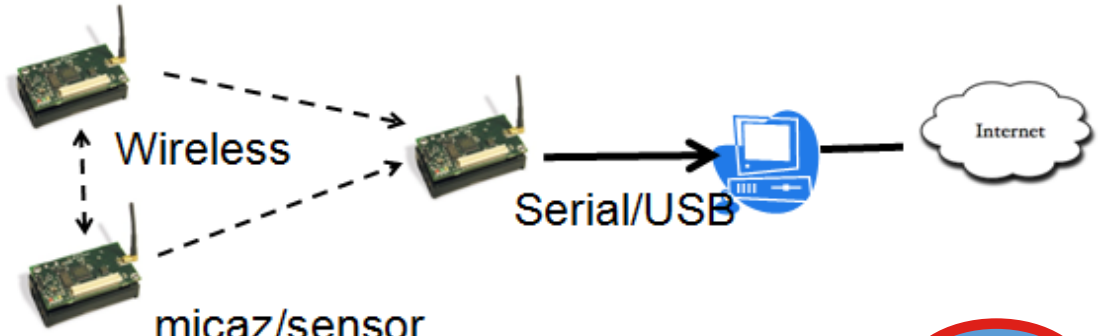
automates nesC, C compilation,
mote installation

The Toolchain

Sensor code (nesC/TinyOS)

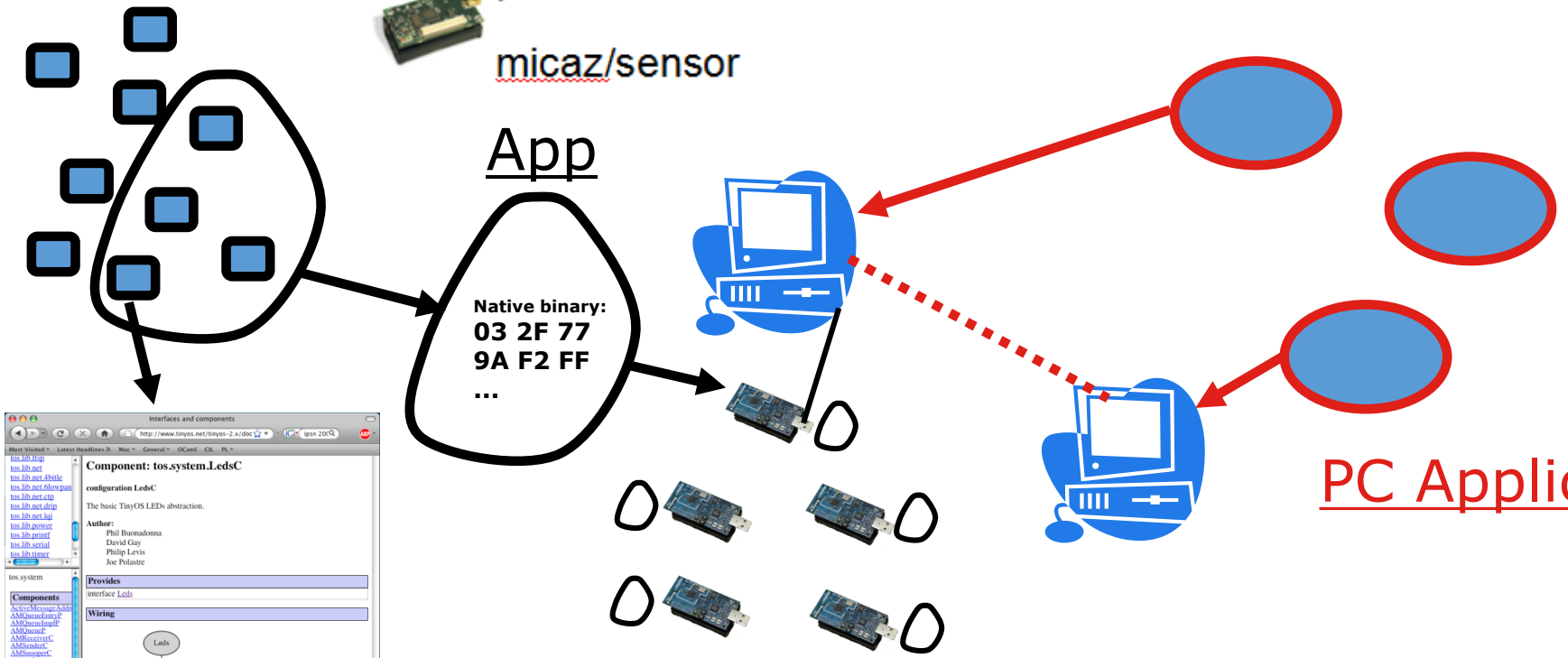
Base station code (nesC/TinyOS)

Gateway code (Java, c, …)



TinyOS

App



PC Applications

Build PC applications

https://github.com/tinyos/tinyos-main

apps	worked around tinyos.jar dependency
deprecated/tosthreads	mv Deprecated to deprecated to conform to TEP 3
doc	Docs: Updated READMEs
licenses	licences/bsd: bump year and tweak a touch
packaging	workaround for #311 - parellel compilation creates empty jar
support	generic NeighborhoodC (for multiple independent rfxlink) - resolves #273
tools	Corrects the list of targets for python tools.
tos	rfxlink: remove deprecated ReceiveDefault and SnoopDefault
.gitignore	Updated .gitignore
Makefile.include	Set TINYOS_ROOT_DIR in Makefile, change var names
README.md	Update README.md
README.tinyos	Update instructions for repo use
release-notes.txt	Merge branch 'master' of git://github.com/markushx/tinyos-main into c...

Architectures

- AVR
  - mica2, mica2dot
  - micaz
  - btnode
  - IRIS
- ARM
  - imote2
- MSP430
  - telosb, sky
  - shimmer
  - eyesIFX
  - tinynode
  - epic
- 8051
  - CC2430
  - CC1110/CC1111

TinyOS

Outlines

- Review on different open-source development software platforms
- TinyOS Introduction
- TinyOS Applications
- Running TinyOS Applicaition with WSIM/WSNET/AVRORA
  - Provided Tutorials, Do at home
- Inside IoTs: in-network processing & routing

https://github.com/tinyos/tinyos-main/tree/master/apps/Blink

Blink Application

- README for Blink
- Author/Contact: tinyos-help@millennium.berkeley.edu

- Description:

- Blink is a simple application that blinks the 3 mote LEDs. It tests
- that the boot sequence and millisecond timers are working properly.
- The three LEDs blink at 1Hz, 2Hz, and 4Hz. Because each is driven by
- an independent timer, visual inspection can determine whether there are
- bugs in the timer system that are causing drift. Note that this
- method is different than RadioCountToLeds, which fires a single timer
- at a steady rate and uses the bottom three bits of a counter to display
- on the LEDs.

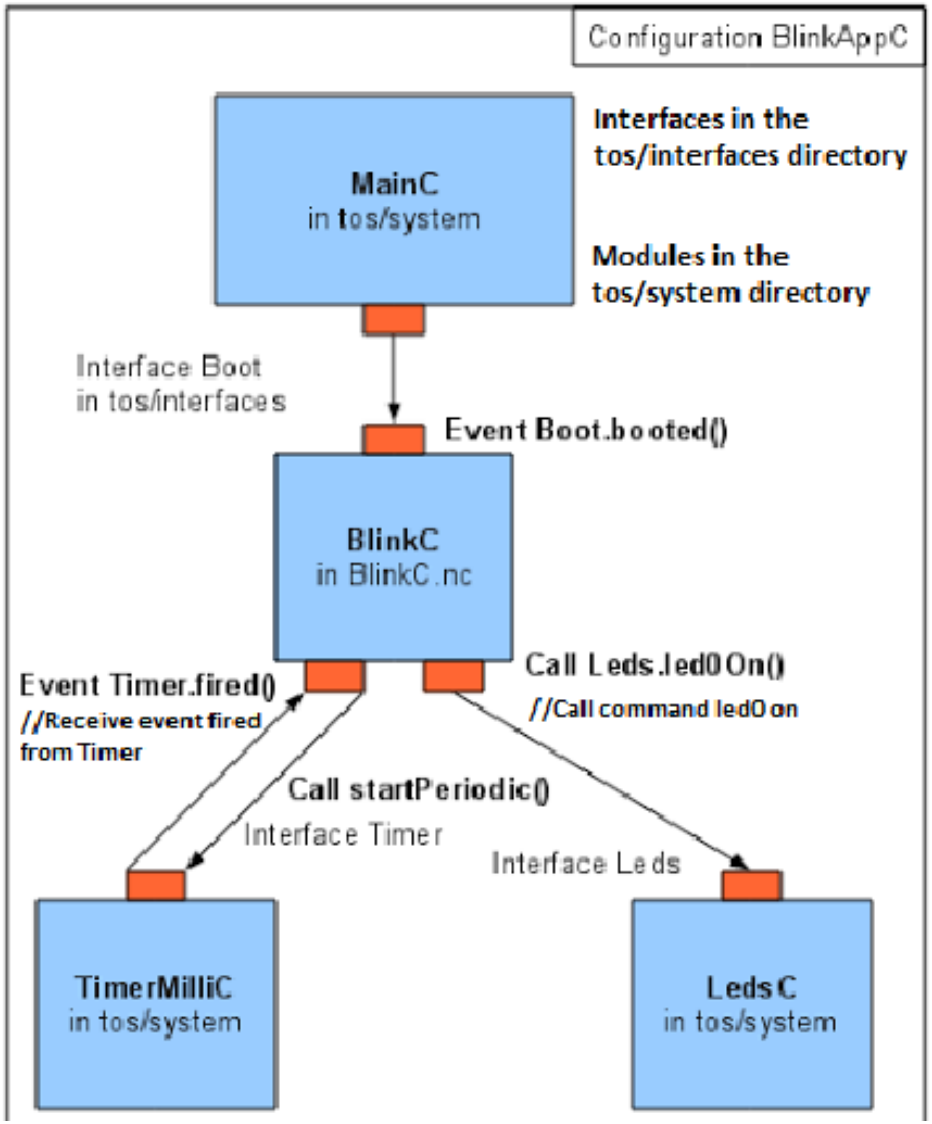
http://www.tinyos.net/dist-2.0.0/tinyos-2.0.0beta2/doc/html/tutorial/lesson1.html
https://github.com/tinyos/tinyos-main/tree/master/apps/Blink

The BlinkAppC.nc Configuration
apps/Blink/BlinkAppC.nc:

```
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;
}
```



The BlinkC.nc Module
apps/Blink/BlinkC.nc:

```
#include "Timer.h"

module BlinkC @safe()
{
  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();
  }
}
```

Other Applications

- https://github.com/tinyos/tinyos-main/tree/master/apps
- /opt/tinyos-2.1.2/apps

Outlines

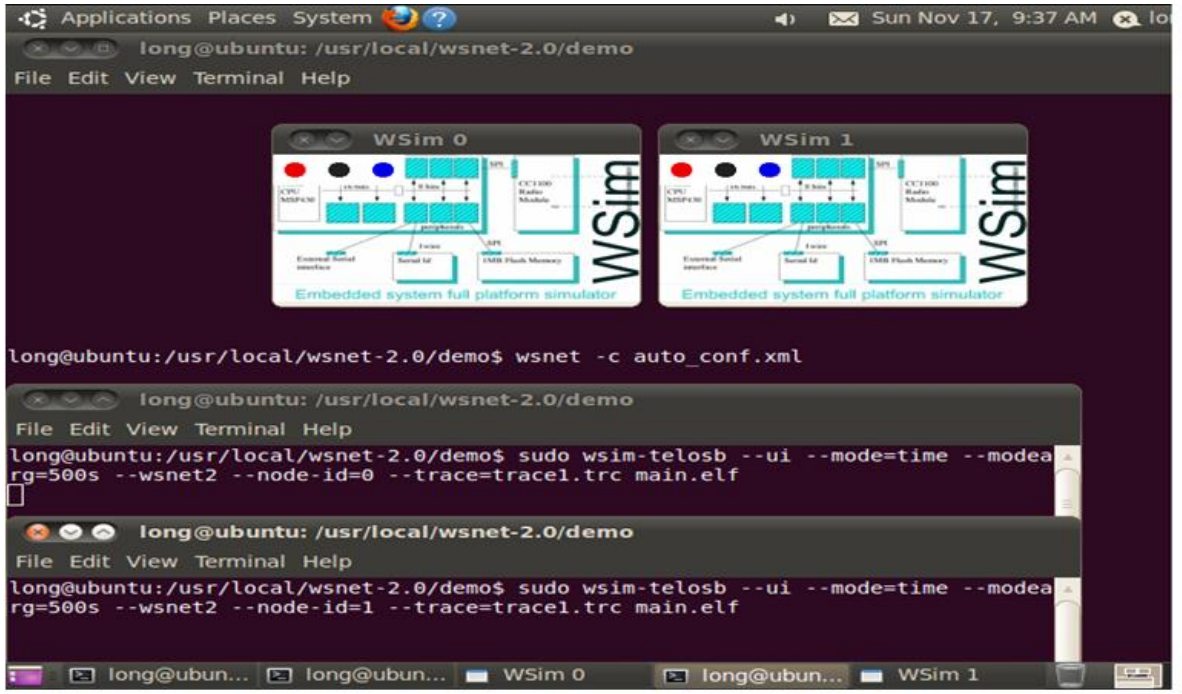
- Review on different open-source development software platforms
- TinyOS Introduction
- TinyOS Applications
- Running TinyOS Applicaition with WSIM/WSNET/AVRORA
  - Provided Tutorials, Do at home
- Inside IoTs: in-network processing & routing

Installation TinyOS/WSIM/WSNET [1/2]

- Linux OS
  - Ubuntu 14.04 [getting a copy of ISO image], or
  - http://www.ubuntu.com/download/desktop
- Other Oses, e.g., Windows/MacOS/…
  - Oracle VirtualBox: https://www.virtualbox.org/wiki/Downloads
  - vmware, https://my.vmware.com/web/vmware/downloads
  - Ubuntu
- TinyOS, http://www.tinyos.net
  - http://tinyos.stanford.edu/tinyos-wiki/index.php/Installing_TinyOS_2.1.1
  - Or steps-by-steps instructions from students

Installation TinyOS/WSIM/WSNET [2/2]

- WSIM, http://wsim.gforge.inria.fr/
- WSNET, http://wsnet.gforge.inria.fr/
- Compile the /apps/Blink application, e.g., “make telosb”
- Run the Blink application with WSIM/WSNET



Outlines

- Review on different open-source development software platforms
- TinyOS Introduction
- TinyOS Applications
- Running TinyOS Applicaition with WSIM/WSNET/AVRORA
  - Provided Tutorials, Do at home
- Inside IoTs: in-network processing & routing

Inside IoTs: in-network processing & routing

- In the next tutorials