TinyOS & RBS

JeongGil Ko
CS 450
Fall 2009
Basics of TinyOS
Why Different OS?

- Think of the characteristics of WSNs
  - Event driven
  - Limited Resources
  - Diversity in applications
  - Unattended long term operations
TinyOS

- TinyOS is...
  - A small sized event driven OS
    - Provides programming environment
    - Language (nesC) + services
  - Collection of drivers and components
Concepts of TinyOS

- COMPONENT-based
  - Module (behavior)
  - Configuration (wiring to other modules)
  - A module USER calls a COMMAND provided by a PROVIDER
  - User offers EVENT handlers to providers
Example: User - Provider

• Interaction with networking layer

• Application (USER) uses COMMANDs of the networking layer (PROVIDER)

• Applications (USER) implements EVENT handlers for networking layer (PROVIDER) to issue
Split-phase

• Separate operation request and completion event for long applications
• Completion events are interrupts
Example

• Radio
  • Start the Radio!
    • In the mean time, let’s do ‘x’!
    • Let’s do ‘y’ as well!
  • Radio has started!
• Timers are also an example
Hierarchy

- Commands go to lower layers
  - Send a packet!
- Events come up layers
  - Received a packet!
Tasks

• Break long operation in multiple pieces
• Making schedules to run in the future
• Good for non-time critical actions
• Not preempted by other task
Active Messages (AM)

- TinyOS’ way of multiplexing the radio
- Think of a port number in IP networks
AM Example
event void Boot.booted(){
    call SplitControl.start();
}

event void SplitControl.start(error_t err){
    call Timer.startOneShot(500);
    call Leds.led2On();
}

task void sendTask(){
    call AMSend.send(0xFFFF, &send_msg, sizeof(send_msg_payload));
}

event void Timer.fired(){
    post sendTask();
}
Folder + Makefile

Top level component!
Put the program on a MOTE!!!!!! or in SIM

make telosb [re]install[,]nodeId] [bsl,/device_port]

make micaz sim

Another useful command: motelist
TOSSIM

• Simulator for TinyOS
• Discrete event simulator
• Supports the MicaZ hardware
• Compile TinyOS sources directly
• Python / C++ used for simulation
• User defined link and environment conditions
from TOSSIM *import

t = Tossim([])
r = t.radio()

n0 = t.getNode(0)
n1 = t.getNode(1)
n0.bootAtTime(long(0.5 * t.ticksPerSecond()))
n1.bootAtTime(long(0.5 * t.ticksPerSecond()))

for j in range(100):
    node0.addNoiseTraceReading(-100)
    node1.addNoiseTraceReading(-100)
    node0.createNoiseModel()
    node1.createNoiseModel()

r.add(n0, n1, -20)
r.add(n1, n0, -20)

while (t.time() < (100 * t.ticksPerSecond())):
    t.runNextEvent()
RBS

• Reference-Broadcast Synchronization

• Presented at OSDI 2002
Goals of RBS

• Synchronize local clocks of receiver nodes
• Why time sync?
  • Clocks can start at different times
  • Clocks can drift
  • Deployments can be long
  • Data-Time relationship can be important
RBS procedures

- Reference Broadcast
- Recording Reference Message Reception
- Exchanging Reference Records
- Comparing Timestamps for synchronization
Pictorial View

RN -> Broadcast -> SN1, SN2
Pictorial View

RN

SN1
Record

SN2
Record
Pictorial View

Exchange time stamp recording information
Pictorial View

RN

SN1

SN2’s local clock should be...

SN2

SN1’s local clock should be...
Good Luck!