

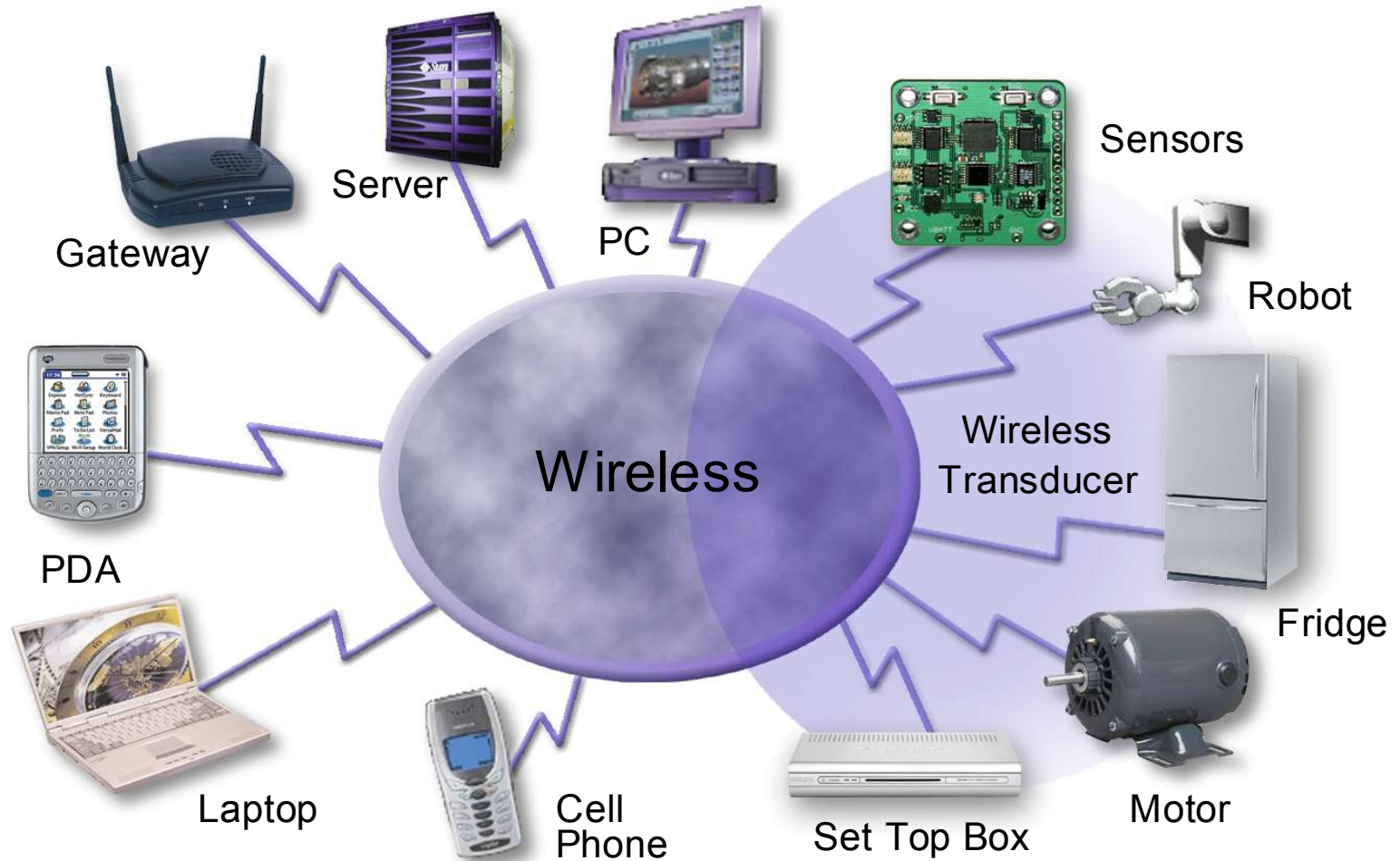


JAVA™ ON SENSORS

John S. Nolan, Cristina Cifuentes
Sun Labs



Wireless Networks



The State of the Art

- Ideas of “Smart Dust”
- Berkeley motes, TinyOS, IEEE 802.15.4
- Sun Labs Anteater project
 - > Research into impact and meaning of such systems to Sun Microsystems
 - > Major customer advantage seen as economics and flexibility
- Most of the work is aimed at infrastructure issues
 - > Size, power, and networking (mesh networking)

Applications: Chicken and Egg

- Not a lot of convincing applications out there based on Wireless Transducer Networks – why?
- Our conclusion: partly due to pain of developing applications using current technologies
 - > Low-level C like languages
 - > Unproductive development tools
 - > Too many low-level concerns in current systems
 - Most high-level software developers do not know how hardware works, or even have an appreciation any more
 - > Not accessible to majority of software developers

What Is Our Solution?

- Provide an opportunity for developers to create applications using Java to run on Wireless Transducer Devices
 - > Java is 4x more productive than C-like languages
 - > VM architecture allows good abstraction of low-level details
 - > VM architecture can protect vital areas of devices from accidental or purposeful corruption
 - > Take advantage of Java's dynamic capabilities for developer productivity

What Is Our Solution? (2)

- Provide a more powerful mid-level device which can be battery powered
 - > Give space to allow exploratory programming
 - Avoid premature optimization issues
 - > Provide a device which will allow more processing closer to the transducer to reduce network traffic
 - Network accounts for majority of power drain
 - Be smart about what to send – requires processing
 - > Enable over-the-air reprogramming
 - Greater developer productivity
 - Leverages dynamic nature of devices and Java

Demonstration

- Demonstration of programming and deploying
- Demonstration of hardware devices

How Can We Provide Java on Next Generation Sensor Devices?

- Squawk VM
 - > Small J2ME VM written mainly in Java
 - > Able to run on-the-metal, without an underlying OS
 - > Simple port to different platform
- Build new wireless sensor device using off-the-shelf hardware components

6 months later ...

The SunSpots System

- Hardware
 - > 32-bit ARM core
 - > Chipcon CC2420 based wireless platform
 - > SPI based peripherals
 - > Simple sensor board
- Software
 - > Squawk: Java VM
 - > Desktop build and deploy scripts
 - > Libraries for
 - Driving hardware: radio, sensor boards, ...
 - Basic 802.15.4 network functionality
 - > SpotWorld: graphical desktop interface

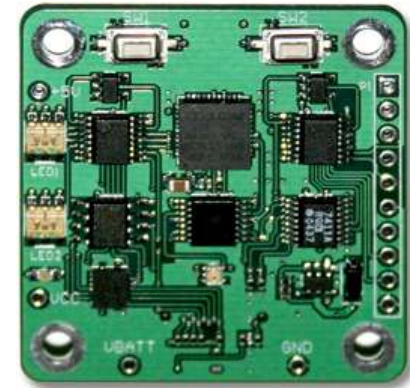
SunSpot Hardware

- ARM7 core
 - > 256K RAM/2M ROM
- CC2420 radio
 - > Strip antenna
- Single LED
- Double sided connector for stackable boards
- Can be powered from single 1.5V battery
 - > Requires 25-90mA depending on operation
- 35x25 mm in size
- Supporting testboard with USB connection to desktop



SunSpot Sensor Board

- All-singing sensor board
 - > 3D accelerometer
 - > 9 I/O Pins (PWM capable)
 - > Temperature sensor
 - > Light sensor
 - > IRDA serial connection
- Mainly for demonstration purposes
 - > All SPI driven peripherals
- Users can build own transducer boards
 - > Experimental board available



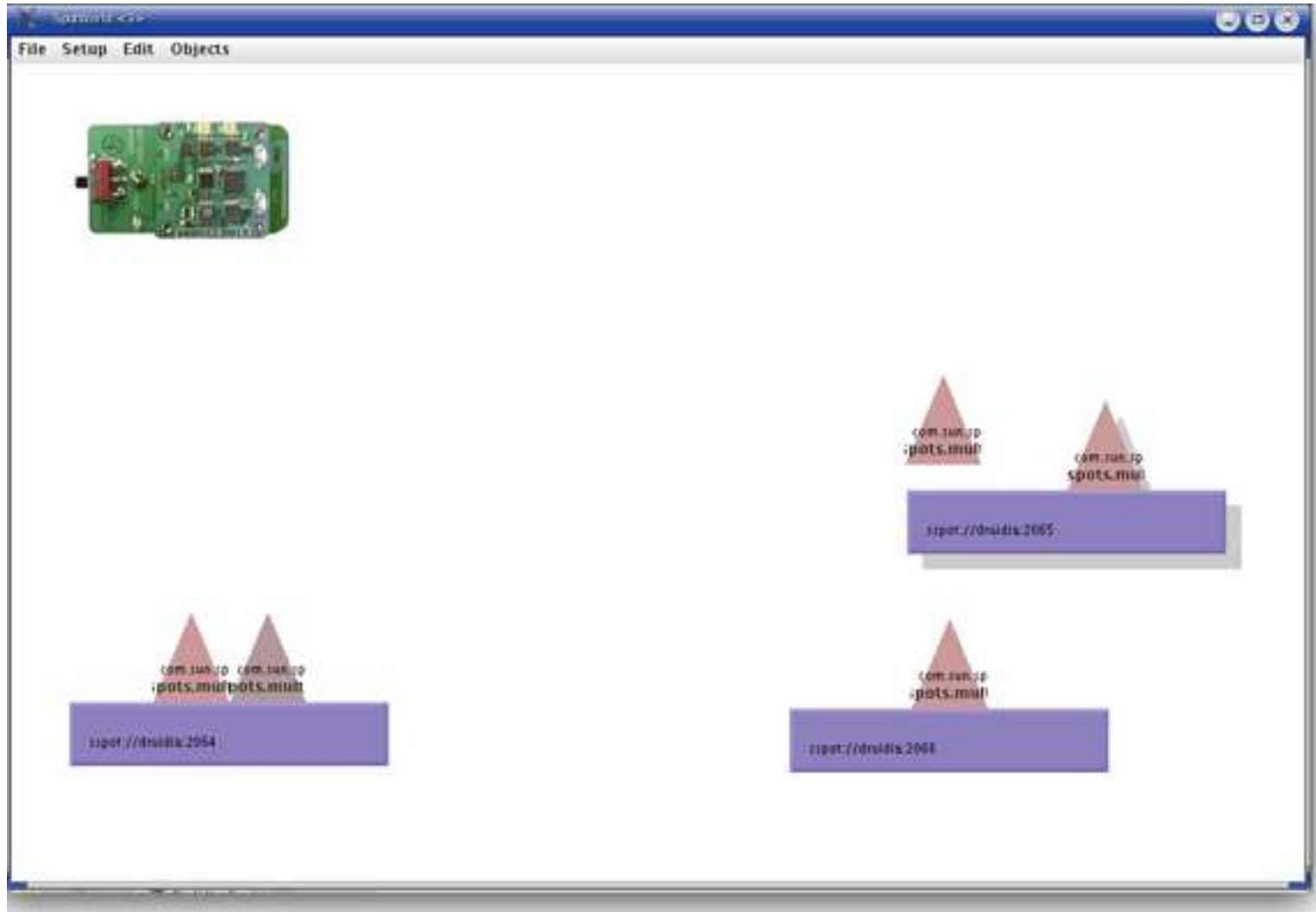
SunSpot Squawk VM

- Fully capable CLDC 1.0 Java VM
 - > GC, threads, etc.
 - > Extra features: isolates, suites, direct interrupts, etc.
 - > Tiny amount of C, mainly written in Java
 - > Currently 80K RAM for VM
 - Can reduce overhead by using some parts from flash memory
 - > Libraries 380K flash
 - Most of the Java components of the VM
 - Full CLDC 1.0

SunSpot Build and Deploy Scripts

- Full range of developer tools
 - > Use standard IDEs to create Java code
 - > Build and deploy scripted to make as simple as possible
 - Ant based
 - > Simple debugger available
 - Working on compliant JDWP debugger
- USB connection to testboard
 - > Testboard provide port for SunSpots
 - > Can program devices on testboard
 - Working on over-the-air programming of SunSpots

SunSpot SpotWorld



SunSpots: Enabling Developers To ...

- Build wireless transducer applications in Java
 - > Use simple sensor board for IO
 - > Use simple radio connections to communicate
- Integrate new hardware
 - > Utilize libraries to integrate own hardware devices
- Build new network layers
 - > Access to all levels of hardware via Java
 - > Can implement own protocols
 - > Can implement own MAC if desired

Future

- Collaborate with qualifying partners, July 2005
- Use within Sun Labs
 - > Gesture based interfaces, building instrumentation, self-organising systems, etc.
- Iterate hardware design
 - > Smaller chips, lower power, cheaper, etc.
- Iterate VM
 - > Smaller footprint, faster, smarter interrupts, power management, etc.
- Open schematics and VM to the community?

Conclusions

- Java on “wireless sensor networks” is here
 - > Small Java-based VM
 - Java runs on the bare metal, no underlying OS needed
 - > Better developer experience than the state-of-the-art
 - Standard Java development and debugging tools
 - Simple out-of-the-box experience (SpotWorld)
 - > Mid-level sensor device that can be battery powered
 - Enable exploratory programming
 - Enable more on device computation and reduce network traffic
 - Enable over-the-air programming

The Team

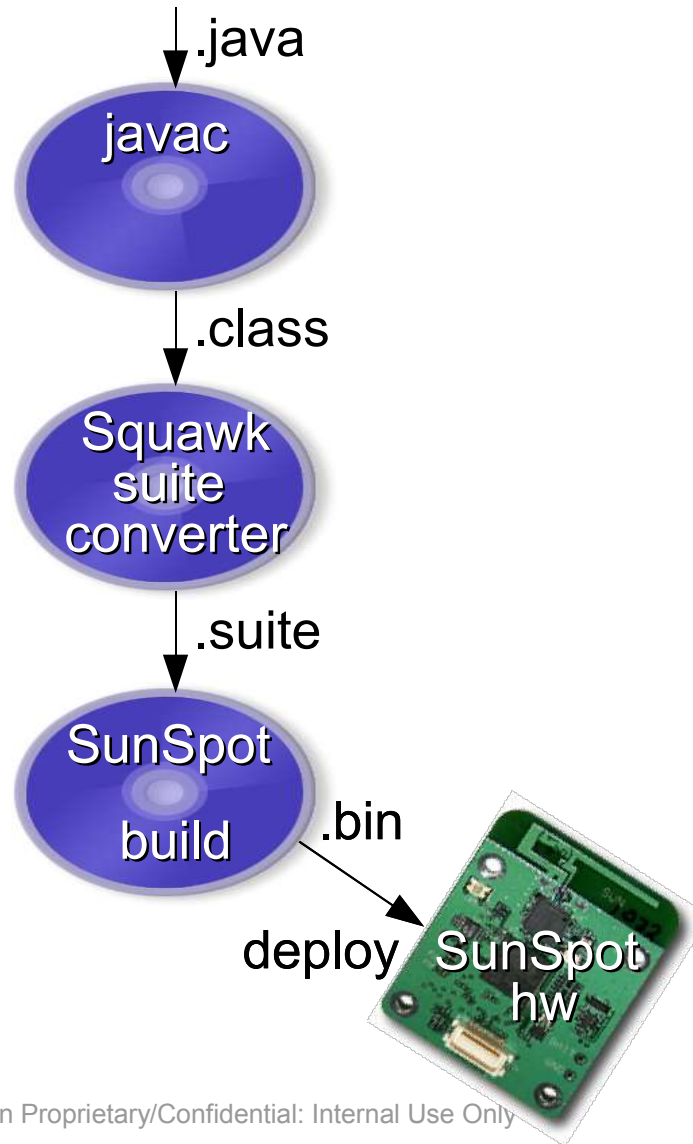


Questions

Appendices

- Following slides are additional material which can be used in support of questions, or purely just FYI

SunSpot Build and Deploy Process



SunSpot Software Libraries

- Standard J2ME Java libraries
 - > CLDC 1.0
- Hardware libraries
 - > SPI, AIC, TC, PIO drivers all in Java
 - > Sensor board hardware driven by Java (no C)
 - ADCs, GPIO, IRDA, etc.
- Radio libraries
 - > Drive Chipcon CC2420 hardware from Java (no C)

Software Libraries (2)

- Network libraries
 - > 802.15.4 MAC layer in Java (no C)
 - > Simple GCF implementations of connections
- Desktop libraries
 - > Create connections from standard J2SE VMs to wireless devices
 - > Utilize Spot in testboard as a gateway

Example: Application

```
//Open a stream over the radio
StreamConnection conn = (StreamConnection) Connector.open
                        ("radio://"+otherSpotAddress+":100");

DataOutputStream output = conn.openDataOutputStream();

//Read pin 4 of the ADC on the Sensor board (ADT7411 is the type of ADC)
RangeInput input = new ADT7411RangeInput(Sensorboard.getADC(),4);

//Loop and send the data
while(true) {
    try {
        output.writeInt(input.getValue());
        output.flush();
        Thread.yield();
    } catch (Exception e) {
        System.err.println("SENDER problem "+e);
    }
}
```

Example: Sensor

```
public synchronized static Accelerometer3D getAccelerometer() throws IOException {
    if (accelerometer == null) {
        //get the ADC inputs
        RangeInput xInput = new ADT7411RangeInput(getADC(), 4);
        RangeInput yInput = new ADT7411RangeInput(getADC(), 5);
        RangeInput zInput = new ADT7411RangeInput(getADC(), 6);

        //get the control pins
        SingleBitOutput selfTest = new MAX6966SingleBitOutput(getIOPort1(), 7);
        SingleBitOutput powerDown = new MAX6966SingleBitOutput(getIOPort1(), 8);
        SingleBitOutput fullScale = new MAX6966SingleBitOutput(getIOPort1(), 9);
        accelerometer = new LIS3L02AQAccelerometer
            (xInput, yInput, zInput, selfTest, powerDown, fullScale);
    }
    return accelerometer;
}
```

Experimental Results (April 15, 2005)

Benchmark	.class	.suite
Richards (Gibbons)	11,770	4,584
Richards (Deutsch)	19,655	6,788
DeltaBlue	27,520	9,724
Game of Life	7,390	3,396

Sampling (samples/sec)	
ARM PIO lines	11,760
Sensor board input lines	300-800

Radio range:	90 mts
--------------	--------

Benchmark	LOC	ms on ARM7 EB40 board
Richards (Gibbons)	410	5,277
Richards (Deutsch)	456	8,382
DeltaBlue	984	4,766
Game of Life	354	4,032



john.nolan@sun.com
cristina.cifuentes@sun.com