Squawk: a Java™ VM for Wireless Sensor and Actuator Devices

Cristina Cifuentes, Derek White, Eric Arseneau

Senior Staff Engineer, Staff Engineer, Staff Engineer
Sun Labs
http://research.sun.com/projects/squawk

TS-1598

Java™ Technology on Wireless Sensors

Java technology in the Java programming language; Java technology on the Java platform

Learn how we designed a Java VM for small wireless sensor devices and how you could program these devices in Java technology

Agenda

Wireless Sensor Networks

The Squawk Java Virtual Machine (Java VM)

The Sun™ Small Programmable Object Technology (SPOT) System

Summary

Agenda

Wireless Sensor Networks

The Squawk Java Virtual Machine (Java VM)

The Sun™ Small Programmable Object Technology (SPOT) System

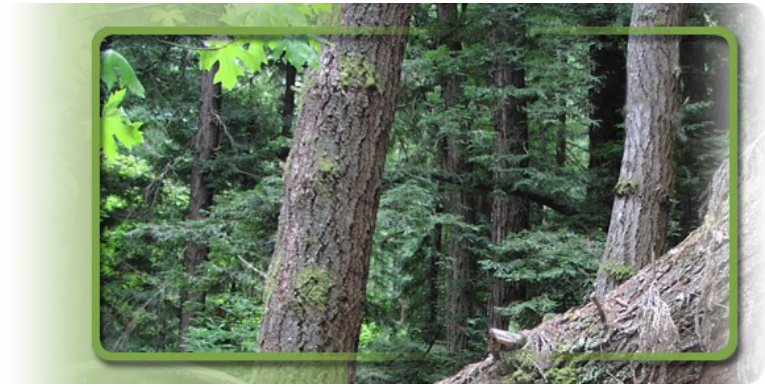
Summary

The State of the Art

- Ideas of “Smart Dust”
  - Berkeley, Kris Pister, 1998–2001
- Berkeley motes, TinyOS
  - Mica2, Mica2Dot: 8-bit microcontroller, 7.37/4.0 Mhz clock, 128 KB flash, 4 KB SRAM, CC1000, 512 KB external flash, 2 AA batteries/3V lithium cell battery
- Intel Mote
  - Zeevo module (ARM7 core, SRAM and flash memory, Bluetooth wireless), TinyOS
  - Mote 2: 32-bit Xscale PXA 271 CPU, large RAM and flash memories, runs Linux and the Java VM

Wireless Sensor Applications

- Structural monitoring
- Cane toad distribution
  - University of NSW, Australia
- Environmental monitoring
  - Redwoods
  - Endangered species



Applications: Chicken and Egg Problem

- Hard to develop applications using current technologies
- Low-level C-like languages
- Unproductive development tools
  - Hardly any debugging support
- 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

Future: Connected Wireless Networks

Future: More Powerful Platform

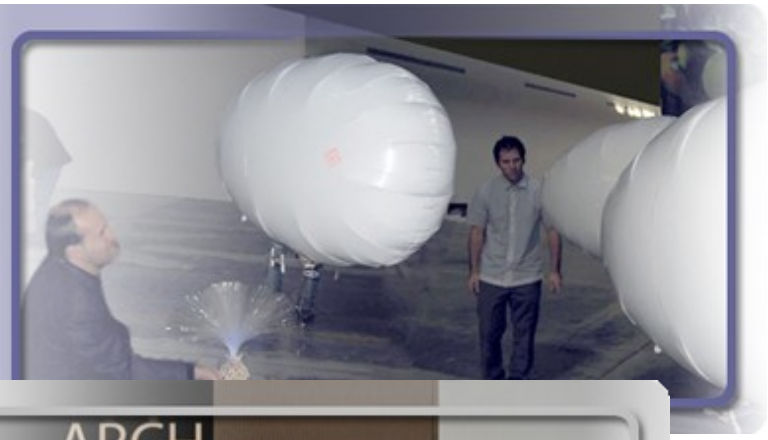
- Mid-level device that can be battery powered
- Enough memory to allow exploratory programming
- More processing closer to the device to reduce network traffic
- Enable over-the-air reprogramming

Future: More Developer-friendly Tools

- Bring Java technology to wireless sensor and actuator devices
  - Ease of development and security of Java
  - Take advantage of the Java programming language dynamic capabilities for developer productivity
- Use standard Java IDEs and debugger tools

Future: The New Ecology of Things

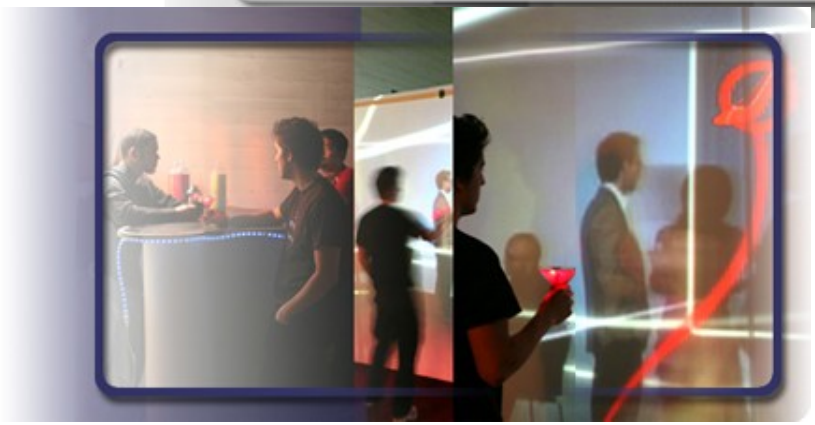
Sponsored by Sun Labs at the Art Center College of Design, USA



- Autonomous light air vehicles

- Retail-smart shoe

- Social interaction icebreakers

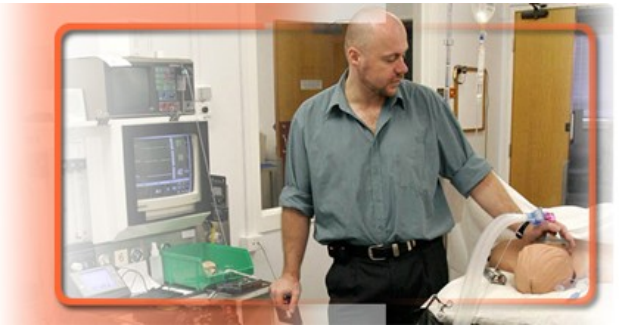


Source: http://people.artcenter.edu/~vanallen/ecology/

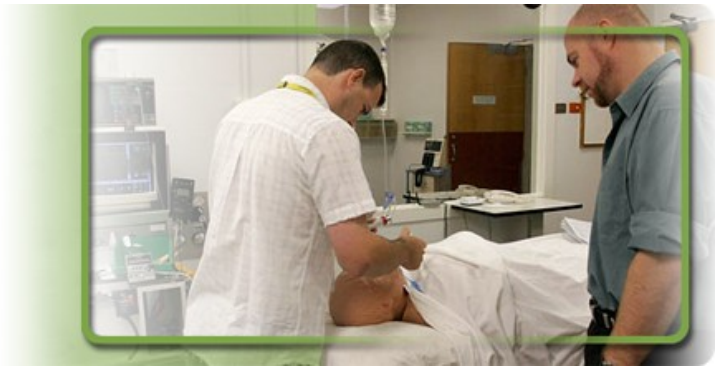
Future: Vibrotactile Alarm System

Patient monitoring

The University of Queensland, Australia



- Breathing mask on patient
- Intubation of the patient
  - Potential reactions to drugs, gases, etc.
  - BUT... monitor out of sight, noise can mask sounds
- Vibrotactile actuators wirelessly alert anesthetist



Source: J. Ng et al, Anesthesia and Analgesia, vol. 101, 2005.

DEMO

Sensor application

Agenda

Wireless Sensor Networks

The Squawk Java Virtual Machine

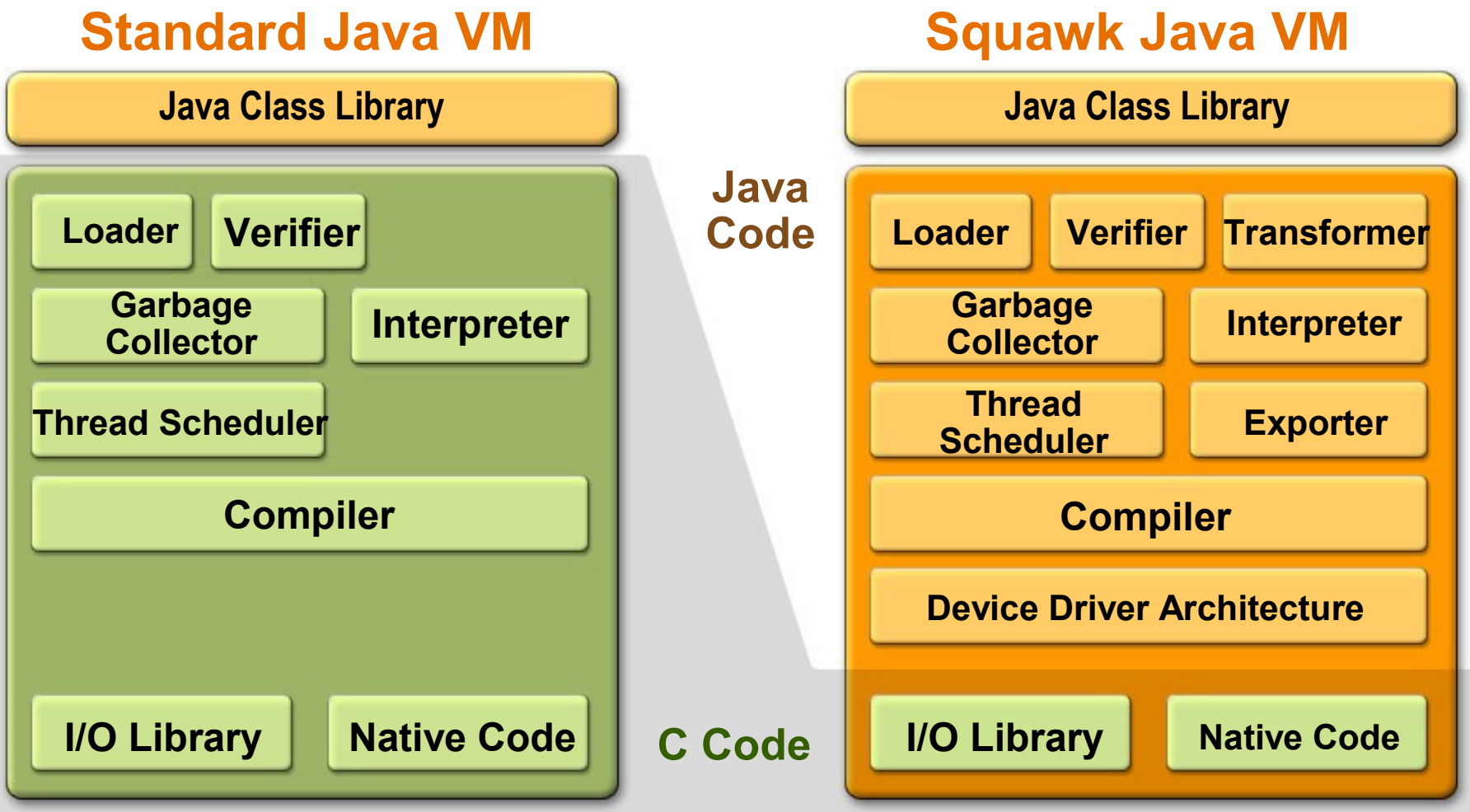
The Sun Small Programmable Object Technology (SPOT) System

Summary

The Squawk Java VM

- Java VM mainly written in the Java programming language
  - Interpreter written in C
  - Garbage collector translated from the Java to the C programming language
- Java ME CLDC 1.1
- Extra features
  - Runs on the bare ARM without an underlying OS
  - Interrupts and device drivers written in the Java programming language
  - Supports isolate application model

Standard Java VM vs. Squawk Java VM



Salient Features of Squawk for Wireless Sensor Devices

- Designed for memory constrained devices
- Runs on the bare metal on ARM
- Isolate application model
- Ease of development

Salient Features of Squawk for Wireless Sensor Devices

- Designed for memory constrained devices
- Runs on the bare metal on ARM
- Isolate application model
- Ease of development

Squawk Bytecodes: Designed for Memory Constrained Devices

Squawk Bytecode Property	Benefit
Commonly used bytecodes are 2 bytes instead of 3 bytes	More compact
References to fields and methods resolve into physical offsets	More efficient for interpretation Enables in-place execution
Local variables are typed	More efficient for compilation
One OOP map per method, nothing on the operand stack at GC points	Simplifies garbage collection Eliminates need for static interpretation to decipher activation frames

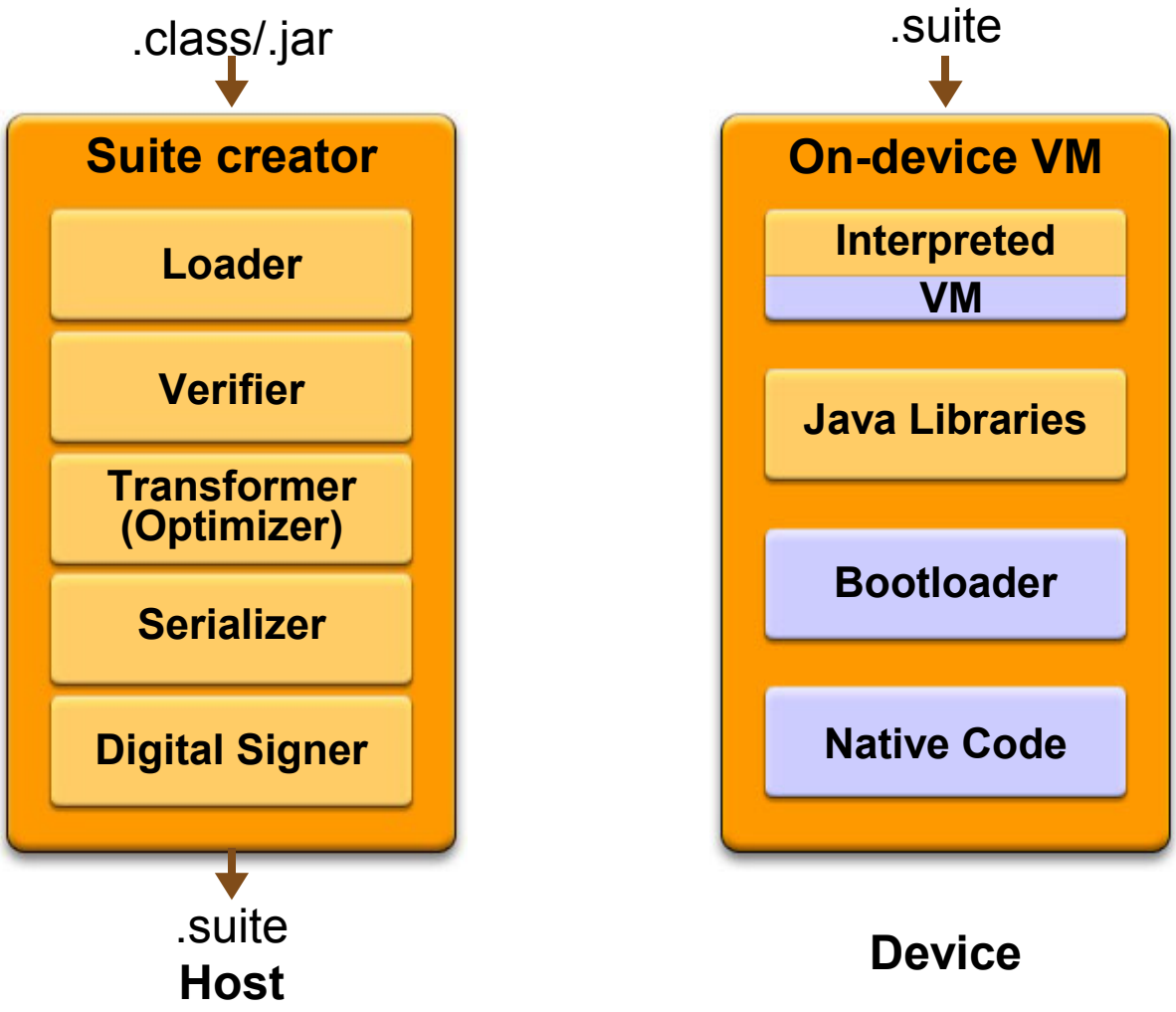
Suite Files

- Preprocessed set of classfiles
- Memory dump of classes, methods, literals and pre-computed objects
- Internally fully linked
  - Everything is an object which points directly to its parts
  - Can be referenced and used directly by the VM
- Execute in-place
- Digitally signed to authenticate content

Class, JAR Files vs. Suite Files

Benchmark	Resources	Class	JAR	Suite	Suite/(Class+Res)	Suite/JAR
Richards (gibbons)	0	10,975	7,968	4,072	0.37	0.51
Richards (gibbons_final)	0	10,981	7,973	4,080	0.37	0.51
Richards (gibbons_no_switch)	0	10,865	7,972	4,156	0.38	0.52
Richards (deutsch_no_acc)	0	16,560	11,637	6,044	0.36	0.52
Richards (deutsch_acc_virtual)	0	21,442	13,180	8,040	0.37	0.61
Richards (deutsch_acc_final)	0	21,440	13,191	8,040	0.37	0.61
Richards (deutsch_acc_interface)	0	22,632	14,131	8,040	0.39	0.63
DeltaBlue	0	27,584	16,478	9,212	0.33	0.56
Game of Life	0	8,467	5,444	3,472	0.41	0.64
Math	0	2,224	2,122	1,264	0.57	0.60
Subtotal	0	153,170	100,096	56,420	0.37	0.56
Chess	58,878	133,435	33,780	33,780	0.25	0.57
Crypto	9,954	89,954	60,690	55,232	0.55	0.91
KXML	19,109	111,346	66,318	57,732	0.44	0.87
Parallel	38,731	99,747	49,848	49,848	0.50	1.29
PNG	15,472	49,401	46,025	33,404	0.51	0.73
Subtotal	142,144	483,883	256,661	229,996	0.37	0.90

Squawk's Split VM Architecture



Salient Features of Squawk for Wireless Sensor Devices

- Designed for memory constrained devices
- Runs on the bare metal on ARM
- Isolate application model
- Ease of development

Runs on the Bare Metal

- Requires no OS on-device
- Minimal glue to
  - Capture interrupts and notify VM
  - Access to low-level hardware
- Device drivers written in the Java programming language
- Interrupt servicing occurs in the Java code

Device Driver and Interrupt Handling

- Device driver written in the Java programming language, with peek and poke interface to device's memory
- Device driver enables the device's interrupt
- Device driver thread blocks waiting for the VM to signal an event
- When an interrupt occurs, an assembler interrupt handler sets a bit in an interrupt status word and disables the interrupt
- At each VM reschedule point, the status word is checked, the event signaled, the scheduler resumes the device driver thread, which handles the interrupt and re-enables it

Interrupt Latency

- Defined as the time from the assembler interrupt handler running until the device driver thread is scheduled
- Optimal case
- VM is idle → no penalty
- Average case
  - VM is executing bytecodes in another thread → VM reschedules after a certain number of back branches
  - 0.15 msec latency
- Worst case
  - VM is executing a GC → VM reschedules after the GC completes; in practice, ← 13 msec for 100 KB heap

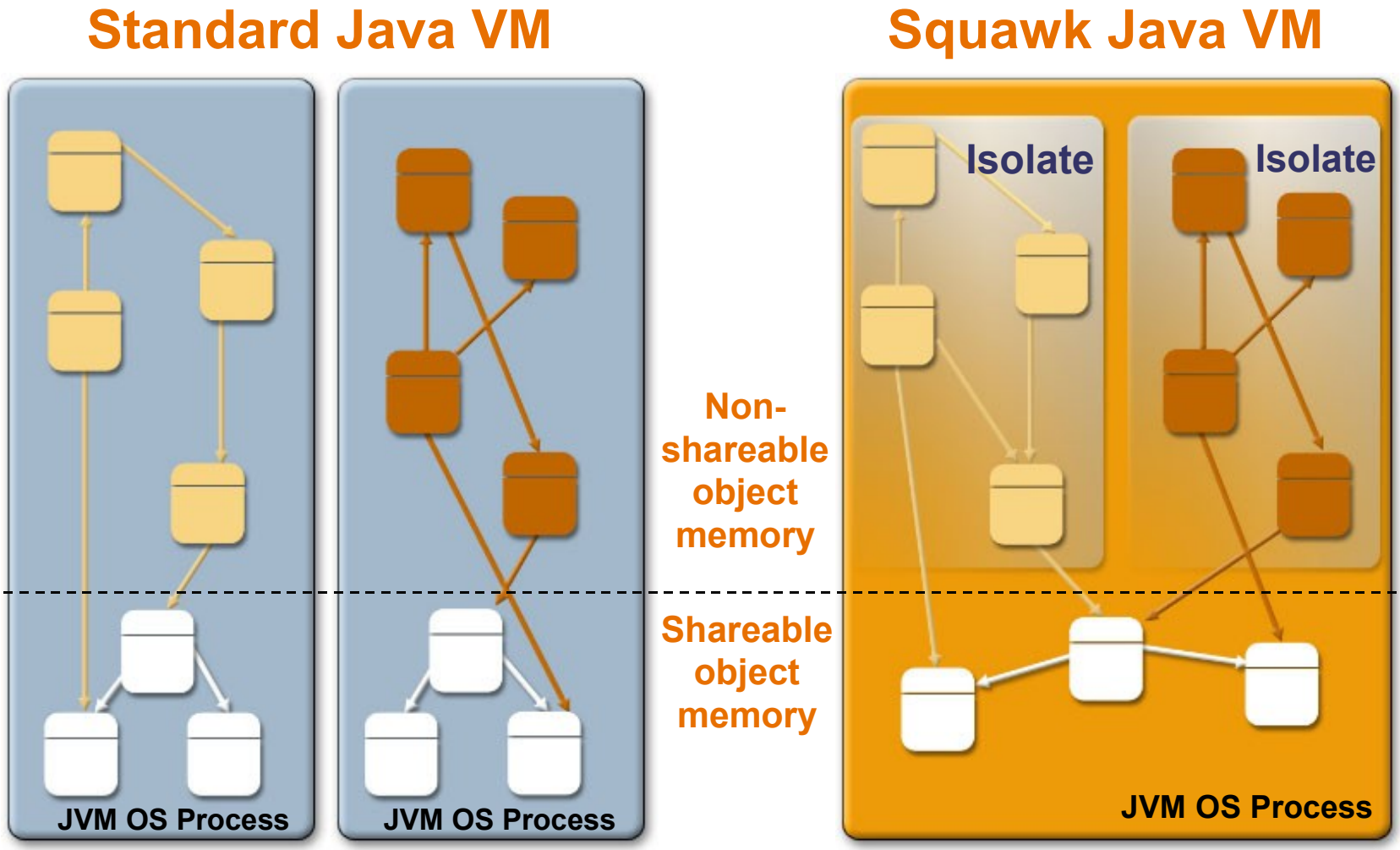
Salient Features of Squawk for Wireless Sensor Devices

- Designed for memory constrained devices
- Runs on the bare metal on ARM
- Isolate application model
- Ease of development

Isolate Application Model

- Sub/super-set of JSR 121: Application Isolation API Specification
- Application state is an object
- Every isolate has its own state for all static variables
- Allows for running multiple applications in one VM
- Inter-isolate communication
  - Provides lower-level asynchronous message delivery
- Can migrate from one device to another

Multiple Isolates (Applications) on the One Java VM



Isolate Migration

- State of a running application is migrated to another device
  - Continues running on target device where it left off
- Target device must have suite of the application
- Constraints on external state
  - There must be none, or
  - I/O sub-system must be homogeneous at both ends (Sun SPOT Squawk), or
  - I/O sub-system must be serializable (desktop Squawk)

Uses of Isolates and Isolate Migration

- Uses of isolates
  - Represent applications as objects
  - Support hardware resource sharing
  - Field hardware replacement
- Uses of isolate migration
  - Load balancing and fault tolerance
  - Local debugging of remote application
  - Seamless client-server programming model

Salient Features of Squawk for Wireless Sensor Devices

- Designed for memory constrained devices
- Runs on the bare metal on ARM
- Isolate application model
- Ease of development

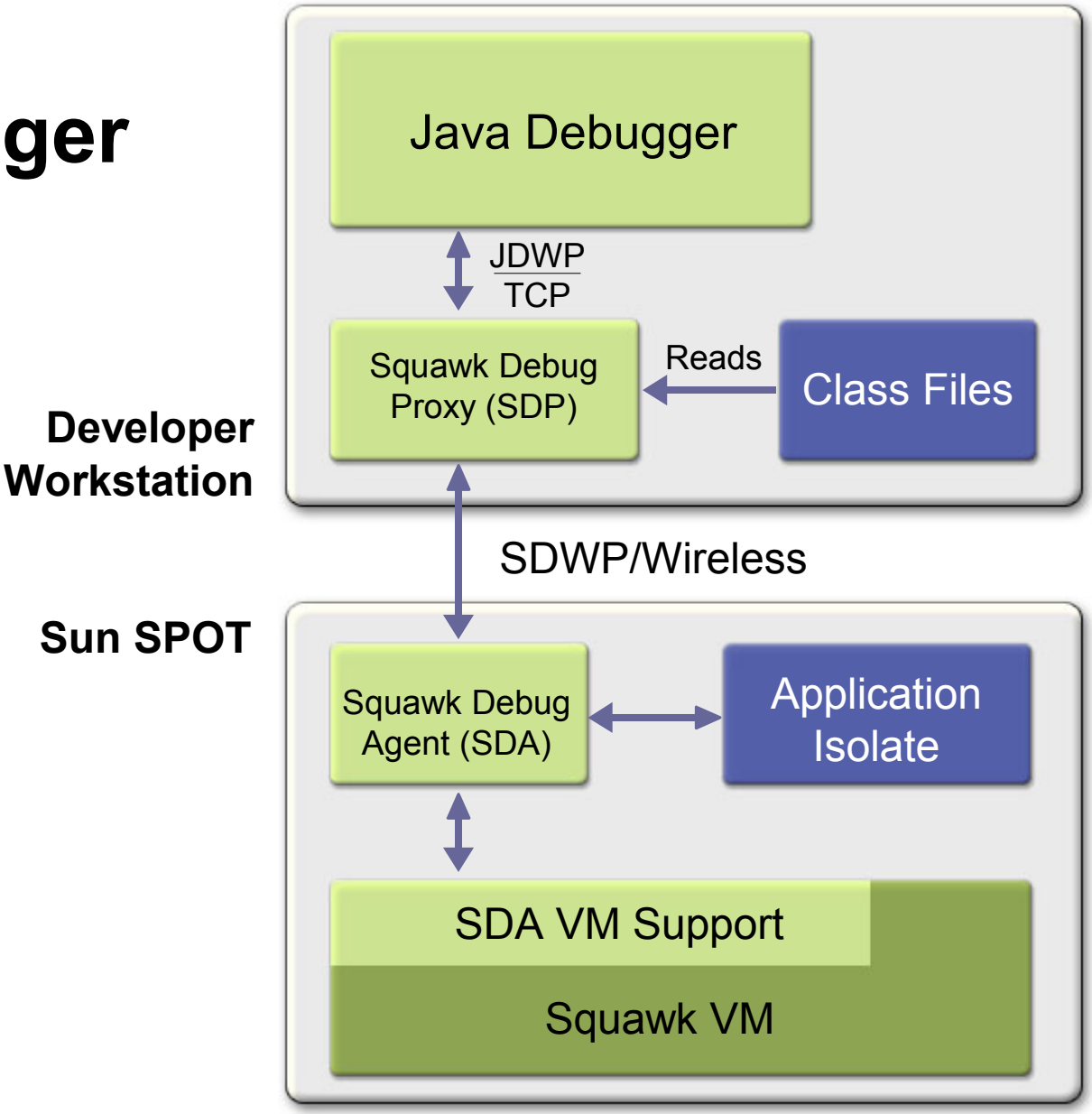
Ease of Development

- Java technology
- Can run on desktop as well as on-device
- Runs with standard IDEs or command-line
- Supports standard Java debuggers
  - Stepping, breakpoint
- On-device debugging

Debugger

- Use standard Java DWP debuggers
- Debug via USB or over-the-air (OTA) on-device
- Challenges
  - Not much memory on-device
  - Slow communication link to debugger client
  - Bytecode patching too slow (stored in flash)

Debugger


Java Debugger
JDWP
TCP
Squawk Debug Proxy (SDP)
Reads
Class Files
Developer Workstation
SDWP/Wireless
Sun SPOT
Squawk Debug Agent (SDA)
Application Isolate
SDA VM Support
Squawk VM

Agenda

Wireless Sensor Networks

The Squawk Java Virtual Machine

The Sun Small Programmable Object Technology (SPOT) System

Summary

Sun SPOT Hardware

- 180 Mhz 32-bit ARM920T core
  - 512K RAM/4M ROM
- ChipCon 2420 radio
  - 2.4 GHz IEEE 802.15.4
- USB interface
- 3.7V rechargeable 750 mAh prismatic lithium ion battery
- 40 uA deep sleep mode, 40 mA to 100+ mA
- 64 mm x 38 mm
- Double sided connector for stackable boards

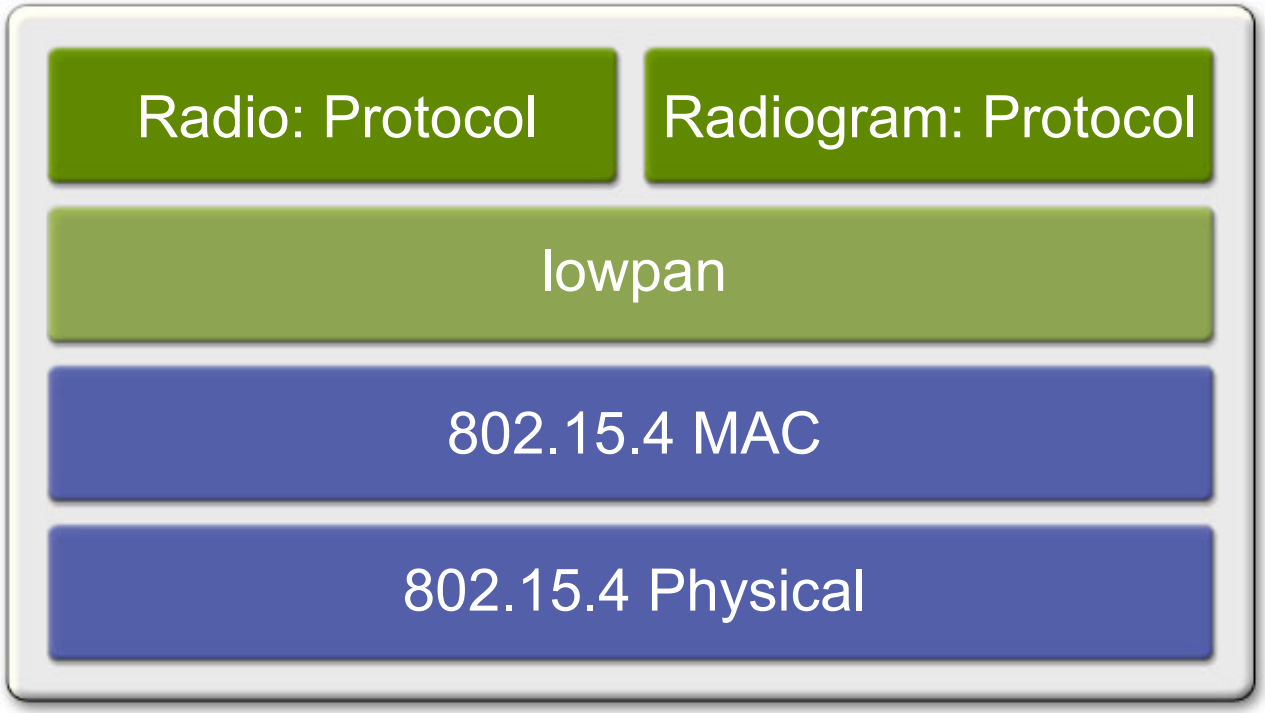
Demo Sensor Board

- 8 tri-color LEDs
- 3D accelerometer
- 5 general purpose I/O pins
- 4 hi current output pins
- 1 A/D converter
- Temperature sensor
- Light sensor

The Sun SPOT SDK—Libraries

- Squawk Java VM: desktop and Sun SPOT
- Libraries
  - Java ME CLDC 1.1 libraries
  - Hardware libraries
    - SPI, AIC, TC, PIO drivers all written in the Java programming language
    - Demo sensor board library
  - Radio libraries
  - Network libraries
    - 802.15.4 MAC layer written in the Java programming language, uses GCF
  - Desktop libraries

The Sun SPOT Radio Stack

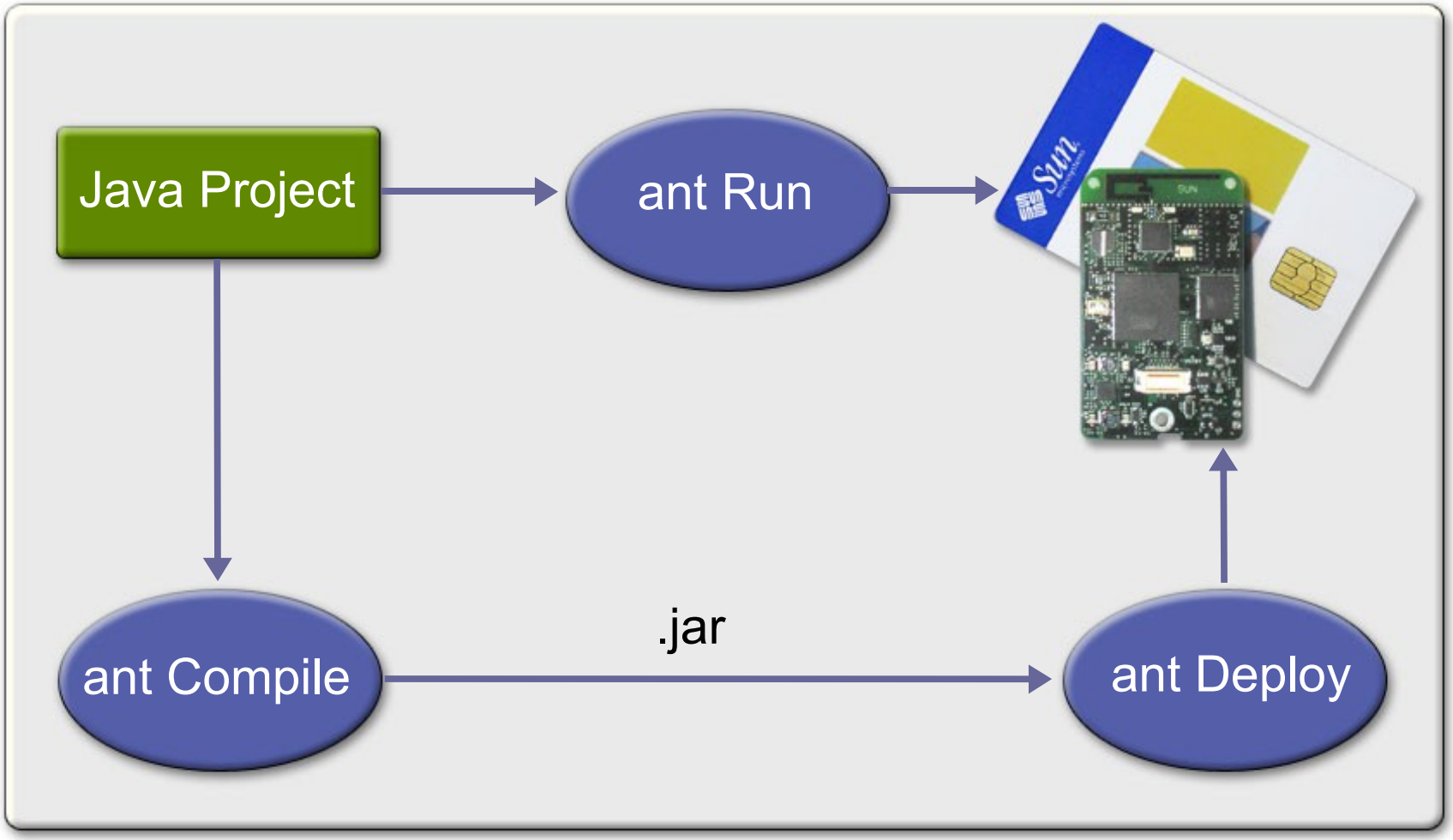


IEEE 805.15.4, 250 kbps OTA

The Sun SPOT SDK—Tools

- DebugClient
- ant tasks
- IDE integrations
- Sample NetBeans™ based projects
- SPOTWorld

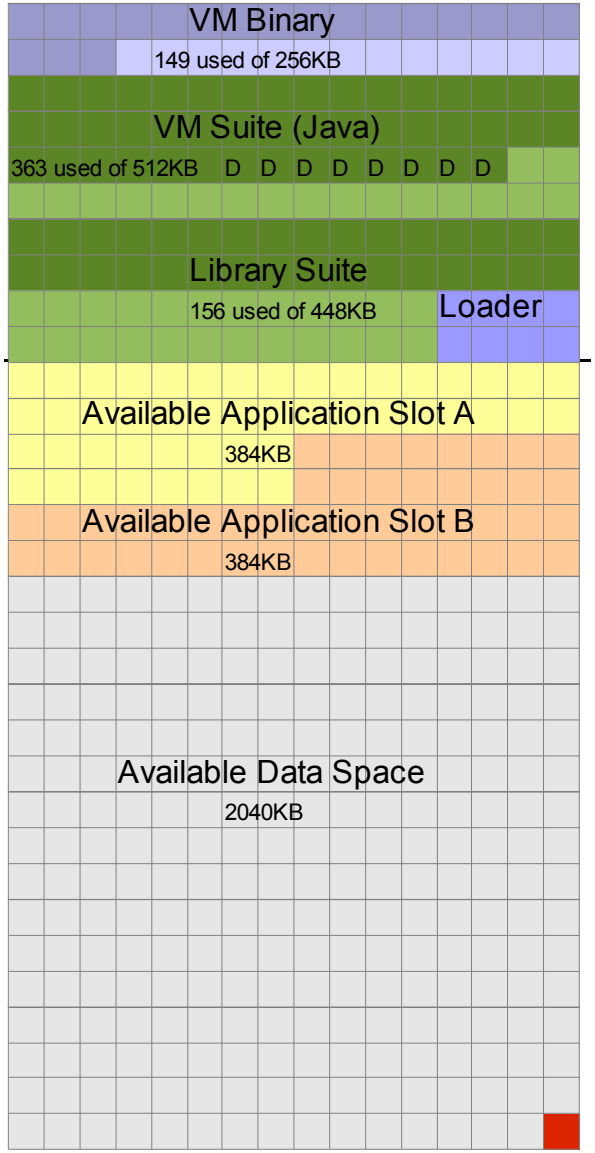
Build and Deploy Process



DEMO

Compile, Deploy, and Run

Squawk on the Sun SPOT: Flash Memory



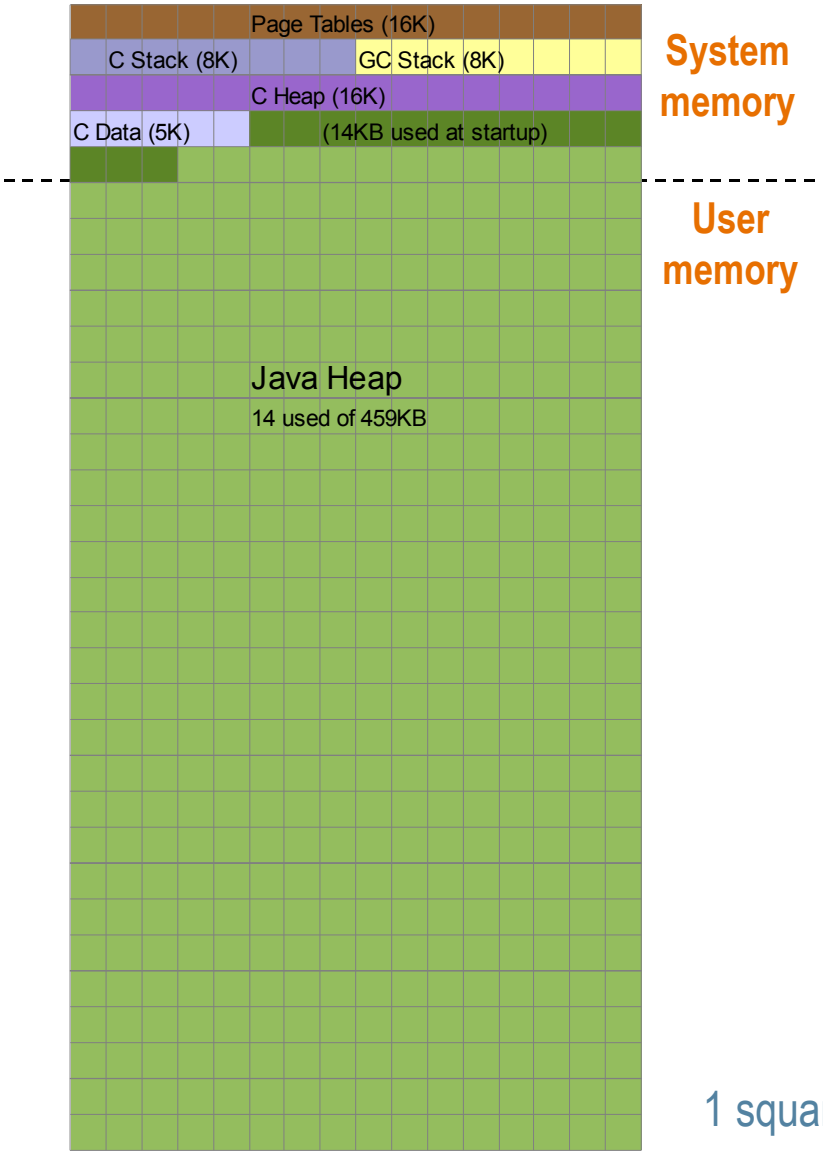
System memory

User memory

- 4 MB flash
  - Very low power
  - 1 million cycles/sector endurance
- 1/3 reserved for System
  - Not all in use
- 2/3 reserved for applications and data

1 square = 8KB

subject to change



Squawk on the Sun SPOT: RAM

- 512 KB pSRAM
  - Active current ≈ low mAs
  - Inactive current ≈ low µAs
- >80% available for application objects

1 square = 1KB

subject to change

Static Footprint of Interpreted Java VMs on the ARM

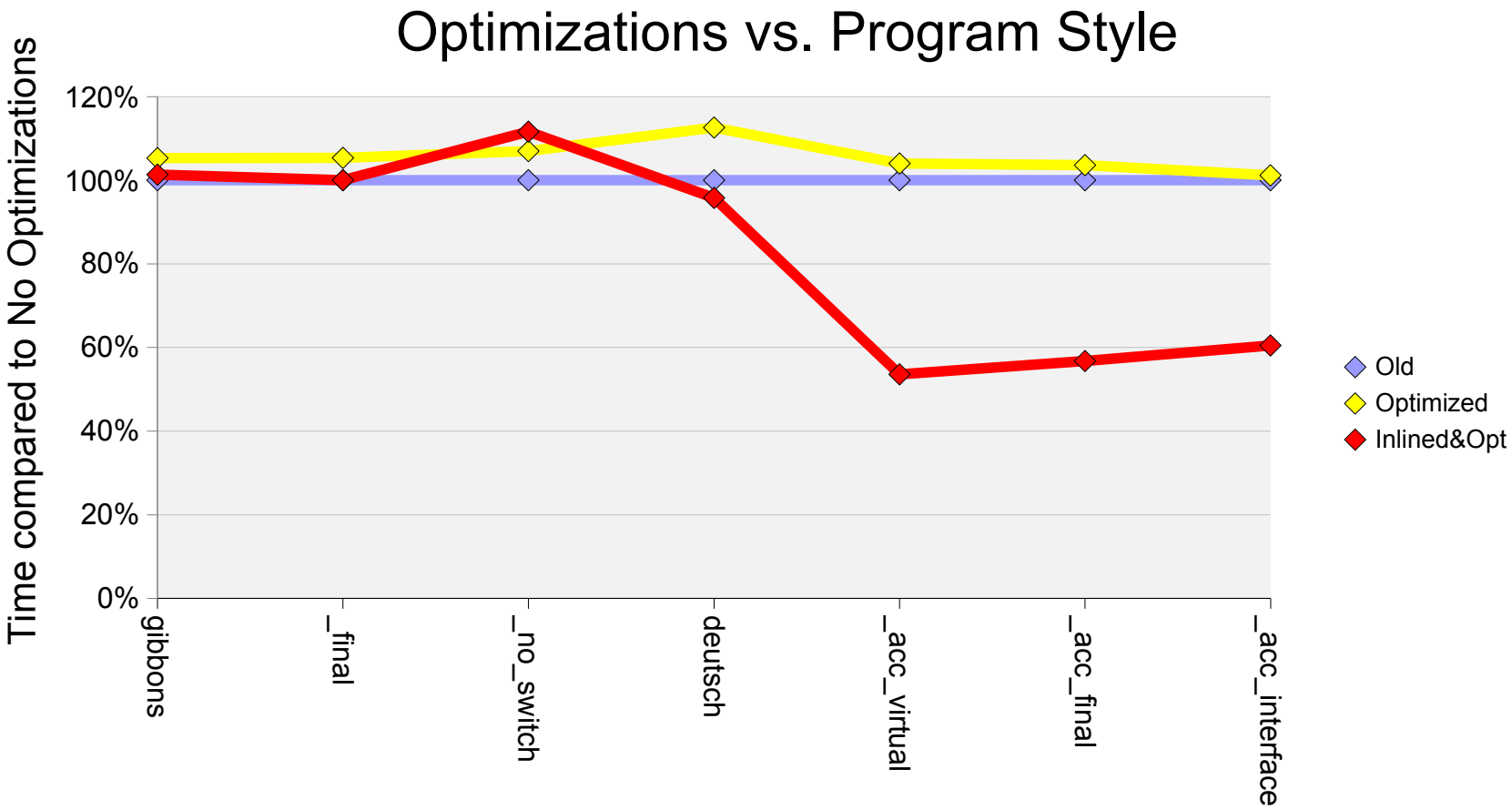
VM	Target	Debugging Support	VM	CLDC Libs	Sun SPOT Libs
Squawk 1.0	ARM7tdmi	No	80 KB	270 KB	
Squawk 1.1	ARM920T	Yes	149 KB	363 KB	156 KB
KVM 1.1	ARMv4l	No	131 KB	504 KB	n/a
KVM_d 1.1	ARMv4l	Yes	198 KB	504 KB	n/a

Runtime Performance of Interpreted Java VMs Running on an ARM

Benchmark	Squawk (ARM920T 180 Mhz) ms	KVM (ARMv4l 200 Mhz) ms
Richards (gibbons)	980	980
Richards (gibbons_final)	969	948
Richards (gibbons_no_switch)	1,067	1,262
Richards (deutsch_no_acc)	1,441	2,118
Richards (deutsch_acc_virtual)	2,557	6,002
Richards (deutsch_acc_final)	2,552	3,119
Richards (deutsch_acc_interface)	2,842	4,555
DeltaBlue	641	470
Game of Life	3,936	5,848
Math int	5,195	4,077
Math long	24,107	12,813

VM	Mhz	Heap	Chess	Crypto	KXML	Parallel	PNG	GrinderMark
Squawk 1.1	180	460 KB	339	714	558	774	727	597.26
KVM 1.0	60	1 MB	288	269	399	272	244	290.01

Bytecode Optimizations Performance Results



Agenda

Wireless Sensor Networks

The Squawk Java Virtual Machine

The Sun Small Programmable Object Technology (SPOT) System

Summary

Summary

- Java technology on “wireless sensor networks” is here
  - Better developer experience than the state-of-the-art
- Squawk: small Java-based VM
- Sun SPOT: 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

Future

- Collaborate with qualifying partners
- Use within Sun Labs
  - Gesture based interfaces, building instrumentation, self-organizing 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

The Teams

- Squawk
  - Nik Shaylor (alumnus)
  - Bill Bush (alumnus)
  - Doug Simon (alumnus)
  - Cristina Cifuentes
  - Derek White
  - Eric Arseneau
- Squawk ARM Support
  - John Daniels
  - Dave Cleal
  - Duncan Pierce
  - Rachel Davies (alumnus)
  - John Wilcox
  - Ivan Moore
- Sun SPOT Hardware
  - Bob Alkire
  - John Nolan (alumnus)
  - Del Peck
- Sun SPOT Software
  - Randy Smith, Bernard Horan (alumnus), David Simmons, Samita Chakrabarti (alumnus), Vipul Gupta, Rob Tow, Arshan Poursohi, Steve Uhler, Pete St.Pierre, Ron Goldman
- Managers
  - Dan Ingalls (Squawk)
  - Roger Meike (Sun SPOT)

For More Information

- JavaOne℠ Conference 2006
  - BOF-0289 “The Sun™ small programmable object technology (Sun SPOT): JavaTM technology-based wireless sensor networks
    - Tuesday May 16th, Argent Olympic, 8:30 pm
  - LAB-7160: “Simplified development of Java sensor and actuator applications using Java™ technology”
    - Wednesday May 17th, Hall E 130/131/132, 9:45–11:15 am
    - Wednesday May 17th, Hall E 130/131/132, 9:50–11:20 pm
  - Pod “Project Sun SPOT: Java technology-based platform for ubiquitous computing”
    - Tuesday May 16th—Thursday May 18th

For More Information

- Squawk
  - http://research.sun.com/projects/squawk
- Sun SPOT
  - http://www.sunspotworld.com
- Papers
  - “Java™ on the Bare Metal of Wireless Sensor Devices—The Squawk Java Virtual Machine”, VEE, June 2006
  - “The Squawk Virtual Machine: Java™ on the Bare Metal”, Extended Abstract, OOPSLA, Oct 2005

Q&A

Cristina Cifuentes
Derek White
Eric Arseneau

Squawk: A JVM for Wireless Sensor and Actuator Devices

Cristina Cifuentes, Derek White, Eric Arseneau

Senior Staff Engineer, Staff Engineer, Staff Engineer
Sun Labs
http://research.sun.com/projects/squawk

TS-1598