LINUX AND REAL-TIME JAVA FOR IOT

Randy Rorden
Software Development Director, PTC Inc.

April 6, 2016
EMBEDDED SYSTEMS EVOLUTION

Hardware:
- ‘80s: CPU, Mem, Peripherals
  Custom board
  $\$ $ Manufacturing
- ‘90s: Chipsets
  Std. Interfaces
  Std. Form Factors
  Commercial Boards
- ‘00s: Multicore SoCs
  On-chip Interfaces
  Stacked DRAM, Flash
  MEMS Sensors
  Open Source Boards

Software:
- ‘80s: RYO kernel/exec
  Custom drivers
  Assembly Code
- ‘90s: Commercial RTOS
  Customized BSP
  C, Assembly
- ‘00s: Embedded Linux
  Open Source Drivers, Libs
  C++, Java, Python, Javascript...
TECHNOLOGY CONVERGENCE TO THE IOT

**Embedded Systems**
SoCs, MEMS, Open Source h/w, C++, Java, Embedded Linux

**Cloud Computing**
Virtualization, Containers, IaaS/PaaS/SaaS, SDN, NFV, OpenStack

**Big Data, Deep Learning**
Hadoop, MongoDB, Spark, TensorFlow, Torch, Leaf, Caffe

**Wireless Connectivity**
LTE, WiFi, Bluetooth LE, ZigBee, 6LoWPAN, LoRa, Sigfox, Weightless
Smart, Connected Products:
- Transportation
- Energy
- Health Care
- Agriculture
- Retail
- Industrial Control
- Home/Building Automation
- Smart Cities
...and many more...
An IoT Gateway:
- Acts as a communications access point between networks
- Provides intelligence and storage at the Edge of the network
- Aggregates, filters, validates data from low-level sensors, forwarding results to the Cloud
- Controls actuators on behalf of Cloud commands and in response to local events
**IoT Gateways can:**

– Connect legacy devices to the Internet of Things
– Assure secure communications with the Cloud
– Provide separation and isolation of end-nodes
– Be easily updated, adapted, configured over the network
– Relieve processing and storage demands from end-nodes
– Reduce overall costs while scaling up
– Help diagnose and troubleshoot device failures
– Offer local autonomy and intelligence
– Respond quickly to time-critical events
Developers can:

Build it from scratch...

Use a high-level language for faster development...

Or leverage frameworks and services...

**Application Code**
- (C/C++)
- (Java)

**OSGi**
- web
- DB
- config
- device
- cloud
- update

**Java SE**

**Linux**

**Hardware**
Java Brings these Great Features:
✓ High-level object-oriented programming
✓ Automatic garbage collection
✓ Robust Security and Exception handling
✓ Built-in threading model
✓ Dynamic loading, unloading, re-loading of code without reboot
✓ Platform independence
✓ Excellent support for networking, concurrency, database, collections, plus thousands of 3rd party libraries
Open Systems Gateway initiative (OSGi):

- A framework for developing, deploying, and managing software components written in Java.
- Components called bundles hide internals and communicate through well-defined services.
- Bundles register, get, and listen to services in a registry. They can adapt their function based on available services.
- Dependencies and versioning among bundles are managed automatically.
- Bundles can be installed, started, stopped, updated, and uninstalled without bringing down the system.
Full-featured OSGi framework with many useful services.

Contributed by Eurotech. Includes services for I/O, Messaging (MQTT, CoAP), Networking (WiFi, Ethernet, Cellular), Watchdog, Web UI. Uses Eclipse Equinox OSGi framework.

From Ghent University. Small-footprint OSGi framework (~300K). Runs Kura and Felix services.

Commercially supported OSGi framework for IoT.

OSGi homepage [https://www.osgi.org/](https://www.osgi.org/)
WHAT ABOUT REAL TIME?

Consider:

A Real-Time System:
- Must Process information and guarantee a response within a specified time.
- Missing the deadline (even if the response is correct) is a failure; and if safety critical, may result in loss of life or limb.
- Real Time is as much about predictability as speed.
WHEN IS REAL TIME REQUIRED?

When is Real Time Required? When Milliseconds Matter!

Real-World, Real-Time Examples:
- **Factory Automation** - Machine tool on a factory floor
- **Logistics** - Automated airport baggage sorter
- **Driver Assistance** - Driver fatigue detection
- **Energy** - Wind turbine gearbox
- **Building Automation** - HVAC fire response
- **Public Access** - Stadium entrance gates

If your brakes engage in 10 milliseconds nine times out of ten but then once after 2 seconds, is the average of 209 ms acceptable?
TRADITIONAL JAVA PROBLEMS

Why Traditional Java can’t do Real Time:
- Garbage Collector Pauses
- JIT Compiler Interference
- Page Fault Delays
- Lack of Precise Timing Control
- Limited Thread Scheduling Control
- Priority Inversions
WHAT ARE MY CHOICES?

Alternatives to Traditional Java for Real Time:
- Go back to C or C++
- Split the application into RT/non-RT
- Try tuning the GC and hope for the best
- Try a “pauseless” GC implementation
- Try an RTSJ implementation
- Use a Real-Time Java SE
Look for these Features in a Real-Time Java:

- Garbage Collector (GC) can be preempted at any time by high priority threads and then resume collecting.
- Ahead-of-Time compilation and early JIT available.
- Can lock memory pages into physical RAM.
- Provides jitter-free timing APIs for critical tasks.
- GC and Java threads are prioritized and strictly scheduled. JVM provides user control over scheduler timing. Can use real-time policies for hard-prioritized Linux scheduling.
- Implements Priority Inheritance Protocol to avoid inversions.
Real-Time Linux Options:
- Get a commercial RT Distribution
- Apply kernel Preempt-RT patches
- Set kernel CONFIG_PREEMPT, CONFIG_HIGH_RES_TIMERS

Tips and Tricks:
- Disable CFS Bandwidth Enforcement
- Boost ksoftirqd thread policy/priority
- Use SCHED_RR or SCHED_FIFO policy
- Call mlockall()
- Pre-allocate memory
Problem Statement:

- You are an astronaut stranded on Mars.
- You need to grow enough food to survive until a rescue mission arrives.
- You build a greenhouse inside your habitat to grow potatoes.
- You need an IoT Gateway to automate the greenhouse.
Mars Greenhouse IoT Gateway Requirements:

- Monitor and display sensor data for available sunlight, internal temperature, and pressure.
- Store sensor data and transmit to the IoT Cloud at Mission Control.
- Round-trip delay for communication with Earth requires local control of internal lights and heat to adapt to changing conditions (i.e. dust storms).
- Bonus: Track harvested potatoes, estimated astronaut TTS (Time to Starvation).
Mars Greenhouse IoT Gateway Hardware:

- BeagleBone Black Rev. C, 1 GHz ARM Cortex-A8, 512 MB DDR3 RAM, 4GB on-board Flash, I2C, AIN, GPIO, Ethernet, USB
- CdS Photoresistor
- BMP085 I2C Temperature/Pressure Sensor
- LEDs, Relays
Mars Greenhouse Software Stack:

- Startlevel
- Package Admin
- Concierge Shell
- Greenhouse Control
- Servlet API
- HTTP Services
- Web Console

- Eclipse Concierge OSGi
- PTC Perc Real-Time Java
- Debian Linux
- BeagleBone Black Rev C
```java
public void run()
{
    float lthresh = 0.5f;
    float tthresh = 21.1f;
    Thread.currentThread().setPriority(10);
    System.loadLibrary("mars_BBB");
    try
    {
        lthresh = Float.parseFloat(System.getProperty("light_thresh", "0.5"));
        tthresh = Float.parseFloat(System.getProperty("temp_thresh", "21.1"));
    }
    catch (NumberFormatException nfe)
    
    initBBBNative();
    running = true;
    while(running)
    {
        light = getBBBLight();
        boolean old_lights = lights_on;
        lights_on = (light < lthresh) ? true : false;
        if (old_lights != lights_on)
        {
            setBBBYellow(lights_on);
        }
        doBBBSample();
        temp = getBBBTemp();
        boolean old_heat = heat_on;
        heat_on = (temp < tthresh) ? true : false;
        if (old_heat != heat_on)
        {
            setBBBRed(heat_on);
        }
        kpa = getBBBkpa();
        updateSim();
        try
        {
            Thread.sleep(100);
        }
        catch (InterruptedException ie)
        
        closeBBBNative();
        done = true;
    }
}
public void doGet(HttpServletRequest request, 
HttpServletResponse response) 
throws ServletException, IOException 
{
    if (request.getParameter("reset") != null) 
    { 
        bbb.resetSim();
    }
    // Set refresh, autolog as 1 seconds 
    response.setIntHeader("Refresh", 1);
    // Set response content type 
    response.setContentType("text/html"); 
    PrintWriter out = response.getWriter();
    float light = bbb.getLight() * 100.0f;
    float temp = bbb.getTemp();
    float kpa = bbb.getKpa();
    
    boolean lights_on = bbb.getLightsOn();
    boolean heat_on = bbb.getHeatOn();
    
    int elapsed = bbb.getElapsedTime();
    int plants = bbb.getPlants();
    int harvest_count = bbb.getHarvestCount();
    int harvested = bbb.getHarvested();
    float consumed = bbb.getConsumed();
    float available = bbb.getAvailable();
    float ttc = bbb.getTTCs();
    
    String lightval = String.format("%.1f" , light);
    String lightpct = String.format("%.1f", (int)light);
    int tpc = (int)((temp + 20.0f) * (100.0f / 80.0f));
    if (tpc < 0) tpc = 0;
    if (tpc > 100) tpc = 100;
    String tempct = String.format("%d", temp); 
    String kpa = String.format("%.1f", kpa);
    int kptc = (int)(kpa % 100.0f / 120.0f); 
    if (kptc < 0) kptc = 0;
    if (kptc > 100) kptc = 100;
    String kppct = String.format("%d", kptc); 
    
    String lights_on_active = lights_on ? " on-active" : " ";
    String lights_off_active = lights_on ? " off-active" : " ";
    String heat_on_active = heat_on ? " on-active" : " ";
    String heat_off_active = heat_on ? " off-active" : " ";
    
    String newhtml = String.format(prehtml, lightval, lightpct, tempval,
    tempct, kpa, kppct,
    lights_on_active, lights_off_active,
    heat_on_active, heat_off_active,
    elapsedval, plantsval,
    harvestcountval, harvestedval, re_seededval,
    consumedval, availableval, ttsval); 
    
    out.println(newhtml);
}

// Method to handle POST method request.
public void doPost(HttpServletRequest request, 
HttpServletResponse response) 
throws ServletException, IOException 
{
    doGet(request, response);
}
Demo Time!
SUMMING UP

In Conclusion:

- Gateways are an important element in building scalable IoT systems.
- Java provides a great foundation for developing IoT Gateways.
- OSGi adds a robust software component model to Java.
- When a gateway or any other embedded system needs to respond quickly and predictably to events, a true Real-Time Java solution is your best choice.
For more information, please visit:

- PTC Perc Real-Time Java:
  
  http://www.ptc.com/developer-tools/perc