# Real-Time Java

• Kelvin Nilsen, "*Adding real-time capabilities to Java*", Commun. ACM 41, 6 (Jun. 1998), Pages 49 - 56.

• Greg Bollella, "*Real-Time Java Architecture and Prototyping*", Embedded Systems Conference, Spring 1999.

## Outline

- Real-Time Systems
  - What are Real-Time Systems?
  - Analysis & Scheduling
- What makes a System NOT Real-Time?
  - Hardware & Software
- Java's Limitations
  - What makes Java not Real-Time?
- Fixing Java
  - What can be done to make Java suitable for Real-Time?

## Real-Time Systems
### Definition

- Correctness
  - Correct Result
  - On Time Result
- Worst Case Analysis
  - Amdahl's Law does not apply
- Bounded and Predictable
- Hard Vs Soft Real-Time
  - Hard → Death & Destruction
  - Soft → Annoying

## Real-Time Systems
### Examples

- Traffic Control
- Air Traffic Control
- Robots (Industrial)
- Networks
  - Telephone Switching
  - ATM / Ethernet
- Planes, Trains, and Automobiles
- Cell Phones and Pagers
- Satellite, Rocket, Missile Control
- Medical Equipment

## Real-Time Systems
### Analysis

- Worst Case Analysis
  - Conservative
- Latency
  - Interrupt latency
  - Scheduler latency
- Tasks
  - Priority
  - Periodic with Deadline
  - Aperiodic - Can happen anytime
  - Sporadic - Bounded inter-arrival time

## Real-Time Systems
### Scheduling Algorithms

- ExecutionTime=C, Period=T, Utilization=U=C/T
- $U_{total} = \sum_{i=1}^{n} U_i = \sum_{i=1}^{n} (C_i/T_i)$
- Static Vs Dynamic
  - Static - Rate Monatomic Analysis (RMA)
    - $U_{total} < \ln(2) = 69\%$
  - Dynamic - Earliest Deadline First (EDF)
    - $U_{total} < 100\%$

## NOT Real-Time
**Hardware**

- Pipelining
- Caches
- DMA
- I/O
  - – Disks
- Virtual Memory

## NOT Real-Time
**Software**

- Memory Allocation / Deallocation
  - – Implicit
  - – Explicit
- Device Access
  - – Polling
  - – Interrupts
- Deadlock
  - – Circular dependence on resources

## NOT Real-Time
**Software**

- Priority Inversion
  - – L = Low, M = Middle, H = High priority task
  - – S = Semaphore
  - – L acquires S
  - – H preempts L and waits on S
  - – M gets scheduled and stops L from being scheduled
  - – M effectively blocks H!

## Java's Limitations

- Task Scheduling
  - – Interrupts
  - – Event Latency
  - – Scheduling Algorithms
- Task Synchronization
  - – No bounded waiting time
  - – Deadlock & Priority Inversion

## Java's Limitations

- Garbage Collection
  - – Conservative
  - – Fragmentation
  - – Scheduling Latency
  - – No Memory Budgeting
- Device access
  - – No direct access to Physical Memory or I/O Devices

## Working With Java

- Task Scheduling
  - – Timer & TimerTask in JDK1.3
    - Periodic non-preemptive tasks
    - Scheduled by Earliest Nextstart First (ENF)
- Task Synchronization
  - – Keep Synchronized code short
  - – Avoid Cyclic Dependencies

## Working With Java

- Garbage Collection
  - Do not Dynamically Allocate memory
  - Allocate pools of memory for later use
- Device Access
  - Use Native Methods to access Physical Memory & I/O Devices

## Fixing Java
### The Real-Time Specification for Java

- Threads
  - Real-Time Threads
  - No Heap Real-Time Threads
    - Priority higher than Garbage Collector
- Scheduling
  - Feasibility Analysis
  - Algorithms
    - Pre-emptive RMA & EDF (RMA default)
  - Task Types
    - Periodic
    - Aperiodic
    - Sporadic

## Fixing Java
### The Real-Time Specification for Java

- Memory Management
  - Raw Memory
    - Device Access
  - Immortal Memory
    - Once allocated it is never freed/Garbage Collected
  - Scoped Memory
    - Memory allocated according to scope, and freed on scope exit
    - Similar to Stack Allocation but more general

## Fixing Java
### The Real-Time Specification for Java

- Memory Management
  - Heap Memory
    - Quotas (Budgeted Allocation)
      - Limits both amount and rate of memory allocation
    - Garbage Collection
      - Bounded Latency & Feasibility Analysis
      - R = Garbage Collector Utilization
      - V = total allocation throughput
      - S = time for complete garbage collection
      - M = total available memory, U = total required memory
      - R >= 2VS/(M-U)

## Fixing Java
### The Real-Time Specification for Java

- Memory Management
  - What Pointers are allowed in Memory areas ?

| | **Reference to Heap** | **Reference to Immortal** | **Reference to Scoped** |
|---|---|---|---|
| **Heap** | Yes | Yes | No |
| **Immortal** | Yes | Yes | No |
| **Scoped** | Yes | Yes | Yes if same, outer, or shared scope |
| **Local Variable** | Yes | Yes | Yes if same, outer, or shared scope |

## Fixing Java
### The Real-Time Specification for Java

- Synchronization
  - Priority Inversion Avoidance
    - Priority Ceiling Protocol
    - Priority Inheritance
  - Wait Free Queue
    - Priority Inversion Avoidance & Having higher priority than garbage collector can clash
    - Protects Data without Synchronizing/Blocking

## Fixing Java

### The Real-Time Specification for Java

- Time
  - High Resolution Time
    - Nanosecond accuracy
  - Absolute Time
  - Relative Time
  - Rational Time
    - N/D → event happens N times in D milliseconds
- Timers
  - Executes a TimerTask based on time
  - OneShot
  - Periodic

## Fixing Java

### The Real-Time Specification for Java

- Asynchrony
  - Event Handlers
    - Handlers share a pool of treads
    - May have to wait for a free thread
  - Bounded Event Handler
    - Handler is bound to a thread
    - Never waits for a thread to execute
  - Interrupts
    - An Object can run a method when interrupted if it implements Interruptible Interface
    - A Method may only be interrupted if it throws the AsynchronouslyInterruptedException
      - This Sucks ☹