Using the Executor Framework
to Implement AEH in the RTSJ

MinSeong Kim & Andy Wellings

Table of Contents

- Role of AEH in the RTSJ
- AEH Facility in the RTSJ
- Implementation Discussion
- Limitations of AEH
- The Executor Framework
- Applying the Framework to AEH in the RTSJ
- Conclusions
Role of AEH in the RTSJ

• Event-based programming
  - An alternative to thread-based programming

• AEH is used to handle the followings:
  a. External events to model parallelism with external objects
     - hardware interrupts, OS signals (SIGALRM), etc.
  b. Asynchronous error conditions detected by RT-JVM
     - a deadline miss or a cost overrun
  c. Application-defined error notification
     - a general error notification or a fault-handling mechanism
  d. Time-triggered events
     - periodic action or scheduled execution
     - OneShotTimer and PeriodicTimer
AEH Facility in the RTSJ

- AsyncEvent (AE)
- AsyncEventHandler (ASEH)
- BoundAsyncEventEventHandler (BoundASEH) extends AsyncEventHandler
Implementation Discussion

- RTTs & ASEHs are both Schedulable Objects,
- RTTs provide the vehicles for the execution,
- ASEHs are designed to be used as a lightweight concurrency mechanism,
- RTSJ does not provide any guidelines
- Major challenges
  - An efficient and predictable AEH implementation model
  - A smaller number of real-time server threads than the number of handlers
Limitations of AEH

• **Lack of implementation configurability**
  - The RTSJ does not provide any configurable facilities to finely tune the components of AEH

• **A single model for all types of non-bound asynchronous event handlers**
  - All asynchronous events must be handled in the same implementation-dependent way
  - Not possible for an application to indicate a different AEH implementation strategy for various handlers with different characteristics (blocking or non-blocking, heap-using or no-heap, hard or soft real-time handlers, and etc.,)

• **These limitations of AEH in the RTSJ severely weaken the configurability and the flexibility of the AEH implementation**
The Executor Framework

• In the `java.util.concurrent` package.

• Provides simple standardized extensible classes which provide useful functionality for using Java threads to control the execution of asynchronous tasks.

• It is extremely *useful* and *convenient* as a configurable server pool:

```java
ThreadPoolExecutor (int corePoolSize,
                    int maximumPoolSize,
                    long keepAliveTime,
                    TimeUnit unit,
                    BlockingQueue<Runnable> workQueue,
                    ThreadFactory threadFactory,
                    RejectedExecutionHandler handler)
```

• Therefore it is a good idea to use the executor framework for the execution of ASEHs in the RTSJ.
Applying the Executor Framework

- AEH cannot directly be used with the framework

- **Three Major Issues to Consider**
  1. **Use of Real-Time Threads**
     - By default, executors use a thread factory that creates normal Java threads
  2. **Use of a Priority Queue**
     - Any blocking queue can be used but they use a FIFO ordering
     - A priority queue must be used as a pending handler queue
  3. **Reflecting Submitted Handlers’ Priorities**
     ```java
     protected void beforeExecute(Thread t, Runnable r)
     protected void afterExecute(Runnable r, Throwable t)
     ```

- These solutions enable the executor framework to work well with AEH in the RTSJ
Conclusions

• Using the configurability and the flexibility of the Executor Framework
  ◇ Static 1:1 mapping
    - Bound ASEHs and non-bound ASEHs in OVM
  ◇ Dynamic 1:1 mapping
    - The RI and jRate
  ◇ Static 1:N mapping
    - Jamaica
  ◇ Dynamic 1:N mapping
    - Java RTS 2.0, and blocking and non-blocking AEH models
    - Not with the default run-time behavior

• Other mapping models for various ASEHs with different characteristics
  ◇ Hard and soft real-time handlers
  ◇ Heap and no-heap using handlers
  ◇ Daemon and non-daemon handlers

• Therefore it provides the programmer with an extremely configurable and flexible environment
Thank you
(Q & A)