Java New Operating System Design Effort

JNode a modern Java operating system

JNode.org

Ewout Prangsma

Java New Operating System Design Effort

Contents

Introduction

- Simple to use & install operating system for personal use: written for and in Java
- Targets:
  - Modern devices
  - Desktop
  - Small servers
- Only actively developed Java OS in the open source world
  - 5 active developers, 27K downloads

History

- Original idea started in 1995
- First attempt: JBS (Java Bootable System)
  - Contained C code, did not work at all
- Second attempt: JBS2
  - Still did not work well, but was better
- Then: JNode
  - No C code anymore, Classpath classlibraries
- Went public in May '03

Characteristics

- All Java, minimal assembler, no C
- All java build system (almost)
- Extensible architecture
- Single flat memory address space, no page swapping
- JVM written in Java, Program isolation in Java
- All Java code is compiled on the fly, no interpreter
- Security is always on
- LGPL license

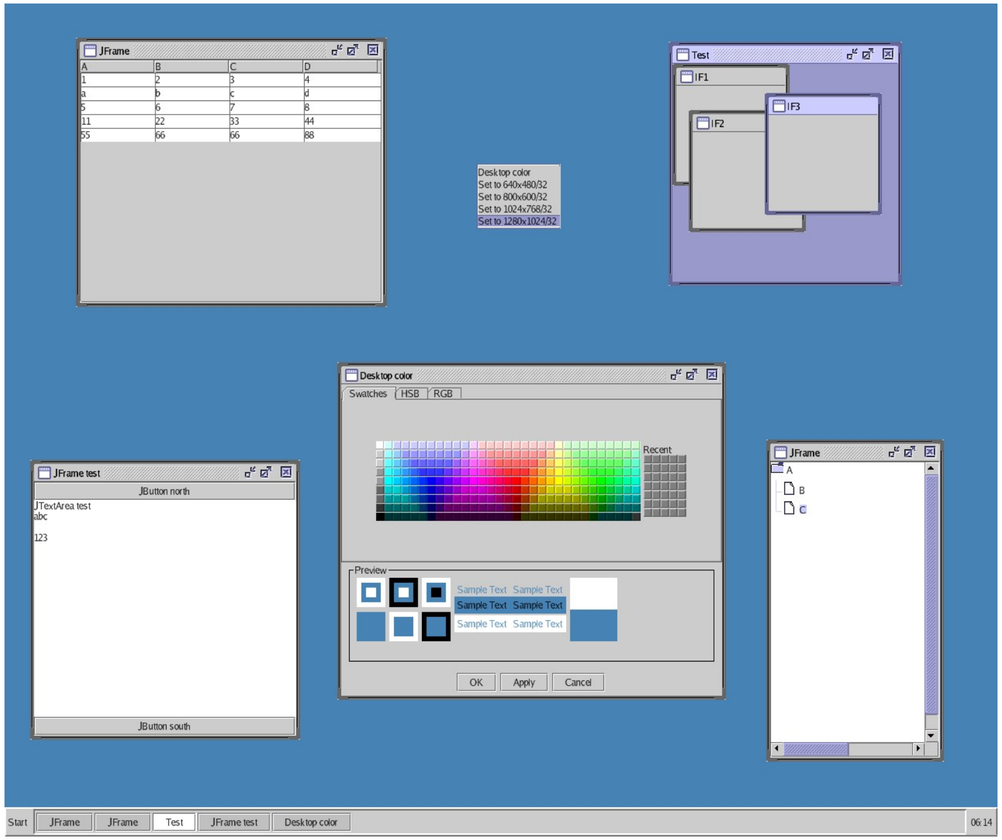
Status (1)

- Release 0.2.3
  - Full device driver model
  - Networking TCP/IP
  - Filesystems EXT2, FAT, NTFS, ISO9660
  - Graphics 2D, Java desktop
  - J2SDK 1.5 support in VM (not classlibs)
  - Simple heap managers & GC
    - MMTk based heap manager & GC in development
  - IA32 & AMD64 platform support

Status (2)

- Release 0.2.4-dev
  - Java Isolate support
    - Java program's run in their isolates space (domain)
    - Implemented in Java with assistance from native code compiler
    - Conforms to JSR 121

Status (3)



JFrame

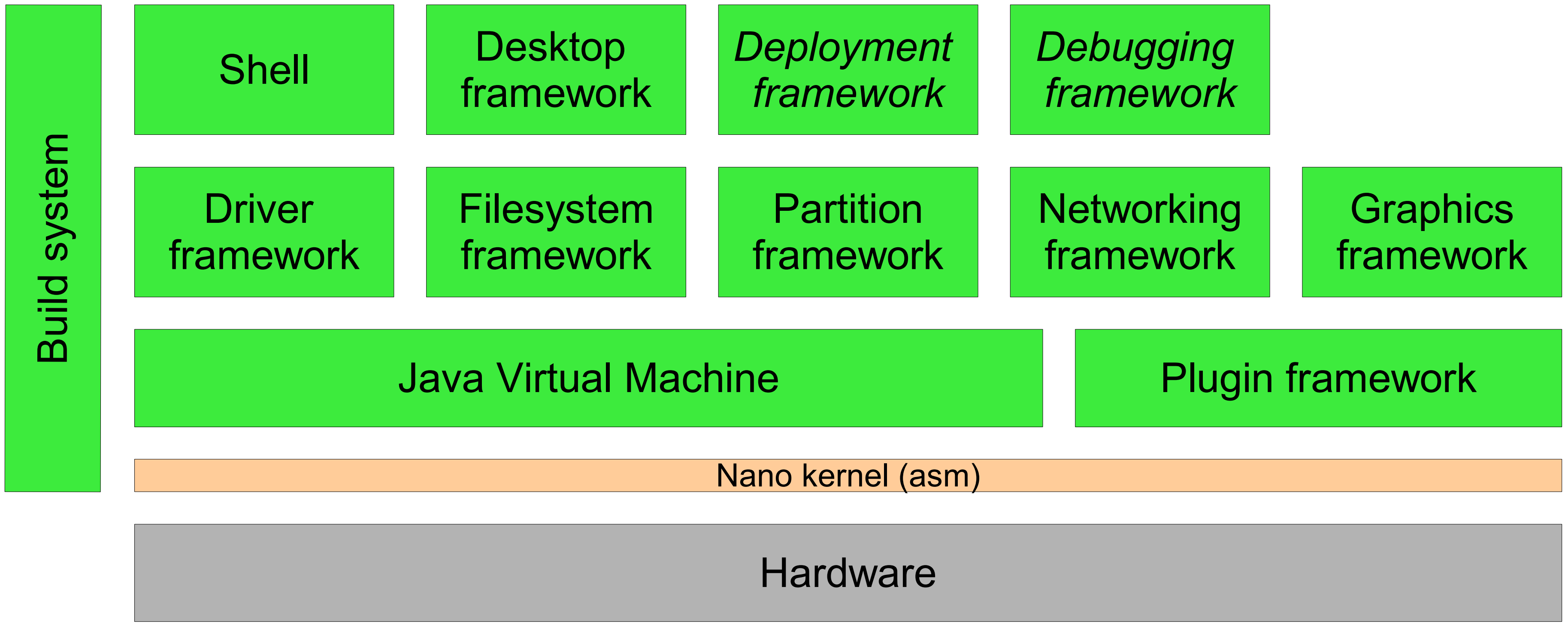
A	B	C	D
1	2	3	4
a	b	c	d
5	6	7	8
11	22	33	44
55	66	66	88

JTextArea test
abc

123

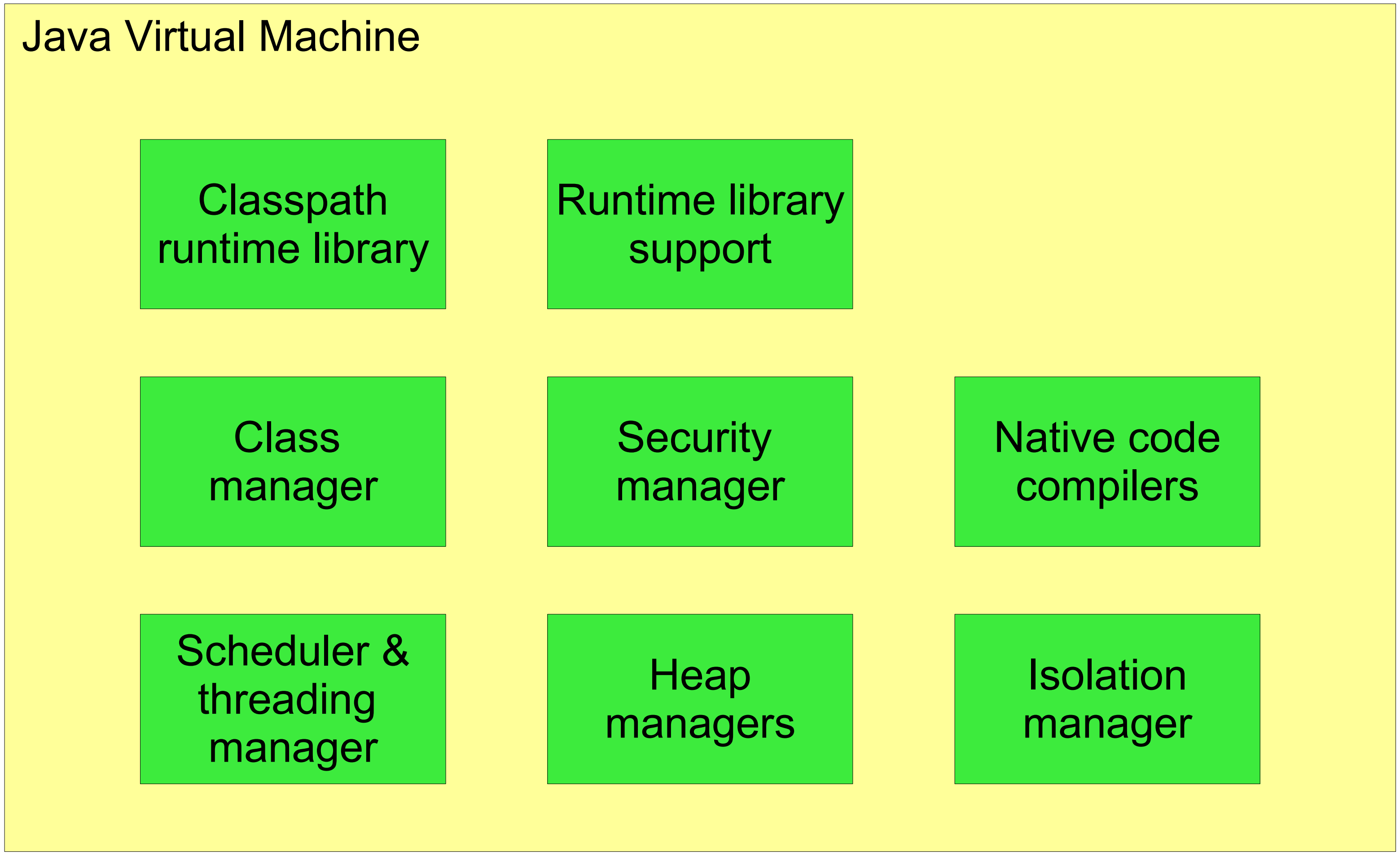
- JNode desktop
- JNode server connected to Internet

Architecture (1)



Java New Operating System Design Effort

Architecture (2)



Plugin framework (1)

- Everything is contained in a plugin
  - code, resources
  - even JVM & the plugin framework itself
- Plugins can:
  - be loaded, unloaded & reloaded (at runtime)
  - depend on other plugins
  - provide well known extension points
  - connect to well known extension points

Plugin framework (2)

- Plugins are:
  - described by a descriptor
    - descriptor also contains license info
  - JAR files
  - inspired by Eclipse plugins

Plugin framework (3)

```
<plugin id="org.jnode.driver" name="JNode Driver Framework" version="@VERSION@"
        provider-name="JNode.org" license-name="lgpl" class="org.jnode.driver.DriverPlugin">

    <requires>
        <import plugin="org.jnode.work"/>
    </requires>

    <runtime>
        <library name="jnode-core.jar">
            <export name="org.jnode.driver.*"/>
            <export name="org.jnode.driver.util.*"/>
        </library>
    </runtime>

    <extension-point id="finders" name="System device finders"/>
    <extension-point id="mappers" name="Device to Driver mappers"/>

    <extension point="org.jnode.security.permissions">
        <permission class="java.util.PropertyPermission" name="jnode.cmdline"/>
    </extension>
</plugin>
```

General info

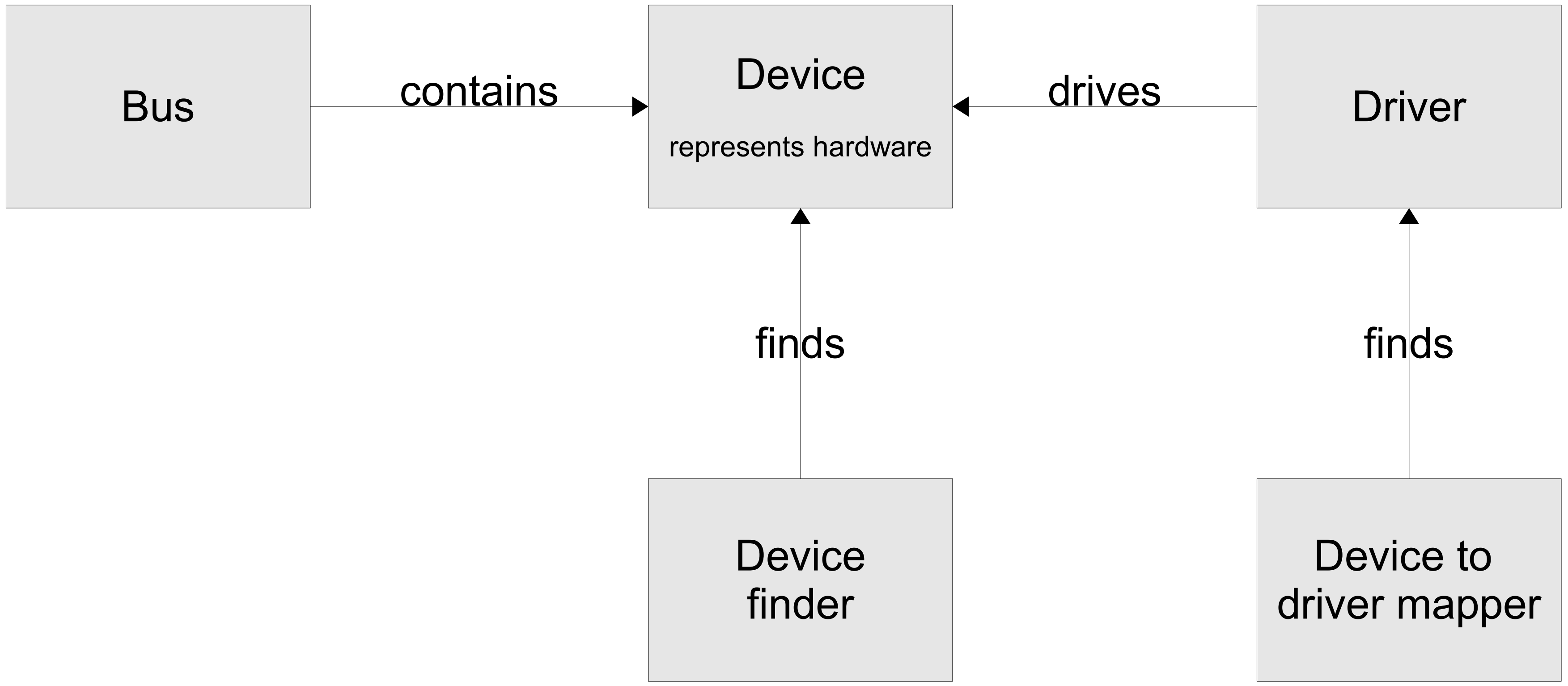
Dependencies

Code & resources

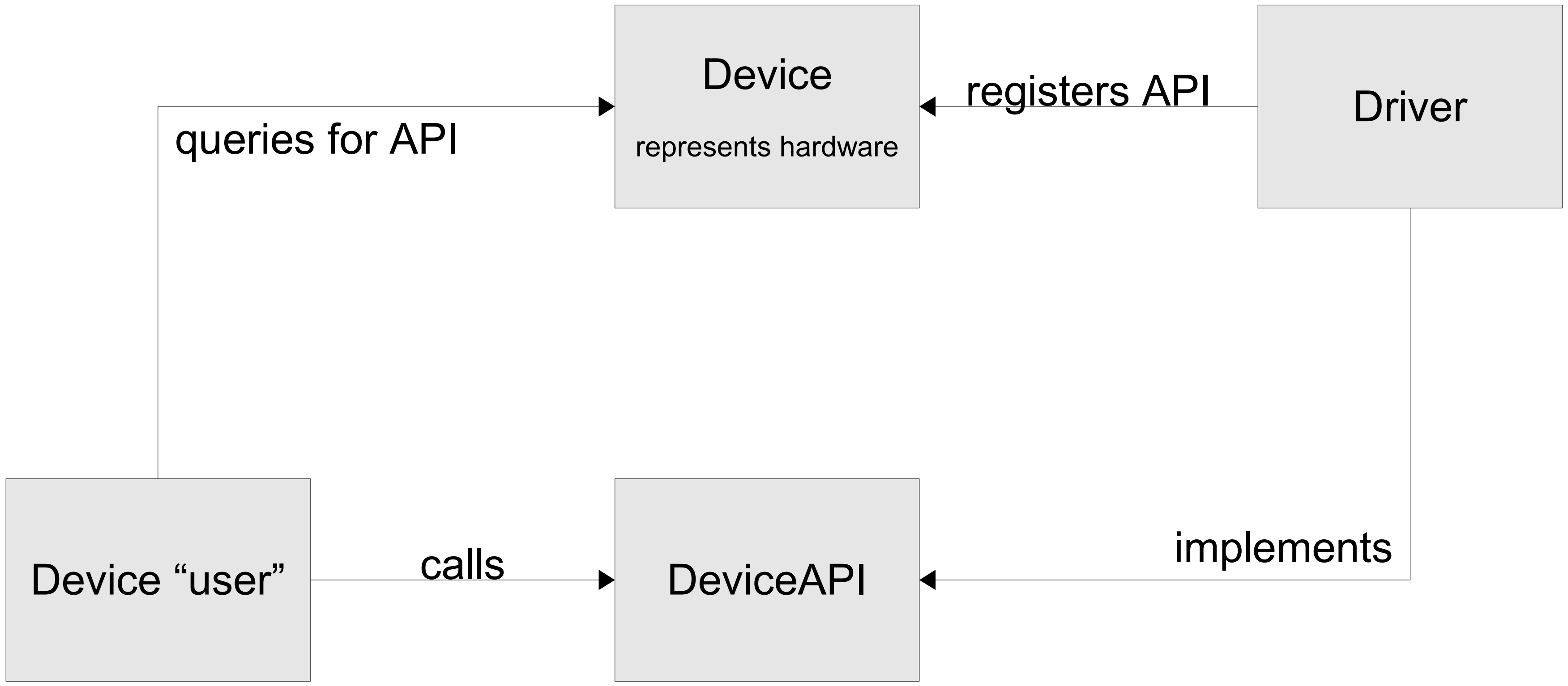
Well known extension points

Connection to well known extension point

Driver framework (1)

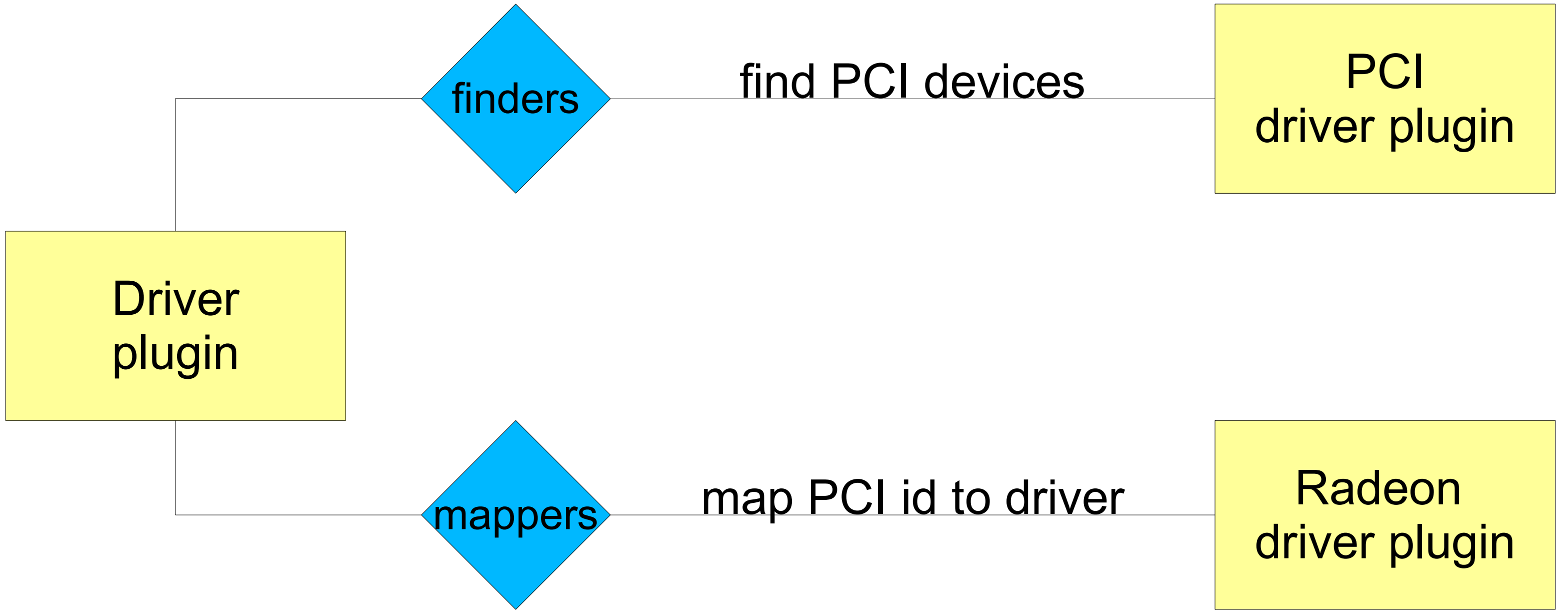


Driver framework (2)

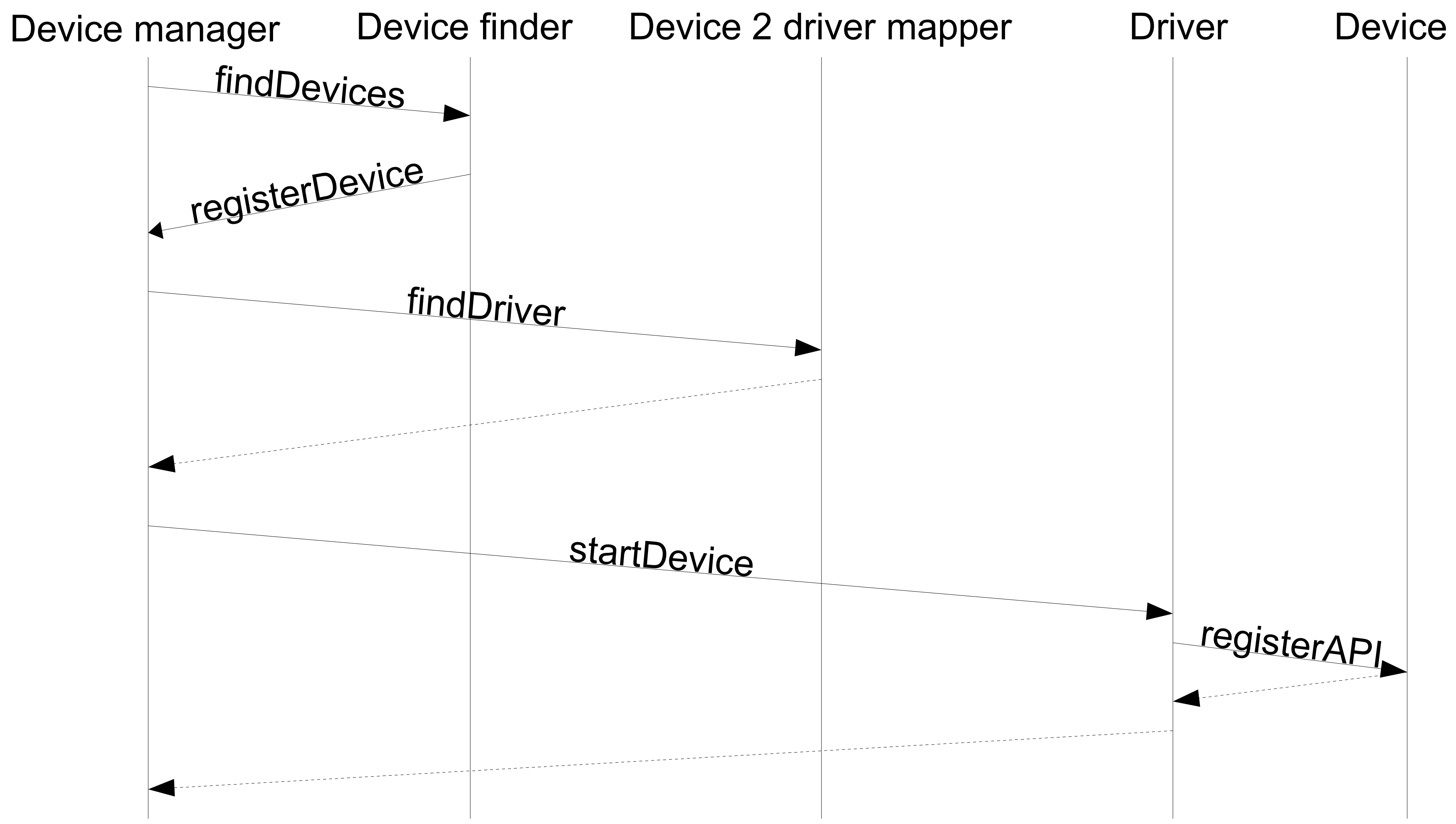


Driver framework (3)

Example: *Radeon Graphicscard driver*



Driver framework (4)



Java Program Isolation

- Goal:
  - Avoid program interference
  - Isolation should be “cheap”
- Isolated v.s. Shared resources
  - Isolated: program visible structures; threads, static variables, Class/Method/Field instances
  - Shared: internal structures; VmThread, VmType, ..., compiled native code

Challenges

- Performance
  - Writing good optimizing compilers is hard
- Perception
  - “Java is slow ...”
  - “ Why not use Linux ...”
- Time and resources
  - JNode is large... getting started is hard
  - All volunteers

Future (short term)

- Finalize Isolate support
- Multi CPU support
- Improved JVM stability & performance
- Improved graphics

Future (long term)

- Simple to use desktop environment
  - Fully document oriented instead of app. oriented
- Java powered servers
  - e.g. Cooperation with ApacheDS
- Technical future / dreams:
  - Easily portable
  - .Net (IL) support

Java benefits

- Dynamic linking
- Type safe language (even more in J2SDK 1.5)
- Security
  - Security manager
  - No uncontrolled memory access
- Great development tools:
  - Eclipse, Ant

Where can you help?

- More registers!
  - X86 register set is so limited, X64 is better
- Optimizing compiler knowledge & expertise
- Good documentation
  - Already pretty good, unlike GPU vendors

- Visit http://www.jnode.org
- Contact me: epr@jnode.org

Questions / Discussion