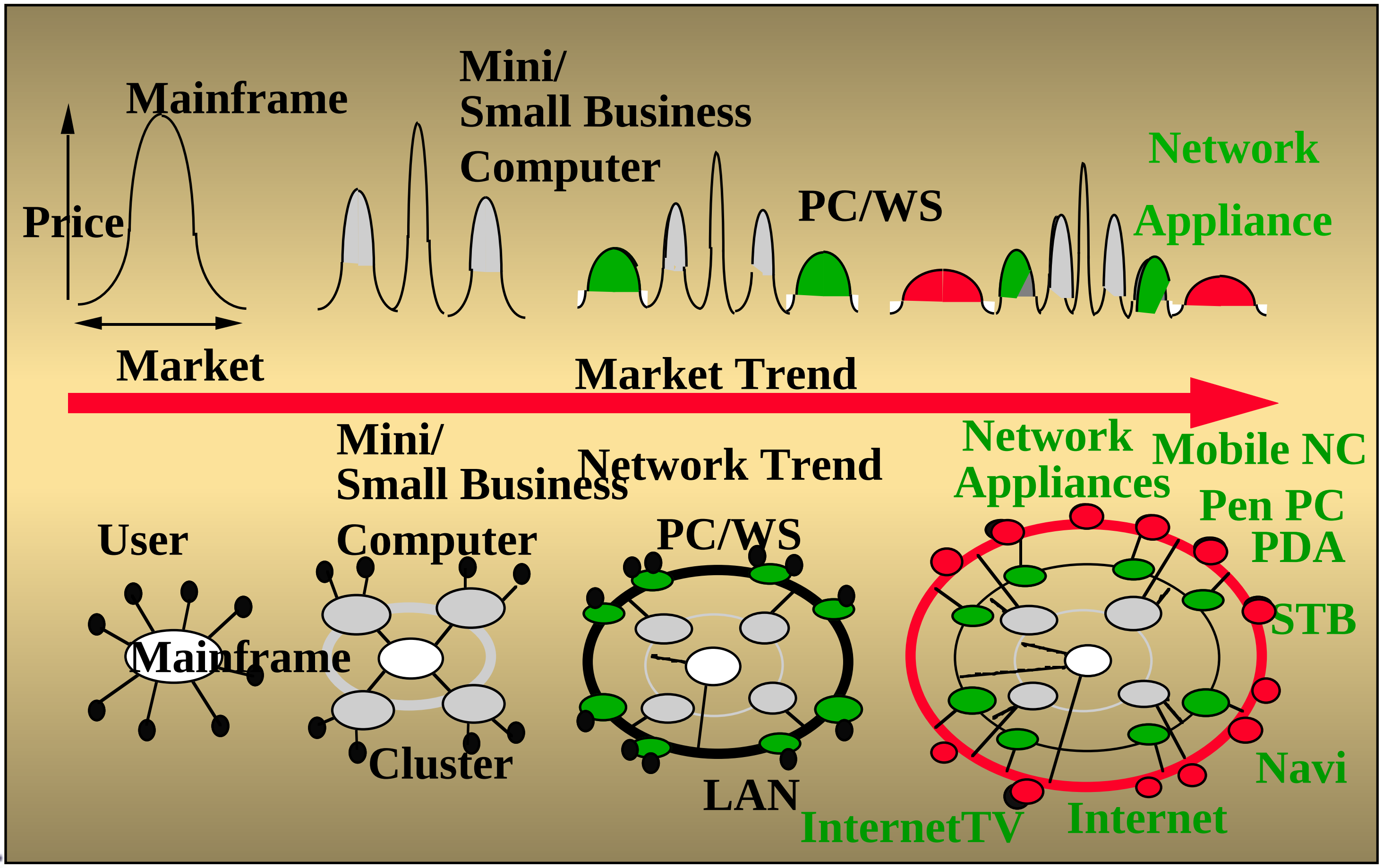
JavaOne℠

Sun's 1997 Worldwide Java Developer Conference

JavaOS™ Based Network Computing

Masahiro Kuroda, Chief Engineer
Scott Hansen, Dep. General Mgr.

Paradigm Shift and the New Wave



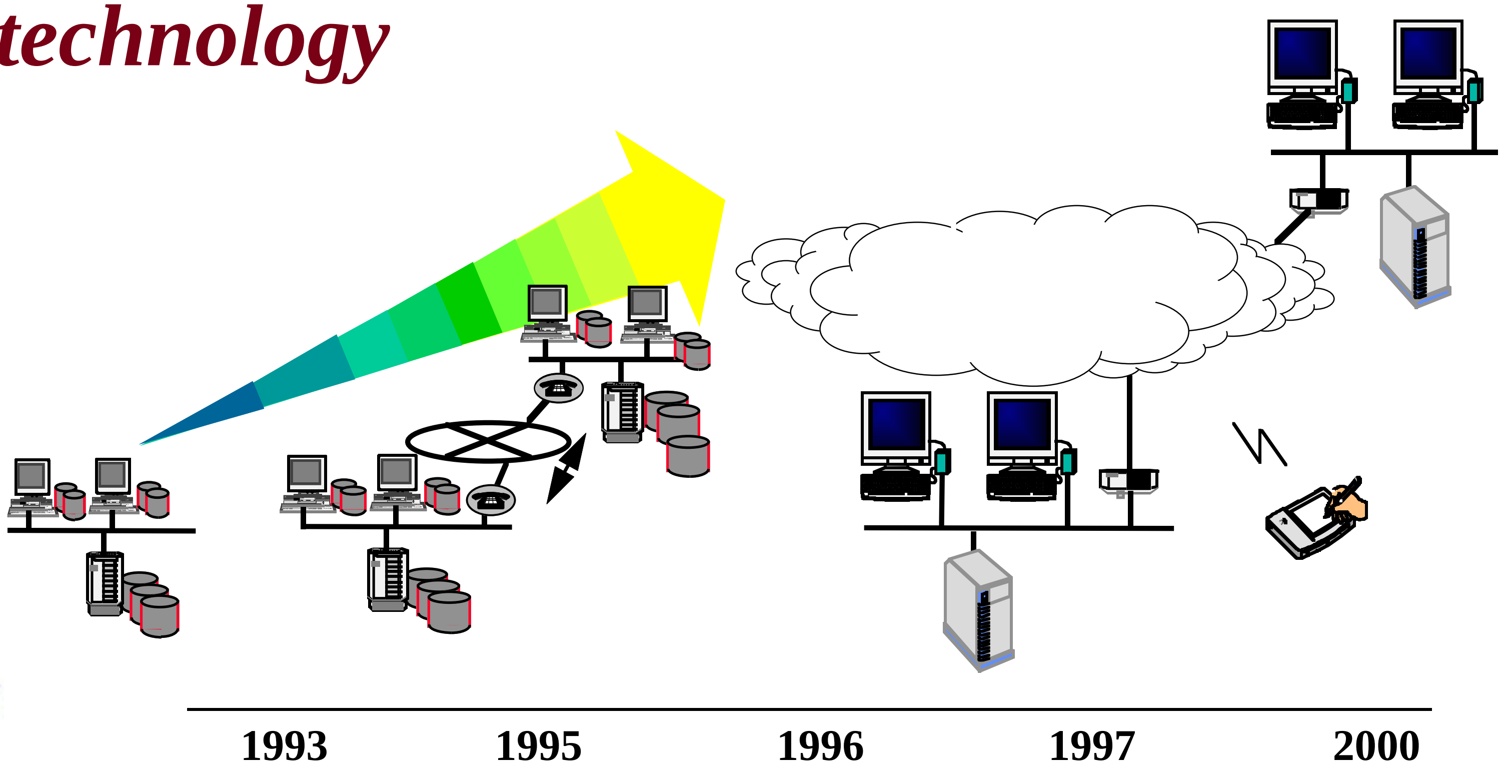
Network Computing Strategy

Focus on new tide of network computing based on Mitsubishi’s technological advantages

- Semiconductor (TFT, M32R/D, etc.)
- Consumer (TV, digital camera, etc.)
- Mobile computing (pen-based PC, etc.)
- Enterprise computing (high-end server)
- System integration
- Basic key technologies

Network Computing Concept

Mobile information systems (anytime, anywhere) using network computing technology



System Concept and Target Application

System Concept

•Seamless Office environment (Wireless LAN, PHS/PCS/CDPD, NC technology)
•Platform independent, intuitive user interface (Java technology)
•Internet/Intranet implementation of flexible systems
  à smooth migration of legacy system (Agent technology)
•Communication and data transfer technology suitable to wireless connectivity
  à (Proxy Server technology, etc)

Target Application Image

•"Anytime, Anywhere" Virtual Mobile Office
•Wide-area Information terminals/Servers System
•Internet/Public Network Information Providing Service
•

Network Computing Types

- Enterprise
- Mobile applications
- Home and consumer product

Office -- Enterprise

- MonAMI/NC
  - JavaOS™ based network terminal
  - NC management kit
- System integration for Enterprise
- Platform for VAR, SI

Mobile Applications

- MonAMI-II
  - JavaOS based mobile terminal
  - Wireless communication
  - Mobile server
- Sales support
- Patient care applications
- In car/train information system

Home -- Home and Consumer Product

- In TV
  - Electronic news
  - Virtual mall and home shopping
- In Telephone
  - Personal cell phone
  - SmartPhone

Why JavaOS ?

- Can support any emerging chips
  - Intel, PowerPC, M32R/D, PicoJava
- Run Java™ with limited resources
- Execute new applications efficiently on old CPUs

What Features Are Added

- System initialize
  - Any boot server -- DHCP/BOOTP
  - Local boot and remote boot
- Communication
  - Wireless LAN
  - Wireless WAN -- CDPD,PCS,PHS,etc
- Management
  - NC management kit -- configuration, User/App manage

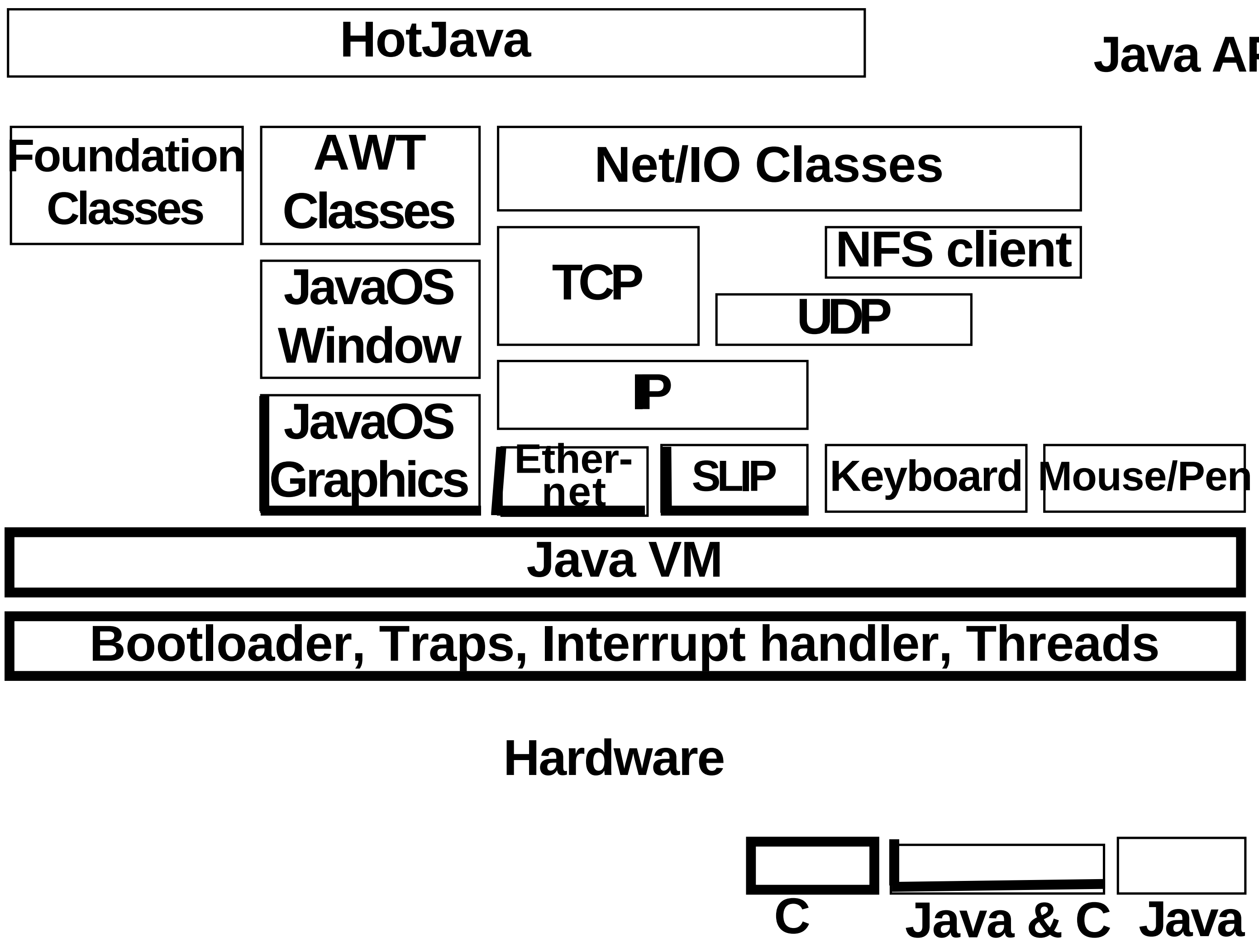
Java Enterprise Network Terminal

- MonAMI/NC -- Compact lunch box
  - At the COMDEX/Fall ’96 exhibition
  - Boot from Unix
- MonAMI/ES -- All-in-One TFT
  - At the JavaOne ’97
  - Boot from WindowsNT

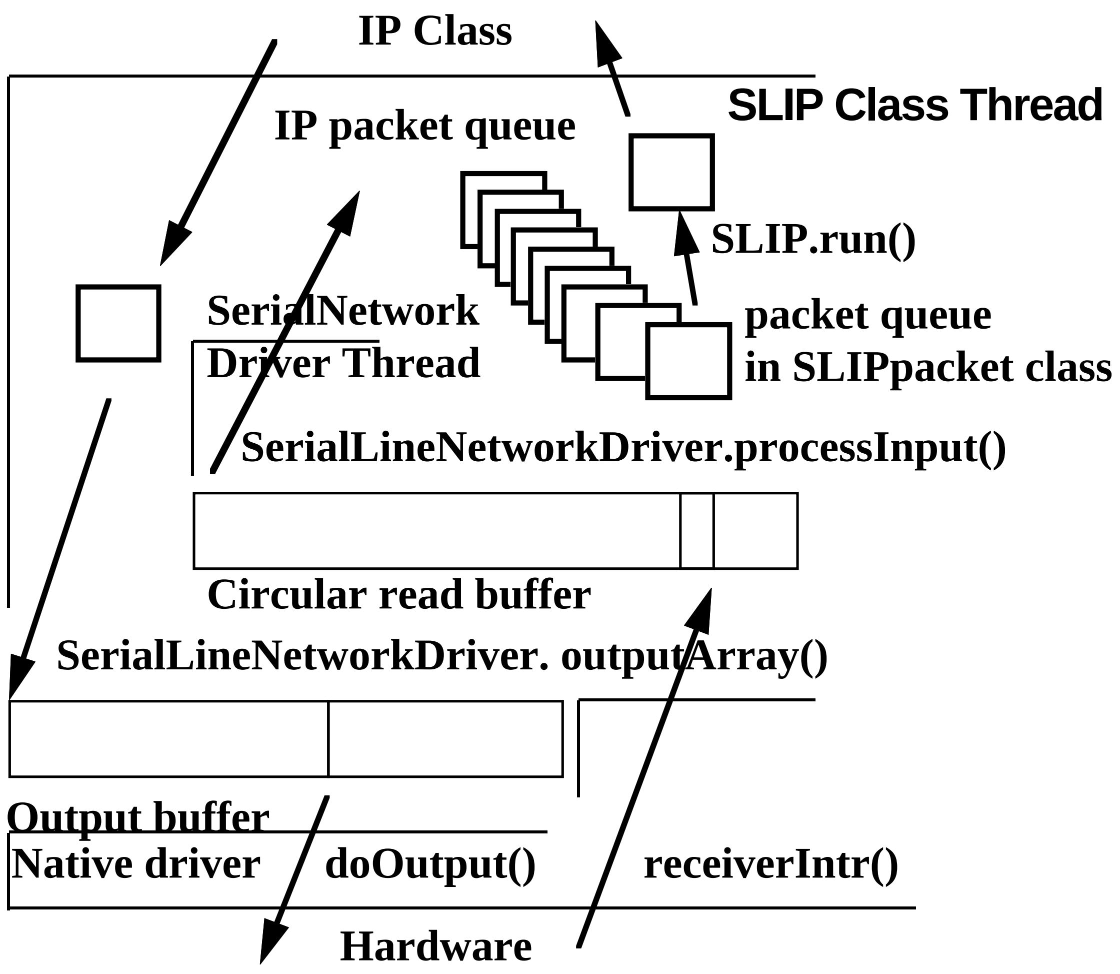
Java Mobile Network Terminal

- MonAMI -- Experimental
  - At the JavaOne ’96
  - At the COMDEX/Fall ’96 with wireless functions CDPD/LAN
- MonAMI-II -- Prototype
  - At the JavaOne ’97

JavaOS Internals



Java Thread Implementation Example



Java Driver or C Driver

- Depends on the interrupt handling
- Currently, trade off between performance and portability

The more written in Java,
the more portable

Speaker Change

JavaOne℠

Sun's 1997 Worldwide Java Developer Conference

Approach to Java™ for Embedded Systems

Mamoru Sakamoto

Approach to Java for Embedded Systems

- Java advantage
- Java cost
  - Memory / CPU usage
- Java for embedded systems
- JVM-M32R/D demo
- JVM-M32R/D
- Conclusion

Java Advantage

- Application development
- Distributed application
- Secure and robust environment to run external code
    - OO language, MultiThreaded, rich APIs, portable bytecode, interpreted, secure, robust, RMI

JVM Cost

Memory usage

- Stacks for threads
- Class information
- Images
- Object heap
- Java bytecodes
- C codes/static data

JVM Cost

CPU usage

- Bytecode interpretation
- Dynamic checking
    - Null pointer
    - Array index

Memory Cost

Stack for threads

- C stack and Java stack
- Unpredictable stack size
- Typically 14-20 threads

Memory Cost

Class information

- Class hierarchies
- Non private methods and variables
- Constant pools
- Strings
- Required for dynamic linking

Memory Cost

Images

- AWT always holds images decompressed

Memory Cost

Object heap

- Unpredictable max size
- Overhead for GC support
- Overhead by non precise GC

Memory Cost

Java bytecodes
C code, C static data

- AWT
- Java core
- Network
- RTOS
- C libraries

Approach to Embedded Systems

RTOS + Java

- RTOS
- Native device drivers
- Communication between Java threads and native threads

Approach to Embedded Systems

Static Java

- Disable dynamic class loading
- Statically link application and library classes
- Strip unnecessary information
- Convert bytecodes into native codes

Approach to Embedded Systems

Others

- Provide ways to estimate stack size
- Single threaded Java
- Alternative GUI packages other than AWT
- Static object memory management

JVM-M32R/D Demo

JVM/M32R

- RTOS (ITRON)
- no AWT
- JPEG decompression on the fly

JVM-M32R/D

M32R/D

- 32b RISC core
- eRAM (on-chip DRAM)
  - 128b internal bus
- 32b x 16b DSP-like multiply and accumulator

JVM-M32R/D

Implementation

- eRAM
  - C codes
  - C/Java stacks
- External DRAM
  - Class information
  - Bytecodes
  - Object heaps, etc.

Conclusion

- JVM memory cost
  - Stacks, class information, images, object heaps, Java bytecodes, C codes/data
- Approach to embedded systems
  - RTOS
  - Static Java