Chapter 3
Real-Time Operating System Overview

Hsung-Pin Chang

Department of Computer Science

National Chung Hsing University

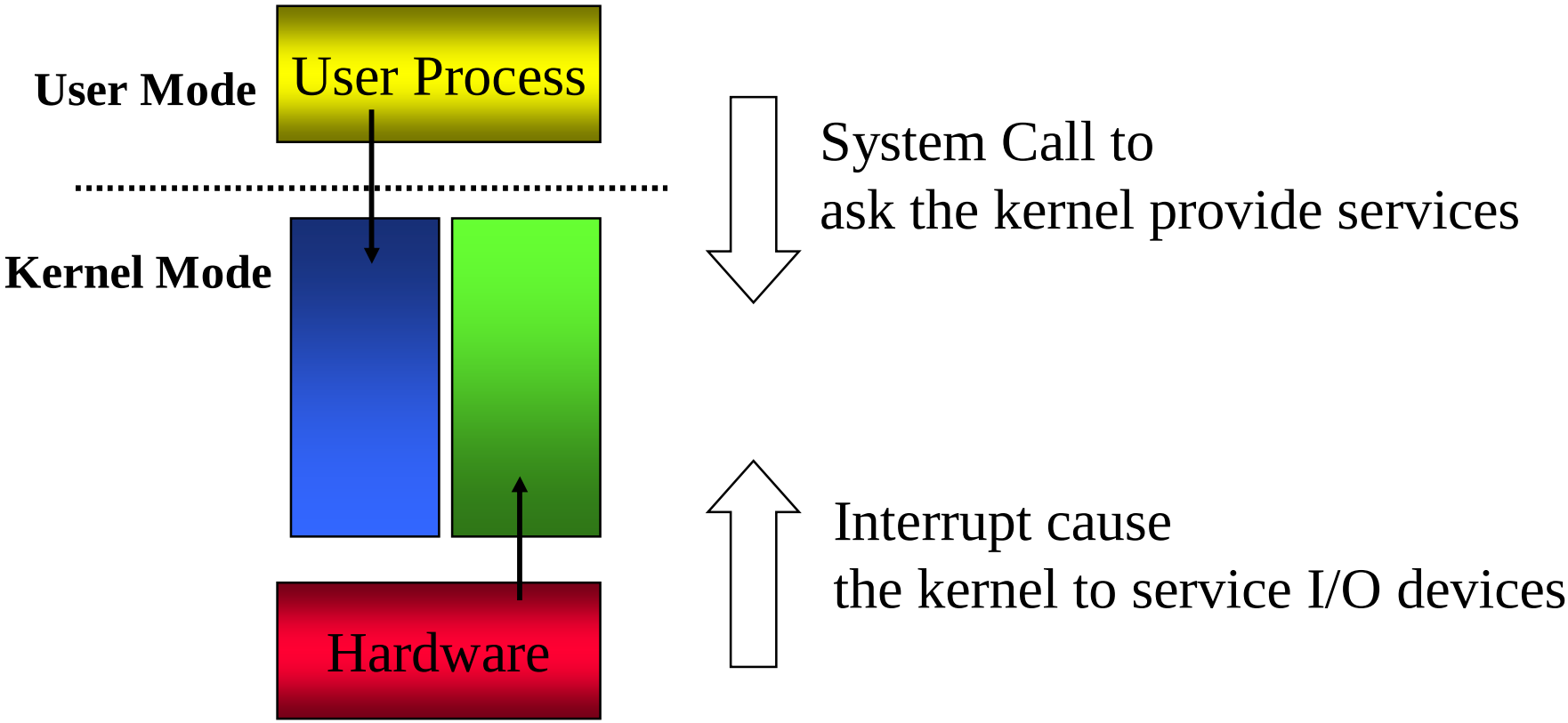
Outline

- 3.1 What Is an Operating System?
- 3.2 What is an Embedded Operating System?
- 3.3 Comparison of an RTOS and GPOS
- 3.4 RTOS Definition
- 3.5 Key Characteristics of an RTOS
- 3.6 Well-known RTOSes

3.1 What Is an Operating System?

- A program that acts as an intermediary between *a user* and *the computer hardware.*
- Goal:
  - Manage system resources
    - Use the computer hardware in an efficient manner.
  - Provide a development environment for applications
    - Via the operating system's API (Application Interface)
    - System API are the *only* interface between user applications and hardware
  - Facilitate program execution
    - Schedule program execution in a system-specific manner

High Level View of an Operating System



Components of an Operating System

- Process Management
- Memory Management
- Synchronization
- IPC (Inter-Process Communication)
- File System
- Hardware Resource Control
- Networking

3.2 What is an Embedded Operating System

- An operating system for embedded system
- Sometimes an embedded operating system is also referred to
  - *Embedded real-time operating system*
  - *Real-time operating system* (*RTOS*)
    - If the target system has real-time characteristics

Embedded RTOS Features

- Small size
- Preemptive kernel
- Respond to external interrupts quickly
- Multitasking
- Inter-Process Communication (IPC) scheme
- Fast context switch
- Minimization of intervals during which interrupts are disabled

3.3 Comparison of a RTOS and GPOS (1/2)

RTOS	GPOS
Optimize worst case	Optimize average case
Predictable schedule	Efficient schedule
Simple executive	Wide range of service
Minimize latency	Maximize throughput

3.3.1 Functional similarities Between a RTOS and GPOS

- Some level of multitasking
- Software and hardware resource management
- Provision of underlying OS services to application
- Abstracting the hardware from the software application

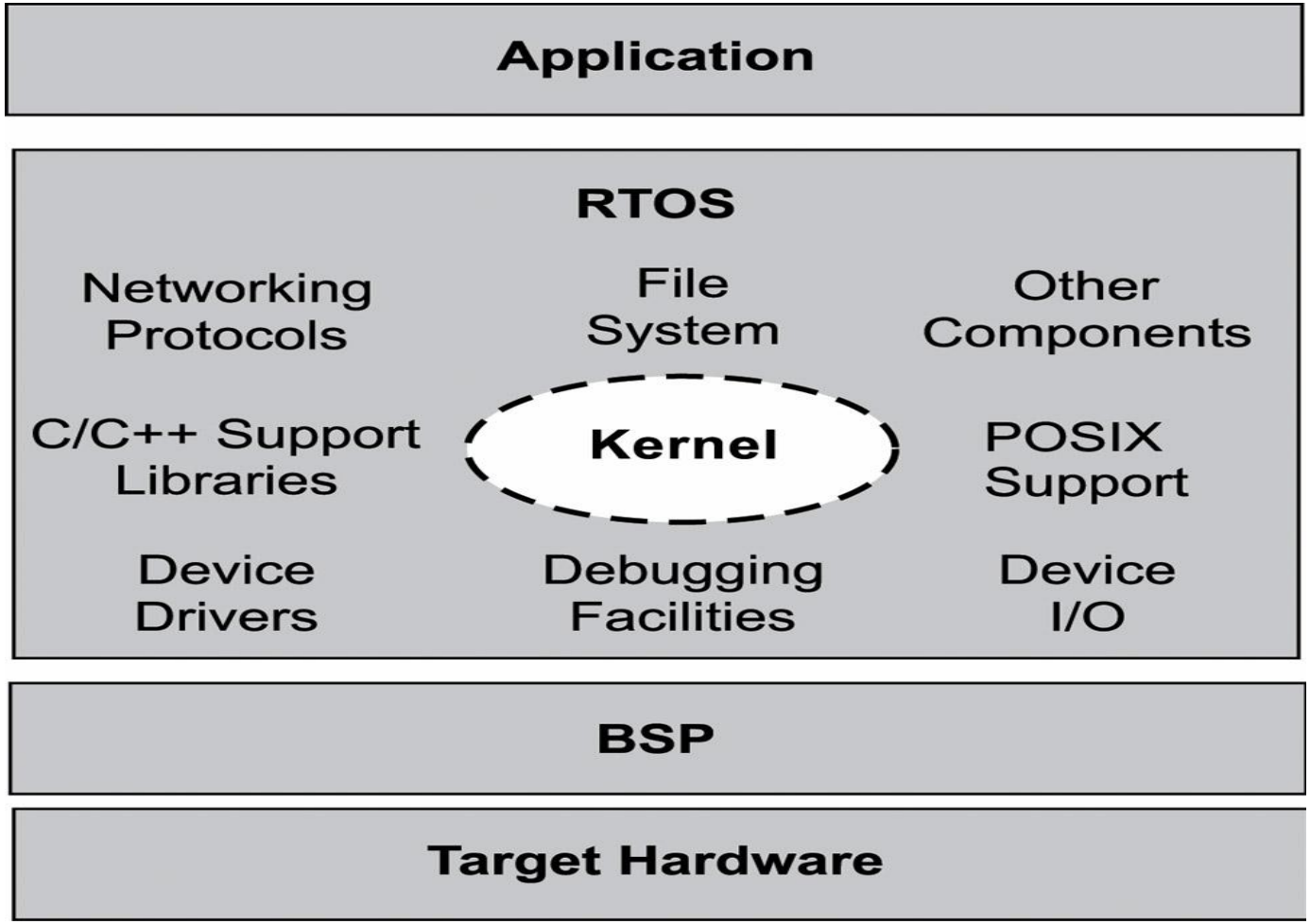
3.3.2 Functional difference Between a RTOS apart GPOS

- Better reliability in embedded application contexts
- The ability to scale up and down to meeting application needs
  - For example, if no disk/network, then we don't need file system and TCP/IP kernel codes
- Faster performance
- Reduced memory requirements
- Scheduling policies tailored for real-time embedded systems
- Support for diskless embedded systems by allowing executables to boot and run from ROM or RAM
- Better portability to different hardware platforms

3.4 RTOS Definition

- In some applications, an RTOS comprises only a kernel
  - The core supervisory software that provides minimal logic, scheduling, and resource-management algorithms.
- However, in most applications, an RTOS can be a combination of various modules
  - Kernel
  - File system
  - Networking protocol stacks
  - Other components required for a particular application

High Level View of an RTOS

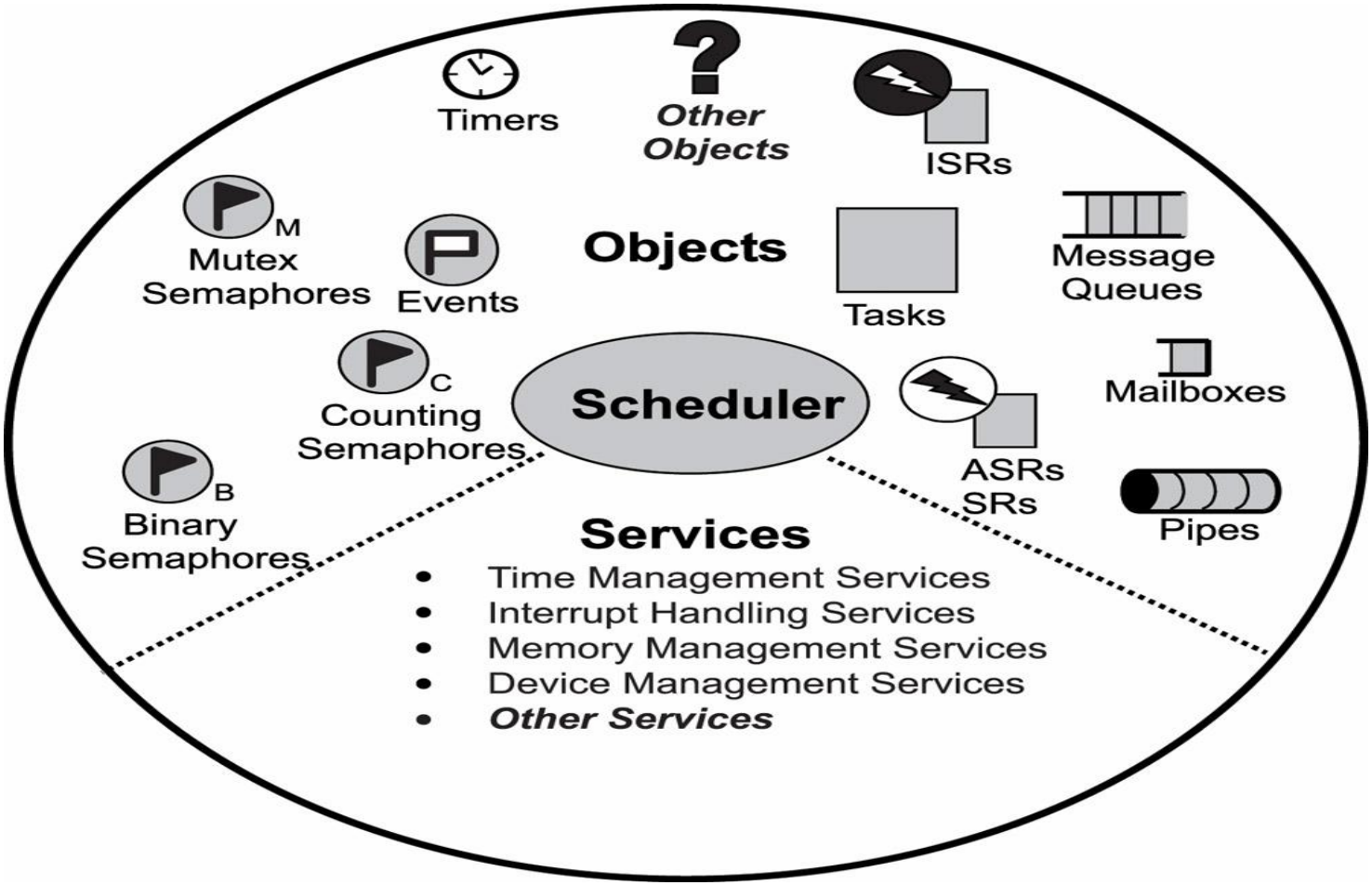


Source: Qing Li and Caroline Yao, "real-time concepts for embedded systems"

3.4.1 RTOS Kernel

- Most RTOS kernels contain the following components
  - Scheduler
    - Determine which task executes when
  - Objects
    - Special kernel construct that helps developer create applications
      - Tasks, semaphores, and message queues
  - Services
    - Operations that the kernel performs on an object
      - Timing, interrupt handling, and resource management

Common Components in an RTOS kernel



Source: Qing Li and Caroline Yao, "real-time concepts for embedded systems"

3.4.2 Kernel Objects

- Kernel objects are special constructs that are the building blocks for application development
  - Tasks
    - Concurrent and independent threads of execution that can compete for CPU execution time
  - Semaphores
    - Token-like objects that can be incremented or decremented by tasks for synchronization or mutual exclusion
  - Message Queues
    - Buffer-like data structures that can be used for synchronization, mutual exclusion, and data exchange by passing messages between tasks

3.4.3 Kernel Services

- Along with objects, most kernels provide services that help developers create applications for real-time embedded systems.
- Sets of API calls that can be used to
  - Perform operations on kernel objects
  - Facilitate
    - Timer management
    - Interrupt handling
    - Device I/O
    - Memory management

3.5 Key Characteristics of an RTOS

- Reliability
- Predictability
- Performance
- Compactness
- Scalability

3.5.1 Reliability

- A reliable system
  - Be available (continue to provide service) and does not fail
- Different degrees of reliability might be acceptable
- Quantified by the download per year
  - Number of 9s: the percent of the total time that a system must be available
- To determine system reliability
  - The combination of all system elements determines the reliability of a system
    - Hardware, BSP, RTOS, and applications

Categorizing Highly Available Systems by Allowable Downtime

Number of 9s	Downtime per year	Typical Application
3 Nines (99.9%)	~9 hours	Desktop
4 Nines (99.99%)	~1 hours	Enterprise Server
5 Nines (99.999%)	~5 minutes	Carrier-Class Server
6 Nines (99.9999%)	~31 seconds	Carrier Switch Equipment

3.5.2 Predictability

- Many embedded systems are also real-time systems
  - Meeting time requirements is key to ensuring proper operation
- For real-time systems, the RTOS needs to be predictable
  - The completion of operating system calls occurs within known timeframes
  - The variance of the response times for each type of system call should be small

3.5.3 Performance

- An embedded system must perform fast enough to fulfill its timing requirements
- Processor's performance
  - MIPS: million instructions per second
- Through may used to measure the overall performance of a system
  - the rate at which a system can generate output based on the inputs coming in
- Call-by-call method *may be* used to measure RTOS performance
  - Produce timestamps when a system call starts and when it completes

3.5.4 Compactness

- To determine how compact an embedded system be
  - Application design constraints
  - Cost constraints
- For example, a cell phone must be small, portable and low cost
  - Limit system memory
    - Limit the size of application and operating system
- To meet total system requirements, designers must understand both the static and dynamic memory consumption of the RTOS and the application that will run on it.

3.5.5 Scalability

- RTOSes can be used in a wide variety of embedded systems
  - Be able to scale up or down to meet application-specific requirements
- An RTOS should be capable of adding or deleting modular components, including file systems and protocol stacks.
  - Depending on how much functionality is required
- For example, a RTOS may be used in both a cellular phone project and a base station project
  - Save time and money

3.6 Well-Known RTOSes (1/3)

- Nucleus Plus
  - Manufacturer: Accelerated Technology
- eCos (embedded Configurable operating system)
  - Manufacturer : redhat
  - Characteristics
    - Open source (GNU License)
    - Highly configurable
      - eCos provides source-level configuration
- VxWorks
  - Manufacturer: WindRiver System

3.6 Well-Known RTOSes (2/3)

- QNX
  - Manufacturer : QNX Software Systems Ltd (Canada)
  - Characteristics
    - Microkernel architecture
    - Full MMU support
- RTLinux
- Embedded Linux
- MicroC/OS-II
- …

3.6 Well-Known RTOSes (3/3)

- How to select a RTOS for your own system ?
  - What are the requirements of your applications?
  - Which OS features are really necessary?
  - How hard are the real time constraints?
  - Is the OS available for the chosen hardware platform?
  - What are the overall cost of using the OS including the license fee, tool cost, training and royalty?
  - How about the supporting tools?
    - Development software, debugging facility, documents…

Reference

- Qing Li and Caroline Yao, “Real-Time Concepts for Embedded Systems”, CMP Books, ISBN: 1-57820-124-1, 2003
- Anthony J. Massa, Embedded Software Development With eCos, Prentice Hall, ISBN: 0-13-035473-2, 2003.
- Jean J. Labrosse, "MicroC OS-II: The Real Time Kernel," ISBN: 1578201039, CMP Books, June 15, 2002.
- Silberschatz, Galvin, and Gagne, “Operating System Concepts”, John Wiley, 2002.
- David Stepner, Nagarajan Rajan, and David Hui, “Embedded Application Design Using a Real-Time OS,” Design Automation Conference, pp. 151-156, 1999