CS343: Operating System

OS Structure and Types

Lect08 : 14th Aug 2023

Dr. A. Sahu

Dept of Comp. Sc. & Engg.

Indian Institute of Technology Guwahati

Outline

- OS Structures
  - How the OSs are structured/organized
- Different type of OS
  - Desktop, Android, Cloud, Peers, Embedded, RealTime

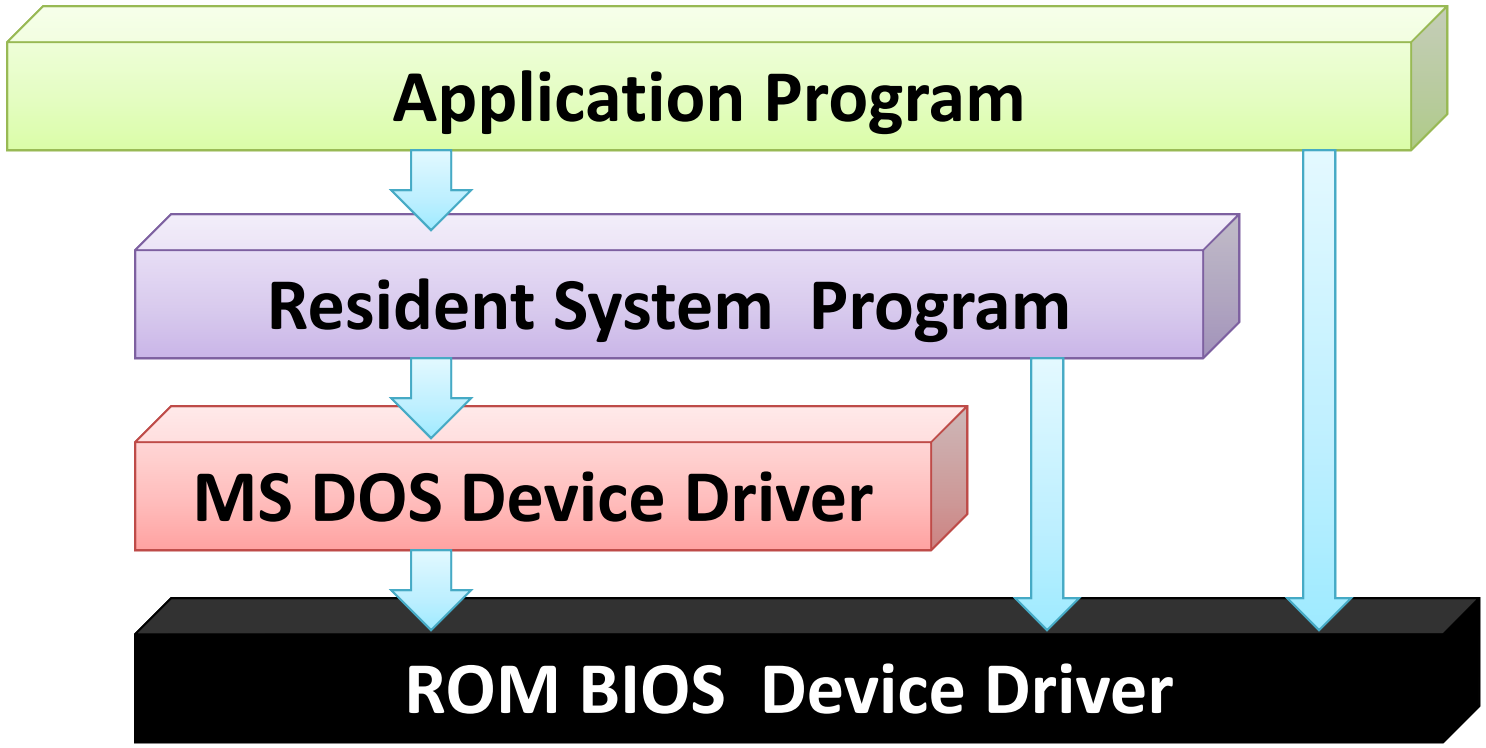
Operating System Structure

Operating System Structure

- General-purpose OS is very large program
- Various ways to structure ones
  - Simple structure – MS-DOS
  - More complex -- UNIX
  - Layered – an abstraction
  - Microkernel -Mach

Simple Structure : MS-DOS

- Written to provide : most functionality in least space
  - Not divided into modules: **Monolithic Architecture**
  - Although MS-DOS has some structure, interfaces and levels of functionality are not well separated



Monolithic Structure

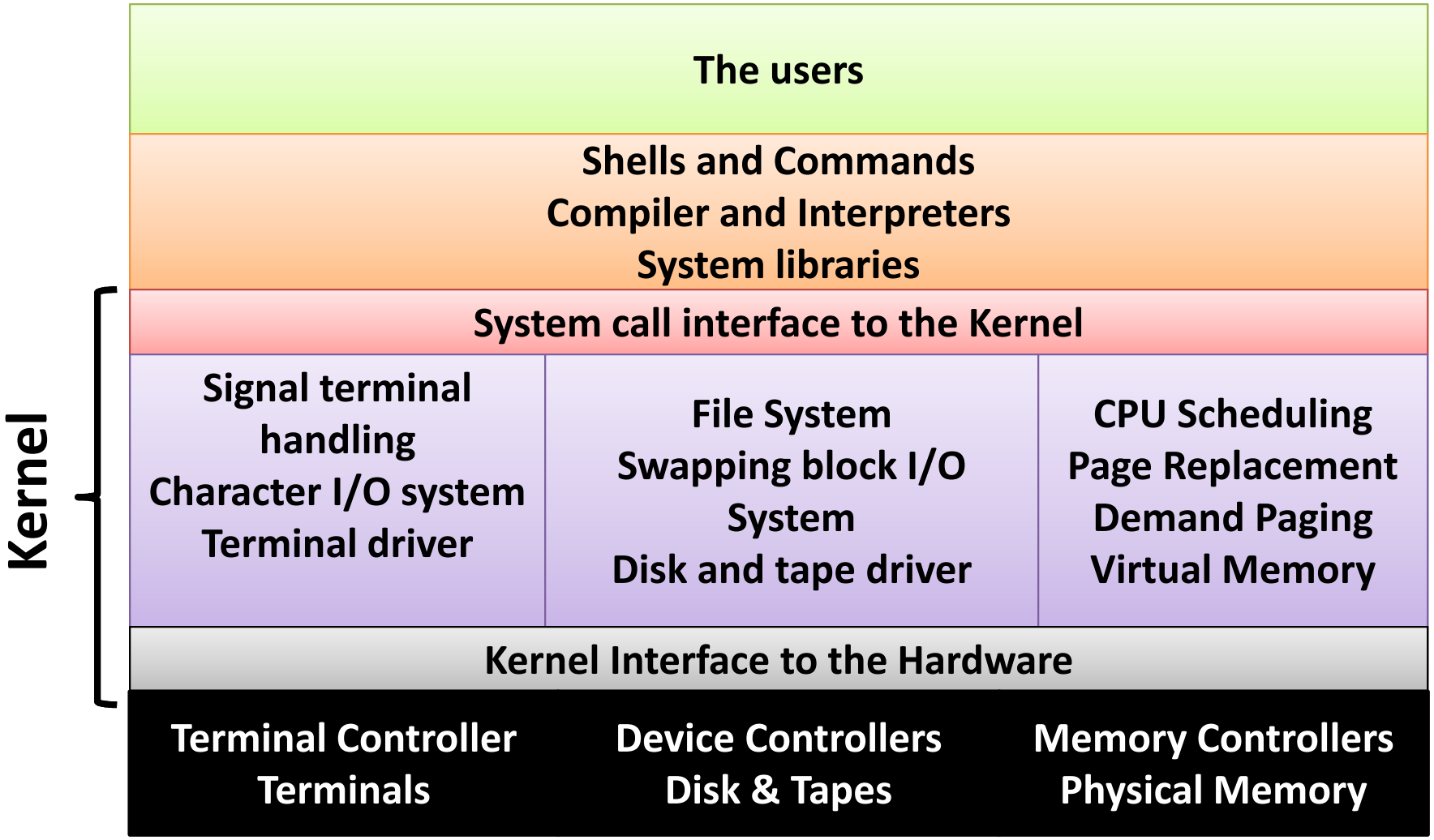
- Monolithic English Definition
  - **Formed of a single large block of stone**
  - Organization/system: large, powerful, indivisible, and slow to change.
- A software system is called "monolithic"
  - If it has a monolithic architecture
  - In which functionally distinguishable aspects
  - For example data input and output, data processing, error handling, and the user interface),
- Are not architecturally separate components but are all interwoven.
- MS DOS architecture is monolithic

Non Simple Structure: UNIX

- UNIX – limited by hardware functionality
- Original UNIX OS had limited structuring.
- The UNIX OS consists of two separable parts
  - Systems programs
  - Kernel
    - Consists of everything below the system-call interface and above the physical hardware
    - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level

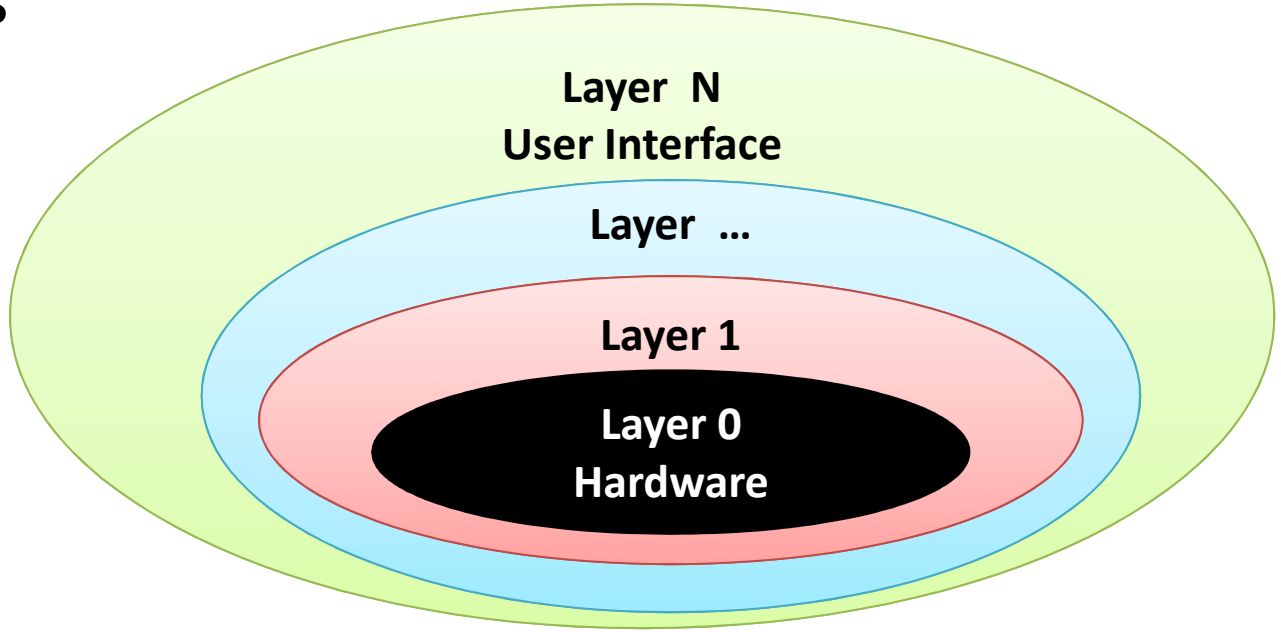
Traditional UNIX System Structure

Beyond simple but not fully layered



Layered Approach

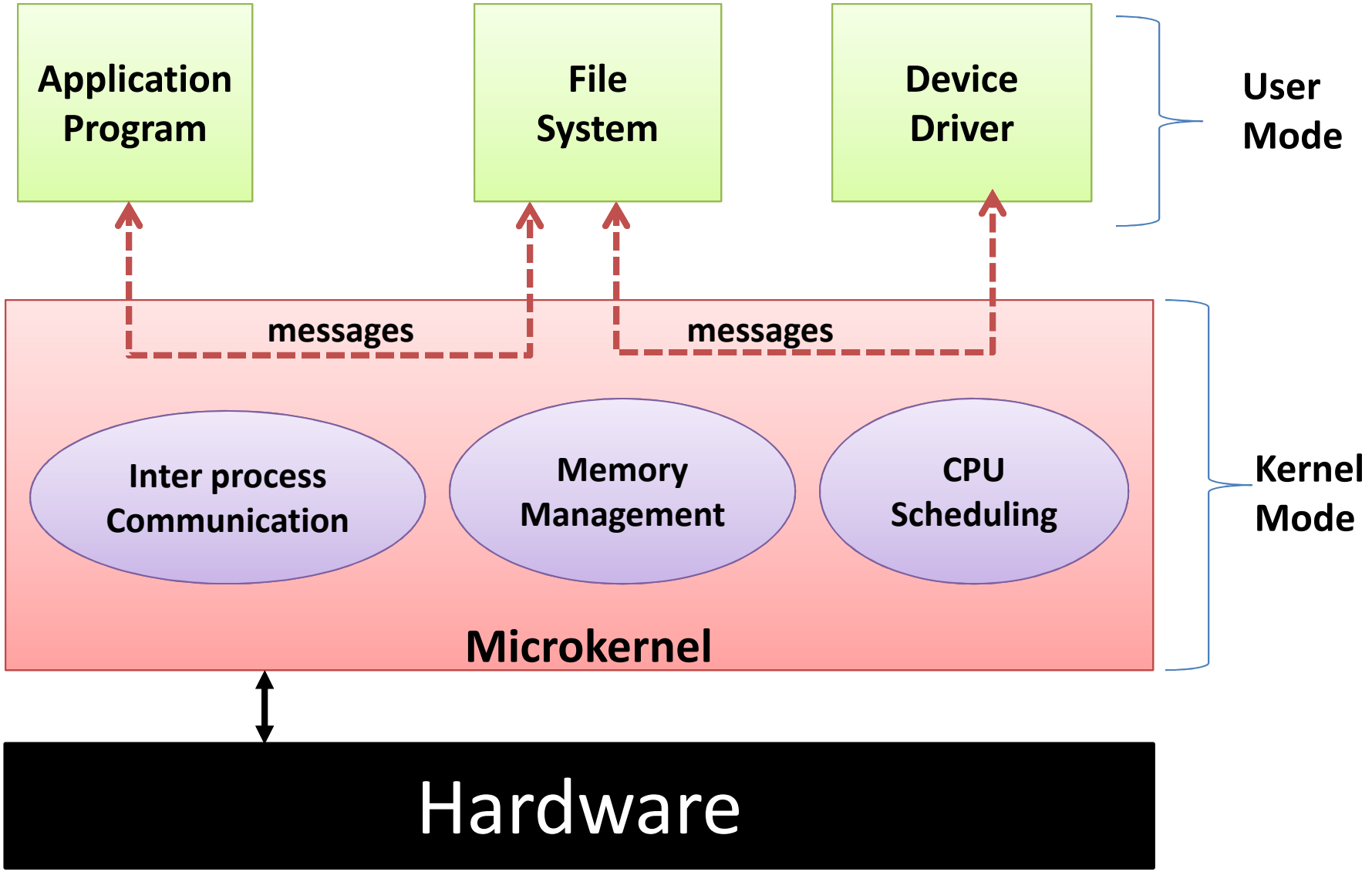
- OS is divided into a number of layers (levels)
  - Each built on top of lower layers.
  - bottom/0 layer: hardware, highest/N =user interface.
- With modularity, layers are selected such that
  - Each uses functions and services of only lower-level layers



Microkernel System Structure

- Moves as much from the kernel into user space
- Communication between user modules
  - Using **message passing**
- **Mach** example of **microkernel**
  - Mac OS X kernel (**Darwin**) partly based on Mach

Microkernel System Structure



Microkernel System Structure

- Benefits:
  - Easier to extend and port to new architectures
  - More secure
  - More reliable (less code is running in kernel mode)
- Demerits:
  - Performance overhead of user space to kernel space communication

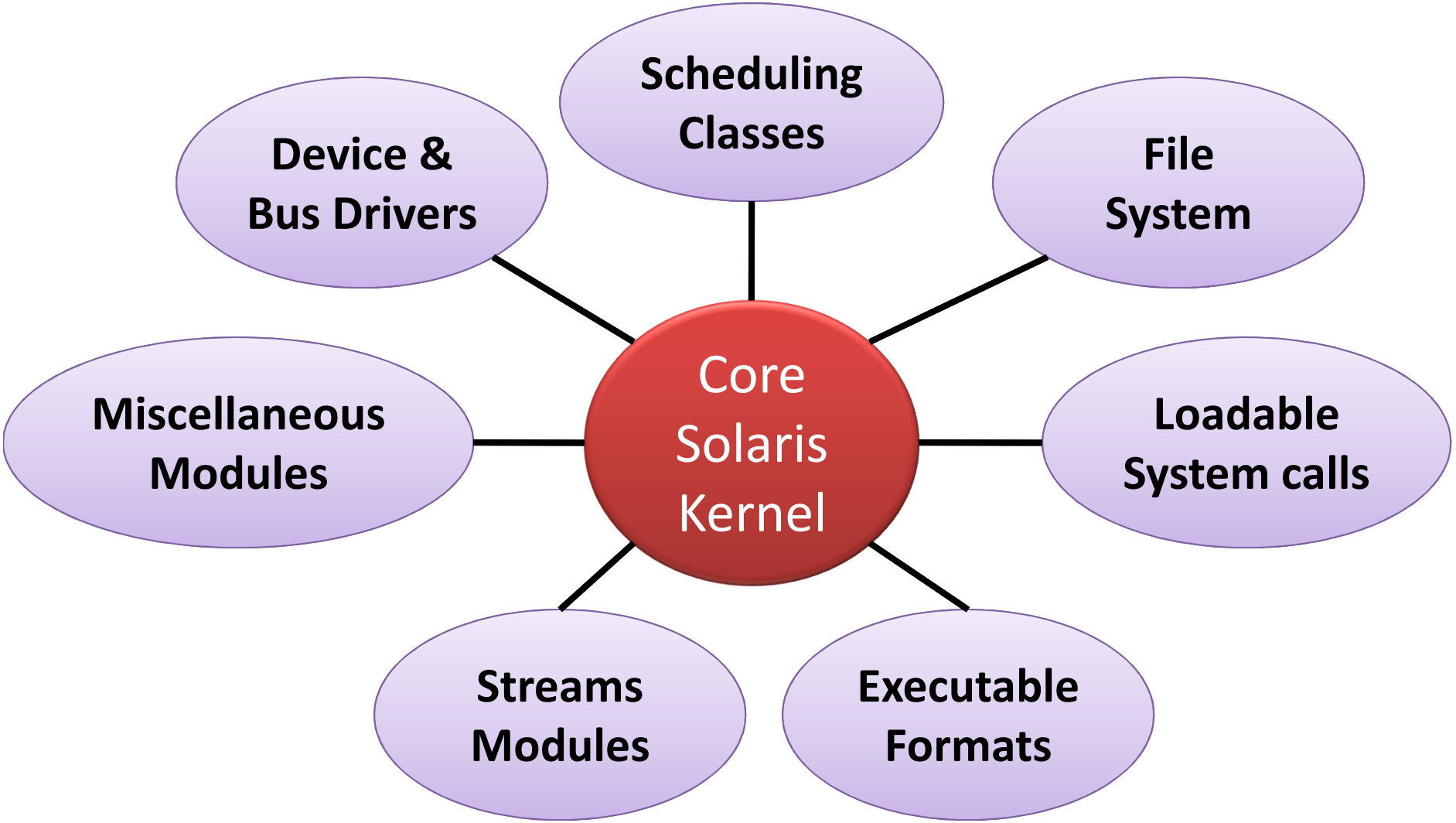
Modules

- Many modern OS implement **loadable kernel modules**
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel
- Overall, similar to layers but with more flexible
  - Linux, Solaris, etc
  - **Linux Kernel Module**

    #lsmod

    you can write simple hello world kernel module and insert to linux kernel

Solaris Modular Approach



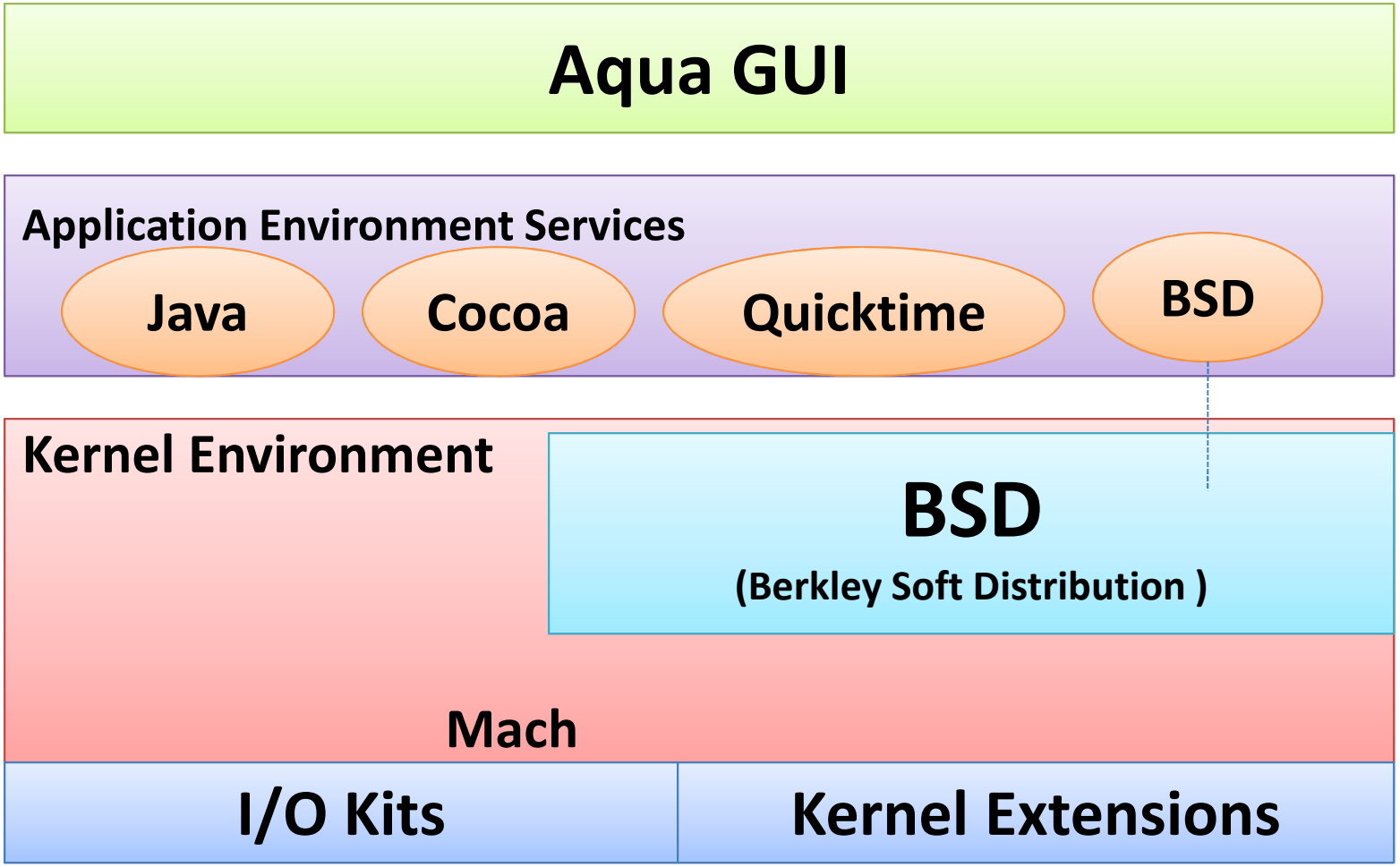
Hybrid Systems

- Most modern OS are actually not one pure model
  - Hybrid combines multiple approaches to address performance, security, usability needs
  - **Linux and Solaris** : kernels in kernel address space, so **monolithic**, plus **modular** for dynamic loading of functionality
  - **Windows:** mostly **monolithic**, plus **microkernel** for different subsystem personalities

Hybrid Systems

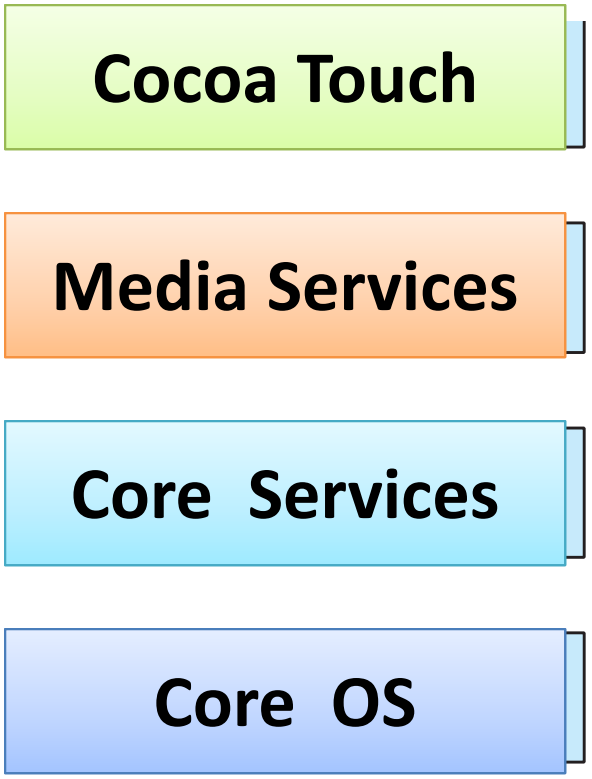
- Apple Mac OS X hybrid, layered, **Aqua** UI plus **Cocoa** programming environment
  - Below is kernel consisting of Mach microkernel and BSD Unix parts, plus I/O kit and dynamically loadable modules (called **kernel extensions**)

Mac OS X Structure



iOS

- Apple mobile OS for ***iPhone***, ***iPad***
  - Structured on Mac OS X, added functionality
  - Does not run OS X applications natively
    - Also runs on different CPU architecture (ARM vs. Intel)
  - **Cocoa Touch** Objective-C API for developing apps
  - **Media services** layer for graphics, audio, video
  - **Core services** provides cloud computing, databases
  - Core operating system, based on Mac OS X kernel



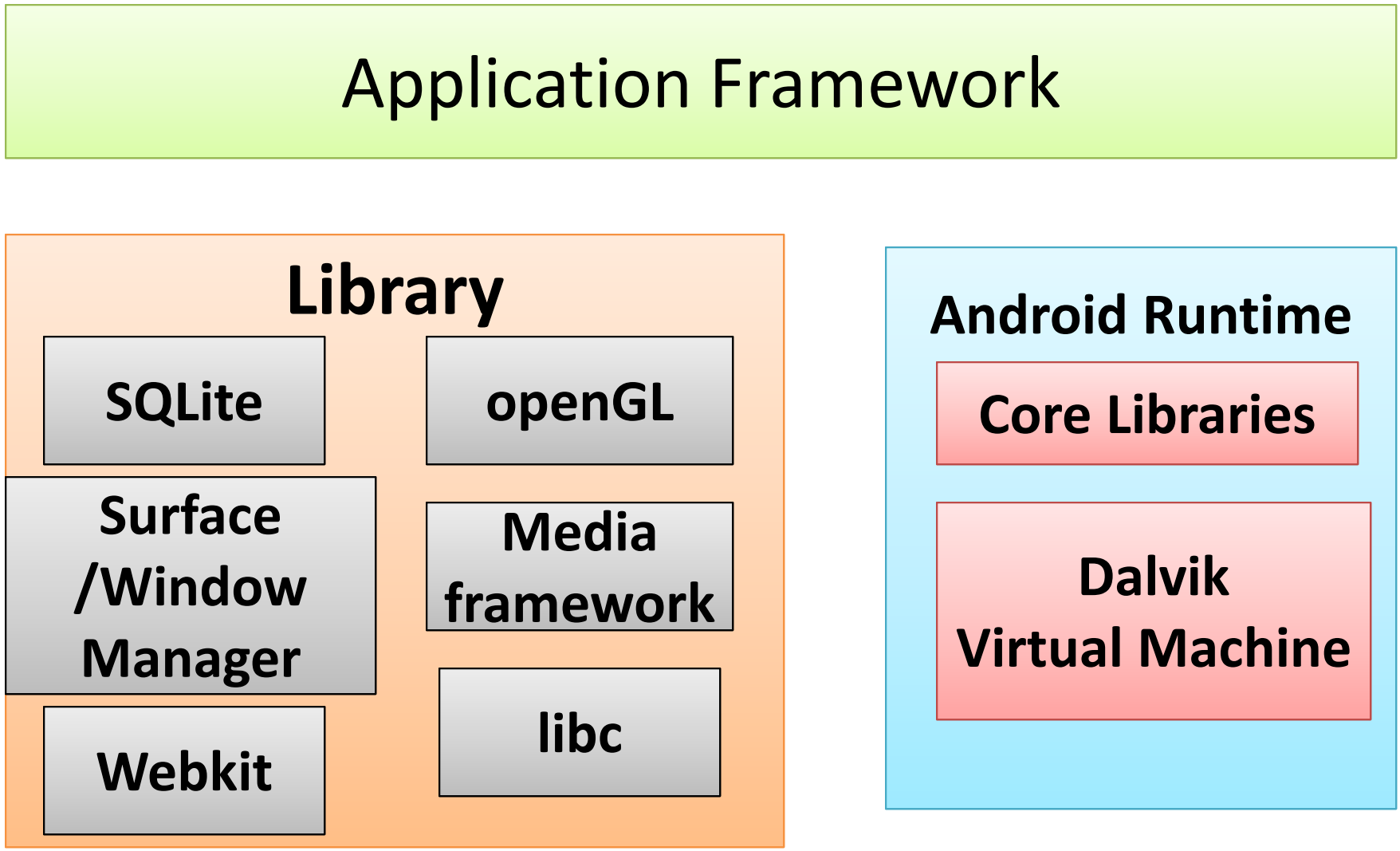
Android

- Developed by Open Handset Alliance (mostly Google)
  - Open Source
  - Similar stack to IOS
- Based on Linux kernel but modified
  - Provides process, memory, device-driver management
  - Adds power management

Android

- Runtime environment includes core set of libraries and **Dalvik virtual machine**
  - Apps developed in Java plus Android API
  - Java class files compiled to Java bytecode then translated to executable than runs in **Dalvik VM**
- Libraries includes
  - frameworks for web browser (webkit),
  - database (SQLite),
  - Multimedia and smaller libc

Android Architecture



Operating System Generation

- OS are designed to run on any of a class of machines
- The system must be configured for each specific computer site
- **SYSGEN** program obtains information concerning the specific configuration of the hardware system
  - Used to build system-specific compiled kernel or system-tuned
  - Can general more efficient code than one general kernel

Another Set of Configuration Example

- Sysgen/HAL configuration
  - **Window in inspecting your hardware**
- $ gcc -dumpspec
- $./configure
- #Pragma in C Program

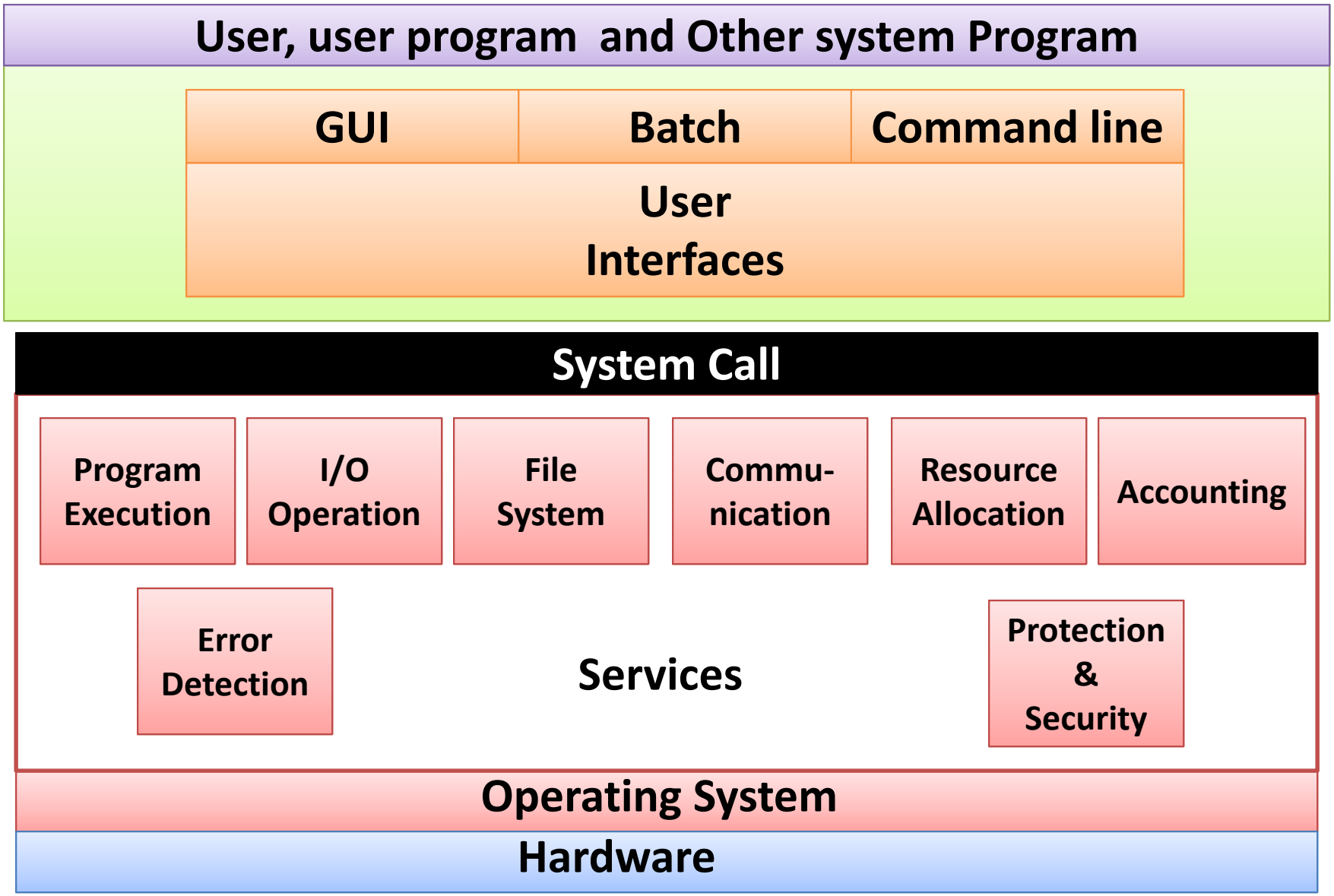
Think

OS : as Service Provider, Manager

User : as Service users

Extending this to much higher level

A View of OS Services



After knowing a bit of OS

Review
of
Different Computing Environment

Computing Environments - Traditional

- Stand-alone general purpose machines
- But blurred as most systems interconnect with others (i.e., the Internet)
- **Portals** provide web access to internal systems
- **Network computers** (**thin clients**) are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming ubiquitous – even home systems use **firewalls** to protect home computers from Internet attacks

Computing Environments - Mobile

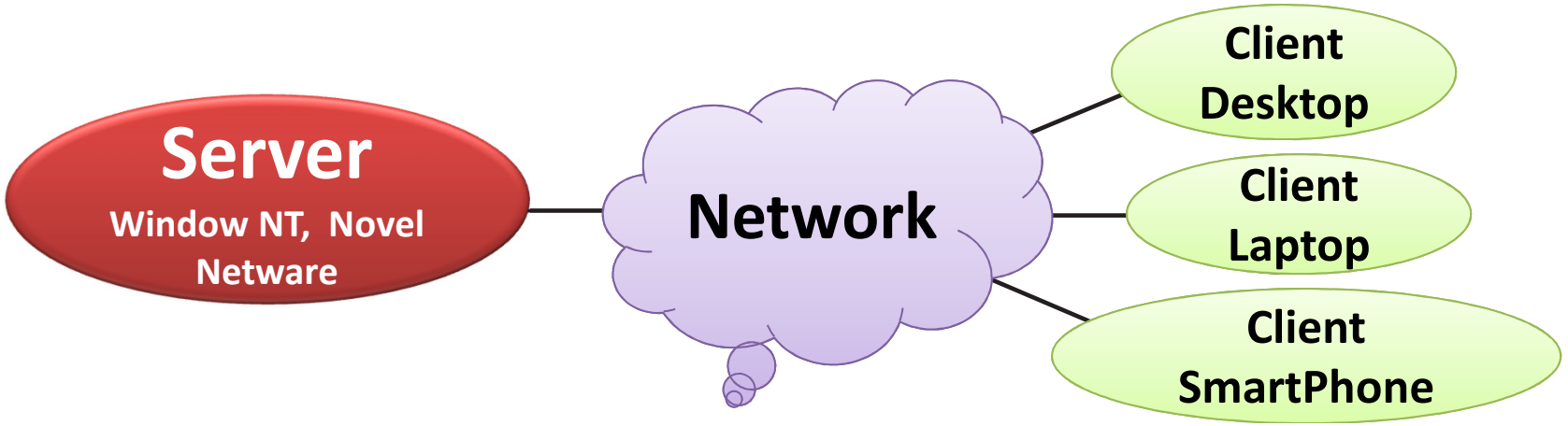
- Handheld smart phones, tablets, etc
- What is the functional difference between them and a “traditional” laptop?
- Extra feature – more OS features
  - GPS, gyroscope, touch screen, etc
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**

Computing Environments – Distributed Computing

- Collection of separate, possibly heterogeneous, systems networked together
  - **Network** is a communications path, **TCP/IP** most common
    - **LAN, Wide Area Network** (**WAN**), **Metropolitan Area Network** (**MAN**), **Personal Area Network** (**PAN**)
- **Network Operating System** provides features between systems across network
  - Communication scheme allows systems to exchange messages
  - Illusion of a single system

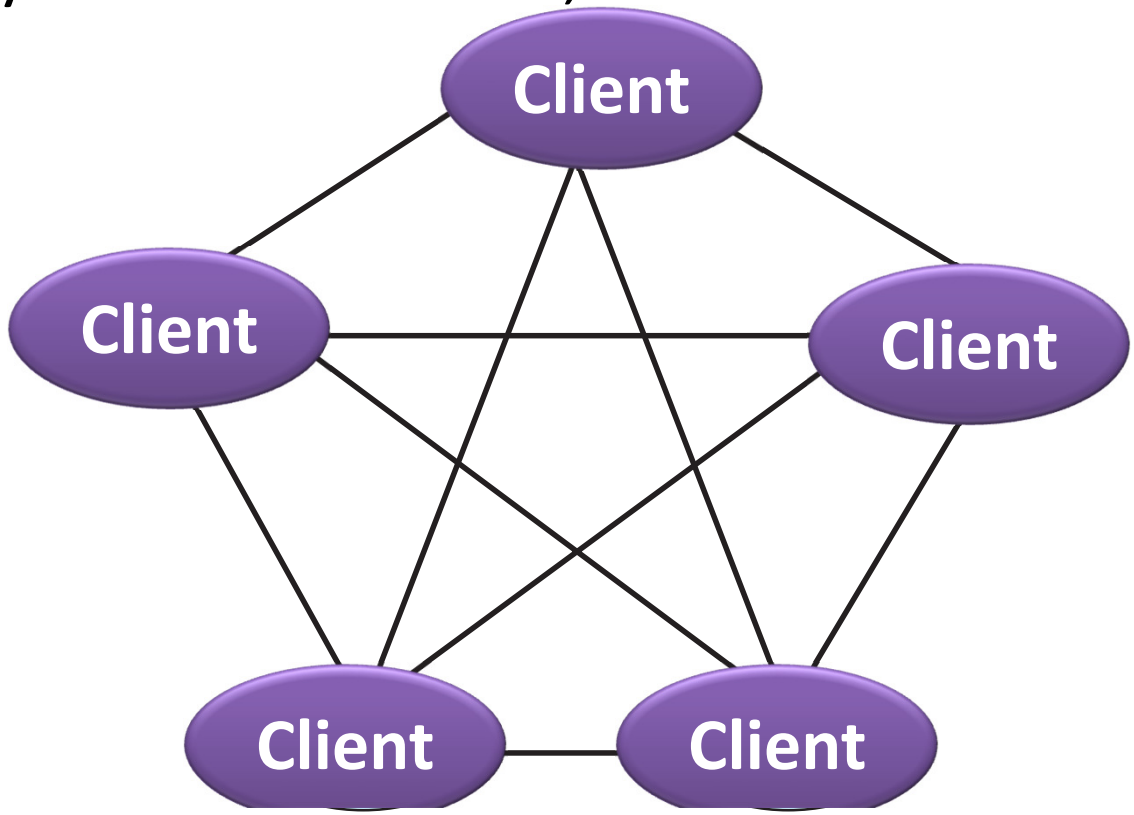
Computing Environments – Client-Server Computing

- Dumb terminals supplanted by smart PCs
- Many systems now **servers**, responding to requests generated by **clients**
  - **Compute-server system** provides an interface to client to request services (i.e., database)
  - **File-server system** provides interface for clients to store and retrieve files



Comp. Envts : Peer-to-Peer

- Another model of distributed system
- P2P does not distinguish clients and servers
  - **Instead all nodes are considered peers**
  - May each act as client, server or both



Comp. Envts : Peer-to-Peer

- Node must join P2P network
  - Registers its service with central lookup service on network, or
  - Broadcast request for service and respond to requests for service via ***discovery protocol***
- Examples include
  - **Napster** ***(music-focused online services)***
  - **Gnutella (*millions for peer-to-peer file sharing*)**
  - **Voice over IP (VoIP) such as Skype**

Comp. Envnt. - Virtualization

- Allows operating systems to run applications within other OSes
  - Vast and growing industry
- **Emulation** used when source CPU type different from target type (i.e. PowerPC to Intel x86)
  - Generally slowest method
  - When computer language not compiled to native code – **Interpretation**

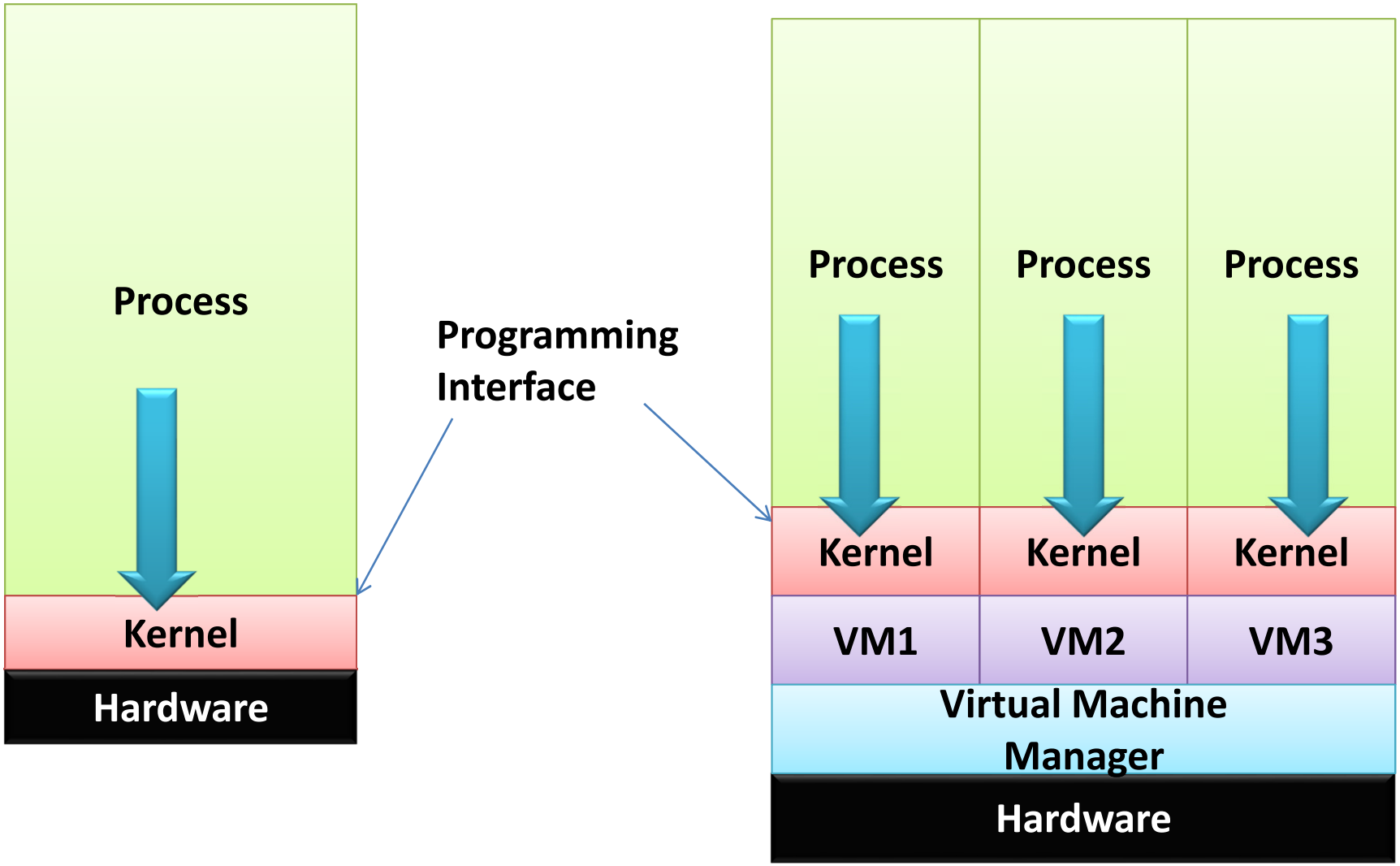
Comp. Envnt. - Virtualization

- **Virtualization** – OS natively compiled for CPU, running **guest** OSes also natively compiled
  - Consider VMware running WinXP guests, each running applications, all on native WinXP **host** OS
  - **VMM** (virtual machine Manager) provides virtualization services

Comp. Envnt. - Virtualization

- Use cases involve laptops and desktops running multiple OSes for exploration or compatibility
  - Mac OS X host, Windows as a guest
  - Developing apps for multiple OSes without having multiple systems
  - QA testing apps without having multiple systems
  - Executing and managing compute environments within data centers
- VMM can run natively, in which case they are also the host
  - There is no general purpose host then (VMware ESX and Citrix XenServer)

Comp. Envnt. - Virtualization



Comp. Envnts: Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization because it uses virtualization as the base for it functionality.
- Amazon **EC2**
  - Has thousands of servers,
  - Millions of virtual machines
  - Petabytes of storage available across the Internet,
  - Pay based on usage

Comp. Envnts: Real Time Embedded System

- Real-time embedded systems most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS, **real-time OS**
  - Use expanding
- Many other special computing environments as well
  - Some have OSes, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
  - Processing ***must*** be done within constraint
  - Correct operation only if constraints met

Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source**
- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has "copyleft" **GNU Public License (GPL)**
- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and many more
- Can use VMM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - http://www.virtualbox.com)
  - Use to run guest operating systems for exploration

Thanks