QNX Overview

Based on material from:
Sebastien Marineau-Mes & Colin Burgess
Jason Clarke, QNX Field Application Engineer

History

QNX SOFTWARE SYSTEMS

- Developed in early ‘80s for the Intel 8088.
- Initially used in “larger” non-embedded projects (44k kernel)
- Migrated to POSIX model / compatibility
- Added Photon GUI
- Rewritten to support SMP (Neutrino)
- Member of Eclipse Foundation (Momentics)
- Sold to Harman International Industries for application in automotive systems
- Purchased by Research in Motion (RIM)
  - Blackberry Playbook Tablet
- Ported to large number of platforms
  - PowerPC, x86, MIPS, SH-4, ARM, StrongARM, XScale

A History of Software Innovation

QNX SOFTWARE SYSTEMS

1980: First commercially available microkernel OS

1982: First RTOS to support a hard disk on a PC

1985: First **memory-protected** RTOS

1990: First **POSIX-certified** RTOS

1992: First RTOS to offer built-in **fault-tolerant networking**

1994: US patent for **scalable microkernel windowing system**

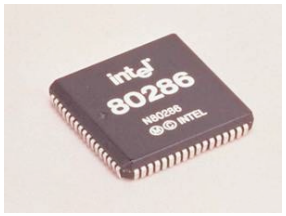
1997: First RTOS to support **symmetric multiprocessing (SMP)**

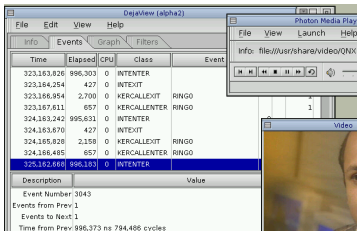
2002: First RTOS vendor to deliver **Eclipse-based IDE**

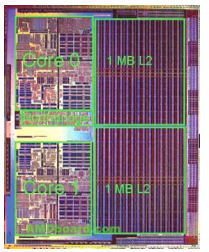
2005: First to offer **"bound" multi-processing**

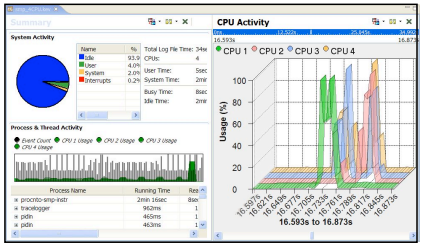
2007: Introduces hybrid software model and **opens source code**

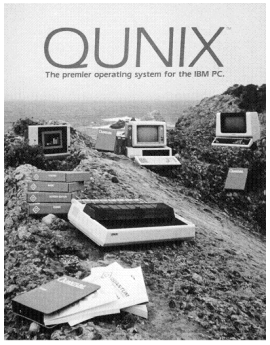
1980 | QNX2 | 1985 | 1990 | QNX4 | 1995 | 2000 | QNX6 | 2005











"QUNIX" Brochure, 1982

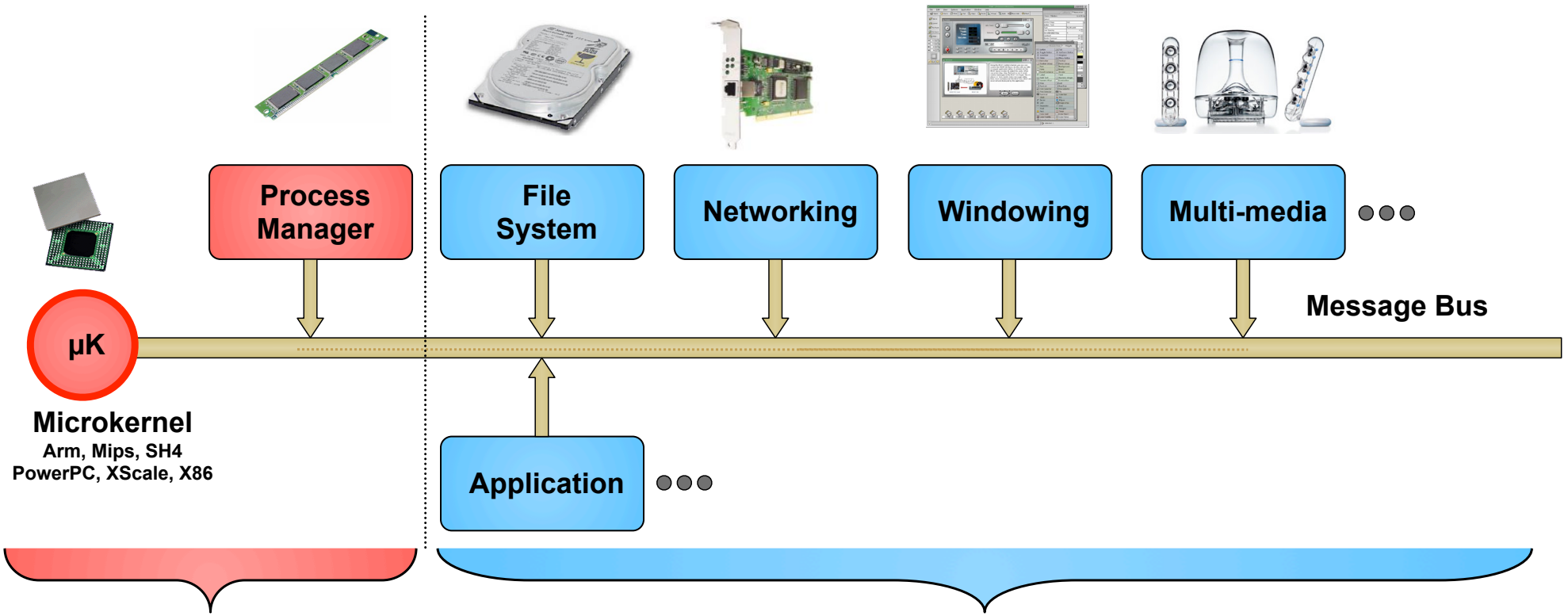
Features

QNX SOFTWARE SYSTEMS

- Micro-kernel Architecture
  - CPU Scheduling
  - Inter-process Communication
  - Interrupt Redirection
  - Timers
- Protected User Process Space
  - Process Lifecycle
  - Memory Management
  - Device Drivers
- Configurable for scalability
- Messaging-based architecture

Microkernel Architecture

QNX SOFTWARE SYSTEMS



Microkernel
+
Process Manager
are the <u>only</u> trusted components

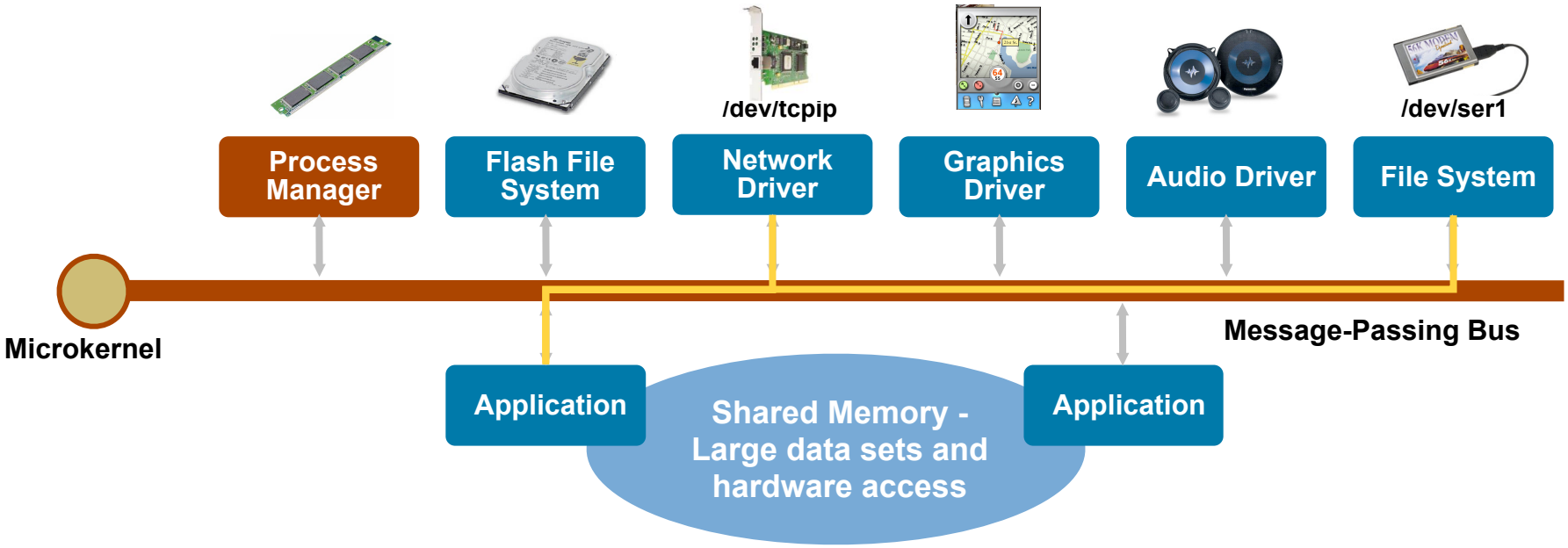
Applications <u>and Drivers</u>

- Are processes which plug into a message bus
- Reside in their own memory-protected address space
- Have a well defined message interface
- Cannot corrupt other software components
- Can be started, stopped and upgraded on the fly

Inter-Process Communication

QNX SOFTWARE SYSTEMS

- Processes communicate by sending messages



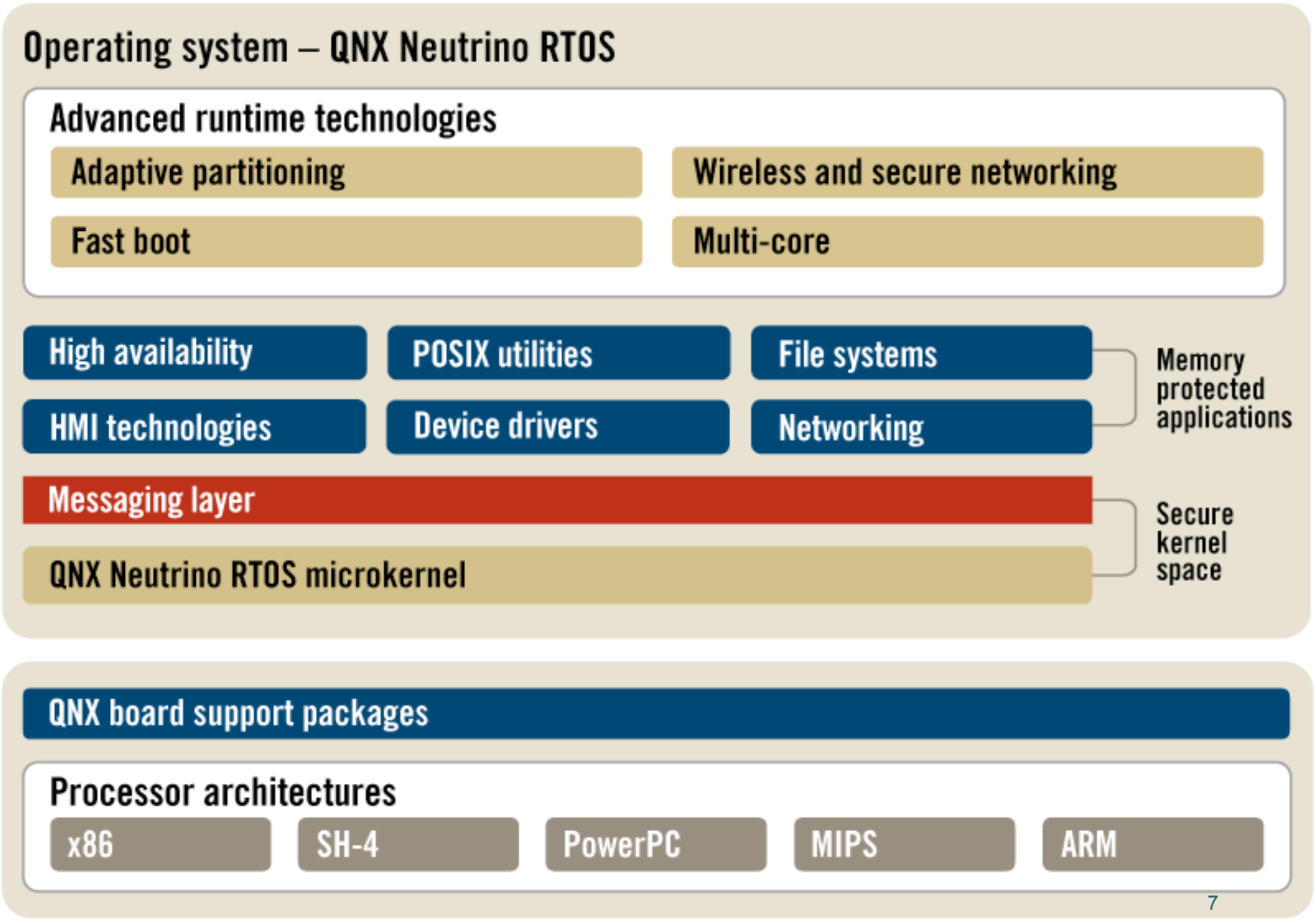
- Using messages cleanly decouples processes
- POSIX calls built on messages

  fd = open("/dev/tcpip", ,,,)
  read, write, stat, devctl, …
  close

- Other POSIX calls as well
  - realtime signals
  - pipes and POSIX mqueues
  - mutexs, condvars, semaphores
  - barriers, sleepon
  - reader/writer locks

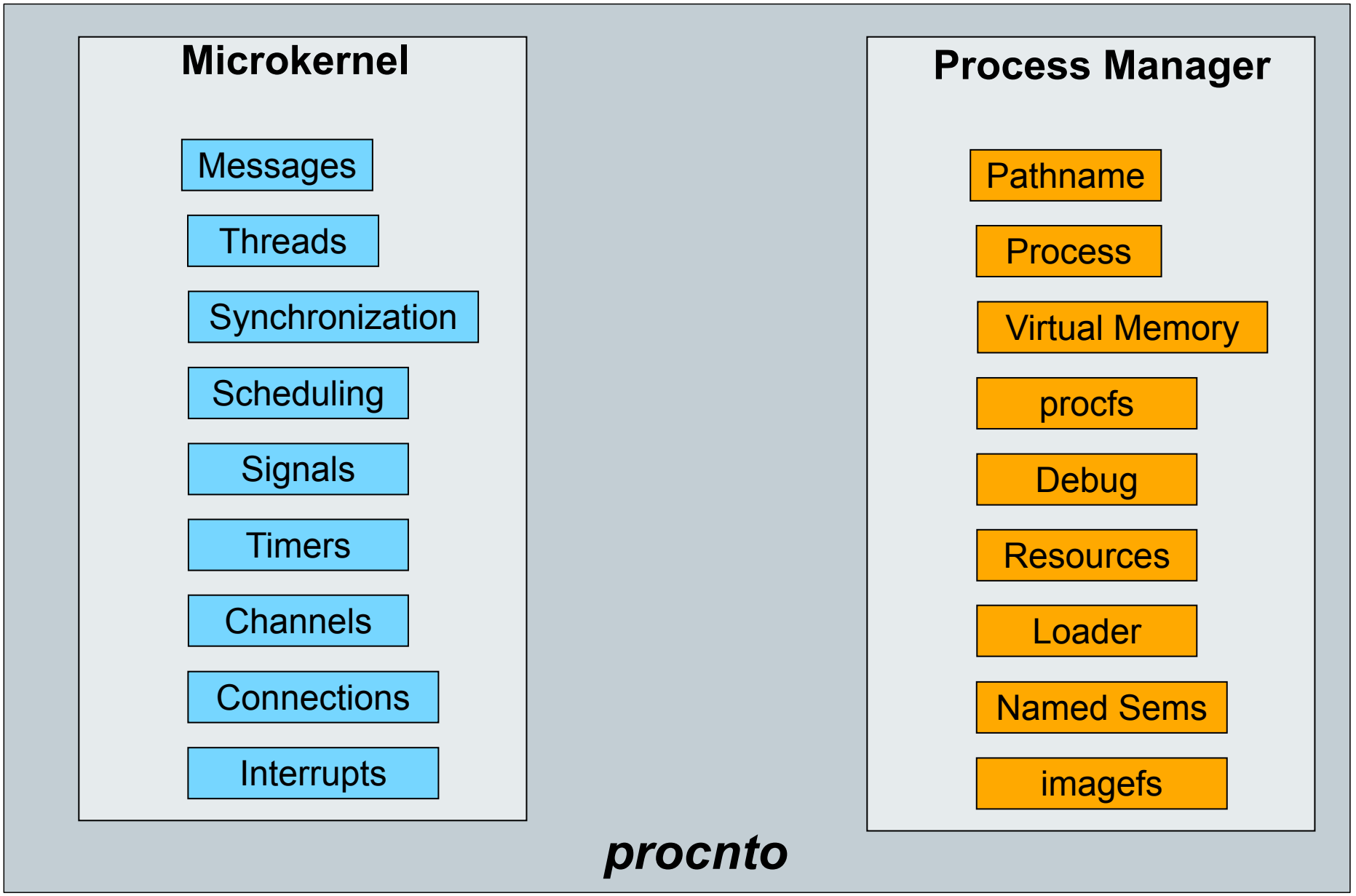
Overview

QNX SOFTWARE SYSTEMS

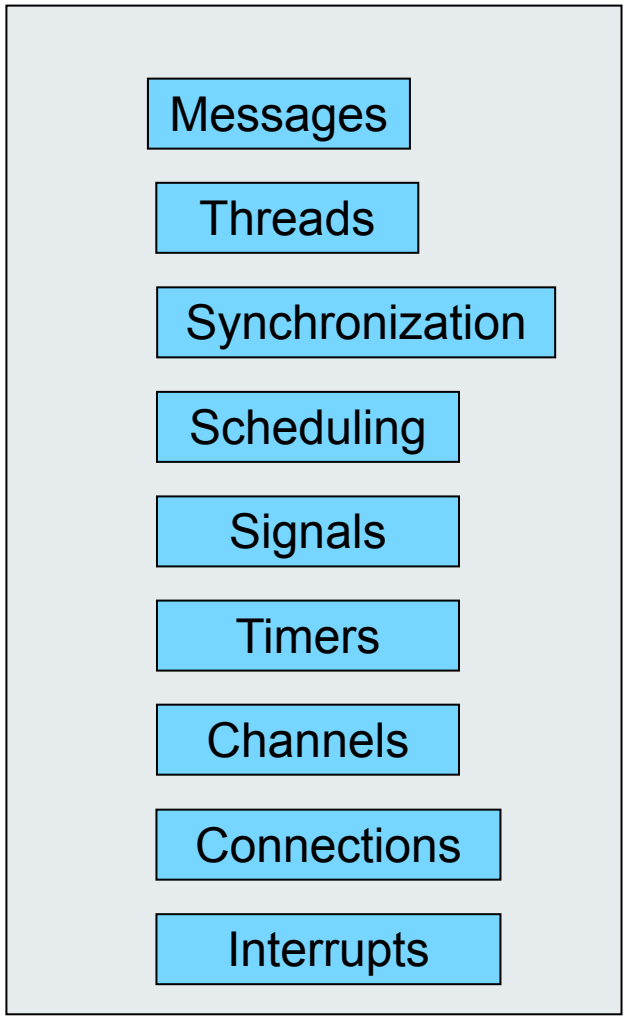


Separation of Duties – Process Manager vs. MicroKernel

QNX SOFTWARE SYSTEMS



Microkernel Services

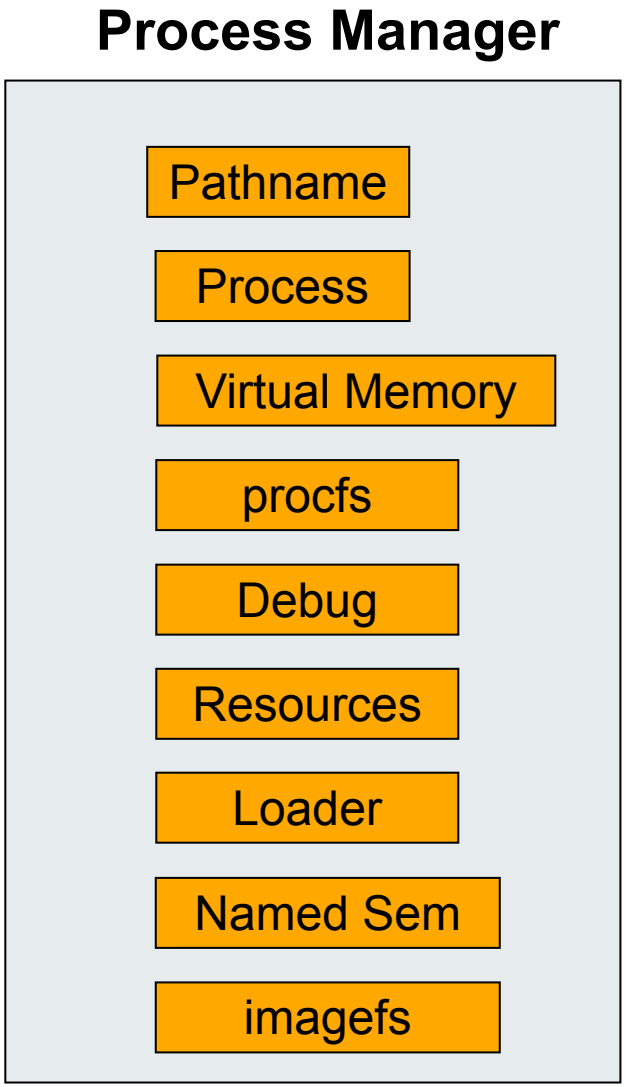


- **Simple pre-emptable operations**
- **Provides basic system services**
  - **Implements much of the POSIX thread and realtime standard**
  - **Interrupt and exception redirection**
  - **IPC primitives**
- **Most of the microkernel is hardware independent**
  - **CPU-dependant layer for low-level cpu interfaces**
  - **CPU-specific optimized routines**
- **Only pieces of code that runs with full system privilege**
- **Microkernel does not run "on its own"**
  - **Only reacts to external events: system calls, interrupts, exceptions**

Process Manager Services

QNX SOFTWARE SYSTEMS

- Implements long, complex operations and services
  - Ex: Process creation and memory management
- Is a multi-threaded process that is scheduled at normal priority
  - Competes for CPU with all other threads in the system
- Message driven server
- More on this later



Process Manager

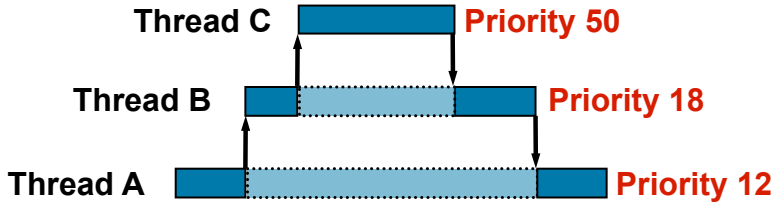
QNX SOFTWARE SYSTEMS

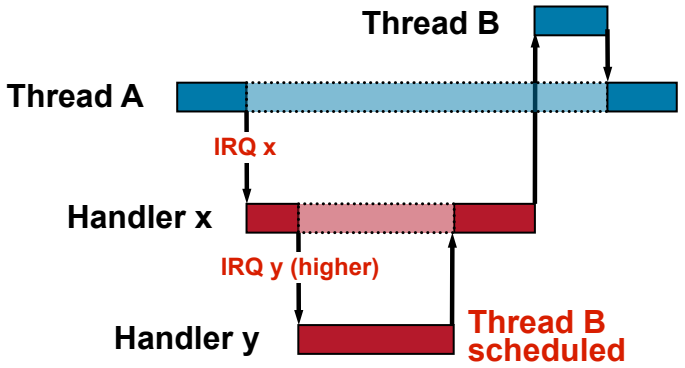
- First process in system
  - Created by kernel (init_objects)
- Provides core services to other processes
- Multi-threaded Process
  - First <ncpus> threads are IDLE threads
  - Additional threads are threadpool worker threads
- Message driven server
- Actually a collection of (almost) independent servers
- 4 message handlers
- 11(!) resource managers
  - These resource managers are actually mini filesystems.

Hard Realtime Performance

QNX SOFTWARE SYSTEMS

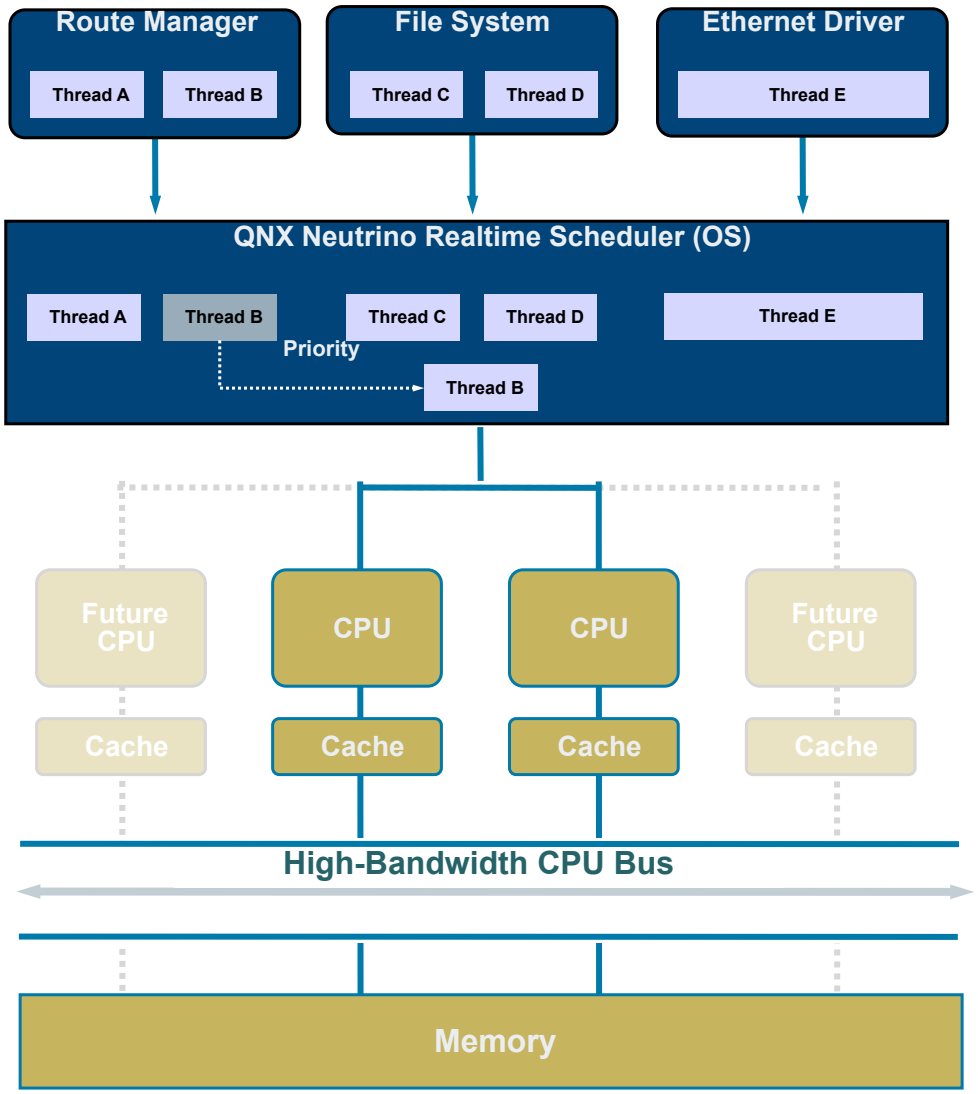
- **Multiple concurrent scheduling algorithms**
  - FIFO, Round Robin, Sporadic
- **Prioritized pre-emptable threads**
  - 256 priority levels
  - Fully pre-emptable and deterministic kernel
- **Prioritized and nested interrupts**
  - Interrupt handlers can schedule a user thread or run custom interrupt code





Symmetric Multiprocessing

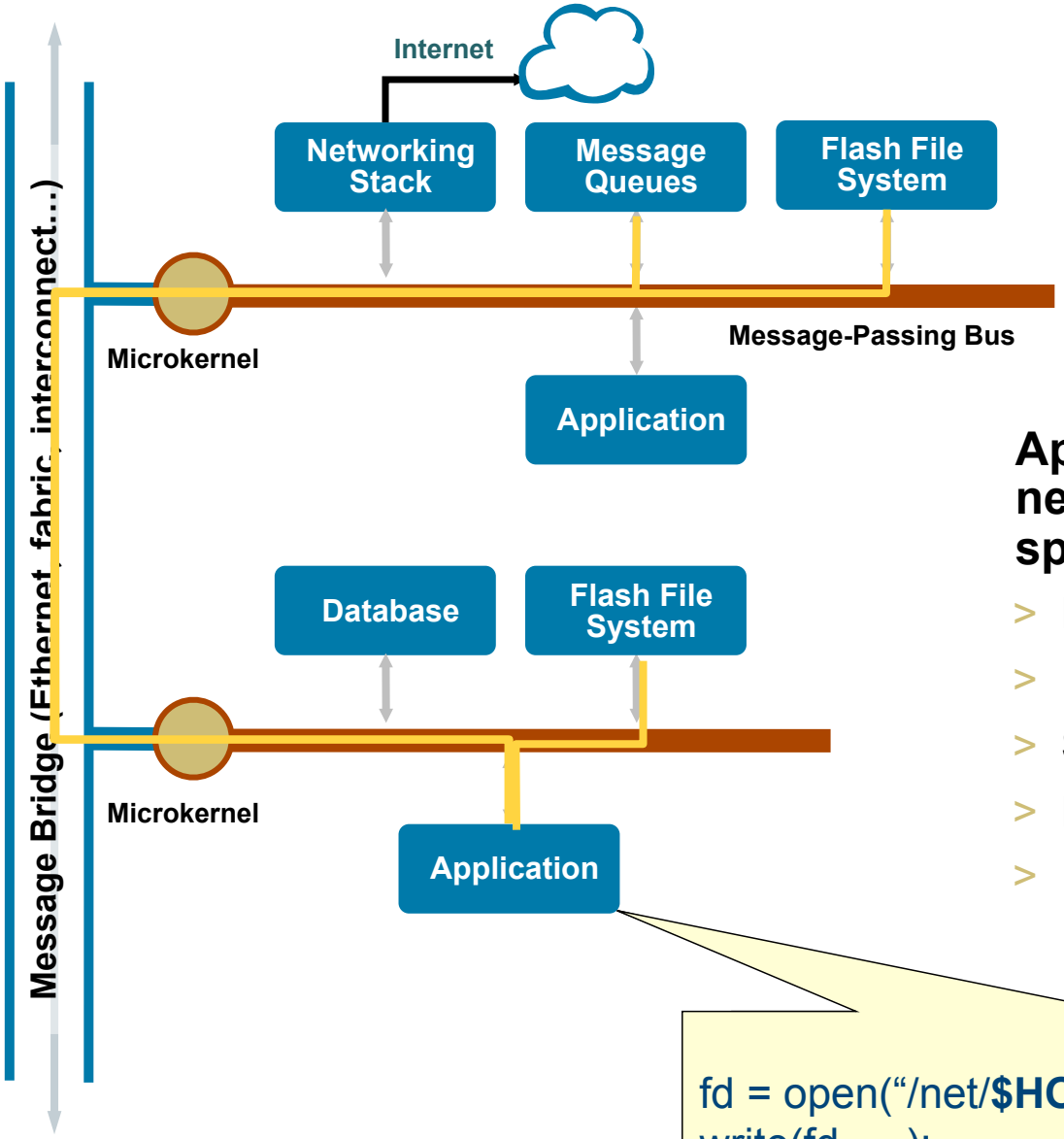
QNX SOFTWARE SYSTEMS



- Multiple processors sharing common hardware
  - Common memory bus and address space
  - Access to all peripheral devices and interrupts
  - OS manages tasks running on processors – true concurrency
- Transparent to application programs
- No application software changes needed
- Automatic thread (~) scheduling across all CPUs

Transparent Distributed Processing

QNX SOFTWARE SYSTEMS



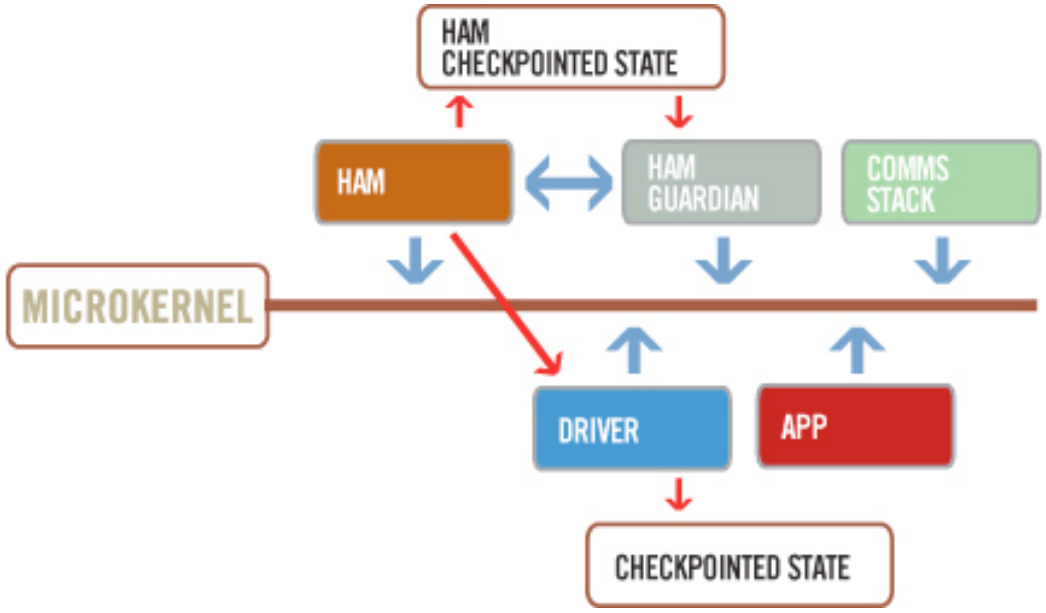
Applications/servers can be network-distributed without special code

- Message queues
- File systems
- Services
- Databases
- …

```
fd = open("/net/$HOSTNAME/etc/log_file.dat",…);
write(fd, …);
```

Critical Process Monitor

QNX SOFTWARE SYSTEMS



- Critical Process Monitor (HAM) monitors components and sends notification of component failure
- Heartbeat services detect component 'hang'
- Core file on crash can be created for debugging and analysis
- Recovery from crash can be:
  - Controlled shutdown or system restart
  - ***Restart of only the failed subsystem (driver)***

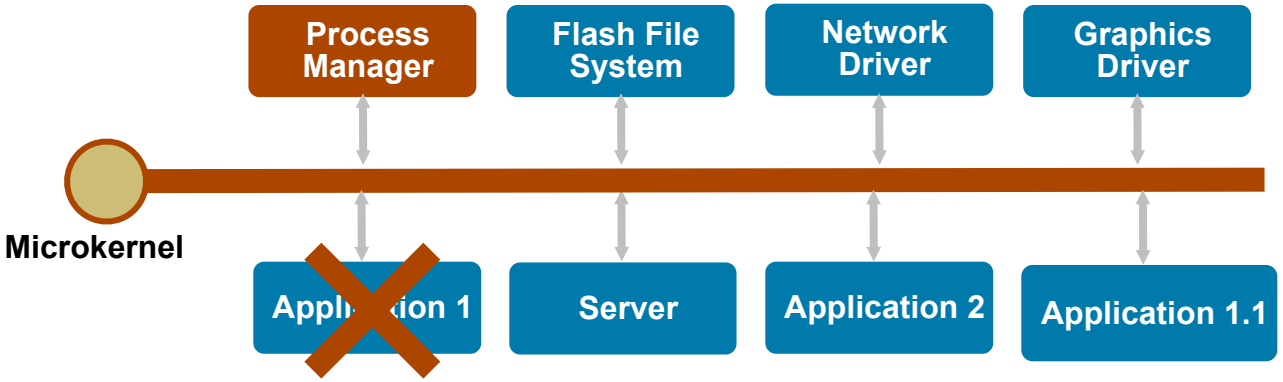
Flash Layout (8260ADS)

QNX SOFTWARE SYSTEMS

- **Boot Image**
  - Contains Kernel
  - Requires only Flash Filesystem to be in Image
- **Flash Filesystem**
  - Fault Tolerant POSIX Compliant Filesystem
  - Once Filesystem is Loaded Everything Else Can be Loaded from the Filesystem, Even Drivers
- **IPL (Initial Program Loader)**
  - Sets Up Board and Loads Boot Image
  - Sits at Reset Vector

System Upgrade

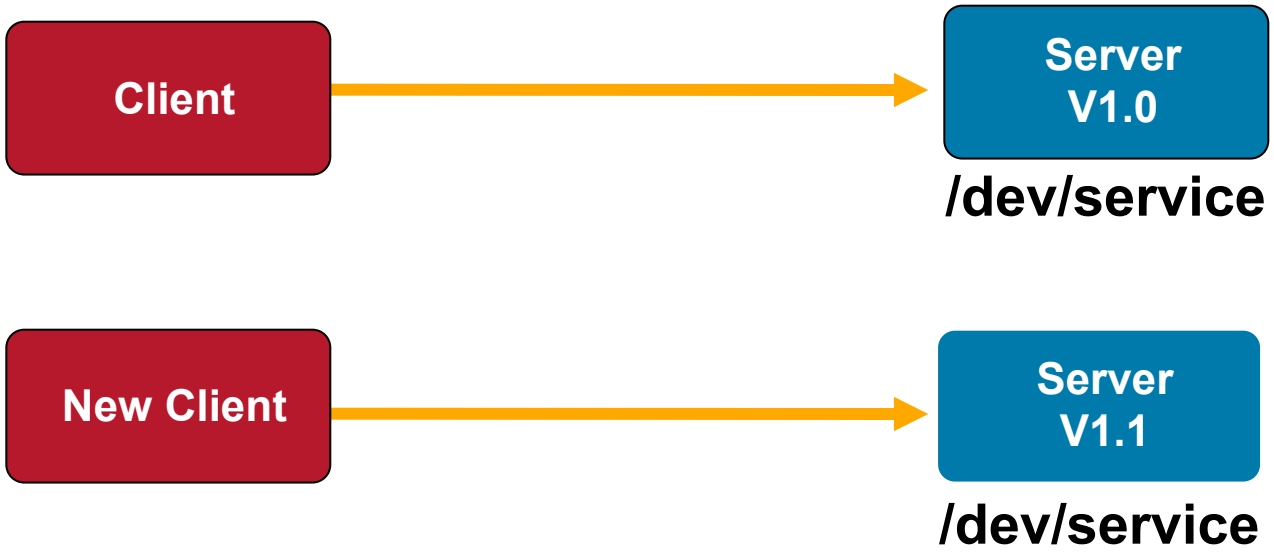
QNX SOFTWARE SYSTEMS



- **Add New Features or Processes on the Fly**
  - Download New Binary into Filesystem or Ram
  - Load New Binary into RAM
- **Replace Existing Processes Without Reboot**
  - Download New Binary to Filesystem
  - Remove Process Running in RAM
  - Load New Binary From the Filesystem

Zero Down Time Upgrade

QNX SOFTWARE SYSTEMS



- New version of the server attaches to the same name
- New clients connect to new server
- Old server exits when all old clients are gone