F9: A Secure and Efficient Microkernel Built for Deeply Embedded Systems

Jim Huang (黃敬群) <jserv.tw@gmail.com>

Dec 9, 2013 / CCU Taiwan

Aug 28, 2013 / JuluOSDev

Aug 3, 2013 / COSCUP

Rights to copy

© Copyright 2013 **0xlab**
http://0xlab.org/

Corrections, suggestions, contributions and translations are welcome!

Latest update: Dec 9, 2013

creative commons
COMMONS DEED

Attribution – ShareAlike 3.0

You are free

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions

BY: **Attribution**. You must give the original author credit.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- or any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

License text: http://creativecommons.org/licenses/by-sa/3.0/legalcode

Goals of This Presentation

- Introduce F9 microkernel, new open source implementation built from scratch, which deploys modern kernel techniques, derived from L4 microkernel designs, to deeply embedded devices.

  `https://github.com/f9micro`
- Characteristics of F9 microkernel
  - Efficiency: performance + power consumption
  - Security: memory protection + isolated execution
  - Flexible development environment

Agenda

- Target: Deeply embedded devices
- Microkernel overview
- Characteristics of F9 Microkernel

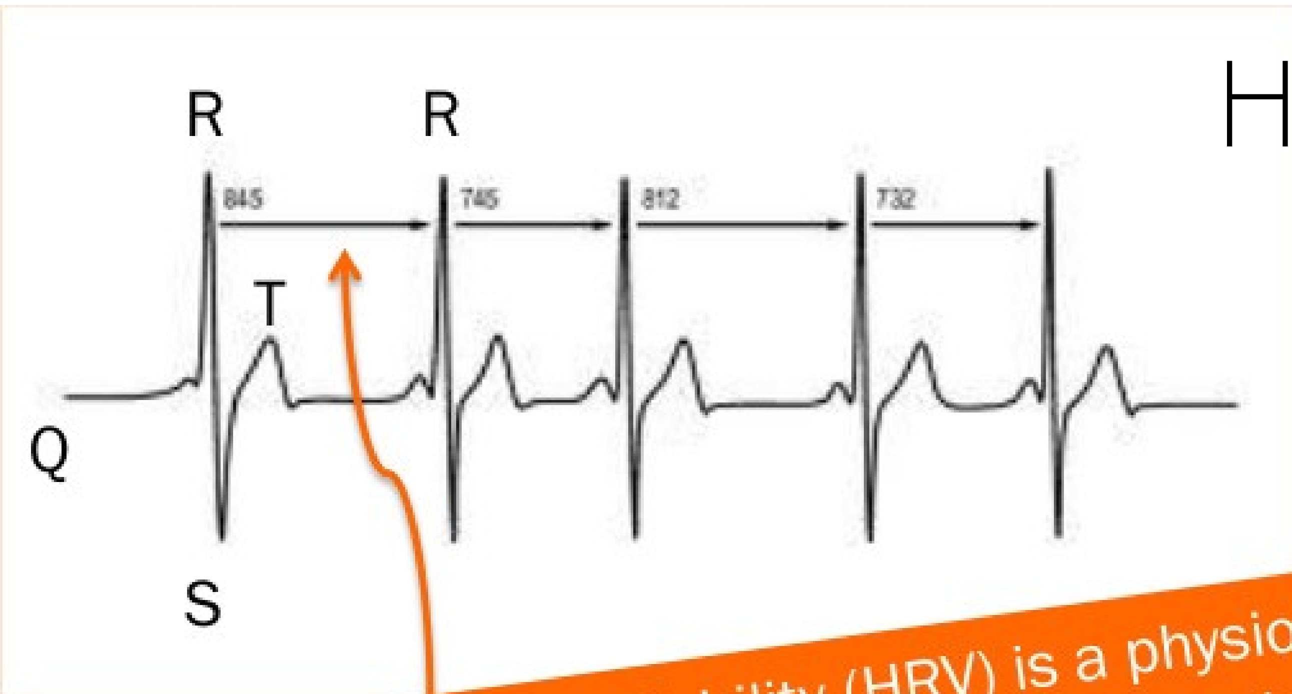
Target: Deeply Embedded Devices

Deeply Embedded Devices

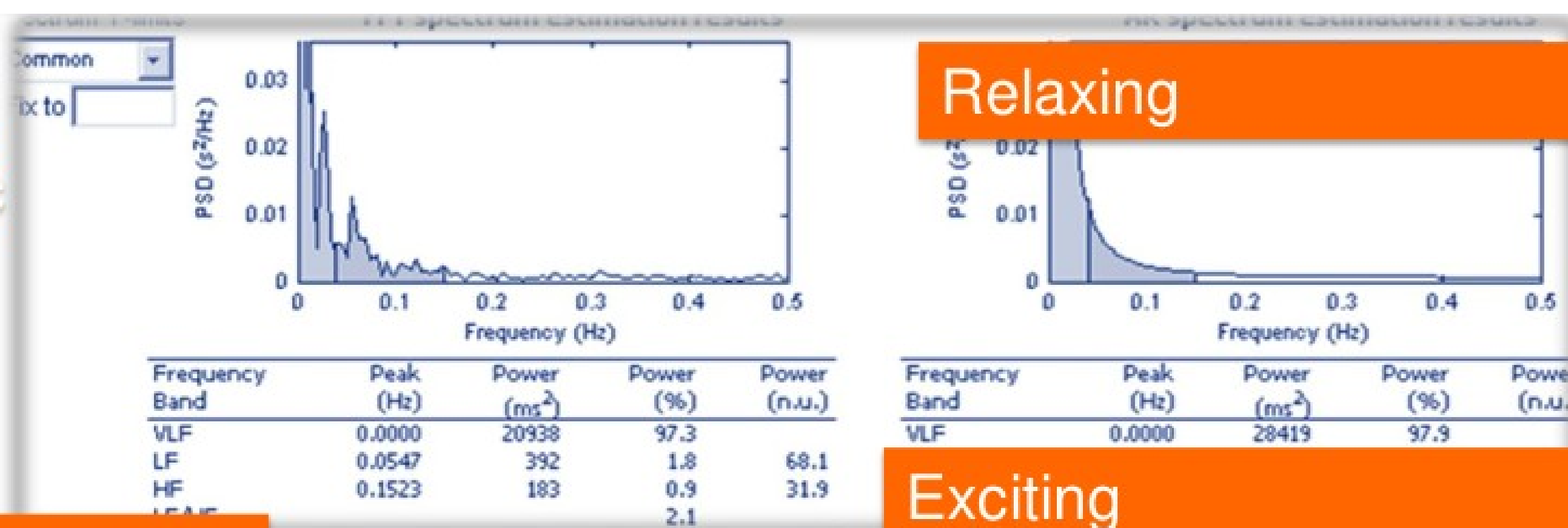
- Power awareness; solid and limited applications
- Multi-tasking or cooperative scheduling is still required
- IoT (Internet of Things) is the specialized derivative with networking facility
- Communication capability is built-in for some products
- Example: AIRO wristband (health tracker)

  `http://www.weweartech.com/amazing-new-uses-smart-watches/`

HRV Knows You



Heart rate variability (HRV) is a physiological phenomenon where the time interval between heart beats varies



Sympathetic

Stress

Exercise Strength

Relaxing

Exciting

We built in-house OS for products and releases the basic part as an open source effort

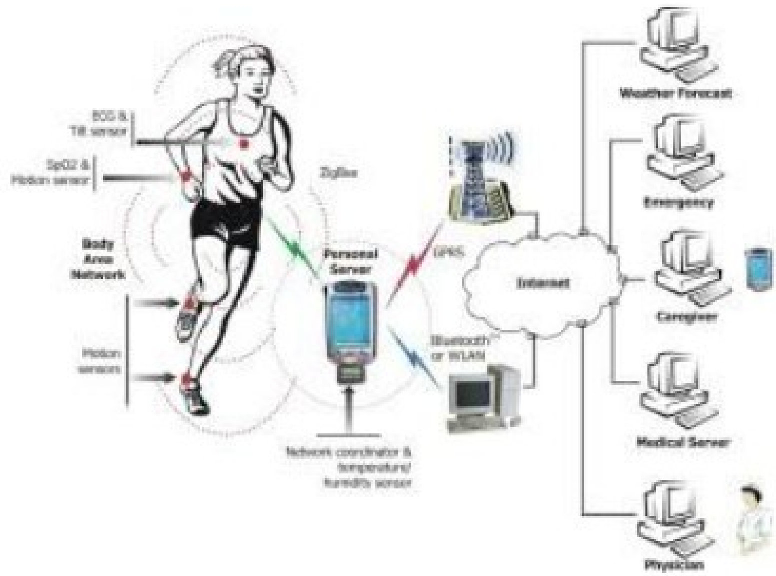


Photo: Philips

(invisible) Medical devices make sense in our life.

:: home-care :: advance warning :: security

Microkernel Overview

Microkernel Concepts

- Minimal kernel and hardware enforce separation
- Only kernel runs in CPU privileged mode
- Components are user!level processes
- No restrictions on component software
- Reuse of legacy software

principle of least privilege (POLA)

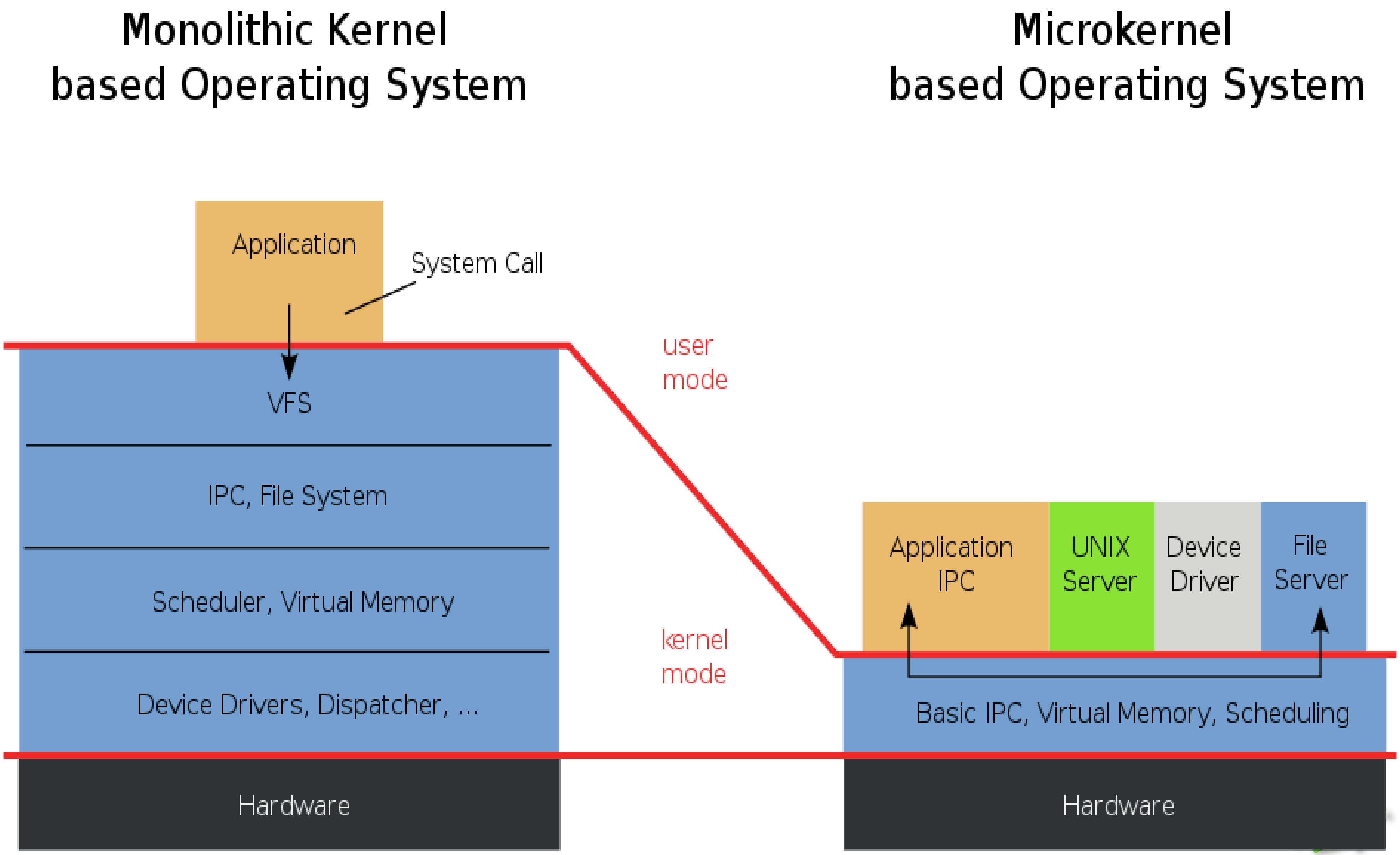
POSIX	POLA
operations allowed by default	nothing allowed by default
some limited restrictions apply	every right must be granted
ambient authority	explicit authority

A capability is a communicable, unforgeable token of authority. It refers to a value that references an object along with an associated set of access rights. A user program on a capability-based operating system must use a capability to access an object.

Case Study: Bugs inside big kernels

- Drivers cause 85% of Windows XP crashes.
  - Michael M. Swift, Brian N. Bershad, Henry M. Levy: “Improving the Reliability of Commodity Operating Systems”, SOSP 2003
- Error rate in Linux drivers is 3x (maximum: 10x)
  - Andy Chou, Junfeng Yang, Benjamin Chelf, Seth Hallem, Dawson R. Engler: “An Empirical Study of Operating System Errors”, SOSP 2001
- Causes for driver bugs
  - 23% programming error
  - 38% mismatch regarding device specification
  - 39% OS-driver-interface misconceptions
  - Leonid Ryzhyk, Peter Chubb, Ihor Kuz and Gernot Heiser: “Dingo: Taming device drivers”, EuroSys 2009

Monolithic Kernel vs. Microkernel



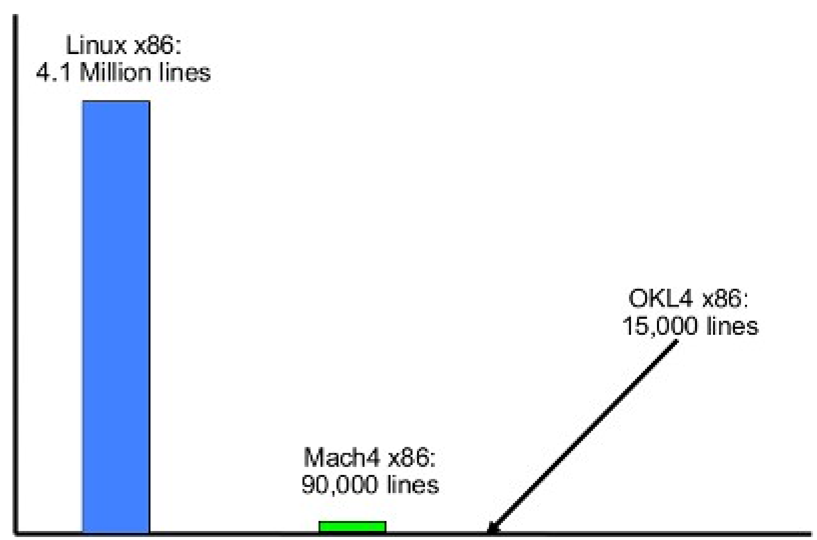
Microkernel Philosophy

A concept is tolerated inside the microkernel only if moving it outside the kernel, i.e., permitting competing implementations would prevent the implementation of the systems' required functionality.

– Jochen Liedtke

Microkernel

- Minimalist approach
  - IPC, virtual memory, thread scheduling
- Put the rest into user space
  - Device drivers, networking, file system, user interface
- Disadvantages
  - Lots of system calls and context switches
- Examples: Mach, L4, QNX, MINIX, IBM K42

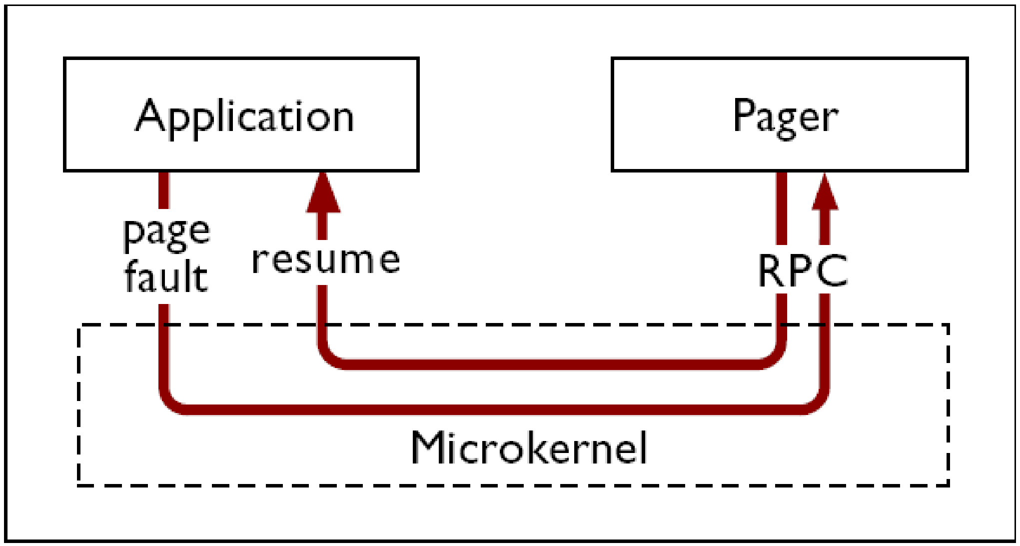


3 Generations of Microkernel

- Mach (1985-1994)
  - replace pipes with IPC (more general)
  - improved stability (vs monolithic kernels)
  - poor performance
- L3 & L4 (1990-2001)
  - order of magnitude improvement in IPC performance
    - written in assembly, sacrificed CPU portability
    - only synchronus IPC (build async on top of sync)
  - very small kernel: more functions moved to userspace
- seL4, Fiasco.OC, Coyotos, NOVA (2000-)
  - platform independence
  - verification, security, multiple CPUs, etc.

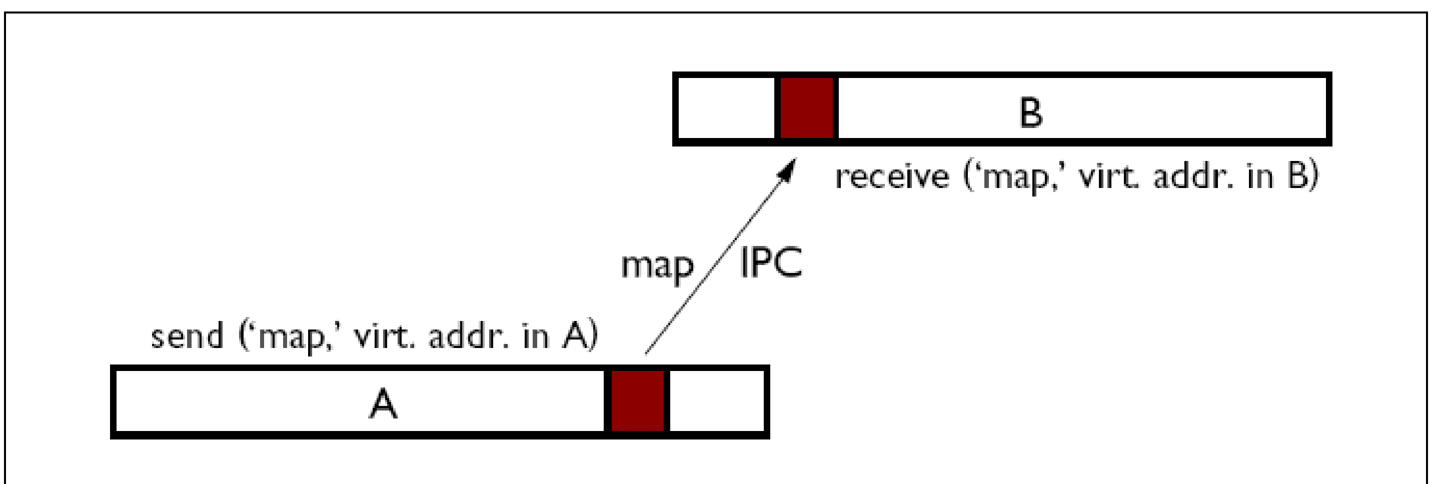
Microkernel Paging

- Microkernel forwards page fault to a pager server.
- Kernel or server decides which pages need to be written to disk in low memory situations.
- Pager server handles writing pages to disk.



Recursive Address Space

- Initial address space controlled by first process.
  - Controls all available memory.
  - Other address spaces empty at boot.
- Other processes obtain memory pages from first or from their other processes that got pages from first.
- Why is memory manager flexibility useful?
  - Different applications: real-time, multimedia, disk cache.



Grant
Map
Flush

Characteristics of F9 Microkernel

4 Aspects

- BSD Licensing (two-clause), suitable for both research and commercial usage.
- Efficiency
  - performance: fast IPC and well-structured designs
  - energy-saving: tickless scheduling, adaptive power management
- Security
  - memory protection: MPU guarded
  - Isolated execution: L4 based, capabilities model
- Flexible development
  - Kprobes
  - profile-directed optimizations

Why are current systems unreliable?

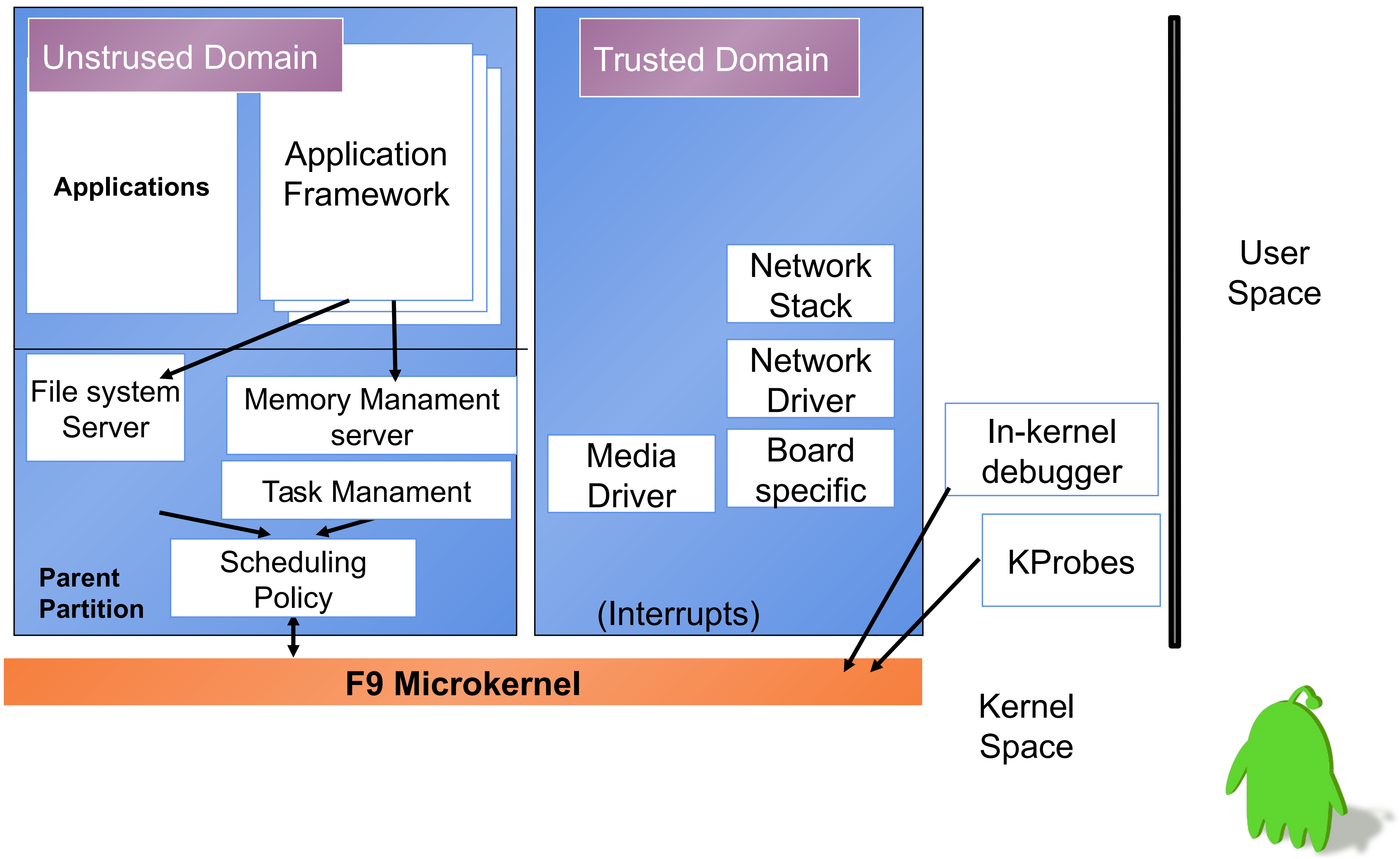
- Systems are huge
  - No single person can understand the whole system

    > F9 Microkernel has only 2K LoC of portable C
- Bug fixes usually introduce new bugs.

    > F9 introduces execution domains and on-the-fly patches
- Poor fault isolation
  - No isolation between system components
  - OS contains hundreds of procedures linked together as a single binary program running on the kernel mode.

    > F9 is built from scratch and well-engineered for isolation

F9 Microkernel Architecture



Principles

- F9 follows the fundamental principles of L4 microkernels
  – implements address spaces, thread management, and IPC only in the privileged kernel.
- Designed and customized for ARM Cortex-M, supporting NVIC (Nested Vectored Interrupt Controller), Bit Banding, MPU (Memory Protection Unit)

Thread

- Each thread has its own TCB (Thread Control Block) and addressed by its global id.
- Also dispatcher is responsible for switching contexts. Threads with the same priority are executed in a round-robin fashion.

Memory Management

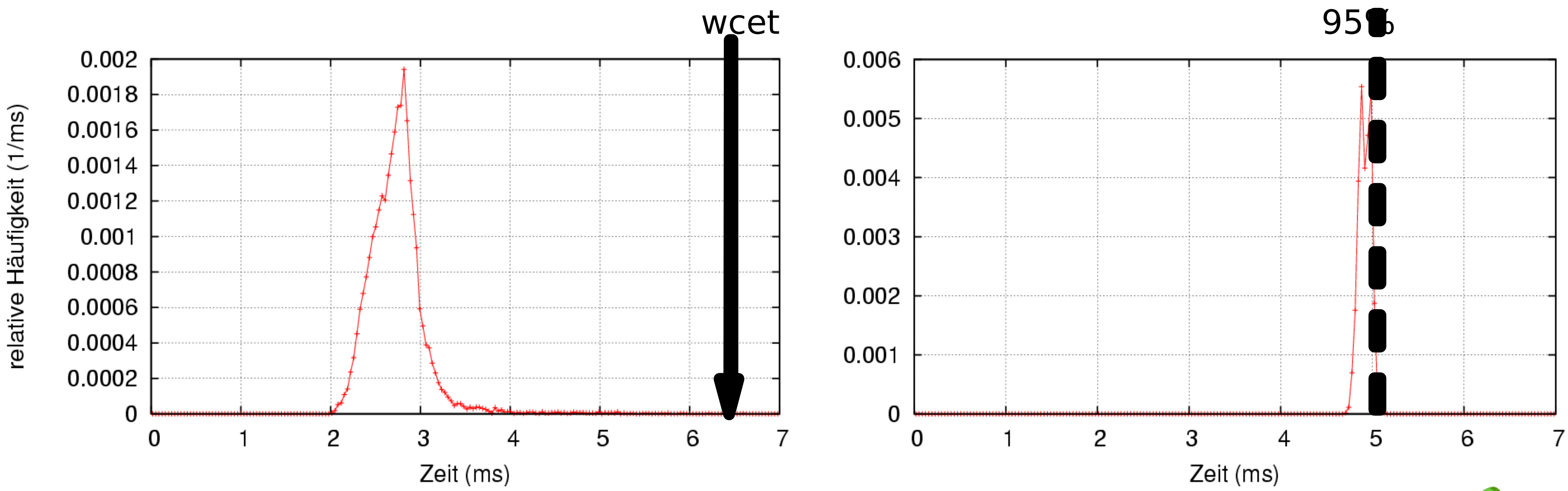
- split into three concepts:
  - **Memory pool**, which represent area of physical address space with specific attributes.
  - **Flexible page**, which describes an always size aligned region of an address space. Unlike other L4 implementations, flexible pages in F9 represent MPU region instead.
  - **Address space**, which is made up of these flexible pages.
- System calls are provided to manage address spaces:
  - **Grant**: memory page is granted to a new user and cannot be used anymore by its former user.
  - **Map**: This implements shared memory – the memory page is passed to another task but can be used by both tasks.
  - **Flush**: The memory page that has been mapped to other users will be flushed out of their address space.

IPC

- The concept of UTCB (user-level thread-control blocks) is being taken on. A UTCB is a small thread-specific region in the thread's virtual address space, which is always mapped. Therefore, the access to the UTCB can never raise a page fault, which makes it perfect for the kernel to access system-call arguments, in particular IPC payload copied from/to user threads.
- Kernel provides synchronous IPC (inter-process communication), for which short IPC carries payload in CPU registers only and full IPC copies message payload via the UTCBs of the communicating parties.

Realtime: overload tolerance

- Hard realtime must be based on worst-case analysis
- overload must be tolerated gracefully and predictable
- Applications can be split into mandatory and optional parts



Energy efficiency: Tickless

- Introduce tickless timer which allow the ARM Cortex-M to wake up only when needed, either at a scheduled time or on an interrupt event.
- Therefore, it results in better current consumption than the common approach using the system timer, SysTick, which requires a constantly running and high frequency clock.

Kprobes: dynamic instrumentation

- Inspired by Linux Kernel, allowing developers to gather additional information about kernel operation without recompiling or rebooting the kernel.
- It enables locations in the kernel to be instrumented with code, and the instrumentation code runs when the ARM core encounters that probe point.
- Once the instrumentation code completes execution, the kernel continues normal execution.

Debugging and profiling mechanisms

- configurable debug console
- memory dump
- thread profiling
  – name, uptime, stack allocated/current/used
- memory profiling
  – kernel table, pool free/allocated size, fragmentation

Reference

- From L3 to seL4: What Have We Learnt in 20 Years of L4 Microkernels? Kevin Elphinstone and Gernot Heiser, NICTA/UNSW
- Microkernel Construction"

  http://os.inf.tu-dresden.de/Studium/MkK/
- Microkernel-based Operating Systems

  http://www.inf.tu-dresden.de/index.php?node_id=1314
- The L4 Microkernel, Hermann Härtig, Technische Universität Dresden
- Microkernels, Arun Krishnamurthy, University of Central Florida