School of Computer Science & Engineering

COMP3891/9283 Extended Operating Systems

2025 T2 Week 10

Microkernels

Gernot Heiser

Copyright Notice

Learning Outcomes

- Understanding the concept of a microkernel
- Understanding of the advantages and drawbacks
- Awareness of historic performance issues and the importance of minimality

Remember from Week 7: 3 OSes

- Ultrix – DEC's version of Unix
  - Traditional OS design
  - Kernel data *physically* addressed (i.e. not in virtual memory)
  - Virtual linear array page table
- OSF/1: New commercial system
  - Most kernel data *virtually* addressed
  - 3-level page table
- Mach 3.0: "Microkernel" OS
  - Small kernel, most OS services in user-mode Unix server
  - Server data virtually addressed (obviously)
  - 3-level page table

Why?

The Origin: Brinch Hansen's Nucleus

P. Brinch Hansen: "The Nucleus of a Multiprogramming System". *Communications of the ACM*, 13, p238–250, 1970.

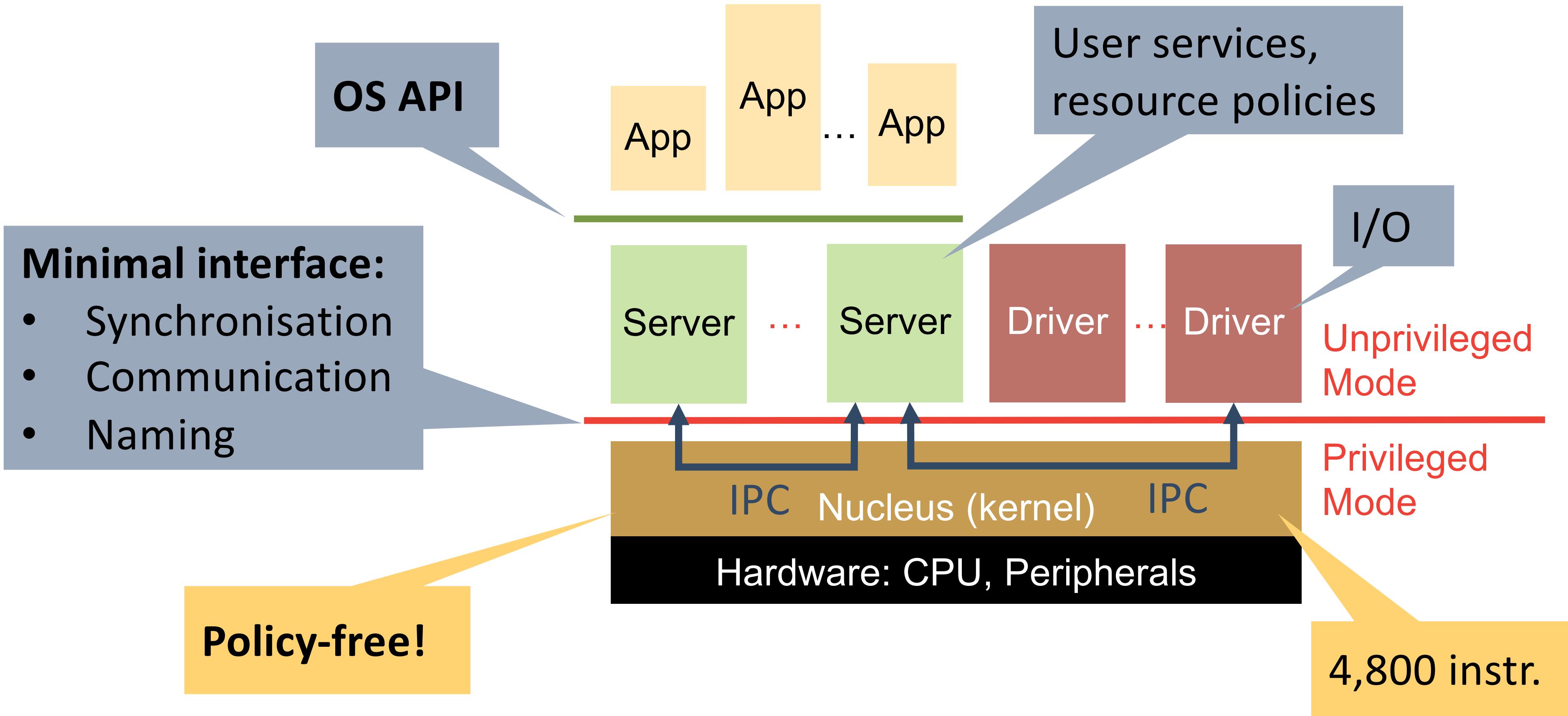
Motivation:
- OS policies difficult to adapt
- System should be extensible!

Approach:
- Stable "nucleus" with minimal abstractions
- Hierarchy of processes provide services & apps

"Internal processes" – System services

"External processes" – Device drivers

OS Structure (In Modern Terminology)



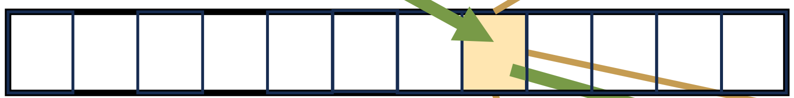
Nucleus IPC: Asynchronous Message Passing

IRQs are delivered to drivers as messages

Nucleus has message buffer pool

- On send_msg, picks free buffer
- Copies data to this buffer
- Returns buffer # as session ID

```
send_msg (dest, msg, &session);
```



```
wait_msg (&sender, &msg, &session);
int send_rply (msg, session);
int wait_rply (&msg, session);
```

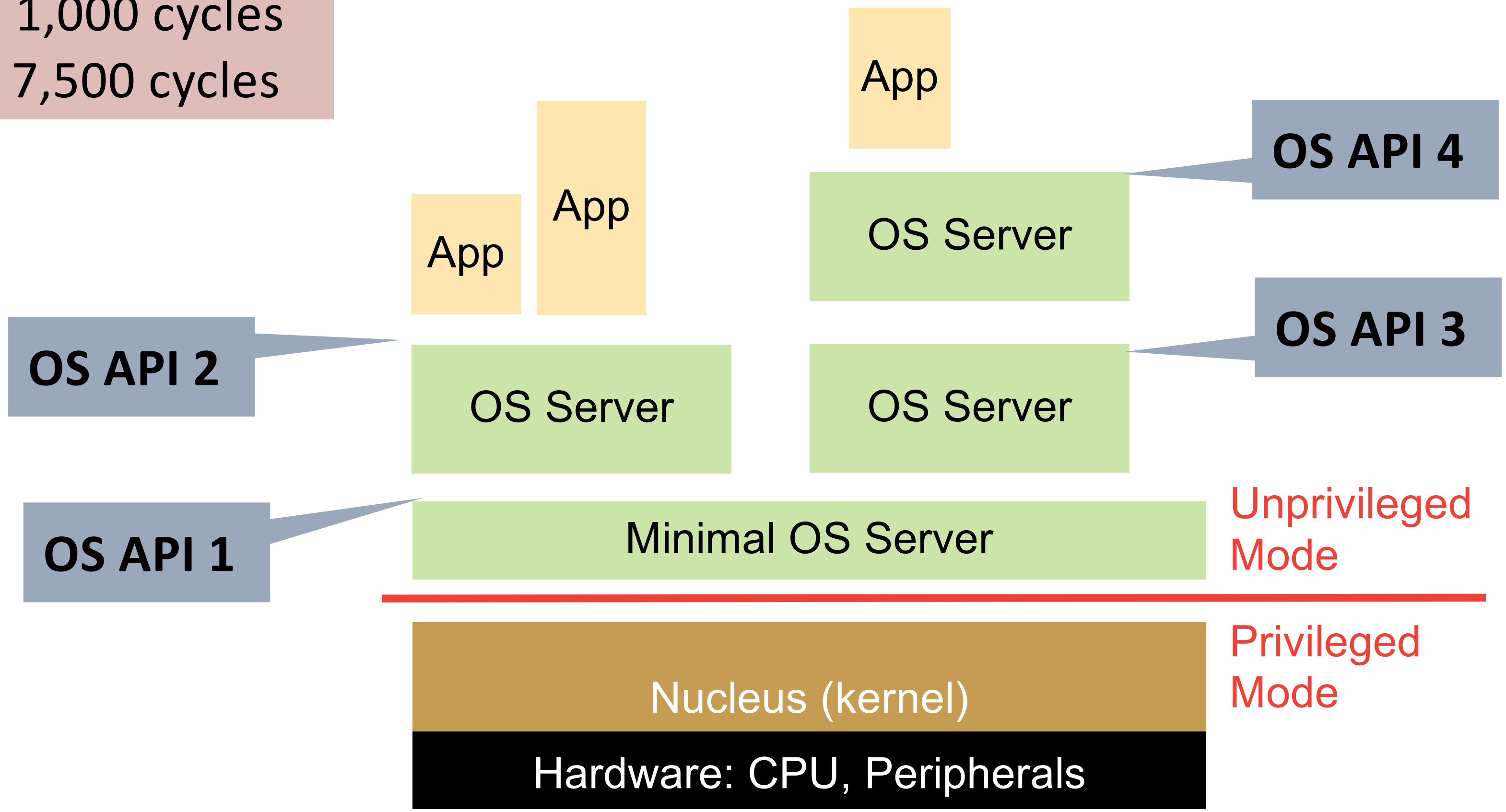
- On wait_msg copies data from buffer
- Returns buffer # as session ID

IPC Cost/message: 100–150 cycles

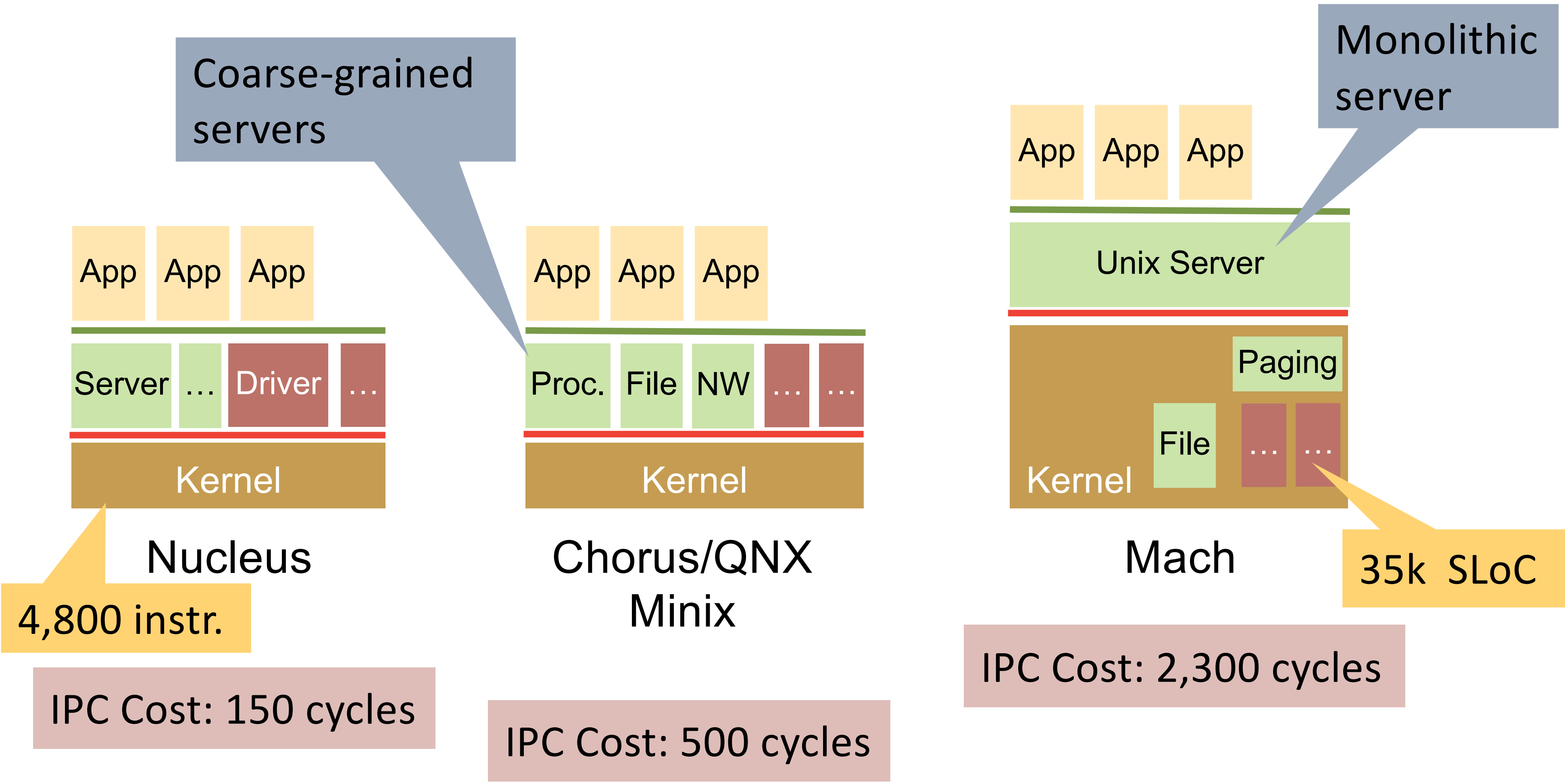
Process Hierarchy – Abstraction Hierarchy

Process management :

- Create: 750 cycles
- Start: 6,500 cycles
- Stop: 1,000 cycles
- Delete: 7,500 cycles



From Nucleus to 1st–Gen Microkernels (1980s)



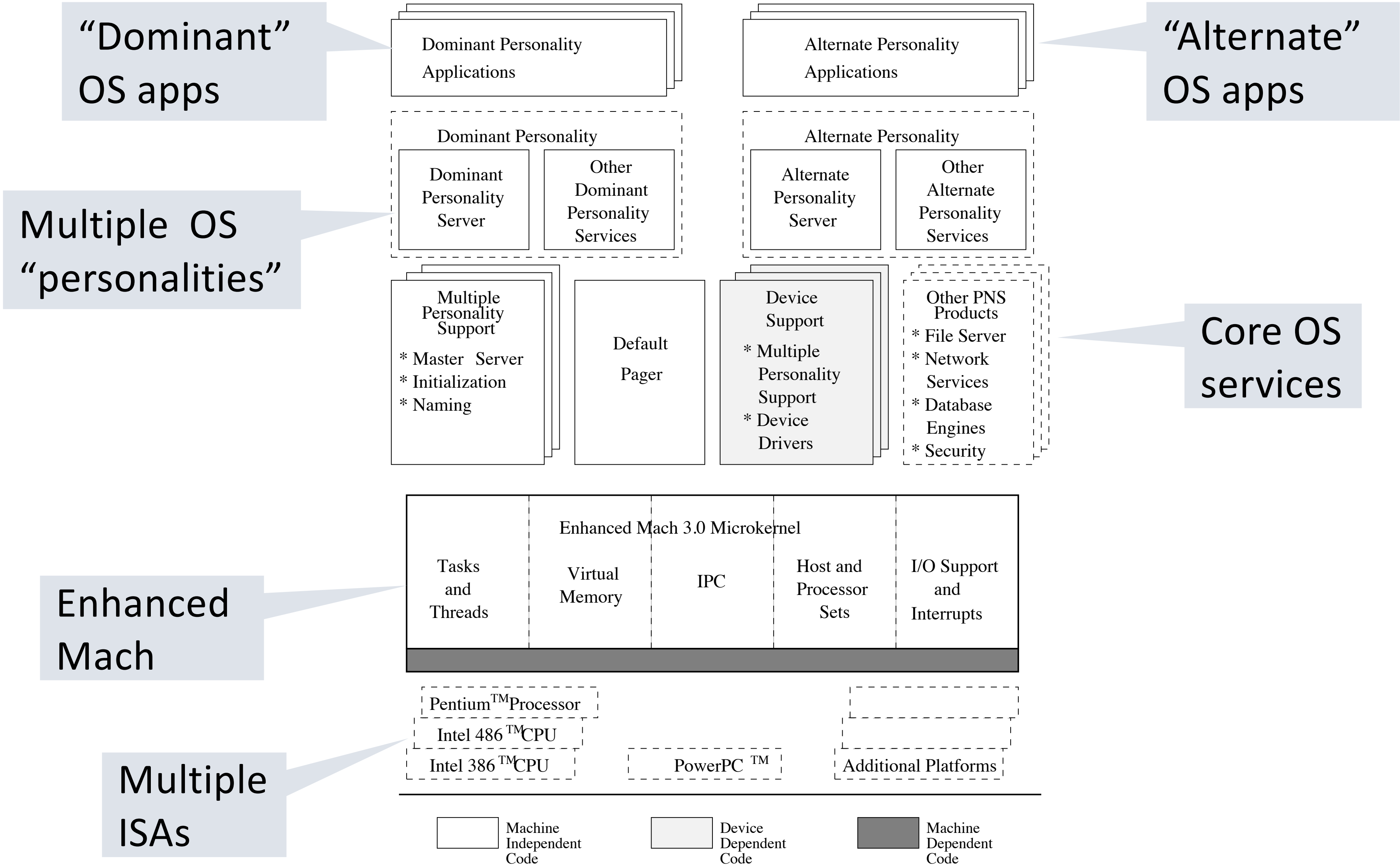
Microkernel Debacle (1990s)

Brett D. Fleisch and Mark Allan A. Co.:
"Workplace microkernel and OS: a case study."
Software: Practice and Experience 28.6 (1998): 569-591.

IBM's Grand Plan: Workplace OS (Jan'91):

- "Improved" Mach microkernel
- Run multiple OSes concurrently
- Scale from handhelds to supercomputers

Workplace OS: Plan



Workplace OS: Delivered

Abandoned late '95 after spending $2G!

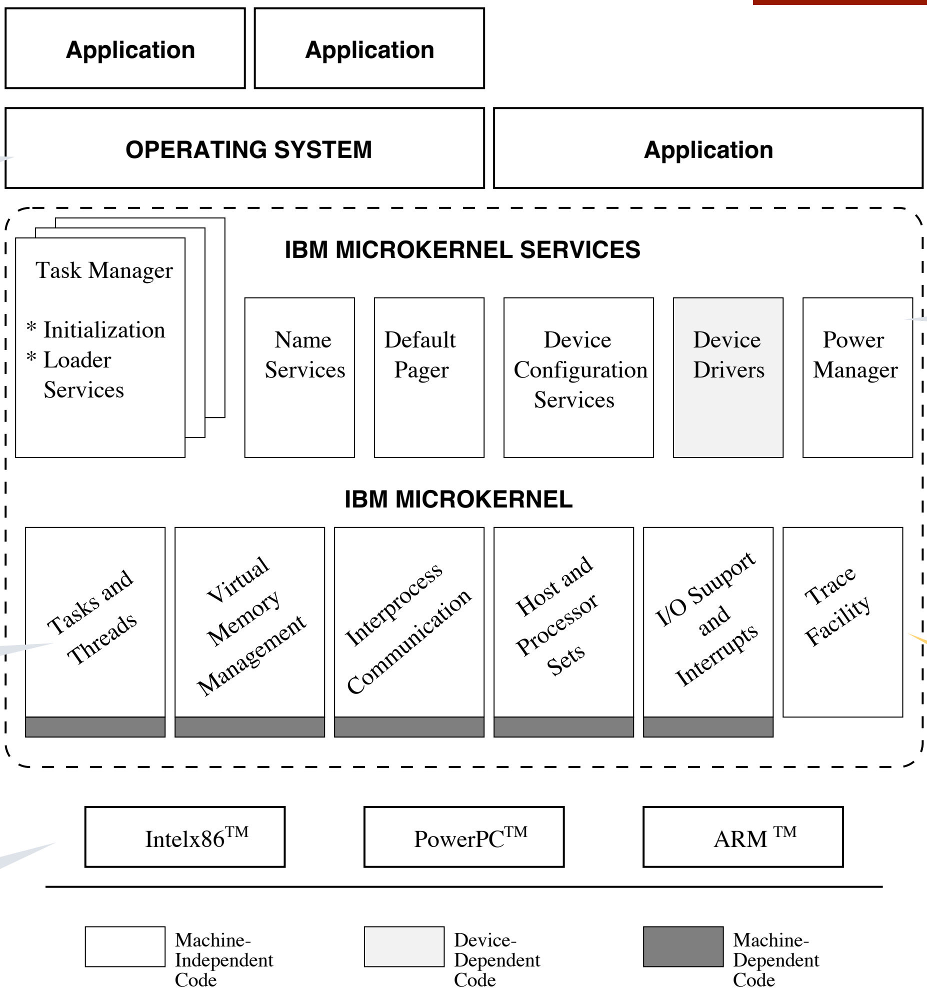
Single OS personality

Core OS services

Enhanced Mach

70k SLoC

Multiple ISAs



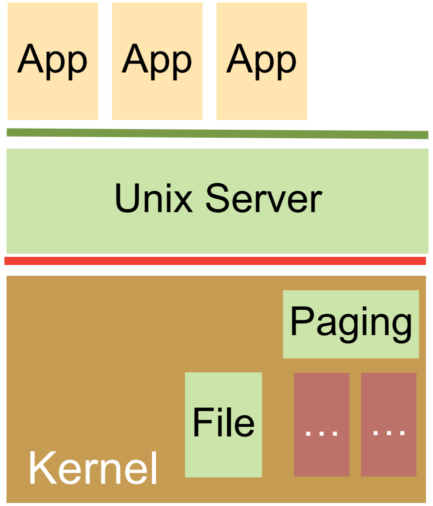
UNSW SYDNEY

Mach Problems

2,300 cycles IPC

Mach was slow, despite:

- Moving functionality back into the kernel
- Running a monolithic Unix server

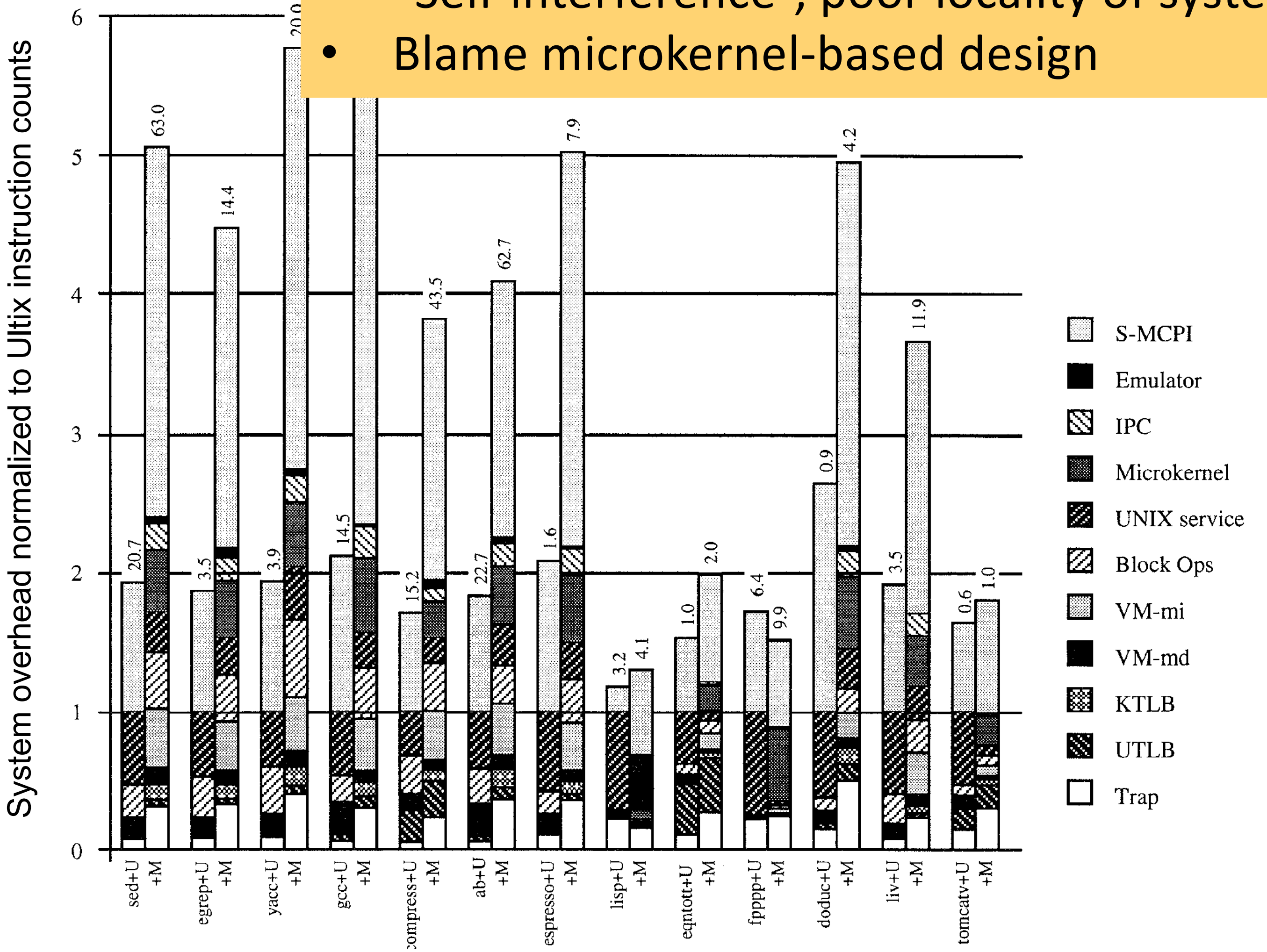


J. Bradley Chen and Brian N. Bershad: "The Impact of Operating System Structure on Memory System Performance." *ACM Symposium on Operating System Principles, 1993, p120–133*.

Mach-Unix vs Ultrix Performance Analysis

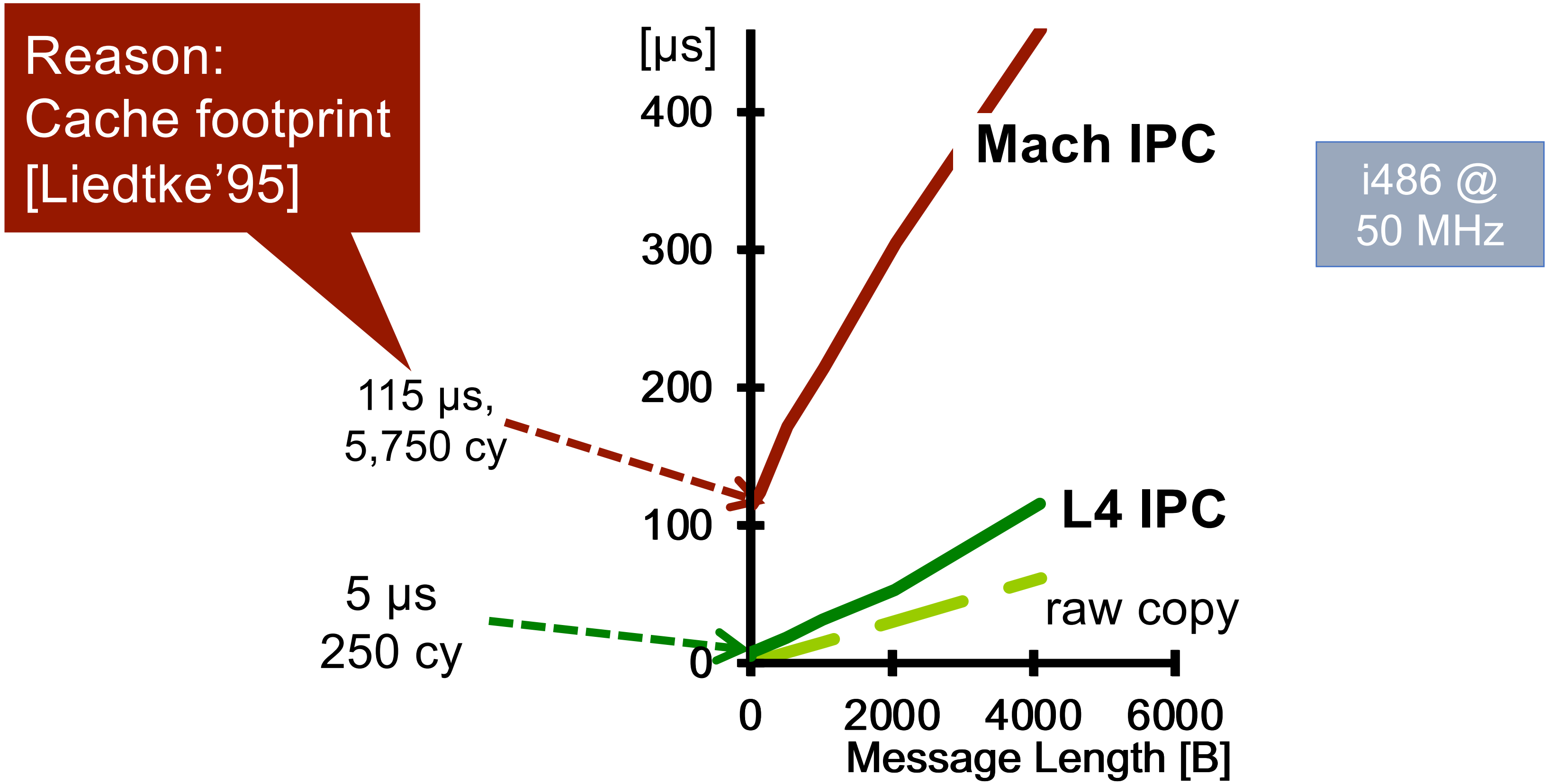
Chen & Bershad Conclusions:

- "Self-interference", poor locality of systems code
- Blame microkernel-based design



Relative system overheads for programs running on Ultix and Mach

Reality Check: Liedtke's L3/L4 Microkernel



Jochen Liedtke: "Improving IPC by Kernel Design."
ACM Symposium on Operating System Principles, 1993, p175–188.

Liedtke's Microkernel Principle: Minimality

Minimality Principle:
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 system's required functionality.

Size comparison:
- Mach: 35 kSLoC
- IBM: 70 kSLoC
- L4: 6 kSLoC

Implications:
- Kernel provides *mechanisms*, not *policies*
- Minimal mechanisms enable simple implementation, aggressive optimisation

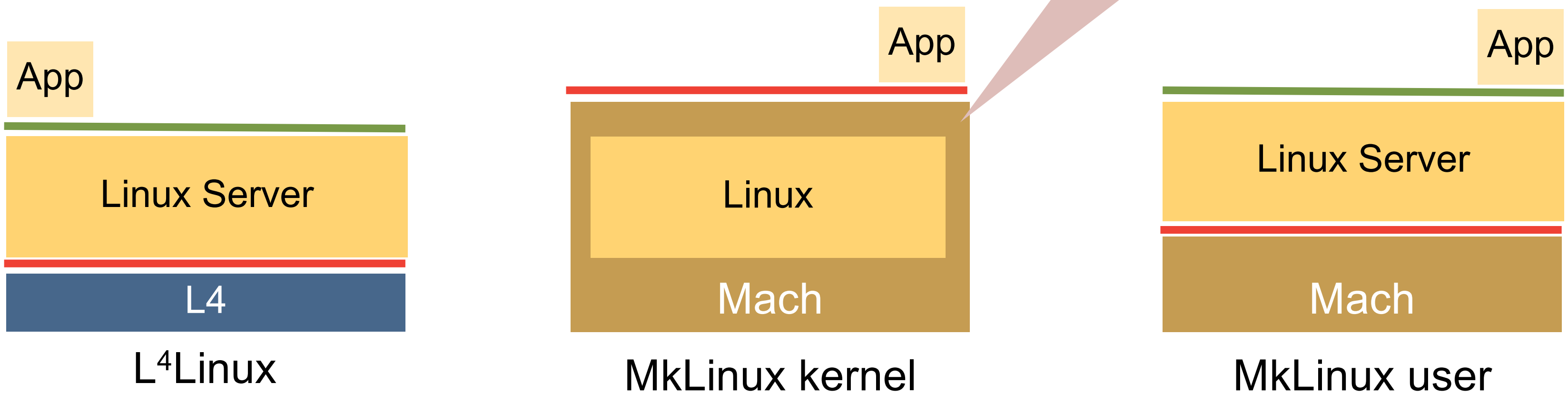
Jochen Liedtke: "On µ-Kernel Construction."
ACM Symposium on Operating System Principles, 1995, p237–250.

L4 vs Mach: Monolithic Linux Server

System	Time	Cycles
Linux	1.68 µs	223
L⁴Linux	3.95 µs	526
MkLinux kernel	15.41 µs	2,050
MkLinux user	110.60 µs	14,710

Cost of **`getpid()`**

macOS similar!



Hermann Härtig and Michael Hohmuth and Jochen Liedtke and Sebastian Schönberg and Jean Wolter: "The Performance of µ-Kernel-Based Systems."
ACM Symposium on Operating System Principles, 1997, p66–77.

Microkernels Are Widely Deployed Today

… especially in safety/security-critical systems:

- QNX [Hildebrand 1992]:
  - Widely used in transport systems (trains, cars)
- INTEGRITY-178 (circa 2000):
  - Avionics: military and civilian aircraft
- PikeOS (née P4, an L4 clone, ca 1999):
  - Avionics, defence systems
- Fiasco.OC/L4Re (TU Dresden, from 1998)
  - National security systems
- L4-embedded (UNSW fork of Karlsruhe L4-Pistachio, 2005)
  - Qualcomm modem chips, iOS secure enclave
- seL4 (NICTA/UNSW, 2009)
  - Defence systems, electric cars

Implications of Minimality

- Challenging to get minimal API right
- But dramatically reduced *trusted computing base* (TCB)

Can we prove it correct?

Security. Performance. Proof.

Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, Simon Winwood: "seL4: Formal verification of an OS kernel." *ACM Symposium on Operating System Principles, 2009, p207–220*.