


The Mach System

from *Operating Systems Concepts*, Sixth Edition by Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne

Presented by Adam Lowry


Design Principles

- Support for multiple architectures
- Varying network speeds
- Simple kernel structure
- Distributed operation
- Integrated memory management & IPC
- Heterogeneous system support


Components

- Task
- Thread
- Port
- Port Set
- Message
- Memory Object

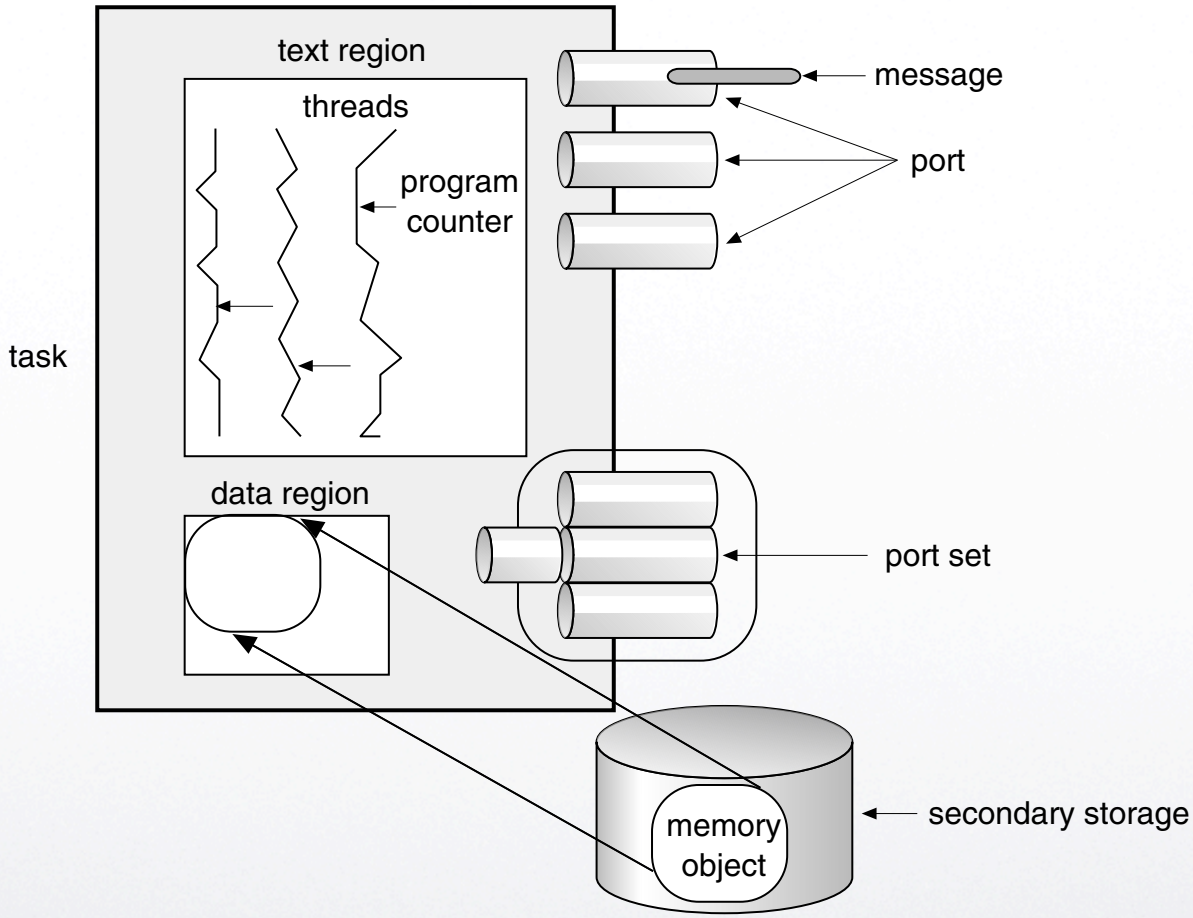

Figure B.2 Mach's basic abstractions.


Process Management

- Task + Threads = Process
- All threads are kernel-supported
- Synchronization implemented with IPC


Threads and Scheduling

- C Threads library provides all standard tasks
- Mutual exclusion implemented with spinlocks
- Blocking waits with condition variables
- Simple model - only threads are scheduled
- Multiple queues


Exception Handling

- Made up of kernel primitives:
  - Handler is a thread, RPC for logic


Internal Exceptions

- Victim notifies handler via IPC
- Victim waits for resolution
- Handler receives notification with information
- Handler clears or terminates


BSD-Style Signals

- Difficult to implement
- Process to process: converted to IPC
- Hardware exceptions handled by kernel


IPC

- Traditional way: global names, untyped streams
- Mach way: location independent *ports*, typed data


Ports

- Managed by kernel
- All objects have standard ports
- Can be grouped into Port Sets

Messages

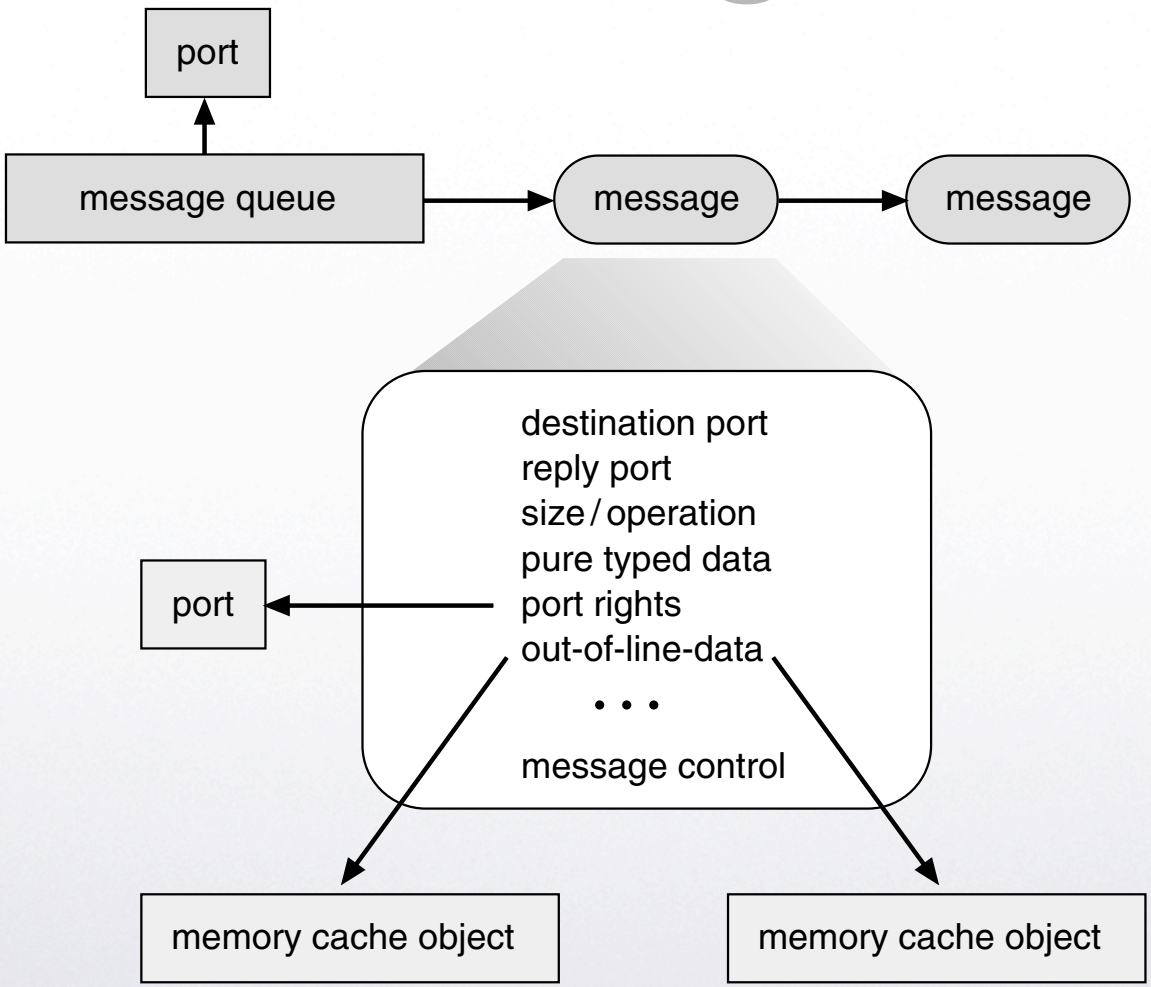

Figure B.5 Mach messages.

Message Passing

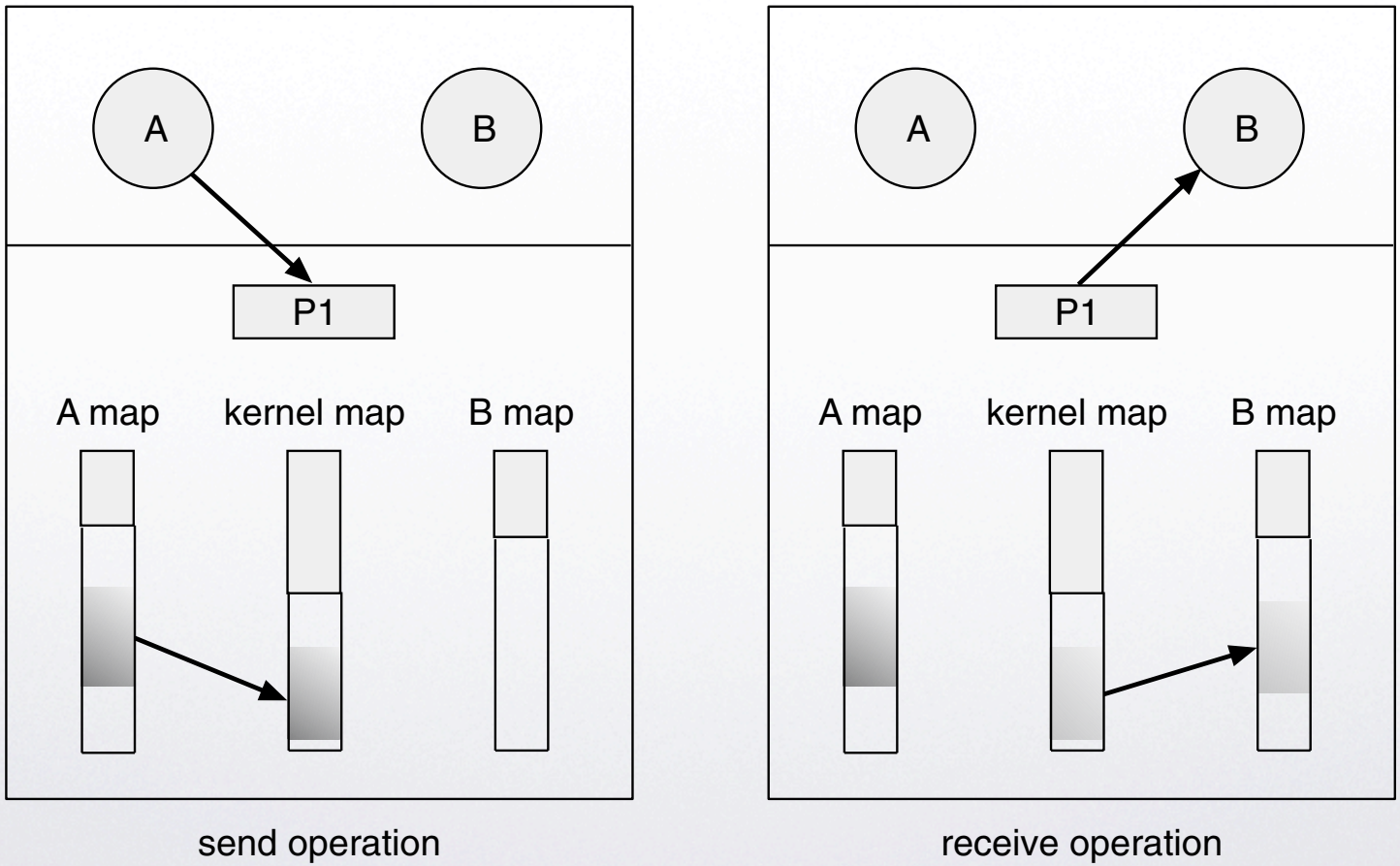

Figure B.6 Mach message transfer.


NetMsgServer

- All objects should be location independent
- NetMsgServer handles naming & transportation in user level


Memory Management

- Everything is a Memory Object
- Control via IPC
- Memory managers are user-level, with default backup


Programming Interface

- BSD system calls
- Emulation and servers at user-level
- Threading library
- Stubs