The MACH Kernel: Architecture and Features

-Nikhil Pradhan

-Nistha Bajracharya

-Prinsa Joshi

-Samman Babu Amgain

Distributed System

-Er. Anku Jaiswal

Content

What is MACH?

Microkernel-based operating system design that separates the kernel's essential functions from less critical services.

In a distributed system, MACH microkernel allows the distribution of these essential functions across multiple nodes, enhancing fault tolerance and scalability while promoting modularity and flexibility in the system.

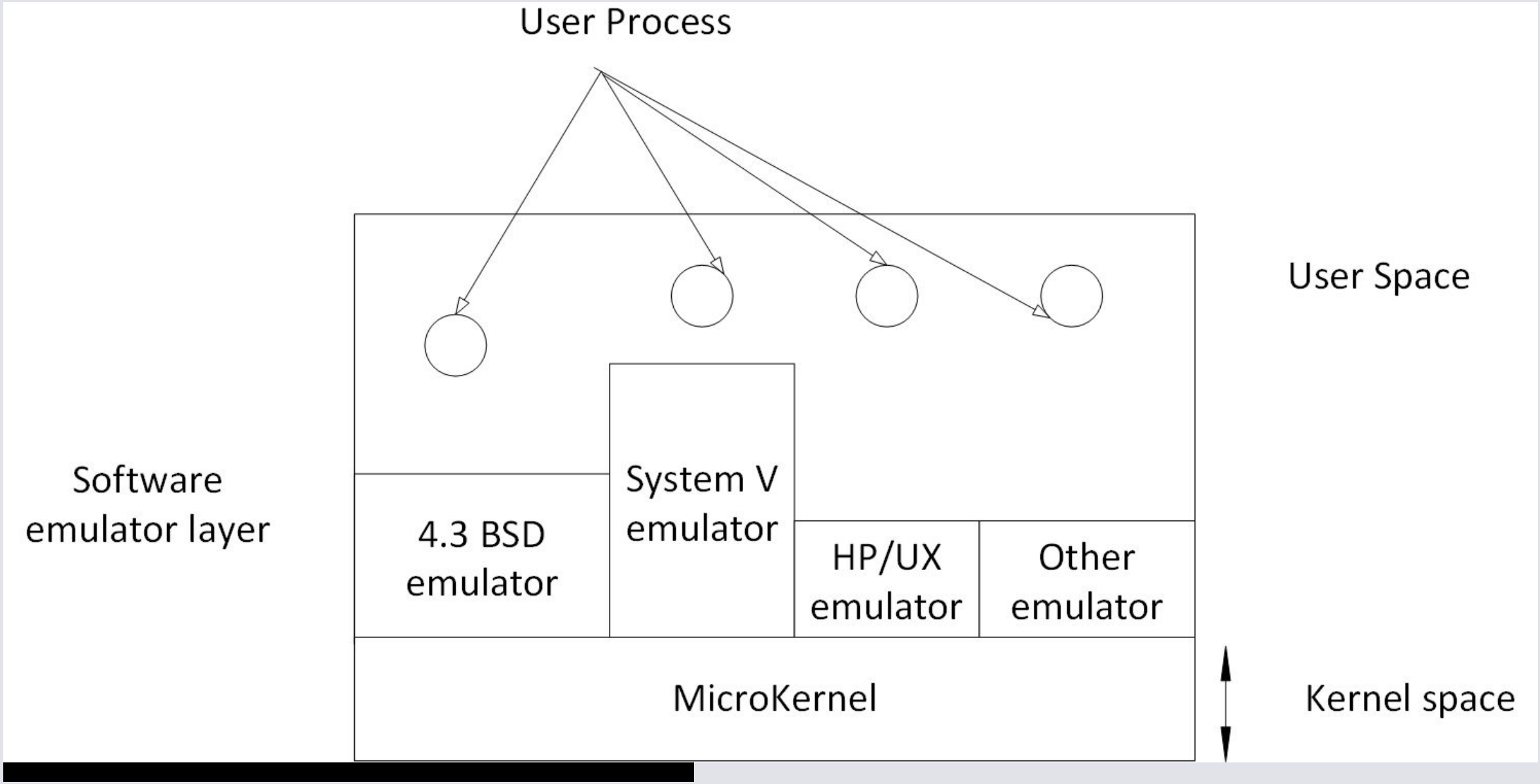


Fig: Mach Microkernel

Evolution of MACH

FIRST VERSION: 1986 FOR VAX 11/784, A FOVR, CPU MULTIPROCESSOR.

SHORTLY THEREAFTER, PART TO IBM PC/RT AND SUN B WERE DONE.

BY 1987 MACH WAS ALSO RUNNING ON THE ENCORE AND SEQUENT MULTIPROCESSOR.

AS 1988, THE MACH 25 KERNEL WAS LARGE AND MONOLITHIC, DUE TO PRESENCE OF LARGE AMOUNT OF BERKELEY'S CODE.

CMU REMOVED ALL BERKELEY CODE FROM THE KERNEL AND PUT IT USER SPACED.

MACH Goals

Basic Architecture of MACH

- MACH has a small and extensible system kernel, providing essential functionalities for an operating system.

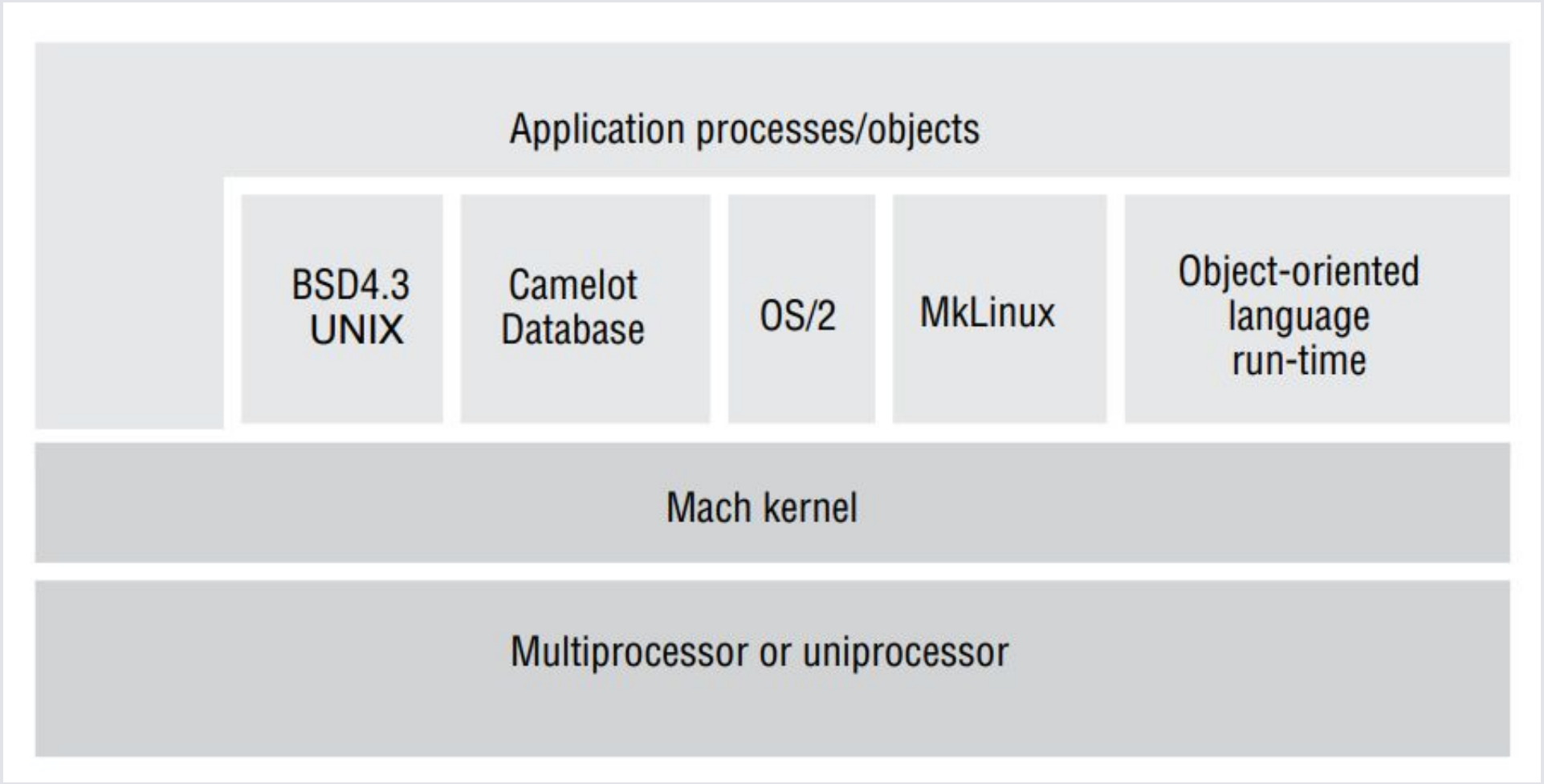
- It includes components like scheduling, virtual memory management, and inter-process communication.

- MACH supports distributed file access for seamless file access across multiple systems.

- Remote execution is possible in MACH, allowing tasks to run on remote systems as if they were local.

- MACH can emulate established operating system environments like UNIX, ensuring compatibility for existing applications.

MACH supports operating systems, databases and other subsystems



Main Mach abstractions

- **Tasks**:

  A Mach task is an execution environment. This consists primarily of a protected address space, and a collection of kernel-managed capabilities used for accessing ports.

- **Threads**:

  Tasks can contain multiple threads. The threads belonging to a single task can execute in parallel at different processors in a shared-memory multiprocessor.

- **Ports:**

  A port in Mach is a unicast, unidirectional communication channel with an associated message queue. Ports are not accessed directly by the Mach programmer and are not part of a task. Rather, the programmer is given handles to port rights. These are capabilities to send messages to a port or receive messages from a port.

Main Mach abstractions(Cont..)

- **Messages:**

  A message in Mach can contain port rights in addition to pure data. The kernel employs memory management techniques to transfer message data efficiently between tasks.

- **Devices:**

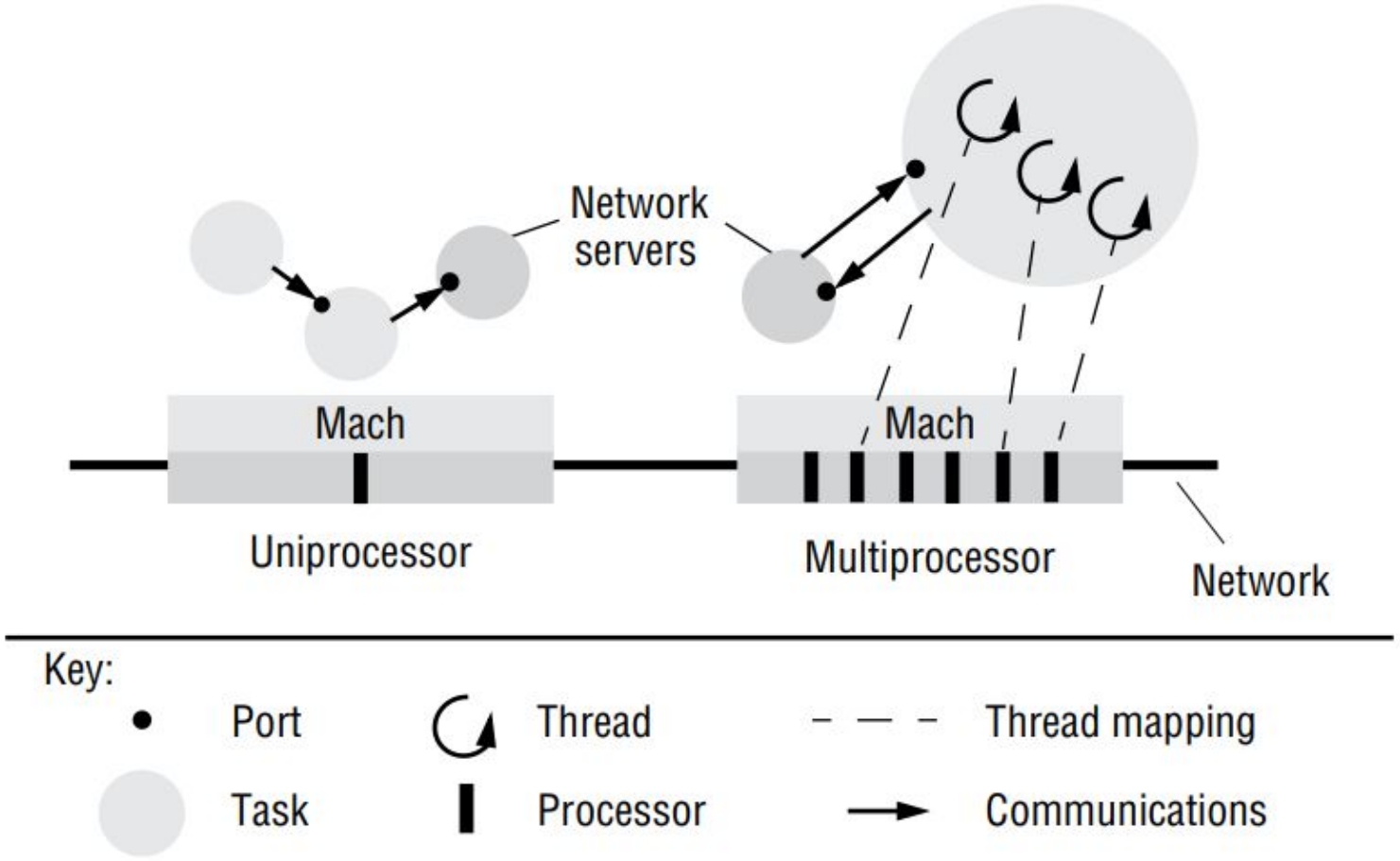
  Servers such as file servers running at user level must access devices. The kernel exports a low-level interface to the underlying devices for this purpose.

Main Mach abstractions(Cont..)

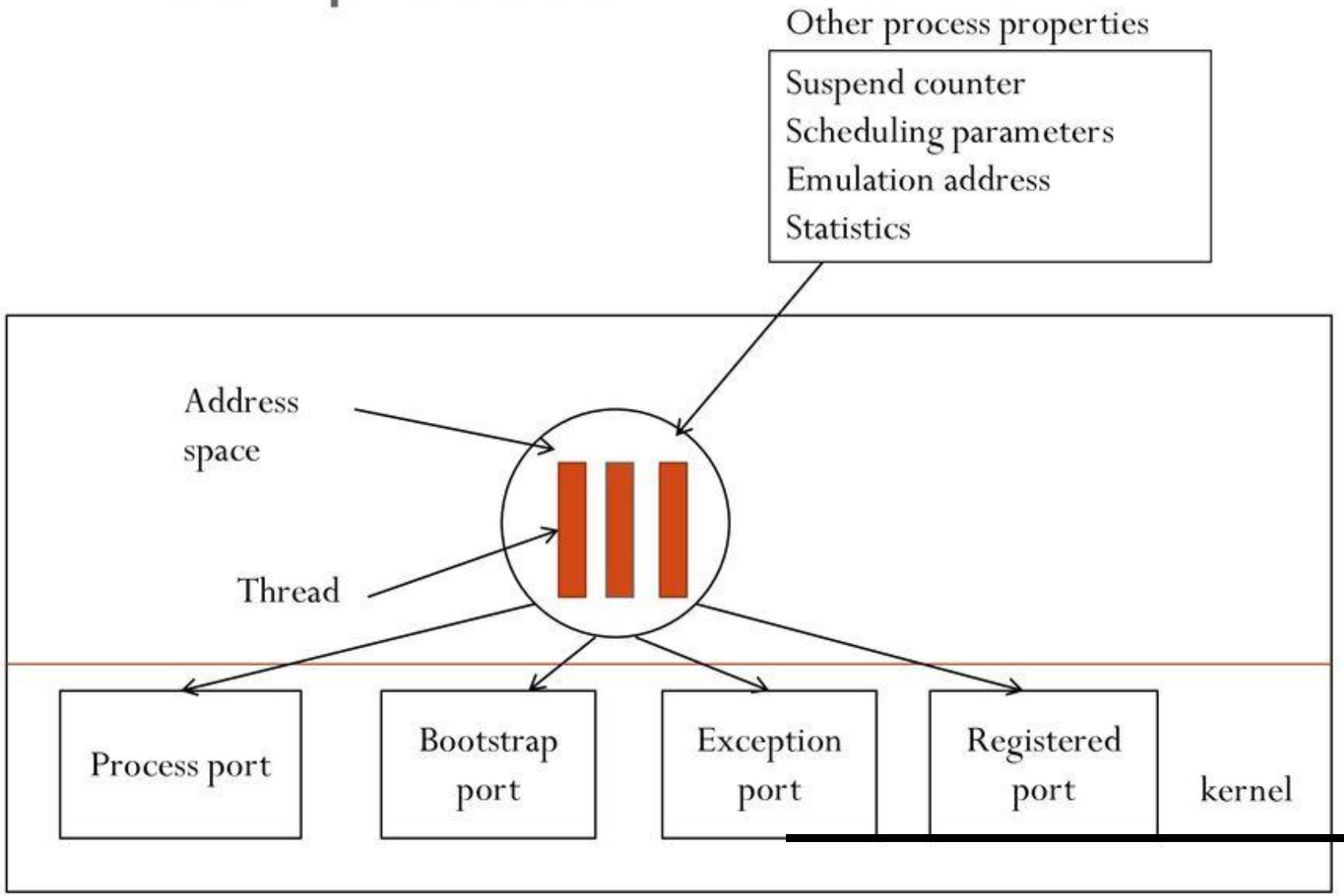
- **Memory object:**

  Each region of the virtual address space of a Mach task corresponds to a memory object. This is an object that in general is implemented outside the kernel itself but is accessed by the kernel when it performs virtual memory paging operations. A memory object is an instance of an abstract data type that includes operations to fetch and store data that are accessed when threads give rise to page-faults in attempting to reference addresses in the corresponding region.

MACH task, threads and communication



A Mach process



Features of the MACH Kernel

- Multiple tasks with large, paged virtual memory spaces
- Multiple threads of execution within each task
- Flexible scheduling facility
- Flexible sharing of memory between tasks
- Port based inter-process communication
- Transparent network extensibility

Process Management

MACH employs a scheduler to determine process execution order based on predefined algorithms.
MACH provides mechanisms for processes to exchange data and synchronize activities, including messaging through ports.
MACH offers synchronization primitives like locks and semaphores to coordinate processes accessing shared resources.
MACH handles process termination by releasing resources, updating accounting information, and notifying parent processes or the system.
MACH maintains states such as running, ready, blocked, or terminated to represent process progress and status.

Inter-Process Communication

In MACH, a conventional process is represented by a task with a single thread of control.
Inter-process communication in MACH is based on ports and messages.
Ports are protected kernel objects used for message sending and receiving.
Port sets are groups of ports combined into a single queue for message reception.
Messages consist of a header and a body, containing typed data, memory copies, and port capabilities.
Tasks communicate by sending messages through ports they have send rights to.
Message receiving is an asynchronous operation, with messages logically copied into the receiving task.
Only one task can hold the receive right for a port, allowing it to read messages from the queue.
Multiple tasks can hold rights to send messages into the queue.

Memory Management

MACH utilizes memory objects. This caching mechanism improves access speed and efficiency.

Memory objects in MACH can be represented by files or any object capable of handling read and write requests. This flexibility allows for diverse types of memory objects to be utilized.

Memory objects are mapped into a task's virtual memory in MACH. This mapping enables tasks to access and manipulate the contents of memory objects as if they were part of their own address space.

The memory manager plays a vital role in MACH's memory management system. It provides memory object ports, which serve as interfaces through which tasks can access and manage memory objects.

The memory manager also oversees the allocation and deallocation of memory object ports, ensuring efficient utilization of memory resources and facilitating controlled access to memory objects.

Other Versions of MACH

Importance of the MACH Kernel in Supporting Distributed and Parallel Computation:

The MACH kernel enables efficient communication and resource sharing, essential for distributed and parallel computation.

It allows for concurrent execution and collaboration among tasks, maximizing system performance.

Potential Applications and Benefits of the MACH Kernel:

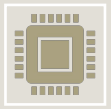
MACH is well-suited for building distributed applications that leverage shared memory multiprocessors and networked uniprocessors.

It facilitates the development of scalable and high-performance computing systems.

The capability-based security approach of MACH ensures controlled access and protects sensitive resources.

Conclusions

Recap of Key Points Discussed:



The MACH kernel is a communication-oriented operating system kernel designed to support distributed and parallel computation.

It incorporates features such as inter-process communication, flexible memory management, and transparent network extensibility.

MACH provides capabilities like multiple tasks, threads, and message-based communication.

Thank You!