Figure 18.1 Mach supports operating systems, databases and other subsystems

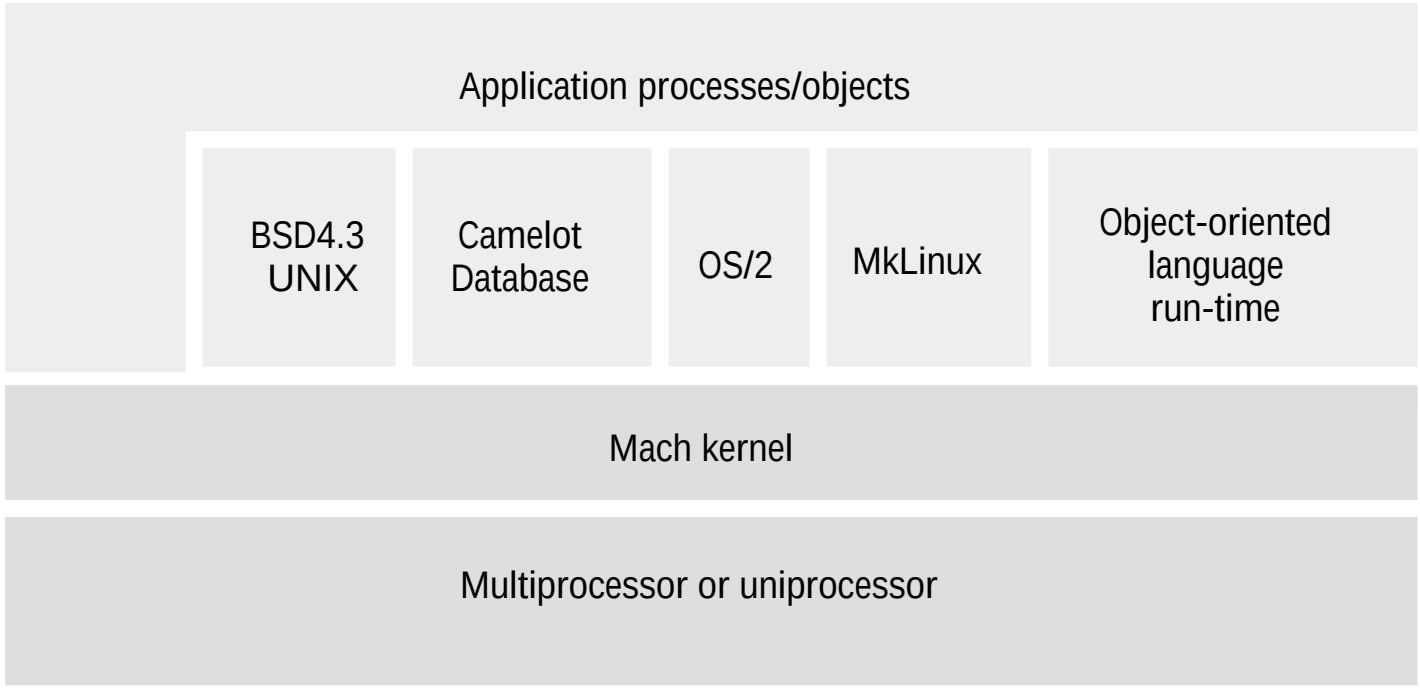

Figure 18.2 Mach tasks, threads and communication

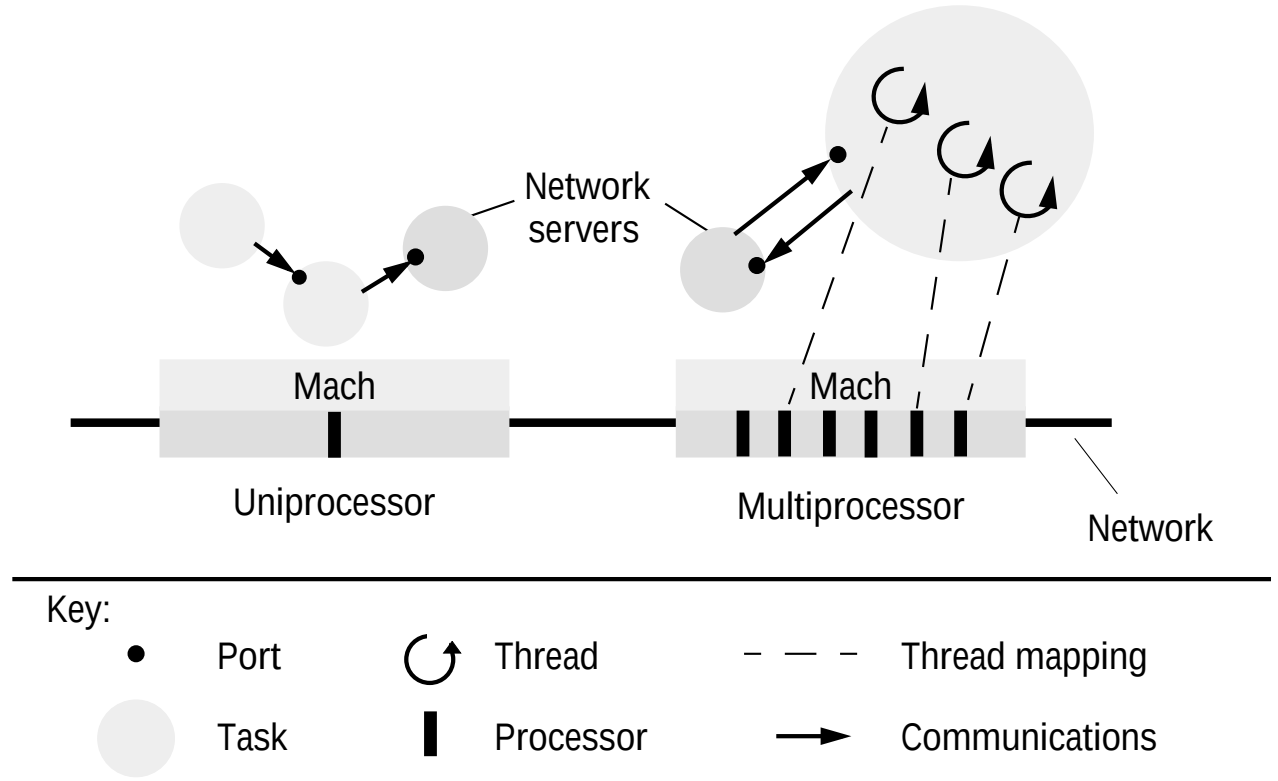

Figure 18.3 A task's port name space

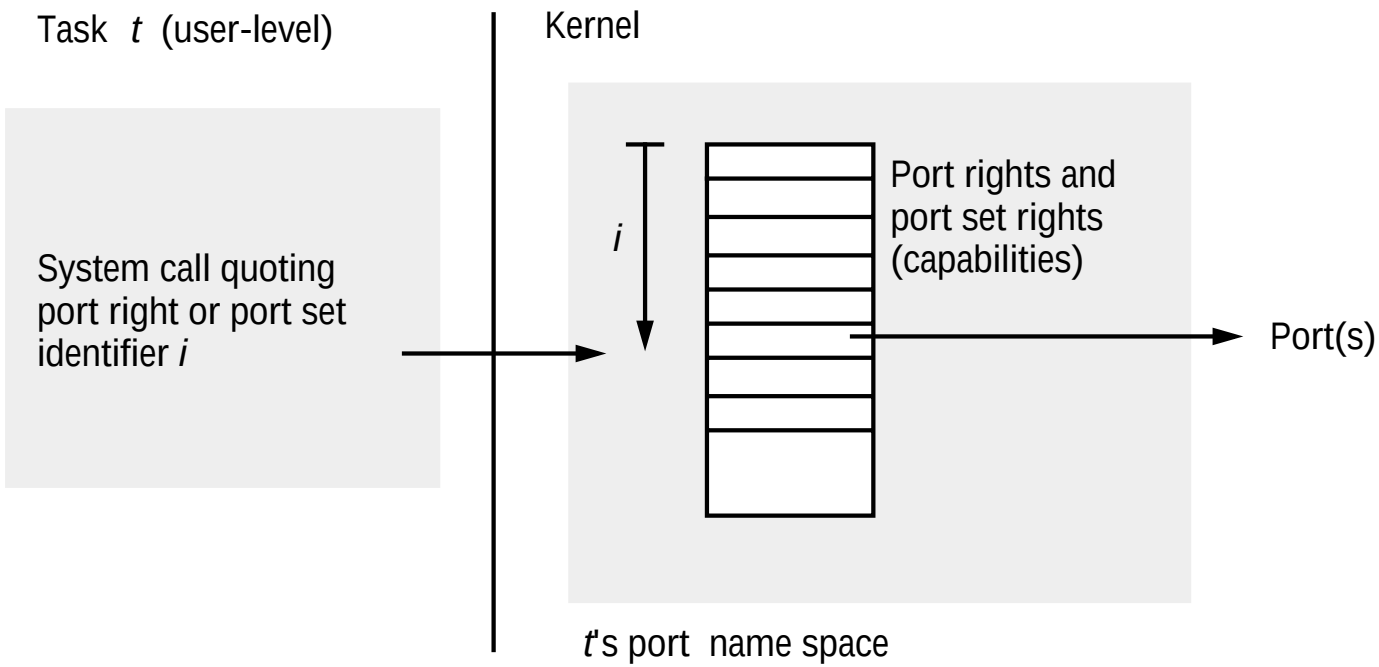

Figure 18.4 Task and thread creation

task_create(parent_task, inherit_memory, child_task)

parent_task is the task used as a blueprint in the creation of the new task, *inherit_memory* specifies whether the child should inherit the address space of its parent or be assigned an empty address space, *child_task* is the identifier of the new task.

thread_create(parent_task, child_thread)

parent_task is the task in which the new thread is to be created, *child_thread* is the identifier of the new thread. The new thread has no execution state and is suspended.

thread_set_state(thread, flavour, new_state, count)

thread is the thread to be supplied with execution state, f*lavour* specifies the machine architecture, *new_state* specifies the state (such as the program counter and stack pointer), *count* is the size of the state.

thread_resume(thread)

This is used to resume the suspended thread identified by *thread.*

Figure 18.5 A Mach message containing port rights and out-of-line data

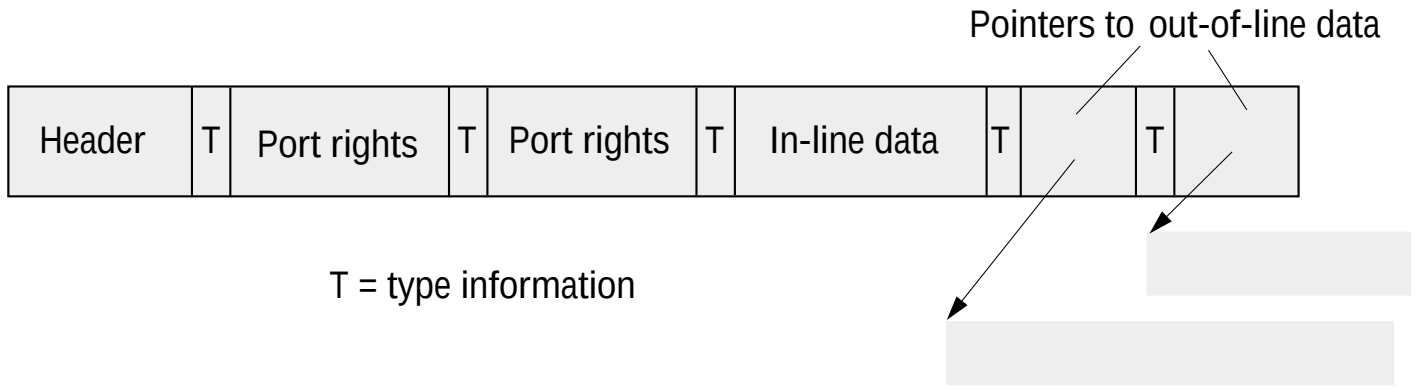

Figure 18.6 Network communication in Mach

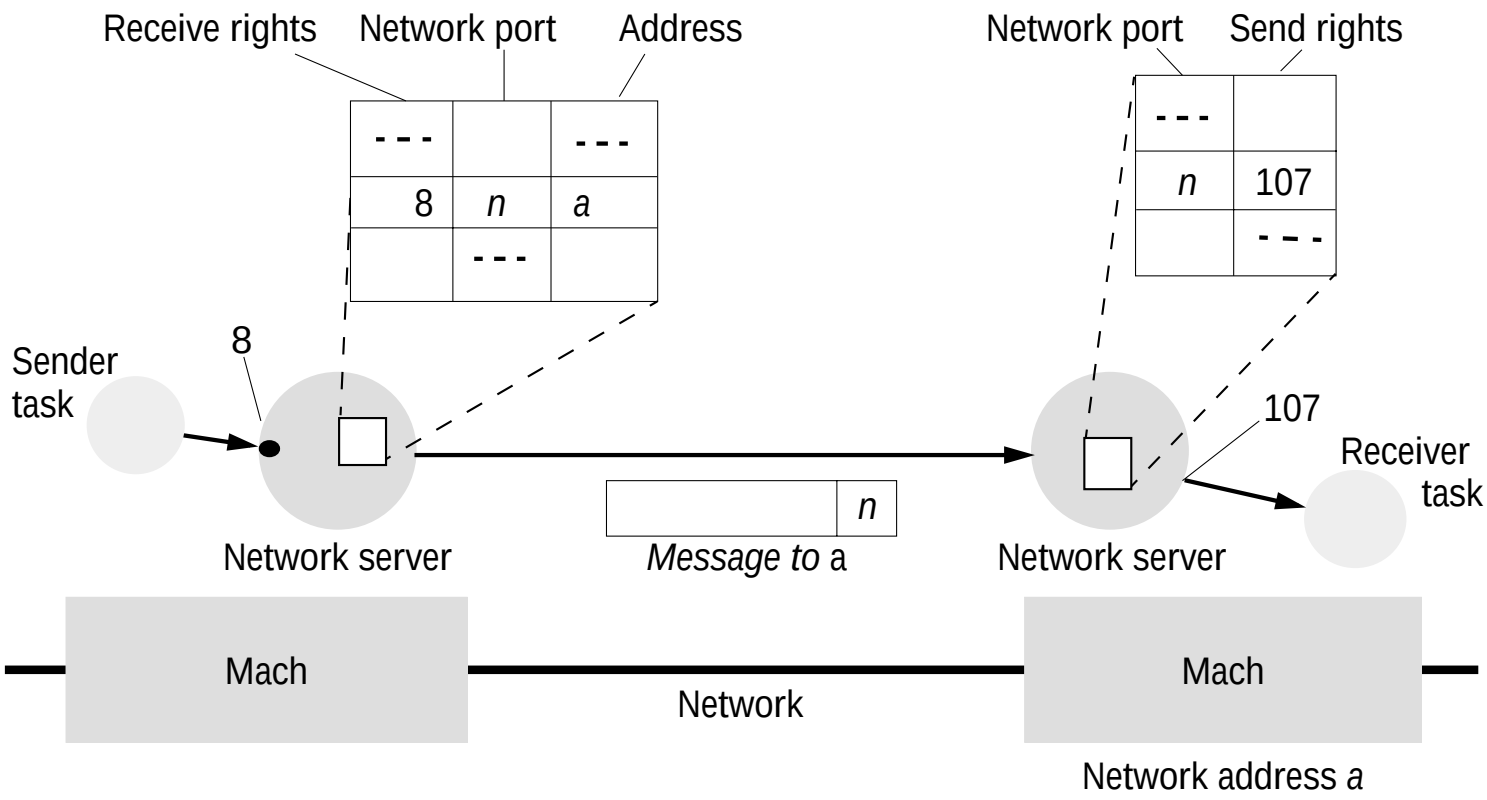

Figure 18.7 Copy-on-write overheads

Region size	*Simple copy*	*Create region*	*Amount of data copied (on writing)*		
			0 kilobytes (0 pages)	*8 kilobytes (1 page)*	*256 kilobytes (32 pages)*
8 kilobytes	1.4	1.57	2.7	4.82	–
256 kilobytes	44.8	1.81	2.9	5.12	66.4

Note: all times are in milliseconds.

Figure 18.8 External pager

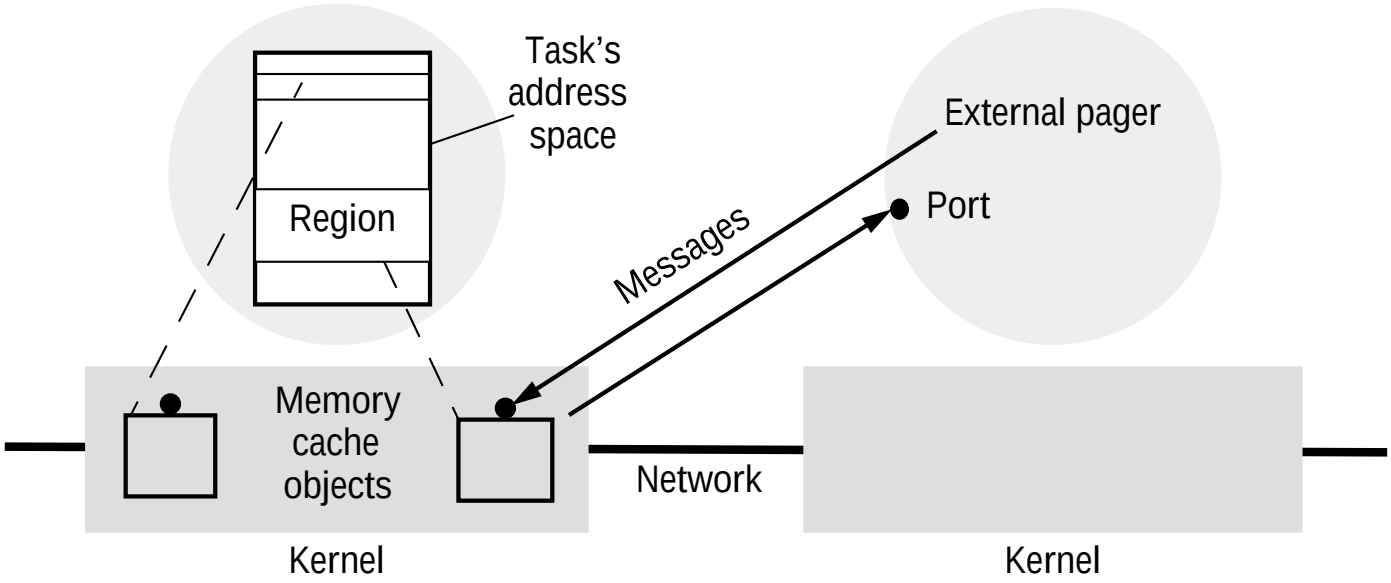

Figure 18.9 External pager messages

Event	*Sender*	*Message*
vm_map called by task	K → EP	*memory_object_init*
	EP → K	*memory_object_set_attributes*, or
	EP → K	*memory_object_data_error*
Task page-faults when no data frame exists	K → EP	*memory_object_data_request*
	EP → K	*memory_object_data_provided*, or
	EP → K	*memory_object_data_unavailable*
Kernel writes modified page to persistent store	K → EP	*memory_object_data_write*
External pager directs kernel to write page/set access permissions	EP → K	*memory_object_lock_request*
	K → EP	*memory_object_lock_completed*
Task page-faults when insufficient page access	K → EP	*memory_object_data_unlock*
	EP → K	*memory_object_lock_request*
Memory object no longer mapped	K → EP	*memory_object_terminate*
External pager withdraws memory object	EP → K	*memory_object_destroy*
	K → EP	*memory_object_terminate*