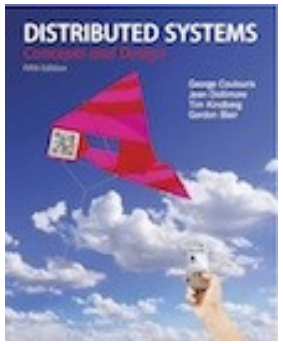
Slides for Chapter 7: Operating System support



From **Coulouris, Dollimore, Kindberg and Blair**

Distributed Systems: Concepts and Design

Figure 7.1
System layers

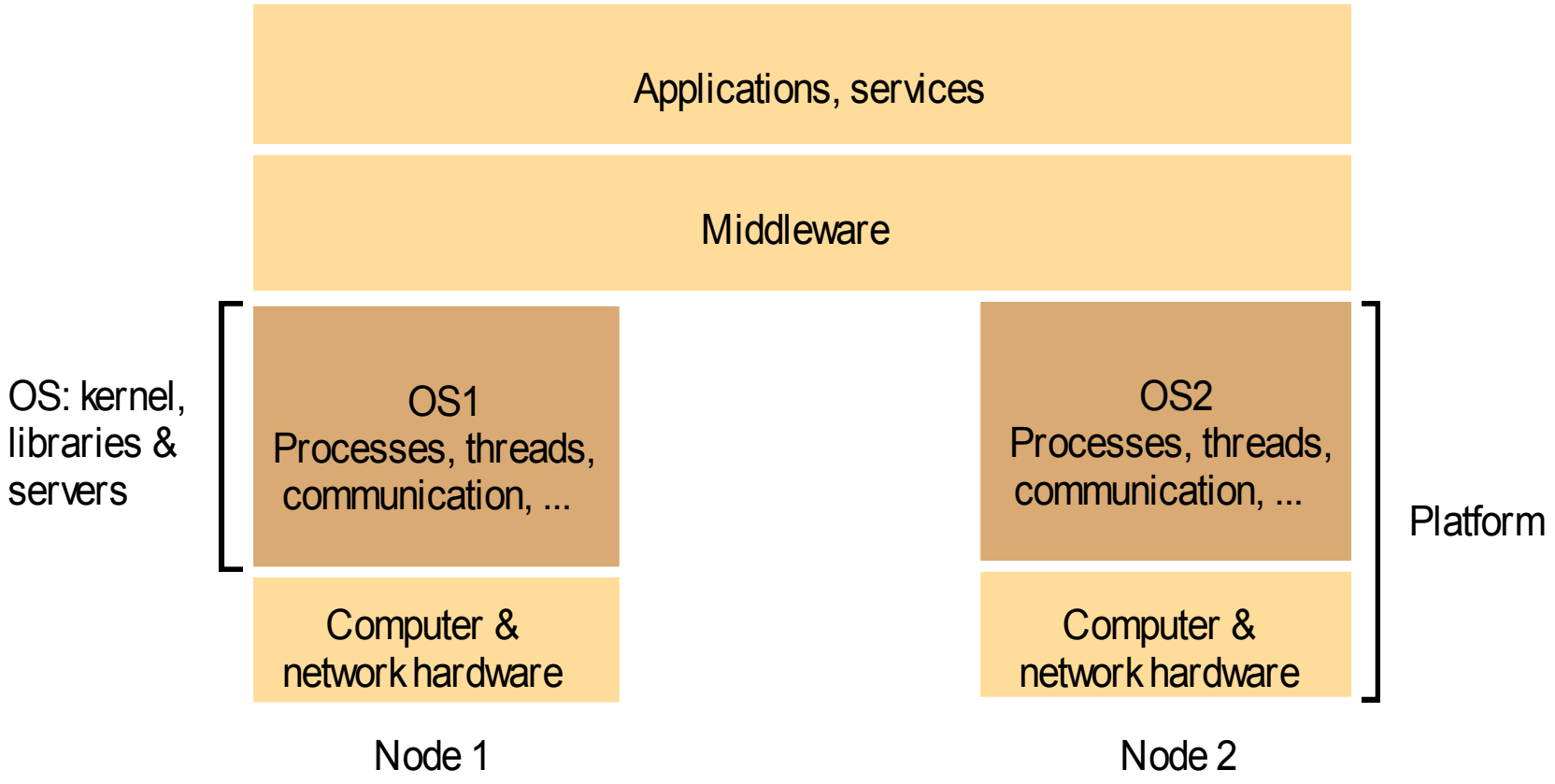


Figure 7.2
Core OS functionality

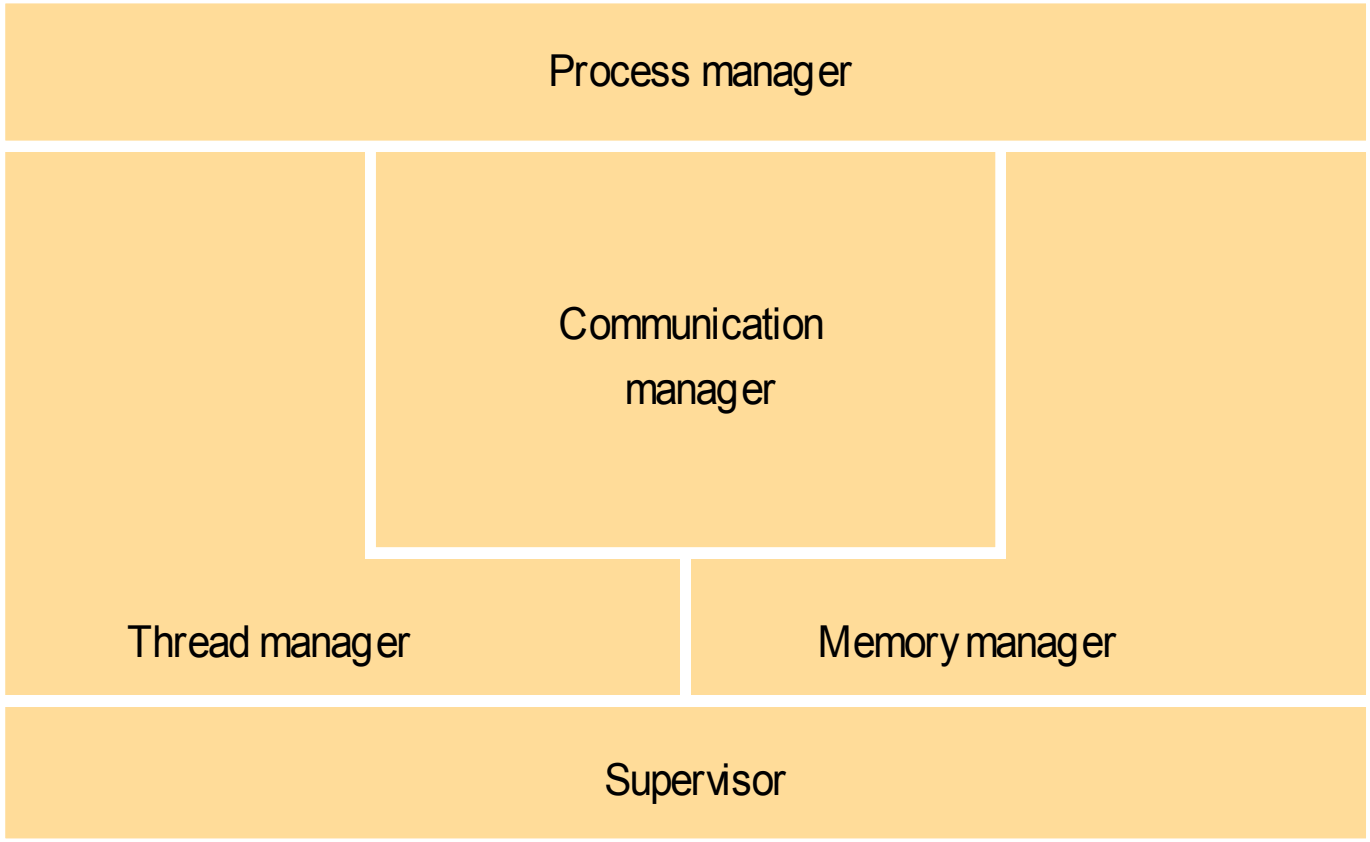


Figure 7.3
Address space

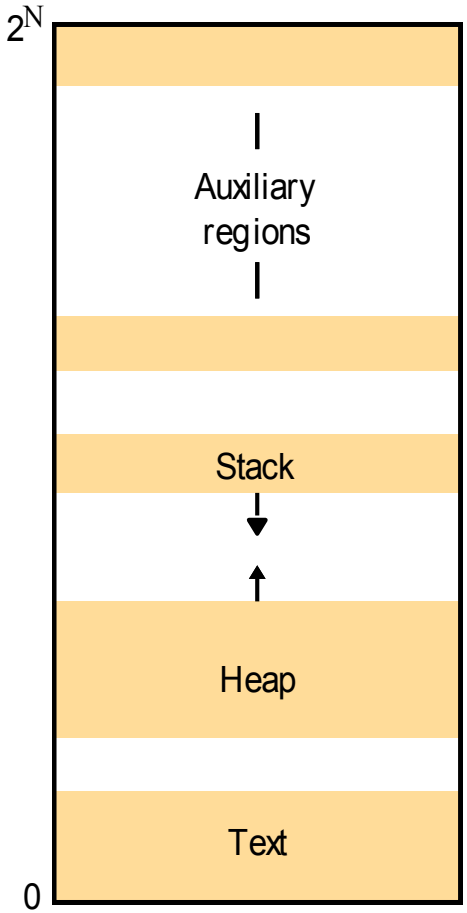


Figure 7.4
Copy-on-write

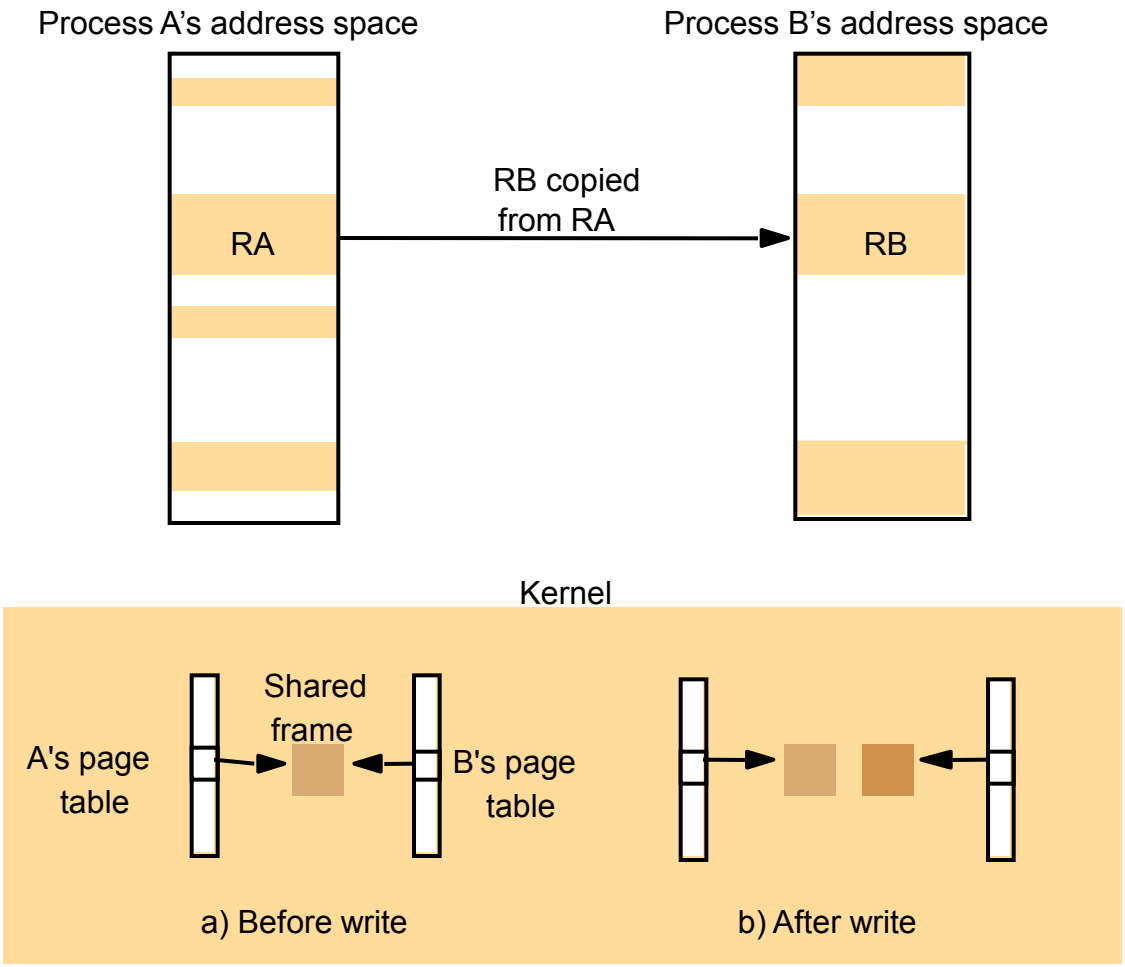


Figure 7.5
Client and server with threads

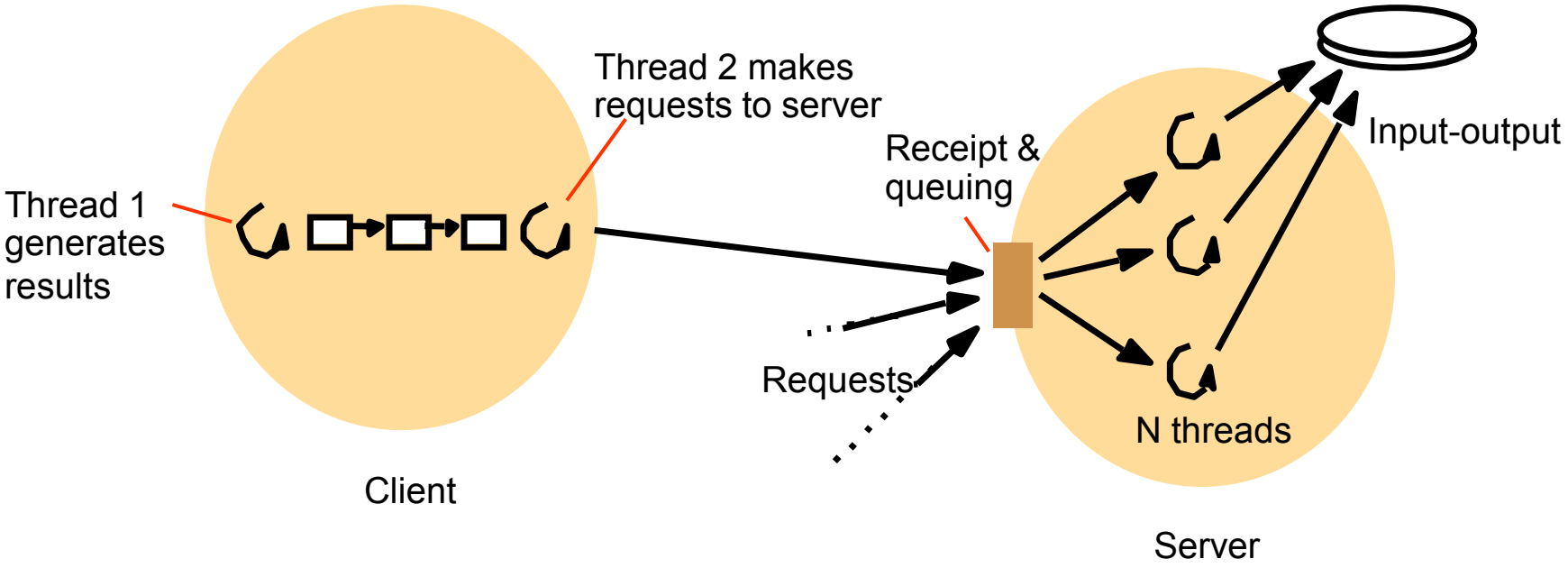
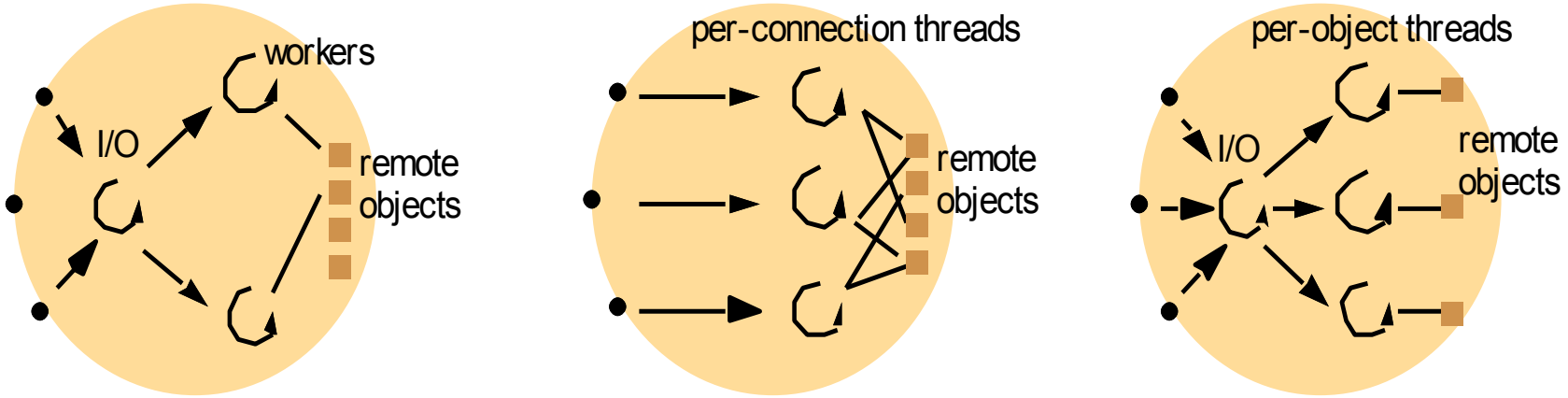


Figure 7.6
Alternative server threading architectures (see also Figure 7.5)



a. Thread-per-request

b. Thread-per-connection

c. Thread-per-object

Figure 7.7
State associated with execution environments and threads

Execution environment	*Thread*
Address space tables	Saved processor registers
Communication interfaces, open files	Priority and execution state (such as *BLOCKED)*
Semaphores, other synchronization objects	Software interrupt handling information
List of thread identifiers	Execution environment identifier
Pages of address space resident in memory; hardware cache entries	

Figure 7.8
Java thread constructor and management methods

Thread(ThreadGroup group, Runnable target, String name)
: Creates a new thread in the *SUSPENDED* state, which will belong to *group* and be identified as *name*; the thread will execute the *run()* method of *target*.

setPriority(int newPriority), getPriority()
: Set and return the thread's priority.

run()
: A thread executes the *run()* method of its target object, if it has one, and otherwise its own *run()* method (*Thread* implements *Runnable*).

start()
: Change the state of the thread from *SUSPENDED* to *RUNNABLE*.

sleep(int millisecs)
: Cause the thread to enter the *SUSPENDED* state for the specified time.

yield()
: Causes the thread to enter the *READY* state and invoke the scheduler.

destroy()
: Destroy the thread.

Figure 7.9
Java thread synchronization calls

thread.join(int millisecs)

Blocks the calling thread for up to the specified time until *thread* has terminated.

thread.interrupt()

Interrupts *thread*: causes it to return from a blocking method call such as *sleep()*.

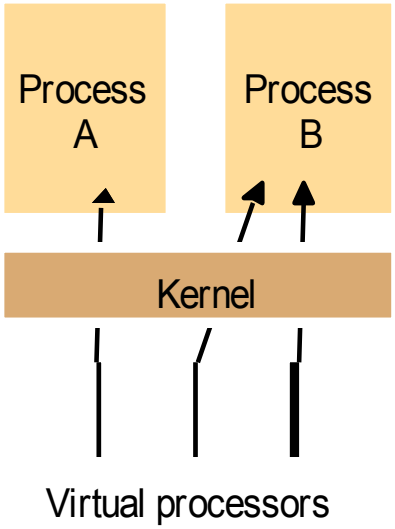
object.wait(long millisecs, int nanosecs)

Blocks the calling thread until a call made to *notify()* or *notifyAll()* on *object* wakes the thread, or the thread is interrupted, or the specified time has elapsed.

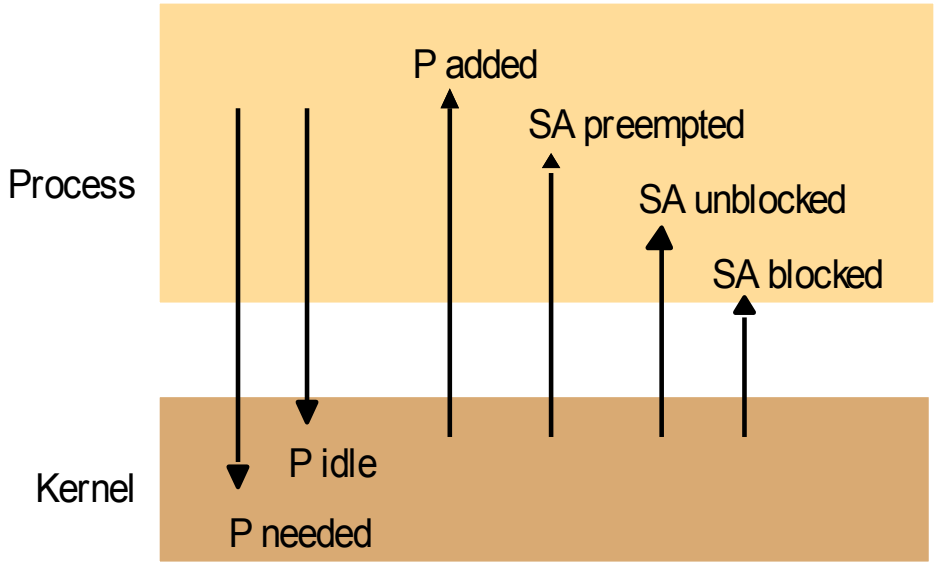
object.notify(), object.notifyAll()

Wakes, respectively, one or all of any threads that have called *wait()* on *object*.

Figure 7.10
Scheduler activations



A. Assignment of virtual processors to processes



B. Events between user-level scheduler & kernel
Key: P = processor; SA = scheduler activation

Figure 7.11
Invocations between address spaces

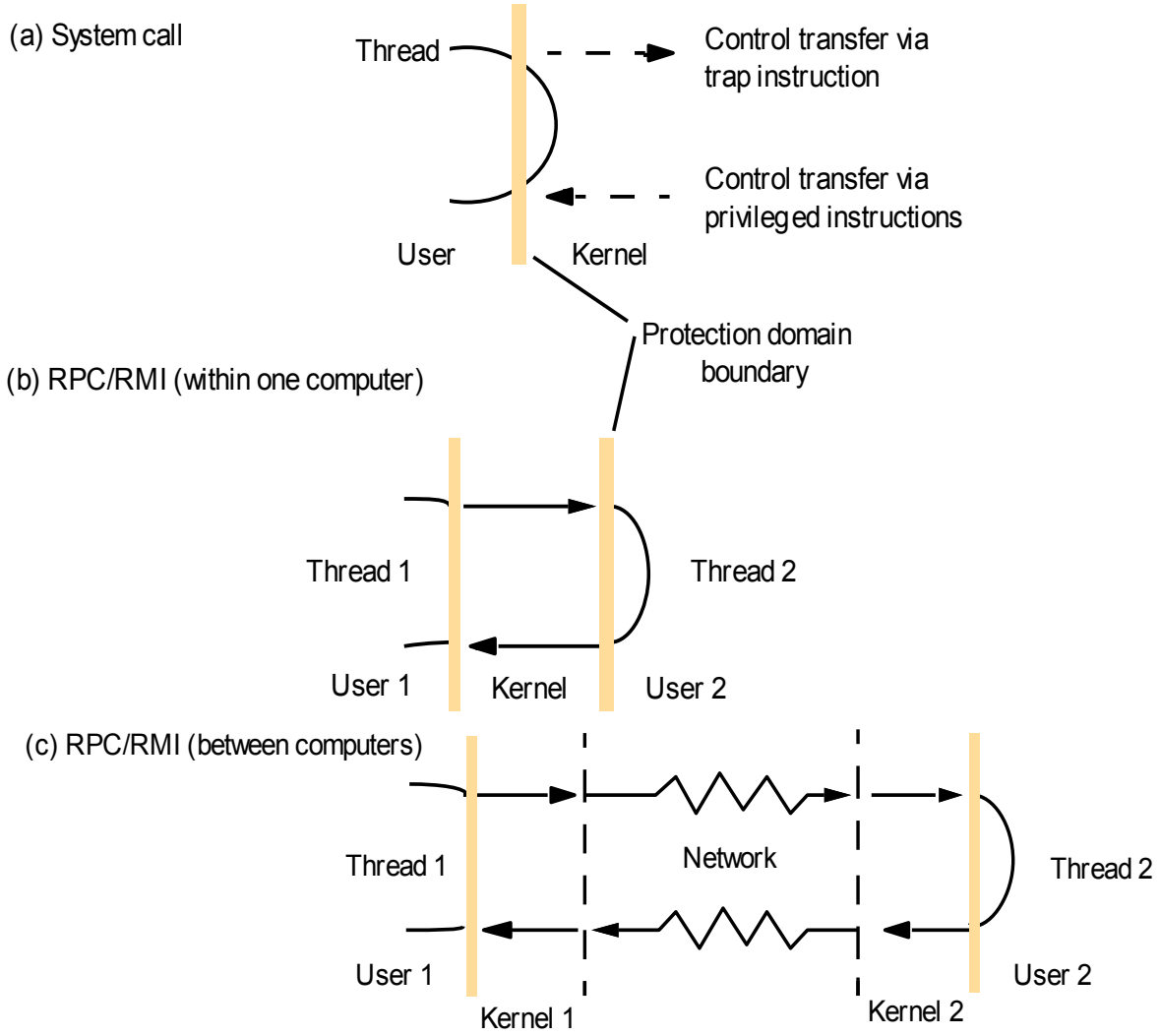


Figure 7.12
RPC delay against parameter size

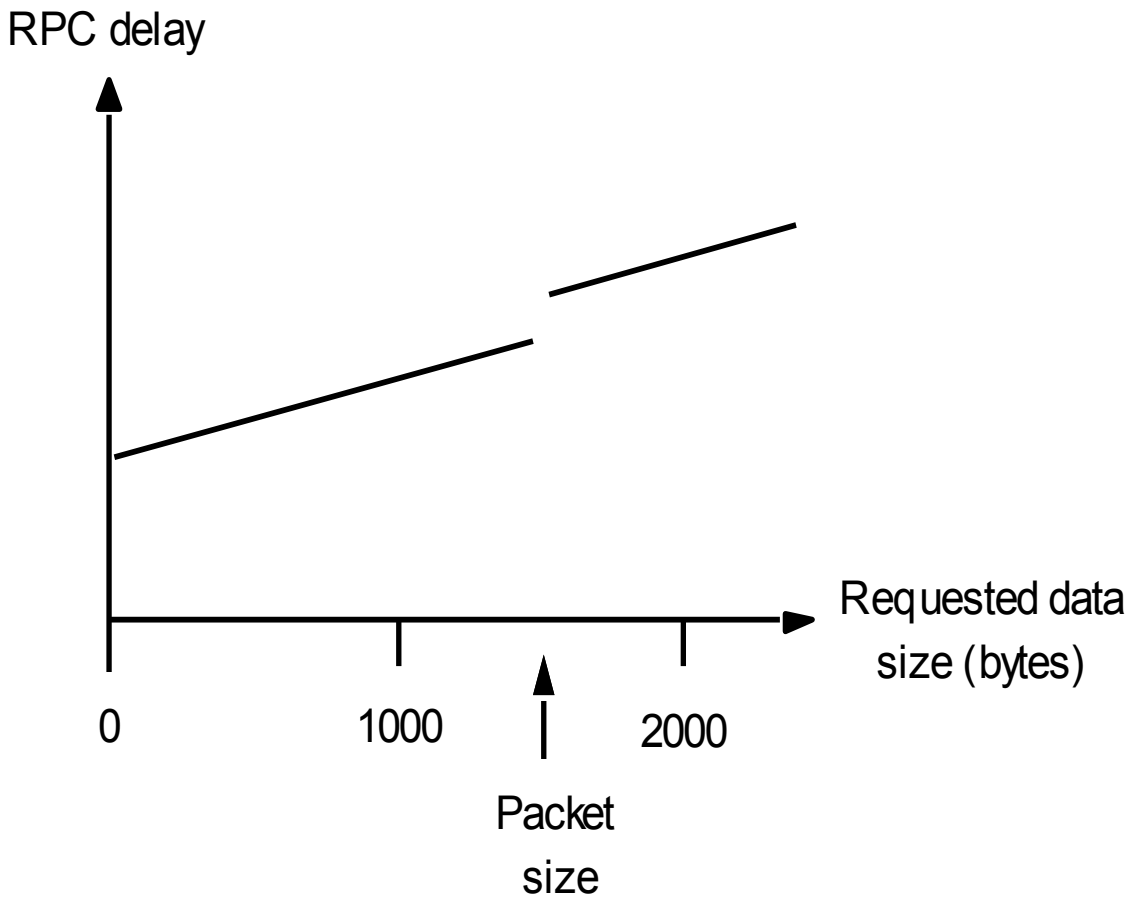


Figure 7.13
A lightweight remote procedure call

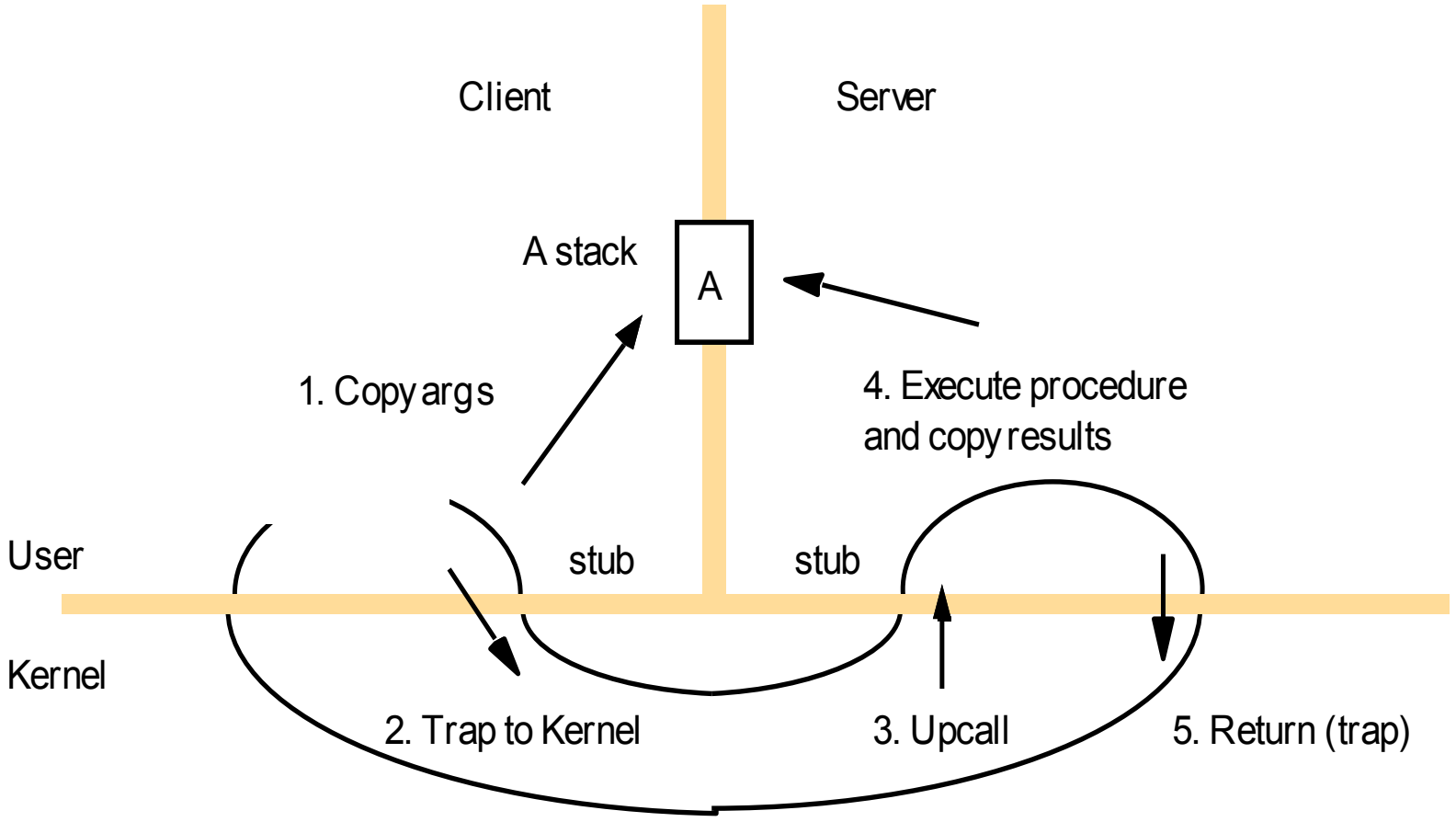


Figure 7.14
Times for serialized and concurrent invocations

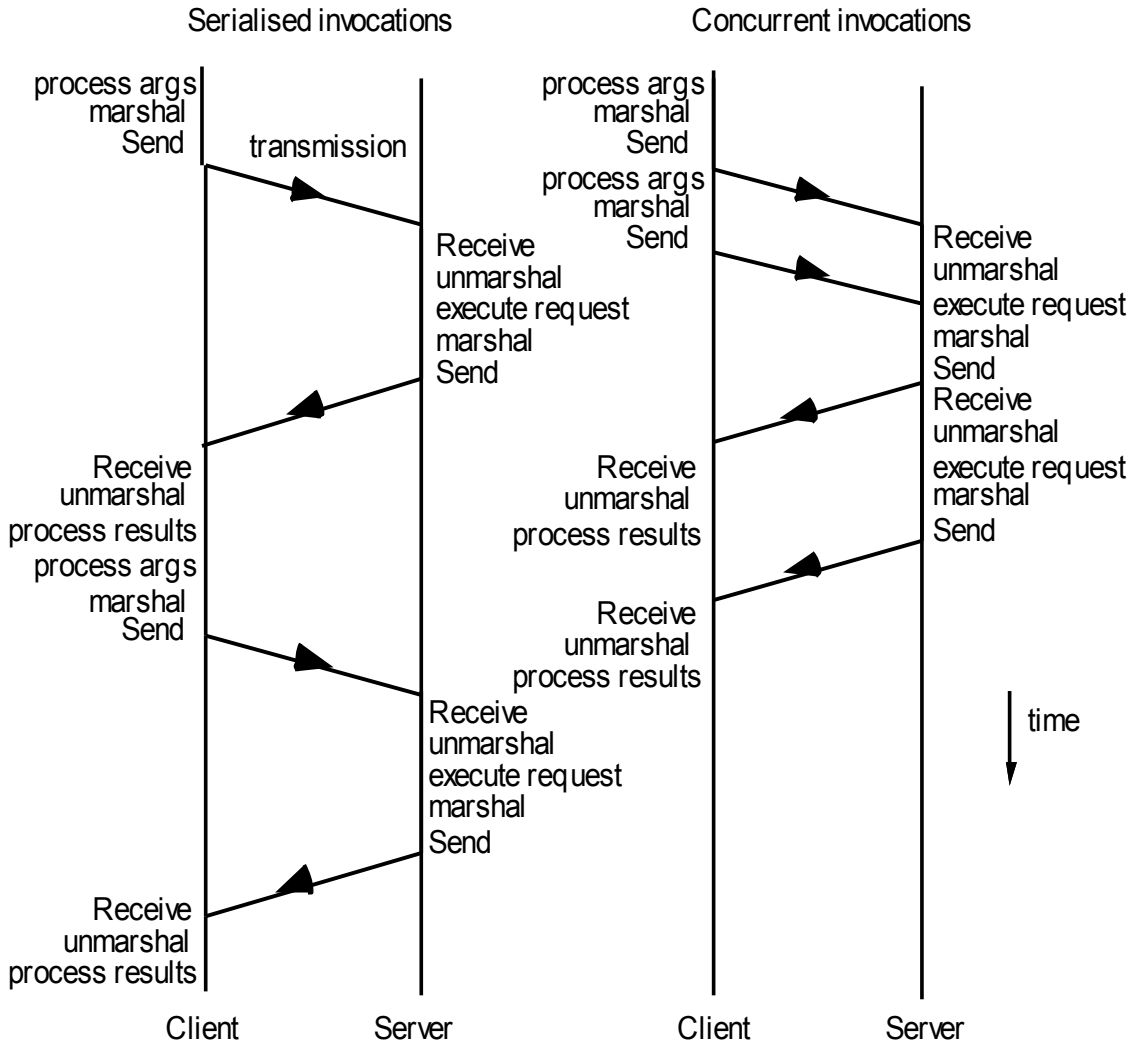


Figure 7.15
Monolithic kernel and microkernel

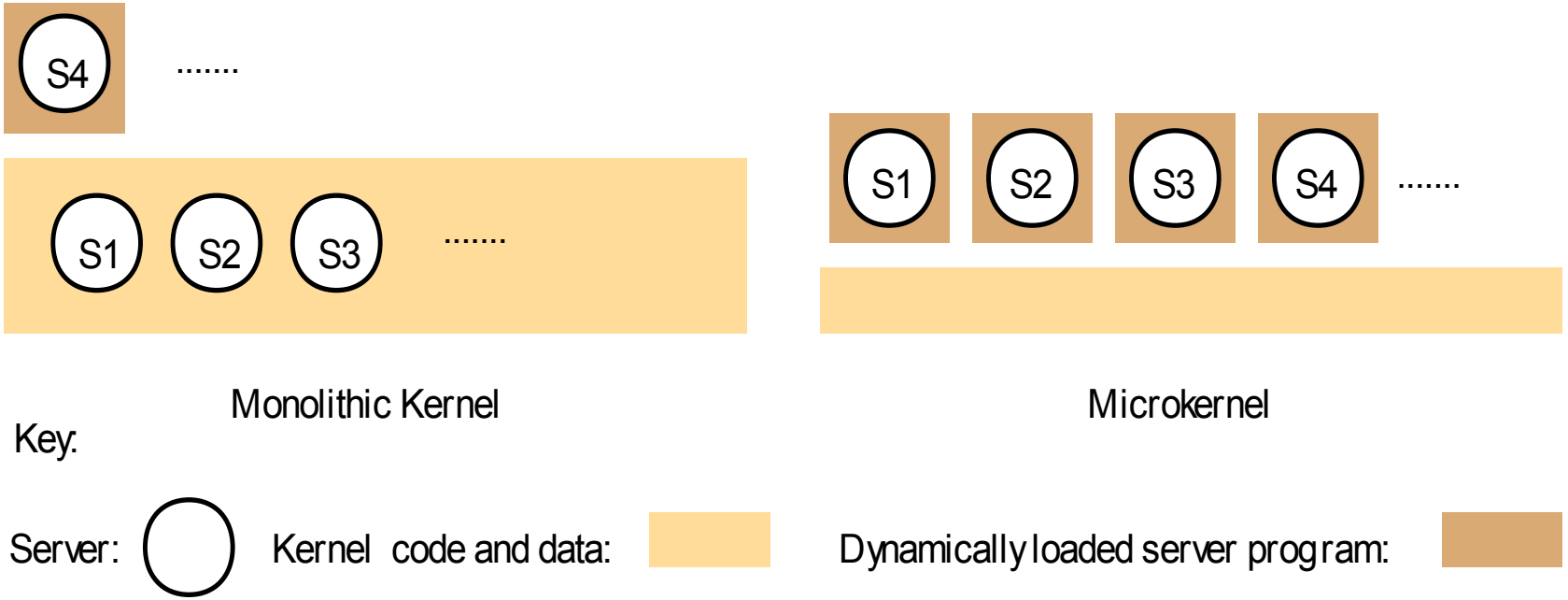
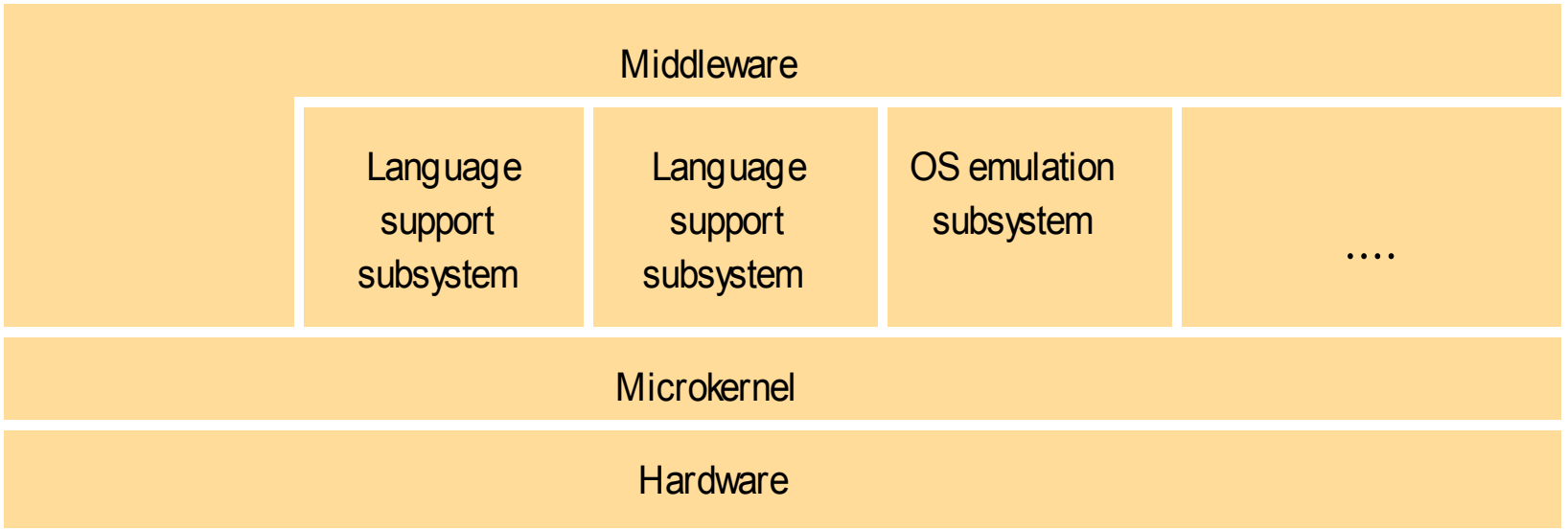


Figure 7.16
The role of the microkernel



The microkernel supports middleware via subsystems

Figure 7.17
The architecture of Xen

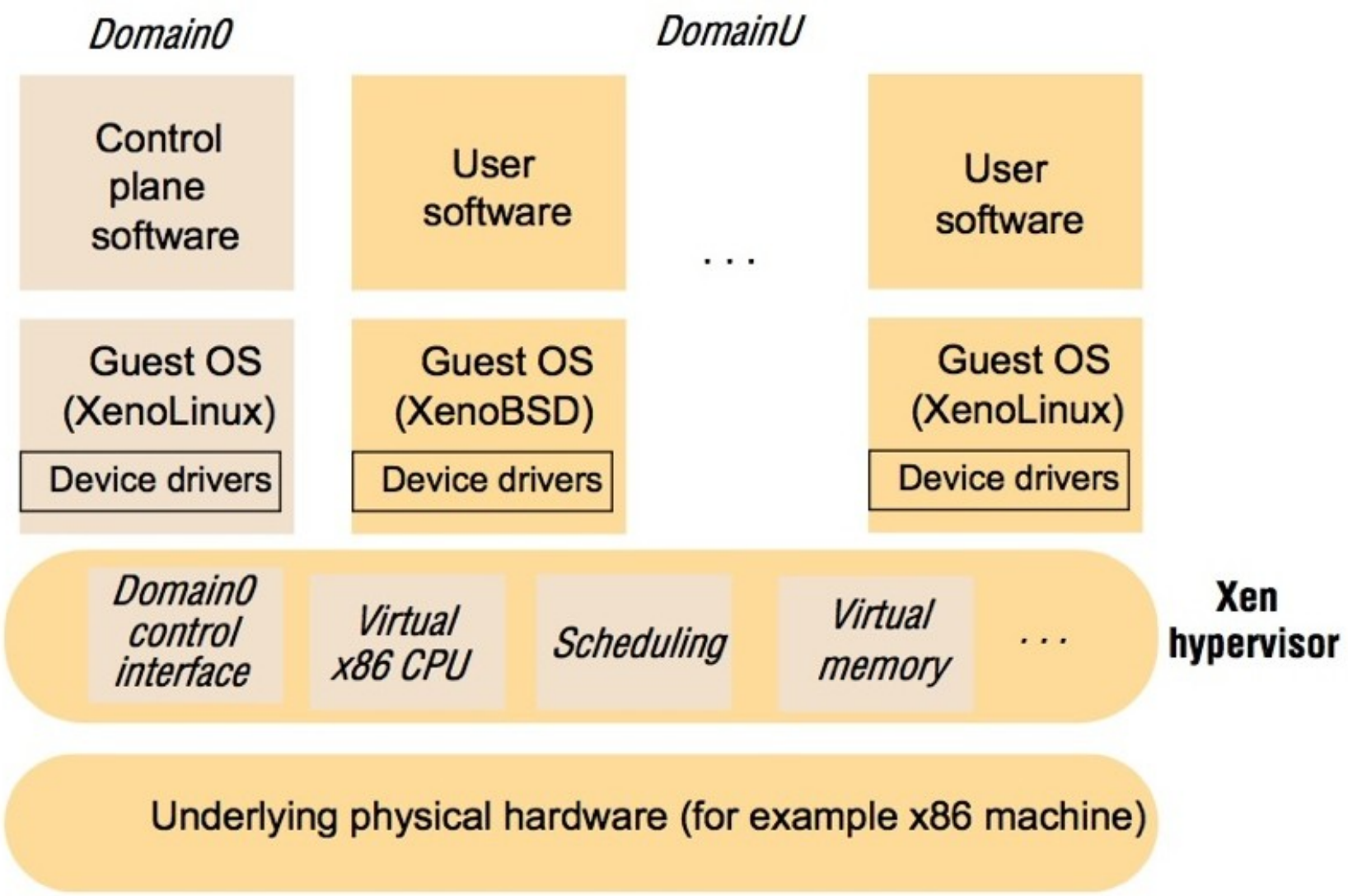


Figure 7.18
Use of rings of privilege

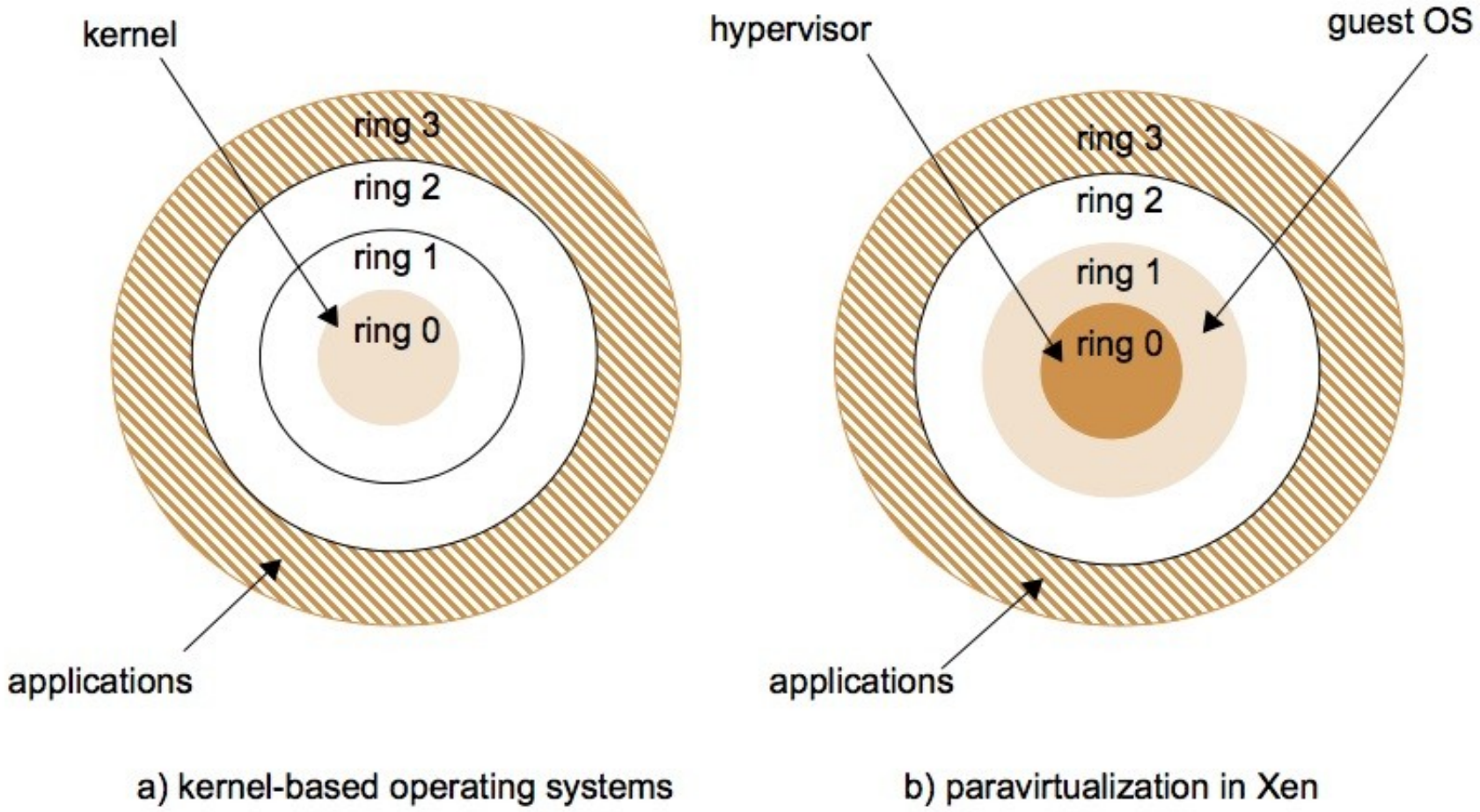


Figure 7.19
Virtualization of memory management

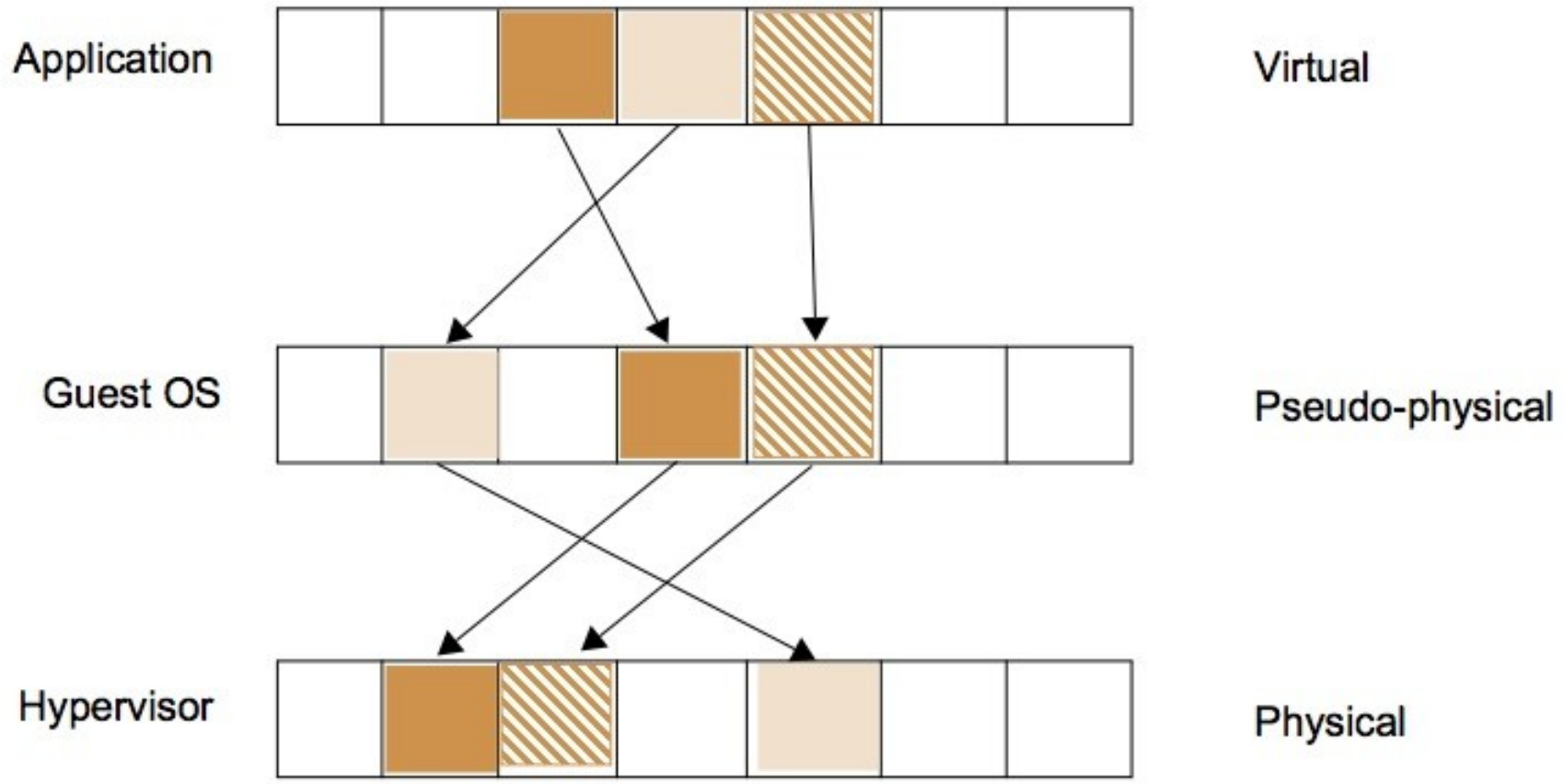


Figure 7.20
Split device drivers

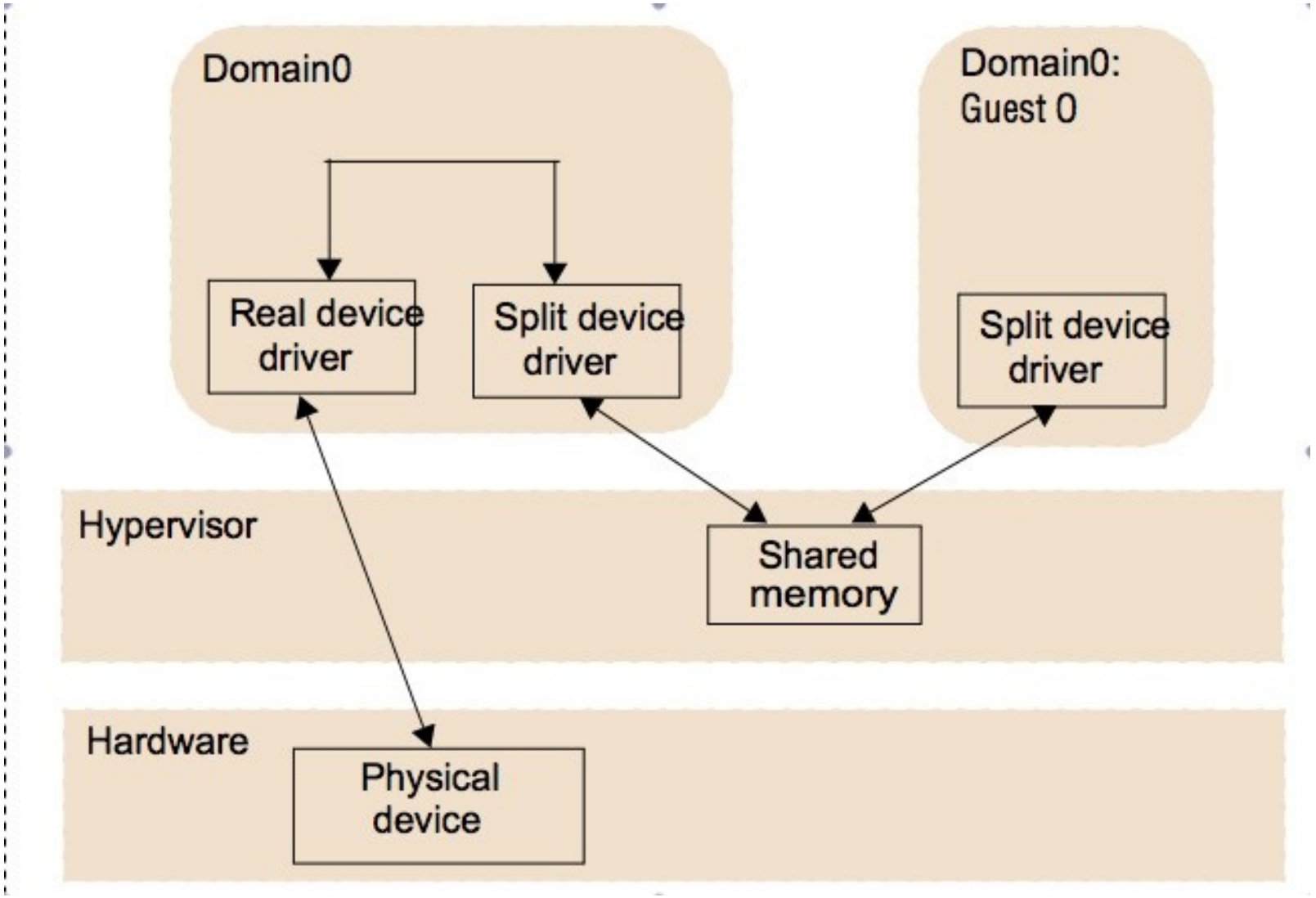


Figure 7.21
I/O rings

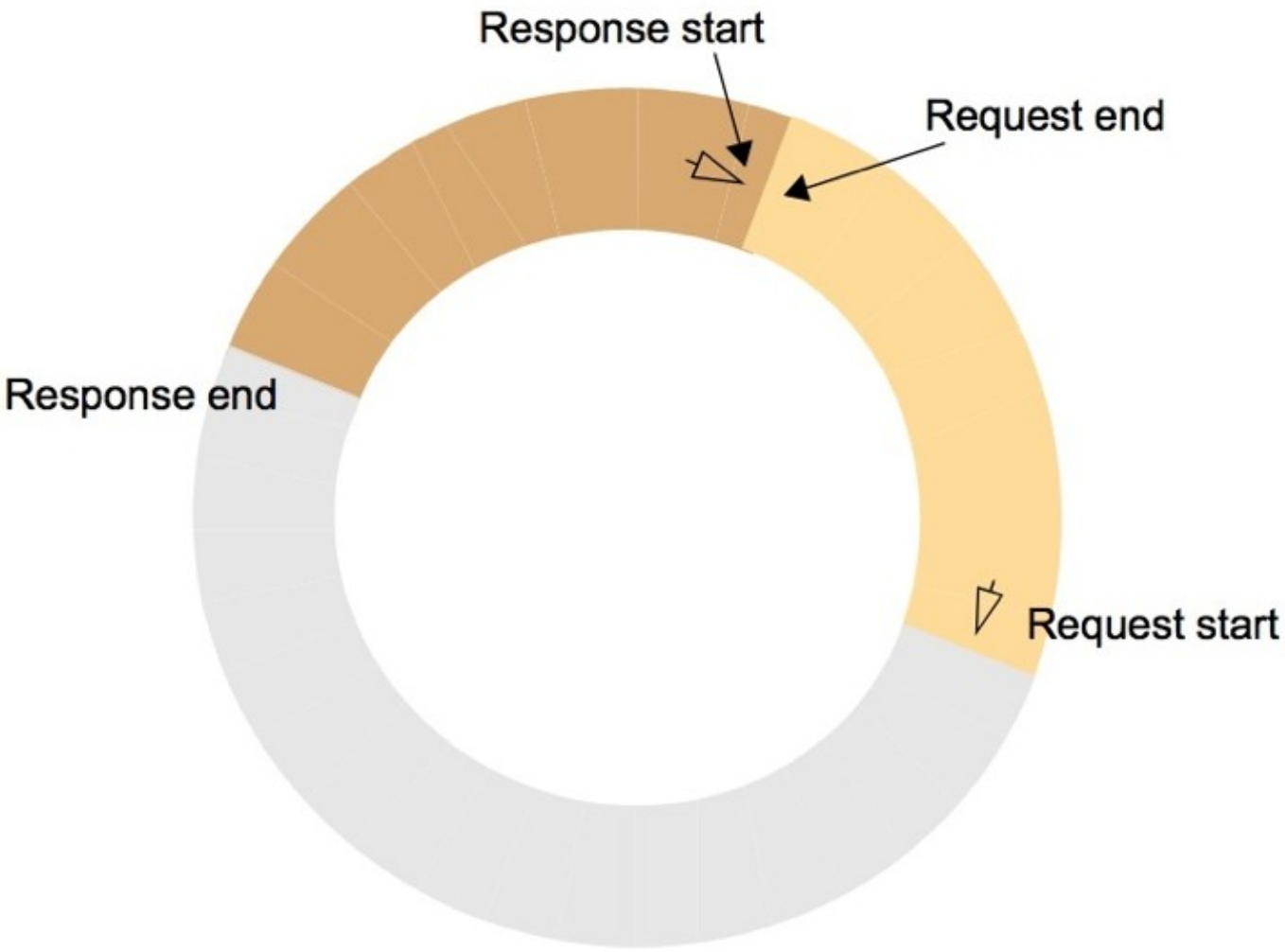


Figure 7.22
The XenoServer Open Platform Architecture