

Chapter 12

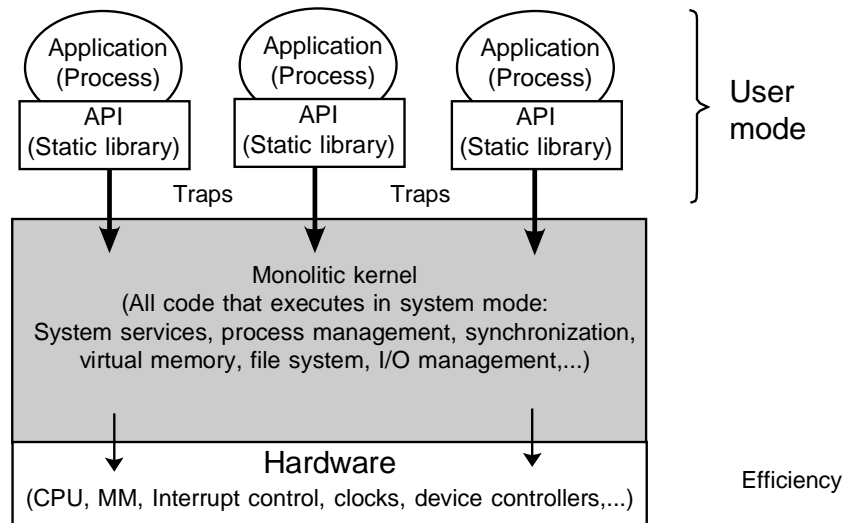
Windows NT/2000

Table of contents:

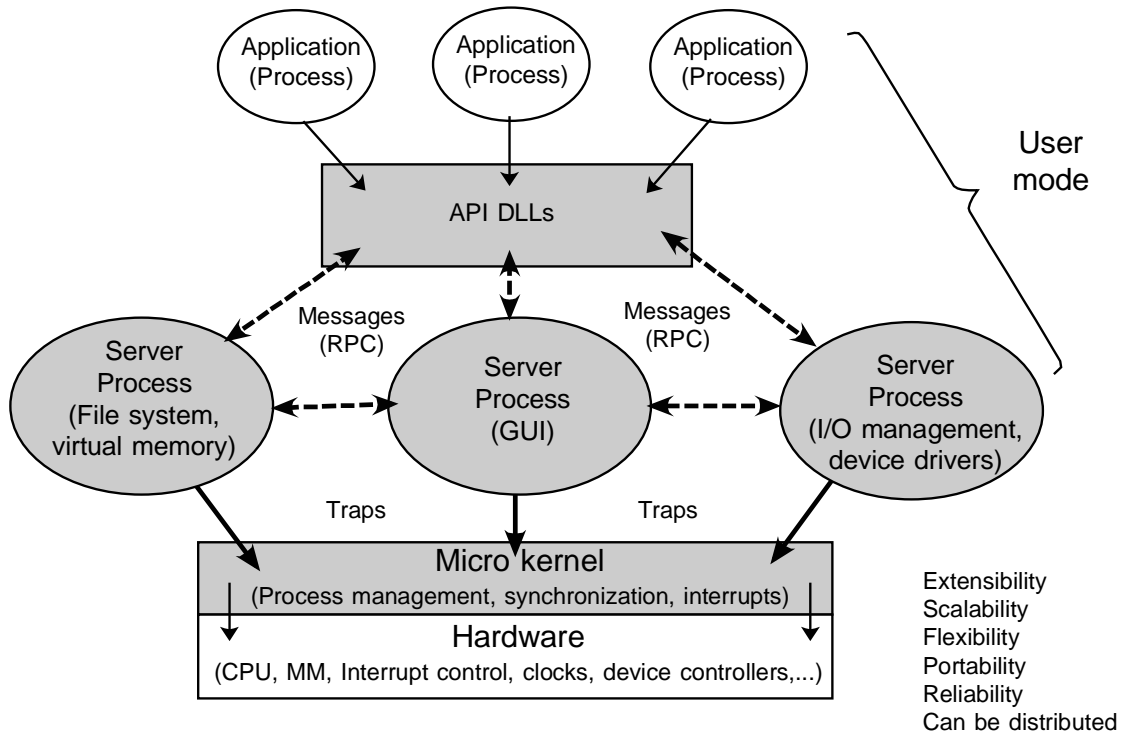
12.1 WINDOWS NT ARCHITECTURE	12-1
12.1.1 Client Server Architecture	12-1
12.1.2 Modified Microkernel Architecture	12-2
12.1.3 Windows NT Operating System	12-3
12.2 WINDOWS NT MECHANISMS	12-4
12.2.1 Local Procedure Calls	12-6
12.2.2 Deferred Procedure calls	12-8

(This chapter is unfinished)

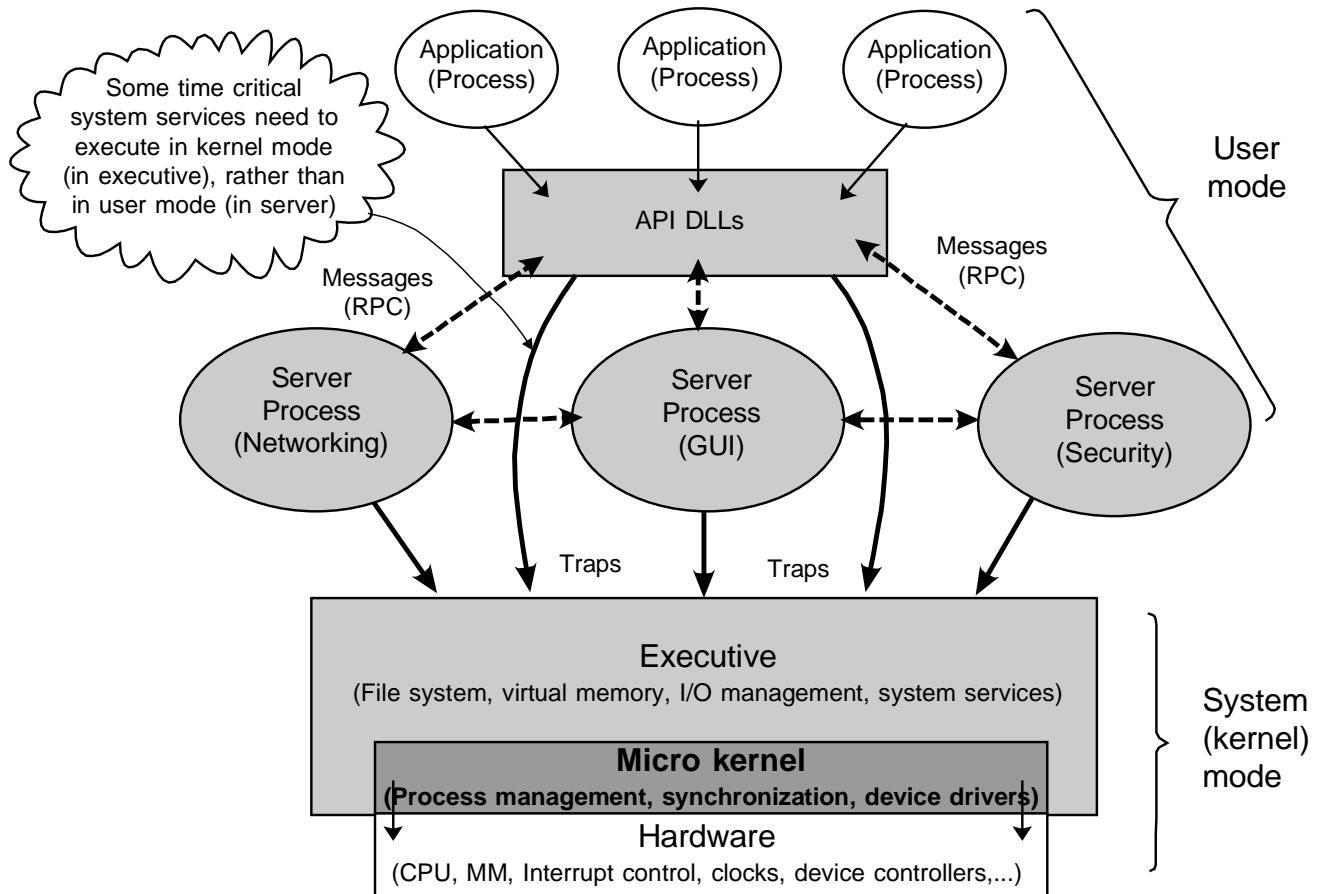
MONOLITIC KERNEL ARCHITECTURE



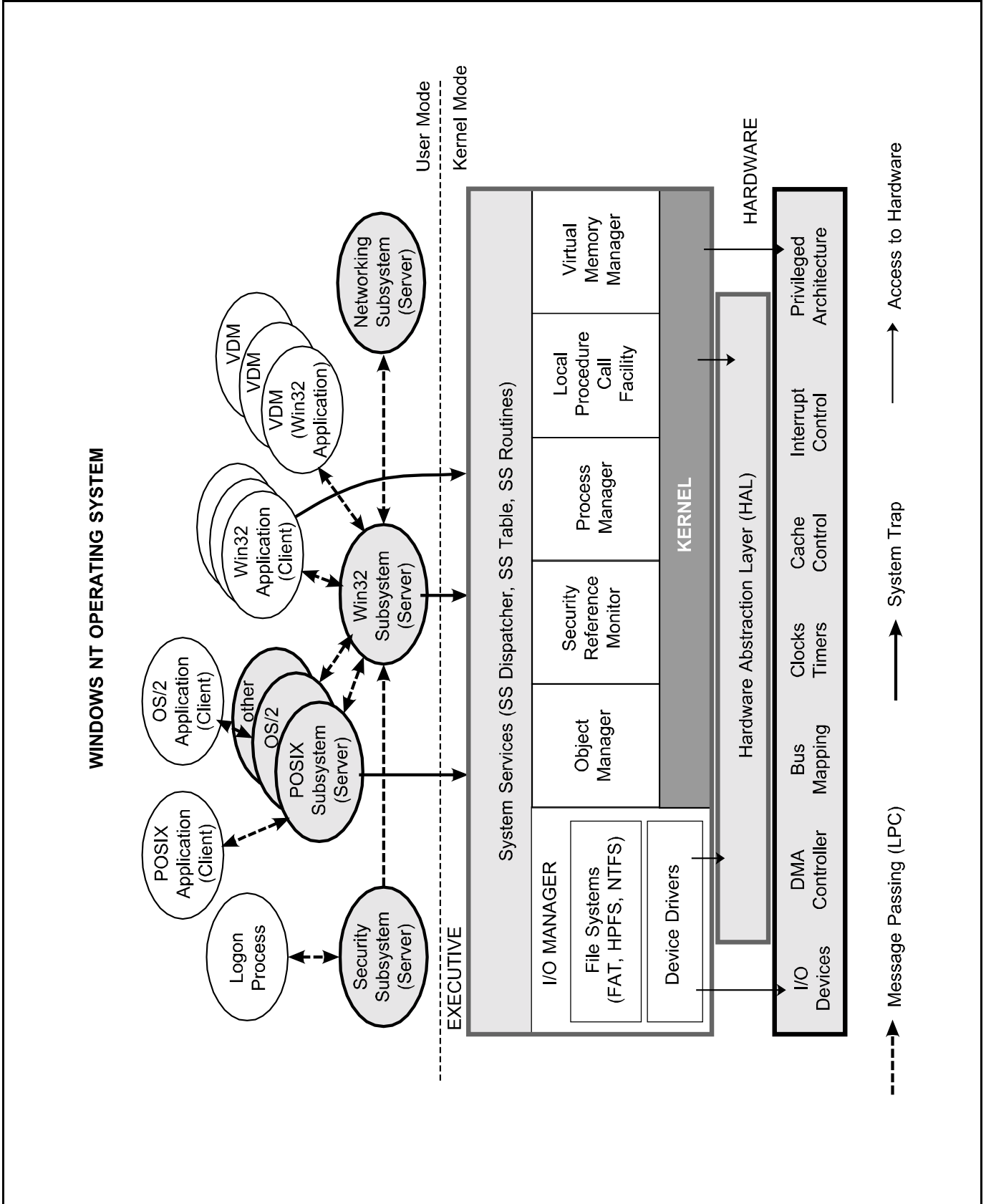
MICRO KERNEL ARCHITECTURE (CLIENT-SERVER MODEL)



MODIFIED MICRO KERNEL ARCHITECTURE

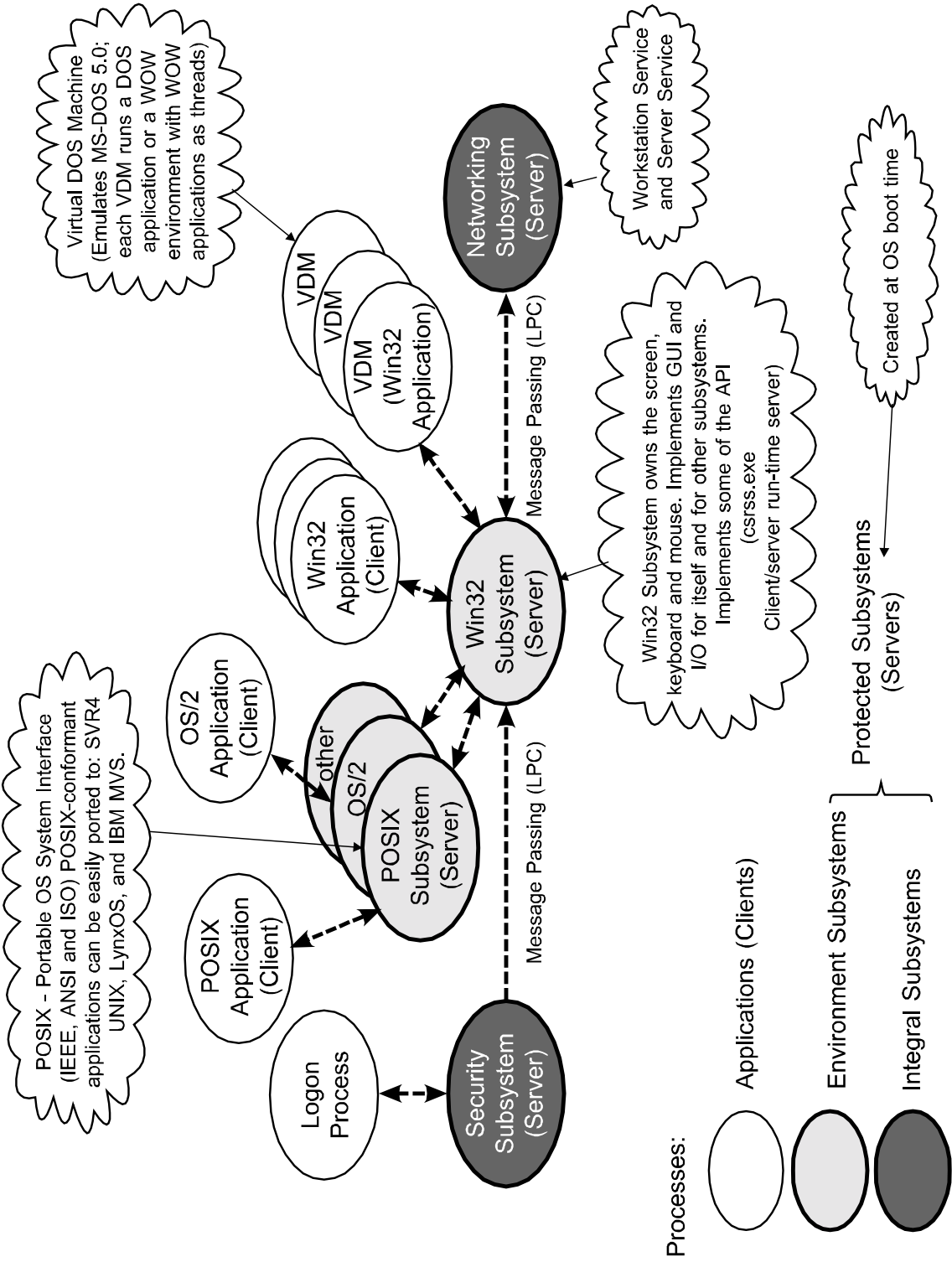


Adds efficiency to the microkernel architecture



WINDOWS NT OPERATING SYSTEM (Cont.)

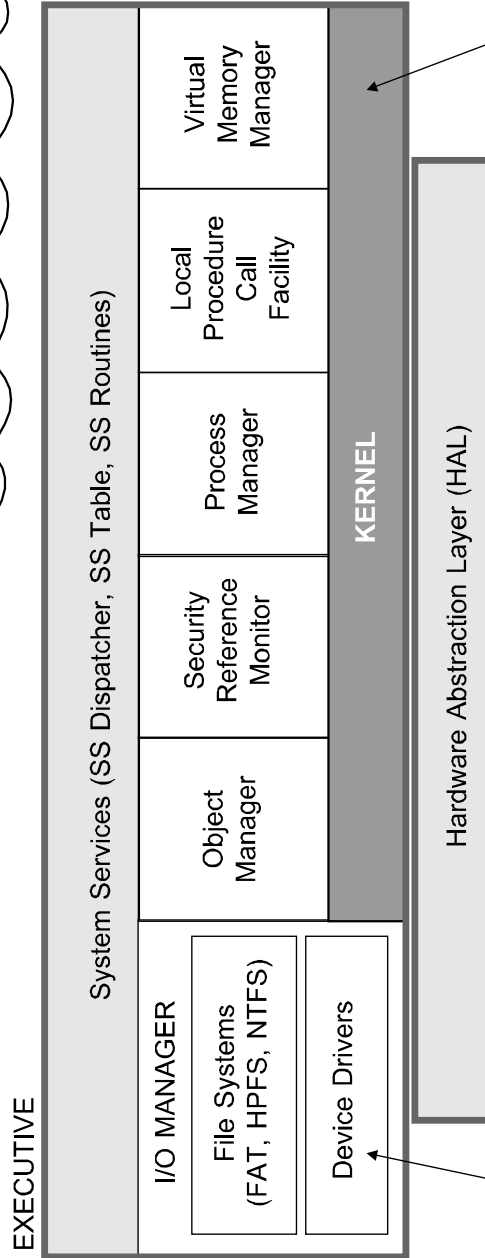
System Components which execute in user mode



WINDOWS NT OPERATING SYSTEM (Cont.)

Windows NT is a layered, client-server oriented OS with micro kernel. Client-server orientation means that the major part of the OS is implemented through a set of user-mode processes (protected subsystems, servers) which ensure integrity of system data and scalability. The client-server approach also facilitates the distributed processing. Micro kernel means that the kernel-mode components are as much as possible kept out of the kernel itself, thus leaving only the architecture dependent functions.

System Components which execute in kernel mode



TCP/IP stack is currently implemented as a network driver, but the tendency is to move it to user mode components.

Exports virtual machine interface that is used by the kernel, device drivers and some other executive components. Abstract model of any hardware that is not CPU. The use of HAL makes the kernel and the device drivers binary compatible across platforms.

Idealized view of the CPU itself, so upper layers of the OS can be architecture independent. Does thread scheduling, interrupt handling, thread synchronization, message passing and power failure recovery. Never paged out. So far supported Intel 386 and above, MIPS R4000, DEC Alpha, PowerPC.

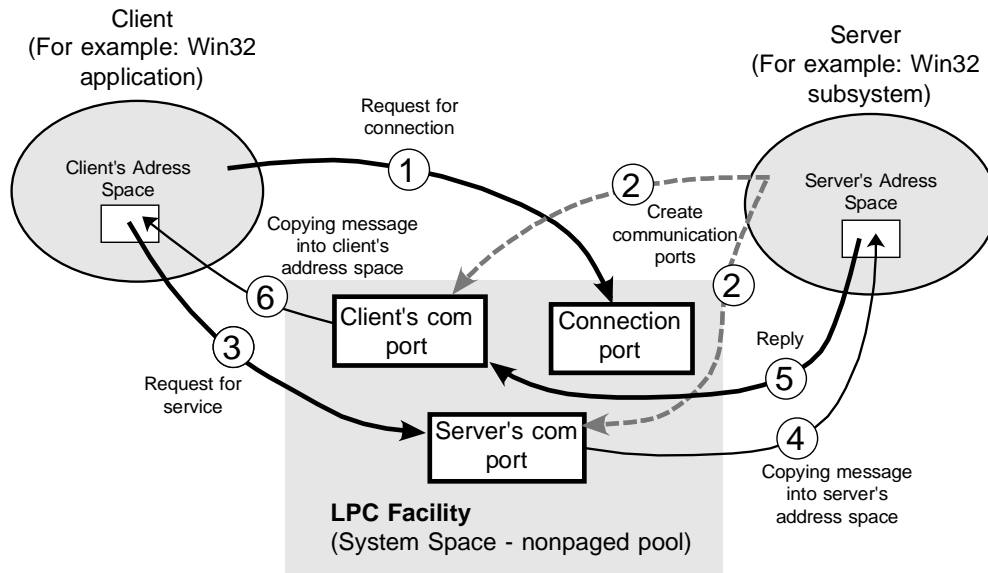
LOCAL PROCEDURE CALL

Generally, the remote procedure call (RPC) is used to communicate the protected sub-systems in a client/server operating system architecture. RPC is transparent to machine boundaries, i.e. it can perform IPC across the network. However, RPC has a significant time overhead and appears to be inefficient. Therefore the Windows NT uses accelerated local procedure calls (LPC) which mimic RPC but only if the server processes are on the same machine.

There are three types of LPC:

- Fixed messages
- Messages with variable length
- Quick LPC

Fixed Messages



Comments:

- (1) Ports are essentially message queues, Connection port is permanent while the communication ports are created by the server. Once they are created they form a communication channel.
- (2) Messages have fixed length of 256 bytes.
- (3) At any time LPC facility has access to the client's or server's address space, but not at the same time.
- (4) The overhead for one message passing includes: two message copies and two process context switches.

LOCAL PROCEDURE CALL (Cont.)

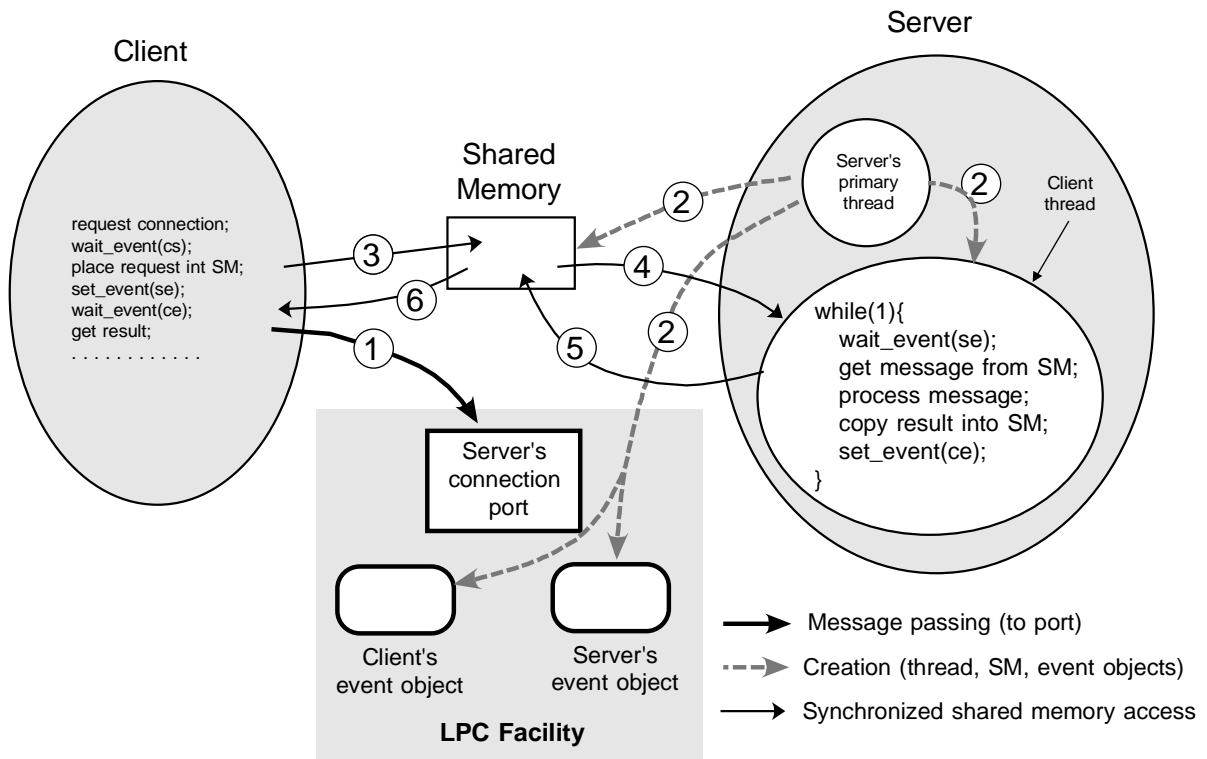
Messages With Variable Length

Used for longer messages. Extension of the previous LPC type - the data are passed via shared memory (mapped file object, also called section object), while the pointer and size information are passed through fixed message channel.

Since there can be many threads that communicate with the server, the messages need to be identified. Therefore the fixed messages also include the thread ID and the message serial number.

Quick LPC

Optimized version of LPC. Doesn't use communication ports. The connection port is used to establish a contact between the client and the server. In response, server creates: a client thread, a shared memory segment (64 KB), and an event pair. The shared memory is used for message passing, while the event pair is for synchronization. Message passing doesn't use copying of messages from port to address space and viceversa, it consists in shared memory access.

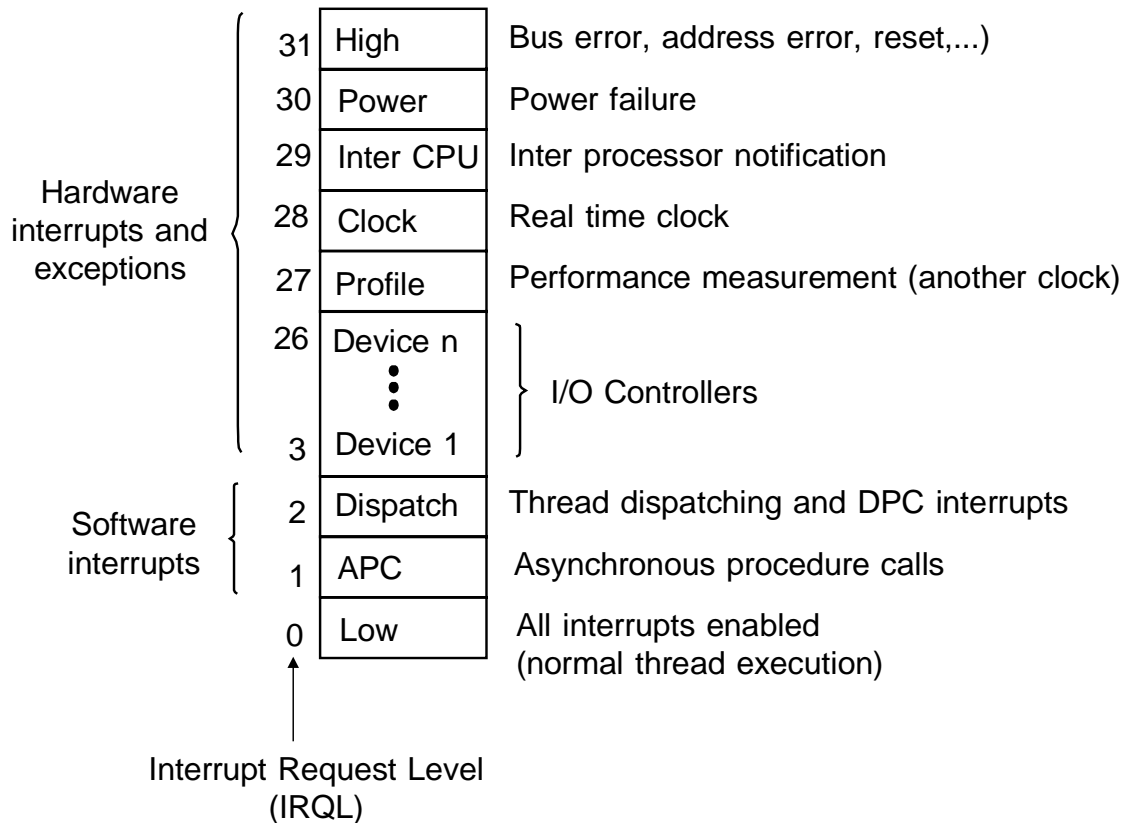


DEFERRED PROCEDURE CALL

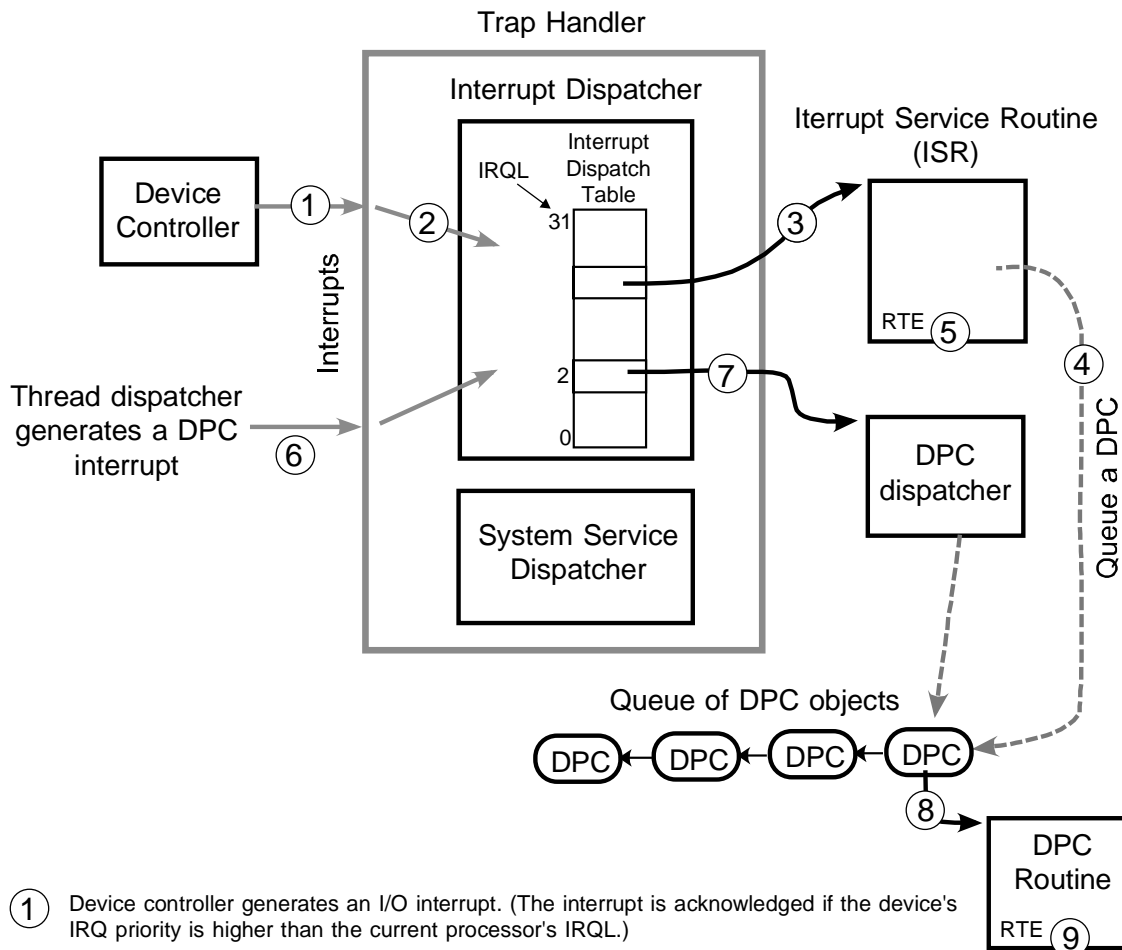
Some actions of an Interrupt Service Routine (ISR) are very critical, while some are less critical. Normally, the ISR executes at an elevated processor's interrupt request level (IRQL) which inhibits all interrupts at the same or lower level. This can degrade the system performance if the execution of ISR takes longer. Therefore, Windows NT introduces deferred procedure calls (DPC) to execute the less important part of an ISR later, when there are no pending hardware interrupts.

Interrupt Request Levels

Example: Ix86 architecture:



DEFERRED PROCEDURE CALL (Cont.)



- ① Device controller generates an I/O interrupt. (The interrupt is acknowledged if the device's IRQ priority is higher than the current processor's IRQL.)
- ② The interrupt is trapped by the trap handler, which immediately raises the processor's IRQL to the level of the interrupting source. Trap handler saves the machine state (trap frame) and calls the interrupt dispatcher which uses the IDT to locate the ISR.
- ③ The ISR executes at the elevated IRQL a critical code that must not be interrupted by other devices at the same or lower interrupt priority.
- ④ If there is a part of the interrupt service that is less critical, the ISR queues a DPC object into DPC queue (DPC object basically contains the address of the DPC routine which can be executed later.) Queuing a DPC object effectively means issuing a DPC interrupt, which cannot be granted immediately since the ISR executes at IRQL higher than that of the DPC level.
- ⑤ After ISR finishes execution it lowers the IRQL to the Dispatch/DPC level and returns to the interrupted thread. This allows that all interrupts can be granted, before the DPC routine is executed.
- ⑥ Since the DPC interrupt is pending, the dispatcher removes a DPC object as soon as the IRQL is lowered, and calls the interrupt dispatcher, which uses IDT to locate the DPC dispatcher, which in turn invokes the DPC routine.
- ⑦ When DPC routine finishes the execution it returns to the interrupted thread. If there are more DPC waiting in the queue, they will all be executed, before the execution finally gets back to the originally interrupted thread.
- ⑧
- ⑨