IOS Architecture Overview

IOS was originally designed to be a small, embedded system for early Cisco routers. At that time, routers themselves were viewed mostly as hardware appliances—little distinction was made between the hardware and software. In fact, initially IOS wasn't even called "IOS"; it was just referred to as "the OS" that ran a Cisco router.

As routed networks gained popularity, demand rose for routers to support an increasing number of protocols and to provide other functionality, such as bridging. Cisco responded to this enormous demand by adding new features into the router software, resulting in the multi-functional routing and bridging software IOS is today. Interestingly, although the functionality of IOS has grown considerably, the basic operating system architecture has remained mostly the same.

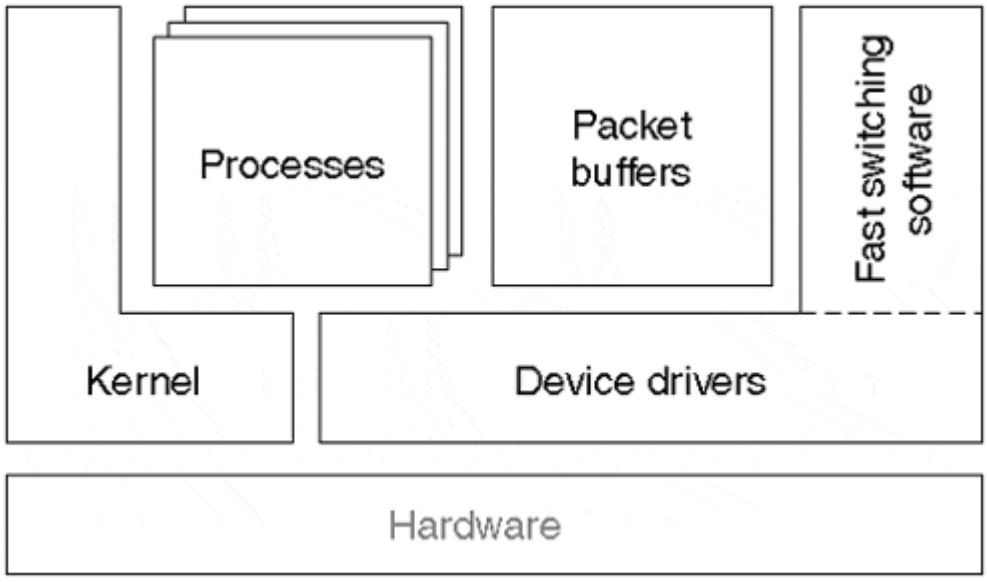
Compared to other operating systems, IOS has a fairly simple architecture. Like most small, embedded systems, IOS was designed to be lean and mean to stay within the memory and speed constraints of the original platforms.

Early routers had limited amounts of memory to share between the software and data (such as routing tables). To limit the size of the executable image, IOS was designed to provide only essential services.

Speed was also a major consideration in the design. To maximize the router's capability to quickly switch packets, a conscious effort was made to design the operating system with a minimum of operational overhead and to allow maximum CPU bandwidth for packet switching. Many safeguards, such as inter-thread memory protection mechanisms, found in other operating systems are missing from IOS because of the CPU and the memory overhead they introduce. In general, the IOS design emphasizes speed at the expense of extra fault protection.

Figure 1-1 shows a conceptual diagram of the IOS architecture.

Figure 1-1. IOS Architecture



As Figure 1-1 illustrates, IOS has five major elements:

- **Processes—**

Individual threads and associated data that perform tasks, such as system maintenance, switching packets, and implementing routing protocols.

- **Kernel—**

  Provides basic system services to the rest of IOS, such as memory management and process scheduling. It provides hardware (CPU and memory) resource management to processes.

- **Packet Buffers—**

  Global memory buffers and their associated management functions used to hold packets being switched.

- **Device Drivers—**

  Functions that control network interface hardware and peripherals (such as a flash card). Device drivers interface between the IOS processes, the IOS kernel, and the hardware. They also interface to the fast switching software.

- **Fast Switching Software—**

  Highly optimized packet switching functions.

Each of these elements, except fast switching software, is discussed in more detail in the following sections. The fast switching software is discussed later in Chapter 2, "Packet Switching Architecture." Before we investigate these architectural elements, let's first look at how IOS organizes memory.

Index terms contained in this section

fast switching software
IOS design influence 2nd