are connected to two CPUs, so each device is controlled by a pair of I/O processes running in those CPUs: a *primary* process that performs the work and a *backup* process that tracks the state of the primary process and waits to fail or hand over control to it voluntarily ("primary switch").

The main issue in the choice of primary CPU is the CPU load, which needs to be balanced manually. For example, if you have six devices connected between CPUs 2 and 3, you would probably put the primary process of three of them in CPU 2, and the primary process of the other three in CPU 3.

Process Pairs

The concept of process pairs is not limited to I/O processes. It is one of the cornerstones of the fault-tolerant approach. To understand the way they work, we need to understand the way messages are passed in the system.

Message System

As we've seen, the biggest difference between the T/16 and conventional computers is the lack of any single required component. Any one part of the system can fail without bringing down the system. This makes it more like a network than a conventional shared memory multiprocessor machine.

This has far-reaching implications for the operating system design. A disk could be connected to any 2 of 16 CPUs. How do the others access it? Modern networks use file systems such as NFS or CIFS, which run on top of the network protocols, to handle this special case. But on the T/16 it isn't a special case; it is the norm.

File systems aren't the only thing that require this kind of communication: interprocess communication of all kinds requires it, too.

Tandem's solution to this issue is the *message system*, which runs at a very low level in the operating system. It is not directly accessible to user programs.

The message system transmits data between processes, and in many ways it resembles the later TCP or UDP. The initiator of the message is called the *requestor*, and the object is called the *server*.[‡]

All communication between processes, even on the same CPU, goes via the message system. The following data structures implement the communication:

- Each message is associated with two *Link Control Blocks*, or *LCBs*, one for the requestor and one for the server. These small data objects are designed to fit in a single IPB packet. If more data is needed than would fit in the LCB, a separate buffer is attached.

‡ These names correspond closely in function to the modern terms *client* and *server*.