Buffer Management Using Particles

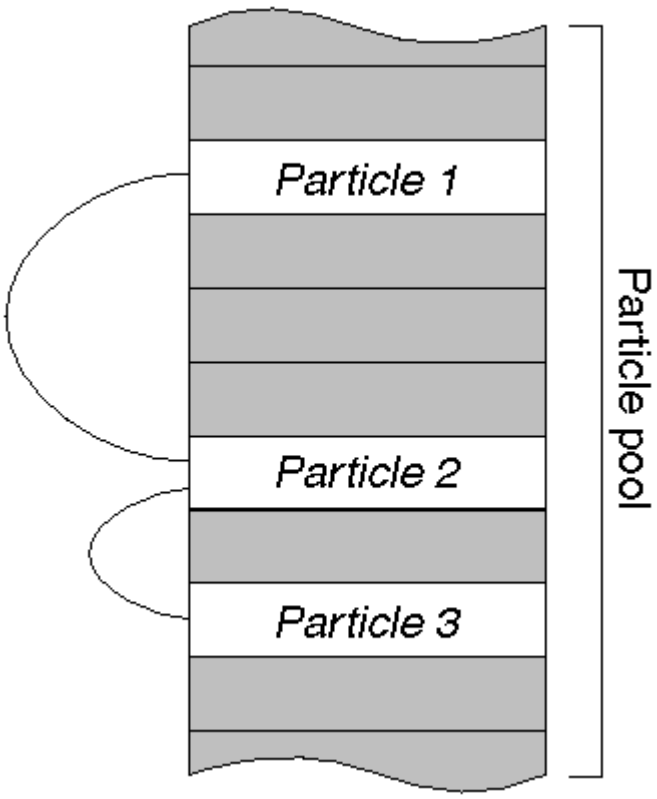
Particle buffering uses a scatter-gather approach for packet buffer memory management. Instead of allocating one piece of contiguous memory for a buffer, particle buffering allocates dis-contiguous (scattered) pieces of memory, called *particles*, and then links them together (gathers) to form one logical packet buffer, called a *particle buffer*. With this scheme, a single packet can be spread across multiple physical buffers.

Particle buffers are built—that is, individual particles are linked together—as each packet is received on an interface, as you'll see when you study the Cisco 7200 router example. This method differs from that of contiguous buffers, which need no assembly before use. As a result, there are no pools of pre-built particle buffers like there are for contiguous buffers. After all, how would IOS know how many particles to add to each buffer? Instead, IOS maintains pools of particles and draws from those pools to build packet buffers as it receives packets. The particles are the basic building blocks and the pools provide the raw materials to build the packet buffers.

Within a given pool, all particles are the same size; so any free particle in the pool can be used to build a buffer without regard to the particle's size. This uniformity simplifies the particle management algorithms and helps contribute to efficient memory use. IOS particle size varies from platform to platform, but within a platform there is usually one primary size. There are exceptions—platforms can have pools with secondary particle sizes as you'll see later in this chapter—but there is usually a primary particle size. The size is chosen so it's large enough to contain the average packet while minimizing wasted memory, typically 512 bytes. This way, most packet buffers can be composed of just one particle.

To understand how the particle buffering scatter-gather approach works, consider the example illustrated in Figure 5-1. In this simple example, there is one particle pool in memory and all packet buffers are built from that pool.

Figure 5-1. Particle Example



Assume that the pool's particle size is 512 bytes and three free particles, labeled P1, P2, and P3, exist in the pool. Now, let's trace what happens when IOS receives a 1200-byte packet.

Step 1. IOS gets the next free particle (P1 in this example) and begins copying the packet data into the particle. The first 512 bytes of the packet are copied.

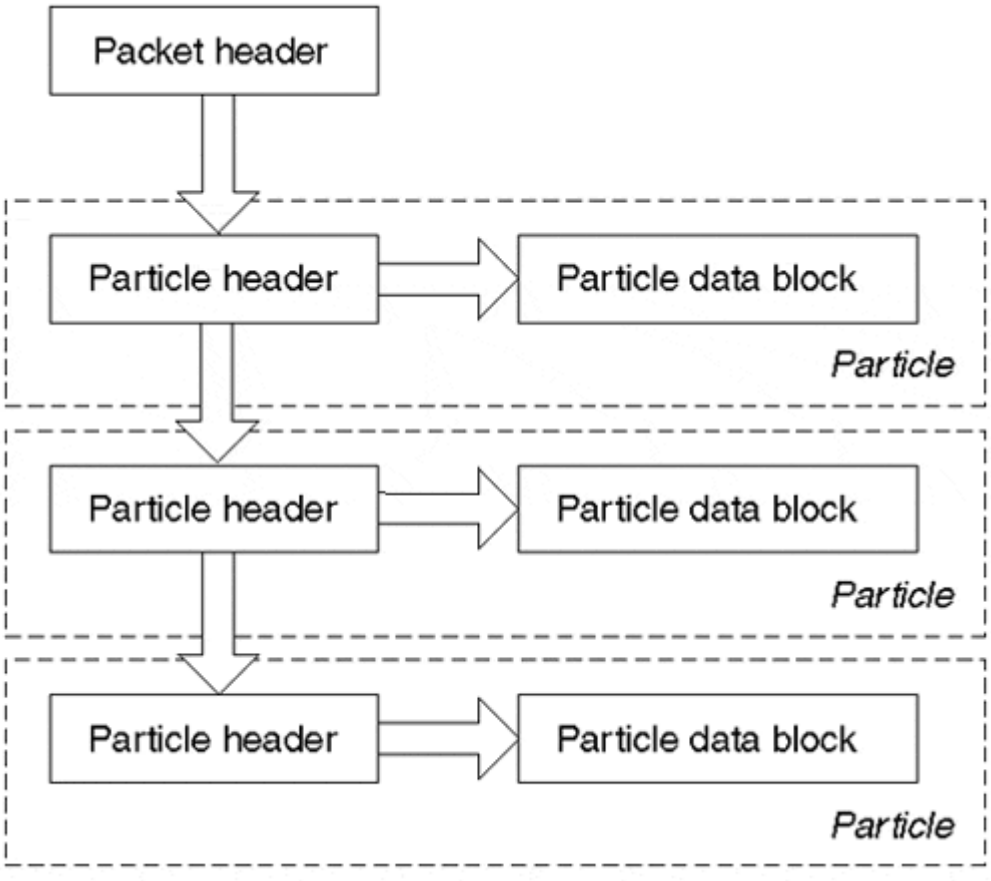
Step 2. When the first particle is filled, IOS moves to the next free particle (P2 in this example), links it to P1, and continues copying packet data into this second particle. The next 512 bytes of the packet are copied.

Step 3. When the second particle is filled, 176 bytes of packet data still are left. So, IOS moves to the next free particle (P3 in this example), links it to P2, and copies the remaining 176 bytes into this third particle.

Upon completion, IOS has copied the entire 1200 bytes into three dis-contiguous pieces of memory that are all logically part of a single packet buffer.

Particle buffers actually consist of more than just the memory that contains packet data. Some amount of overhead is required (in the form of additional components) to maintain data about the packet as well as to link the scattered memory pieces together. Figure 5-2 shows the components of a particle buffer and how each one is related.

Figure 5-2. Particle Buffer Components



Each particle buffer consists of a *packet header* linked to one or more particles. A particle consists of a *particle header* and an associated *particle data block*. The following list examines each of these components in more detail:

- **Packet header—**

  The packet header is the same as the header for contiguous buffers. It contains information about the type and the size of the packet in the buffer as well as pointers to specific fields in the packet. The

packet header also contains a link to the first particle.

- **Particle header—**

  The particle header is a control block containing a link to the particle data block and a link to the next particle in the buffer (if there is one). It also contains other information, such as the number of data bytes in the data block and the pool that owns the particle.

- **Particle data block—**

  The particle data block is the piece of memory containing the actual particle data—basically, this is just a buffer.

Particle Pools

IOS uses the same pool management strategy for particle pools as it does for contiguous packet buffers on shared memory systems. IOS creates a private static particle pool for each interface, and then creates a public dynamic pool, called the *normal pool*, that all interfaces and processes share. The private particle pools use the public pool as a *fall-back* in case they run out of particles.

IOS also creates another public particle pool on most systems, called the *fast switching pool*, containing small 128-byte particles. These particles are used when IOS needs to prepend data to a packet during fast switching—for example, to rewrite a larger media header onto a packet. To prepend the data, IOS must insert a new particle at the beginning of a particle buffer after it already has been linked together. Because the prepended data is usually small, it uses the particles from this special pool.

The partial output from the **show buffers** command in Example 5-1 shows a sample of the public and private particle pools IOS creates. The fast switching pool is labeled **F/S**.

Example 5-1. show buffers Command Output Displays Public and Private Particle Pools

Router-7200#**show buffers** …. Public particle pools: F/S buffers, 128 bytes (total 512, permanent 512): 0 in free list (0 min, 512 max allowed) 512 hits, 0 misses 512 max cache size, 512 in cache Normal buffers, 512 bytes (total 1024, permanent 1024): 1024 in free list (512 min, 2048 max allowed) 0 hits, 0 misses, 0 trims, 0 created 0 failures (0 no memory) Private particle pools: FastEthernet0/0 buffers, 512 bytes (total 400, permanent 400): 0 in free list (0 min, 400 max allowed) 400 hits, 0 fallbacks 400 max cache size, 271 in cache Ethernet1/0 buffers, 512 bytes (total 128, permanent 128): 0 in free list (0 min, 128 max allowed) 128 hits, 0 fallbacks 128 max cache size, 64 in cache Ethernet1/1 buffers, 512 bytes (total 128, permanent 128): 0 in free list (0 min, 128 max allowed) 128 hits, 0 fallbacks 128 max cache size, 64 in cache …

Particle Coalescing

Although particle buffers do improve memory efficiency, they also tend to increase the complexity of the programs that manipulate their contents. For example, with particles, packet fields can be split arbitrarily across particle boundaries. Packet switching methods have to accommodate that possibility.

All fast switching methods, for most protocols, are equipped to handle particle buffers. However, this is not the case for process switching. Because process switching adds little benefit, only complexity, Cisco decided not to equip the process switching method for particle buffers. Process switching still requires a packet to reside in a contiguous buffer. As a result, particle-buffered packets queued to process switching must be transferred to a contiguous buffer through a process is called *coalescing*.

Coalescing essentially involves copying the contents of all the particles in a packet buffer, one by one, into an appropriately sized contiguous buffer. Depending on the platform, this task either can be performed by the main CPU or by a separate DMA engine. The coalescing process works as follows:

Step 1. IOS allocates an appropriately sized contiguous buffer from one of the system buffer pools. IOS must find a buffer large enough for the entire packet or the coalescing process fails.

Step 2. IOS copies the data from each packet buffer particle into the new contiguous buffer.

Step 3. IOS frees all the particles, returning them to their original pool.

Step 4. Finally, IOS unlinks the packet header from the freed particles and links it to the new contiguous buffer.

When the process is complete, the resulting packet looks the same as the original (same packet header) except it resides in a physically contiguous block of memory.

To understand how particle buffers are used in IOS, it's helpful to look at an implementation example. The following sections describe how particle buffers are used on the Cisco 7200 router series, the first platform to support particles.

Last updated on 12/5/2001
Inside Cisco IOS Software Architecture, © 2002 Cisco Press

< BACK Make Note | Bookmark CONTINUE >

Index terms contained in this section

About Us | Advertise On InformIT | Contact Us | Legal Notice | Privacy Policy

reviewed by TRUSTe site privacy statement

© 2001 Pearson Education, Inc. InformIT Division. All rights reserved. 201 West 103rd Street, Indianapolis, IN 46290