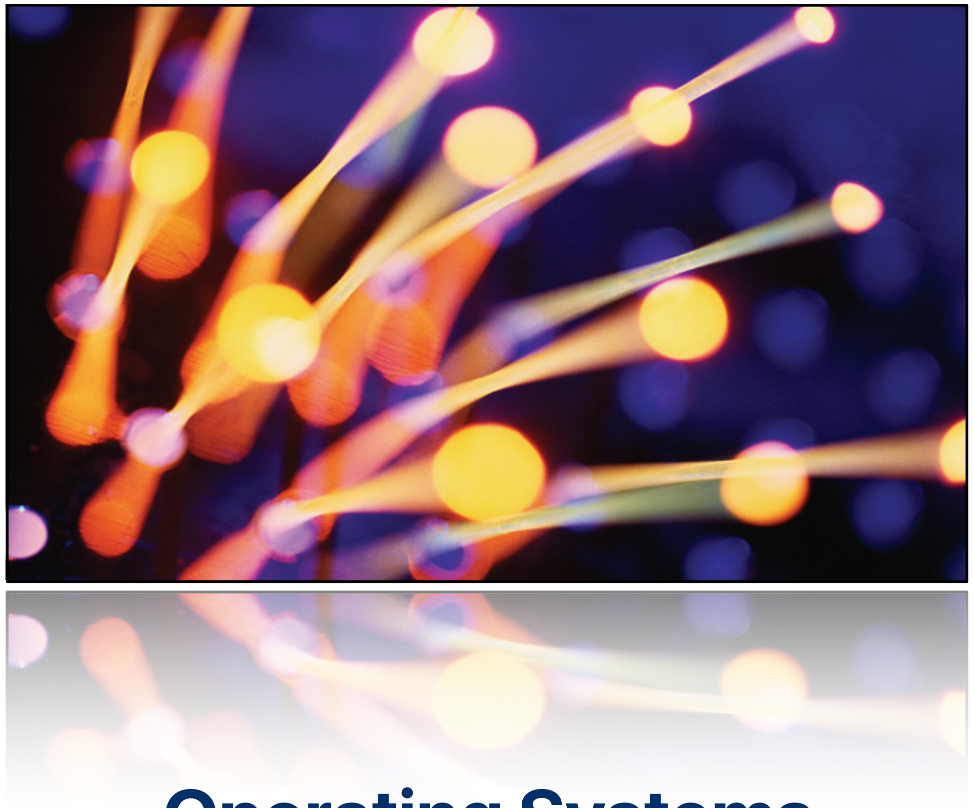
Free Space Management



Operating Systems
Baochun Li
University of Toronto

Another solution: use better algorithms to find available memory space

With variable-sized segments, what are the most suitable algorithms to manage free space?

The problem of free-space management

When using segmentation with variable-sized segments

When allocating memory on the heap in user space

- **malloc(size)** and **free(ptr)**

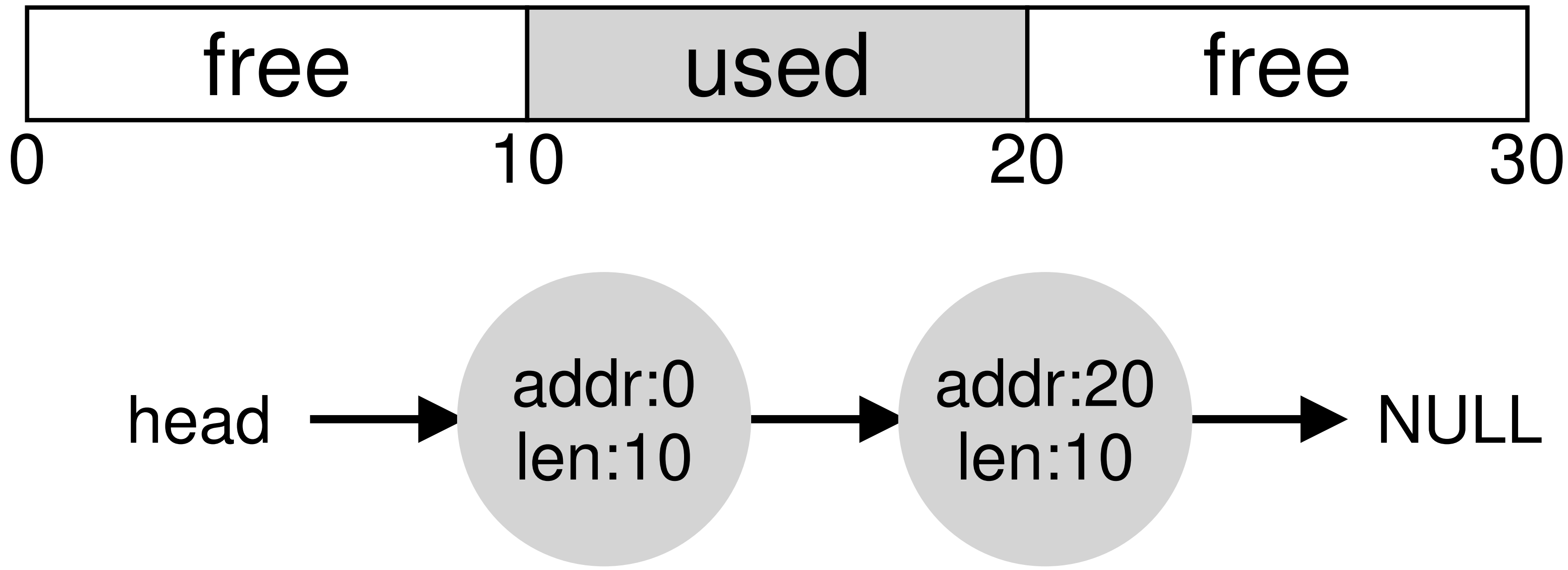
When allocating kernel memory

Objective: minimize external fragmentation

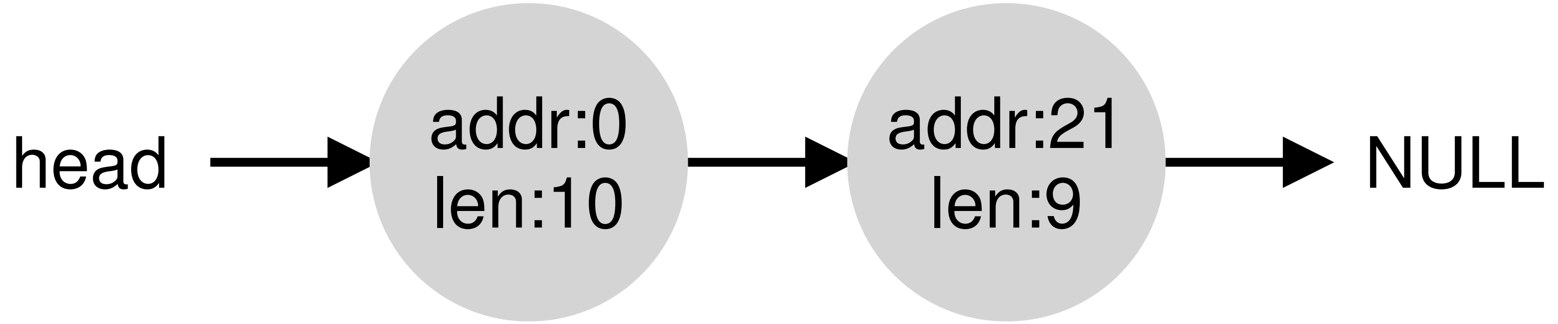
- may introduce internal fragmentation

Basic idea: Use a linked list to manage free chunks of memory

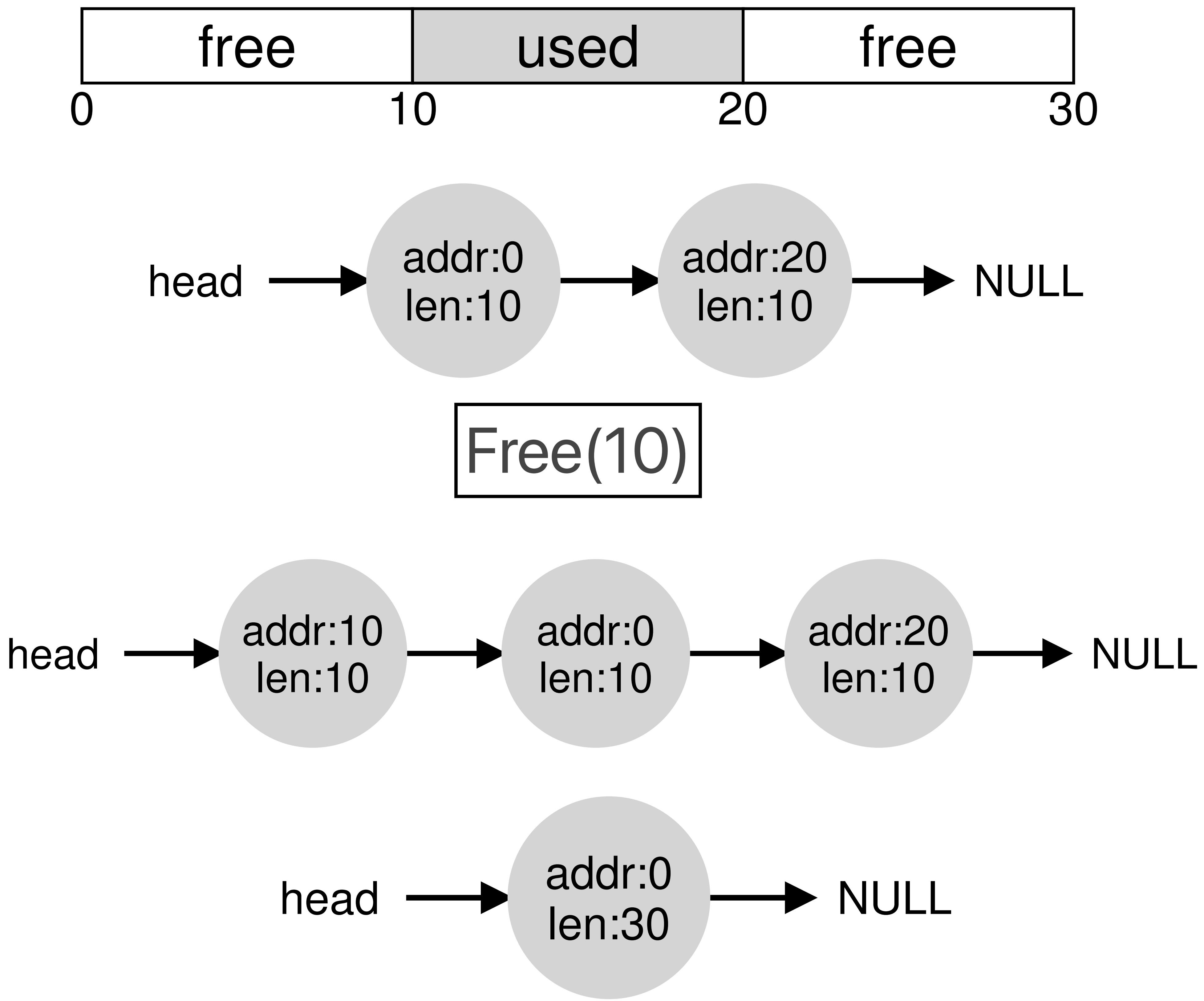
Low-level mechanism: Splitting



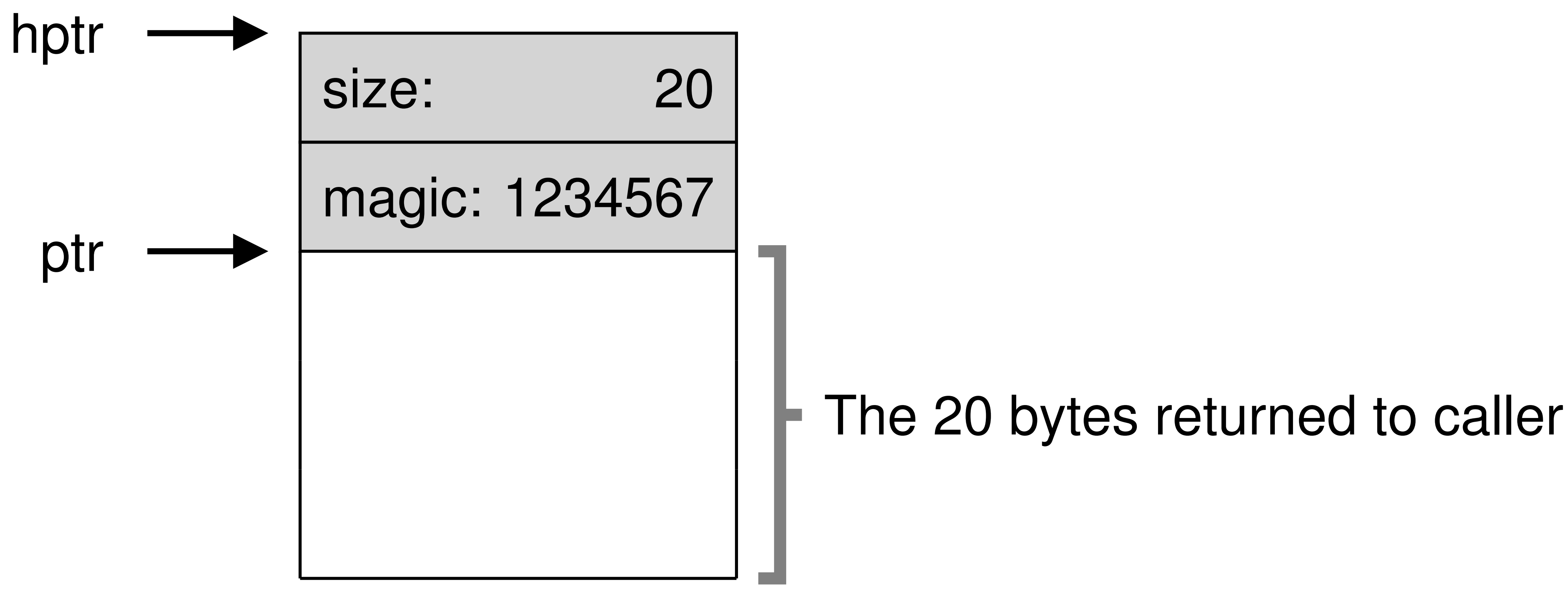
Request: one byte of memory



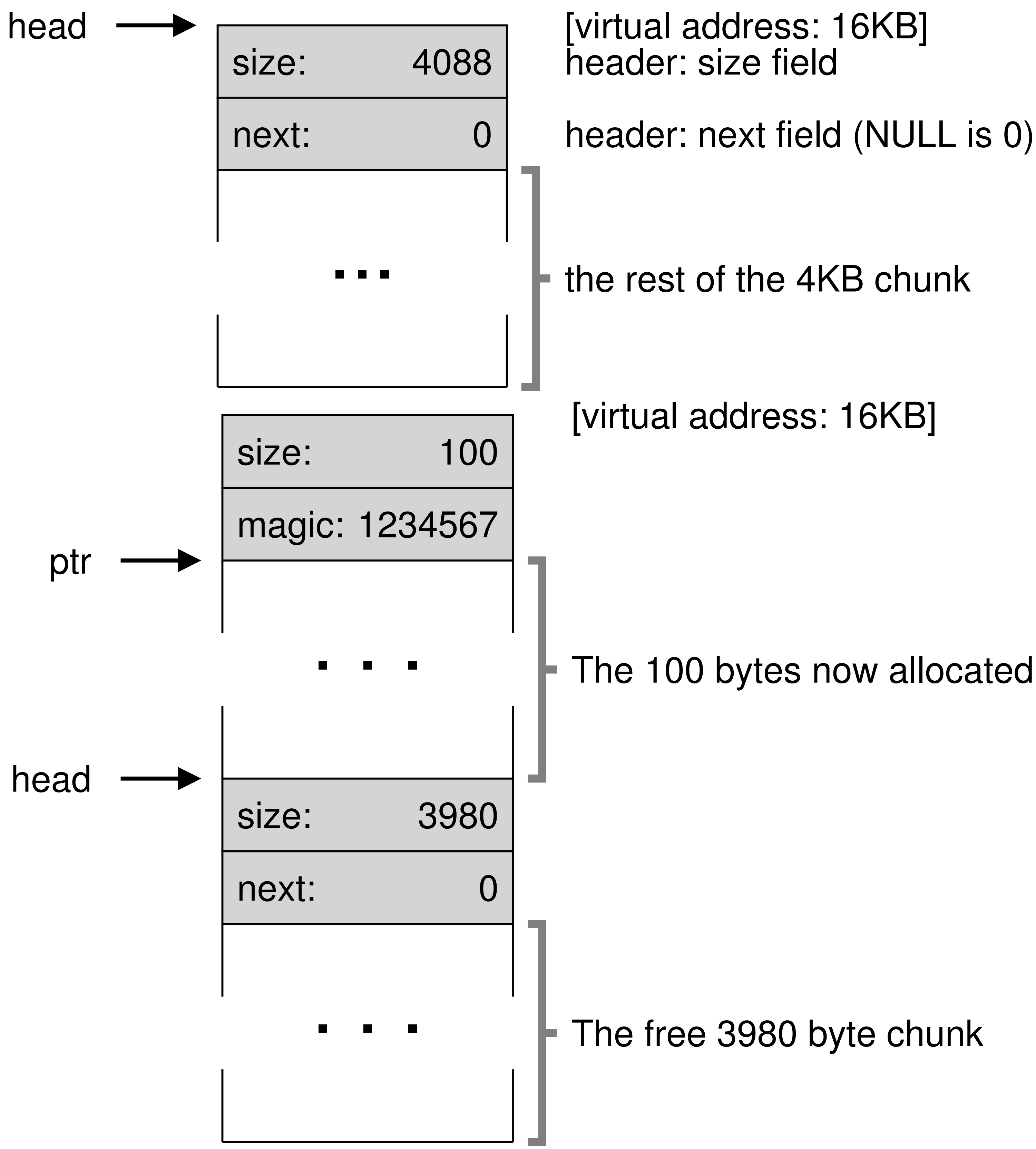
Low-level mechanism: Coalescing



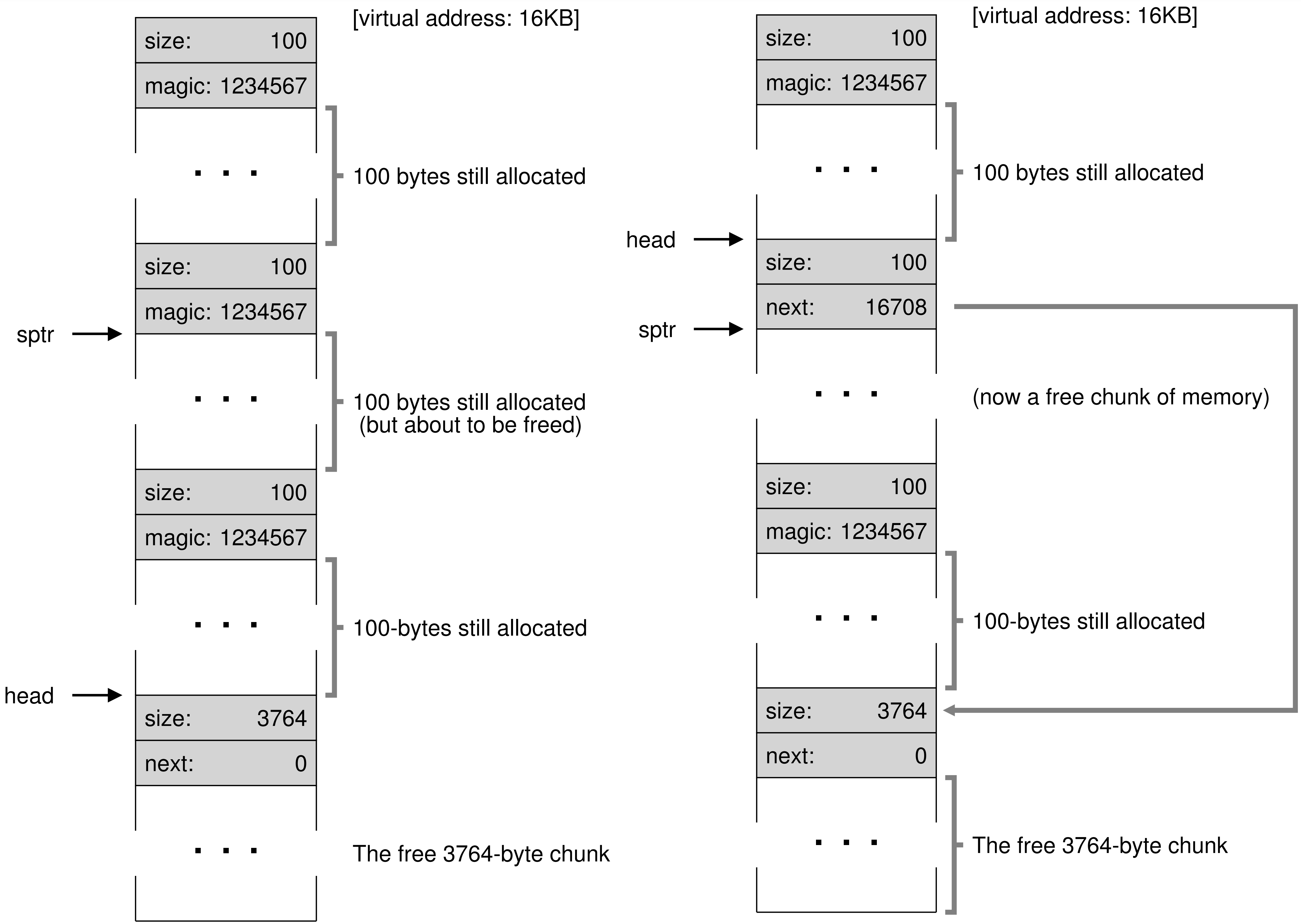
Heap allocator uses a header to store the length



The free list is embedded in the free space



Free(): From three chunks down to two



Basic strategies

Search the free list for a hole with size >= requested size

First Fit and Next Fit: Start from the beginning of the list or the current node

- Stop searching as soon as we find a free hole that's big enough

Best Fit: Find the smallest hole that will fit by searching the entire list

- Produces the smallest leftover hole — reduced wasted space

Worst Fit: Find the largest hole by searching the entire list

Simulations have shown: First Fit and Best Fit are better, but First Fit is simpler and faster

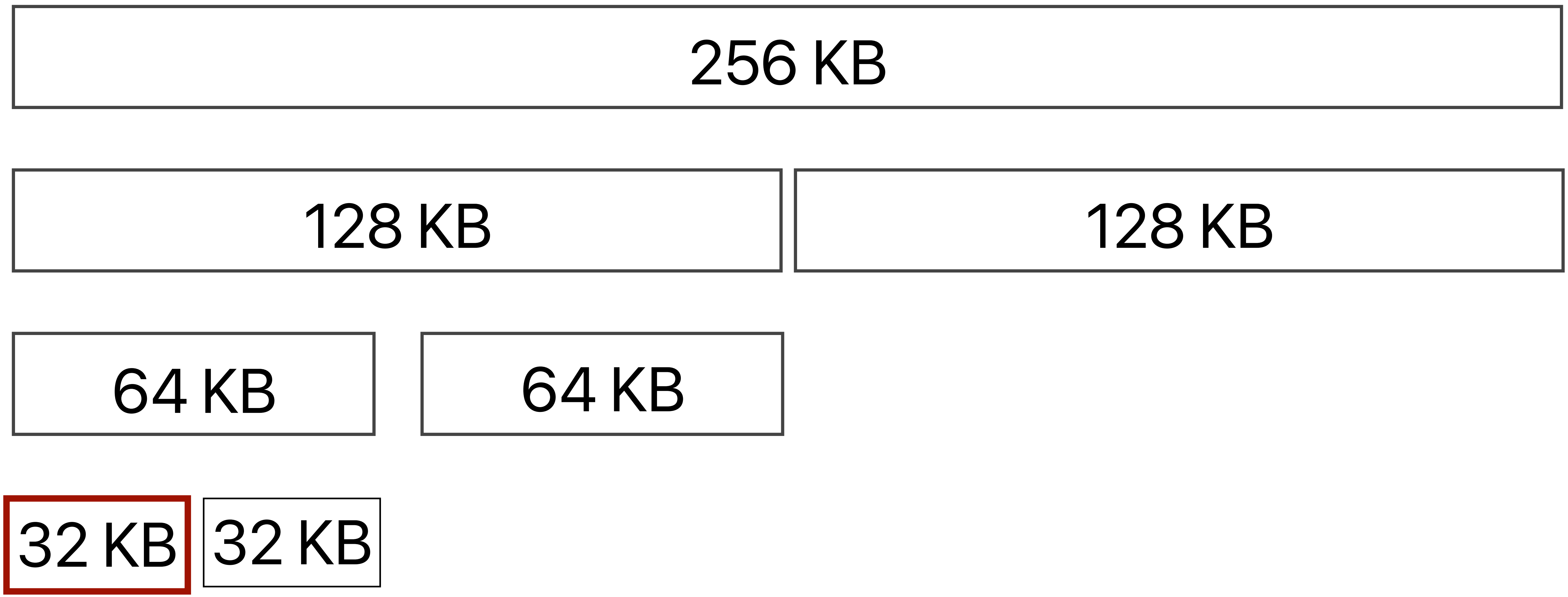
A new idea: Buddy Allocation

Buddy Memory Allocation: allocates sizes in powers of 2 (4 KB, 8 KB, 16 KB, etc.)

- requests in odd sizes are satisfied by rounding up to the next power of 2
- every time a request comes in, existing memory will be recursively divided into two "buddies" till the requested size is satisfied with the smallest "buddy"

Example: Request for 21 KB from 256 KB

contiguous physical memory partition



selected to satisfy
the request of 21 KB

Advantage of Buddy Memory Allocation

Coalescing —

When an allocated partition of memory is released, it can be easily coalesced (recursively, if needed) with adjacent free partitions to a partition doubling in size

In the example, ultimately we end up with the original 256 KB partition

Drawbacks of Buddy Memory Allocation

If we are unfortunate, there will be a large amount of space wasted within the partition

- This unused space within a partition is **internal fragmentation**
- A 33 KB request will need to be satisfied using a 64 KB partition

Any ideas that are even better?

Linux 2.0: Buddy memory allocation

Solaris 2.4 and Linux 2.2: Slab allocation

- Designed by Jeff Bonwick ("100x" engineer)
- Uses slabs to store kernel objects of precise sizes
- An object in a slab can be marked as **free** or **used**
- The slab allocator first attempts to satisfy the request with a free object in a *partial* slab
- If none exist, a free object is assigned from an *empty* slab
- If no empty slabs are available, a new slab is allocated from physical memory by a general allocator

Another problem with segmentation: a segment needs to fit into the physical memory

What we've covered so far

Three Easy Pieces

17 (Free Space Management)