Memory Management

Memory Partitioning

- The primary purpose of memory manegement is to bring processes into main memory so processor can execute them
- There are a number of solutions to handle the coming in and moving out of processes e.g. Virtual memory, paging, segmentation.
- A very simple technique for memory management can be partitioning.

A. Fixed Partitioning

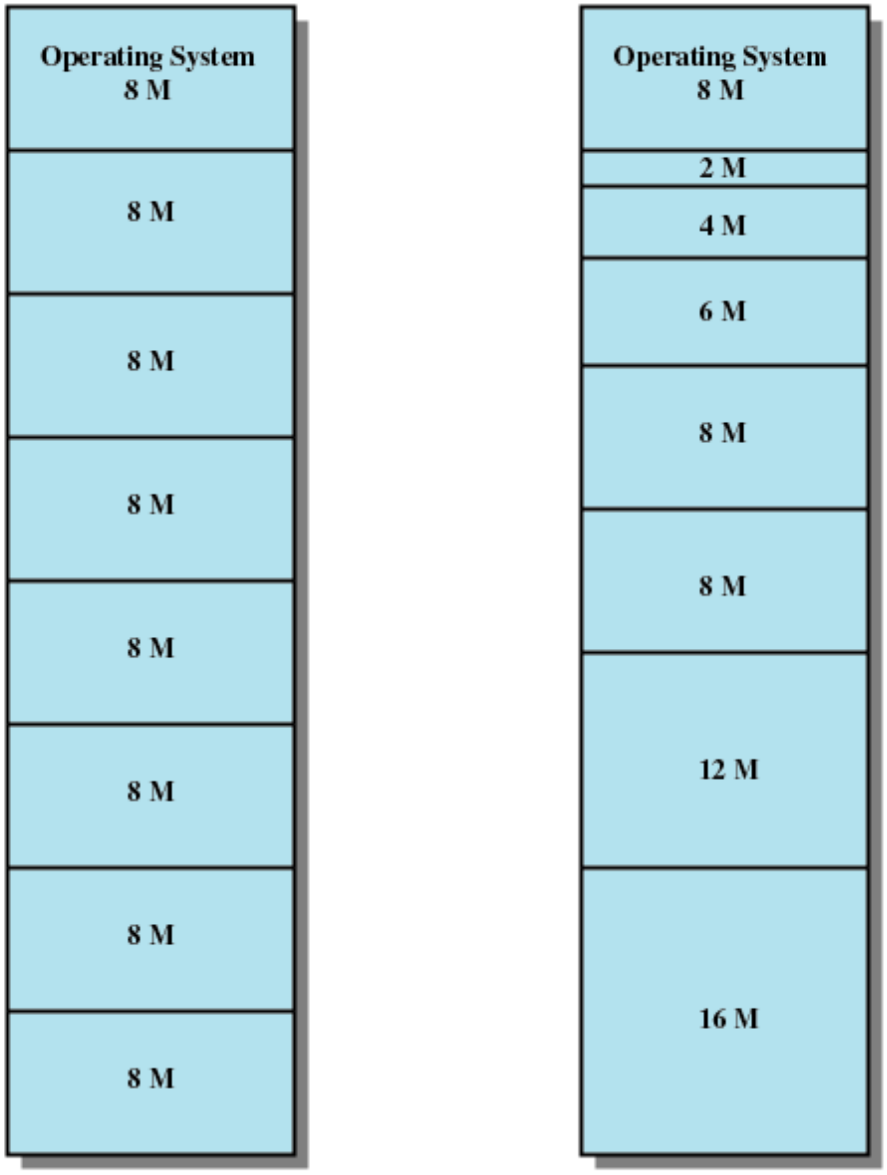
- In most schemes for memory management, we can assume that OS occupies some fixed portion of memory and the rest is available for processes
- The simplest scheme for using this available memory is to partition it into regions with fixed boundaries
- The partitions can be either equal or unequal-sized

A. Fixed Partitioning (cont..)

- Equal-size partitions
  - Any process whose size is less than or equal to the partition size can be loaded into an available partition
  - If all partitions are full, the operating system can swap a process out of a partition
  - A program may not fit in a partition. The programmer must design the program with overlays

A. Fixed Partitioning (cont..)

- Main memory use is inefficient. Any program, no matter how small, occupies an entire partition. This is called **internal fragmentation**.



(a) Equal-size partitions

(b) Unequal-size partitions

Figure 7.2 Example of Fixed Partitioning of a 64-Mbyte Memory

Placement Algorithm with Partitions

- **Equal-size partitions**
  - Because all partitions are of equal size, it does not matter which partition is used
- **Unequal-size partitions**
  - Can assign each process to the smallest partition within which it will fit
  - Processes are assigned in such a way as to minimize wasted memory within a partition

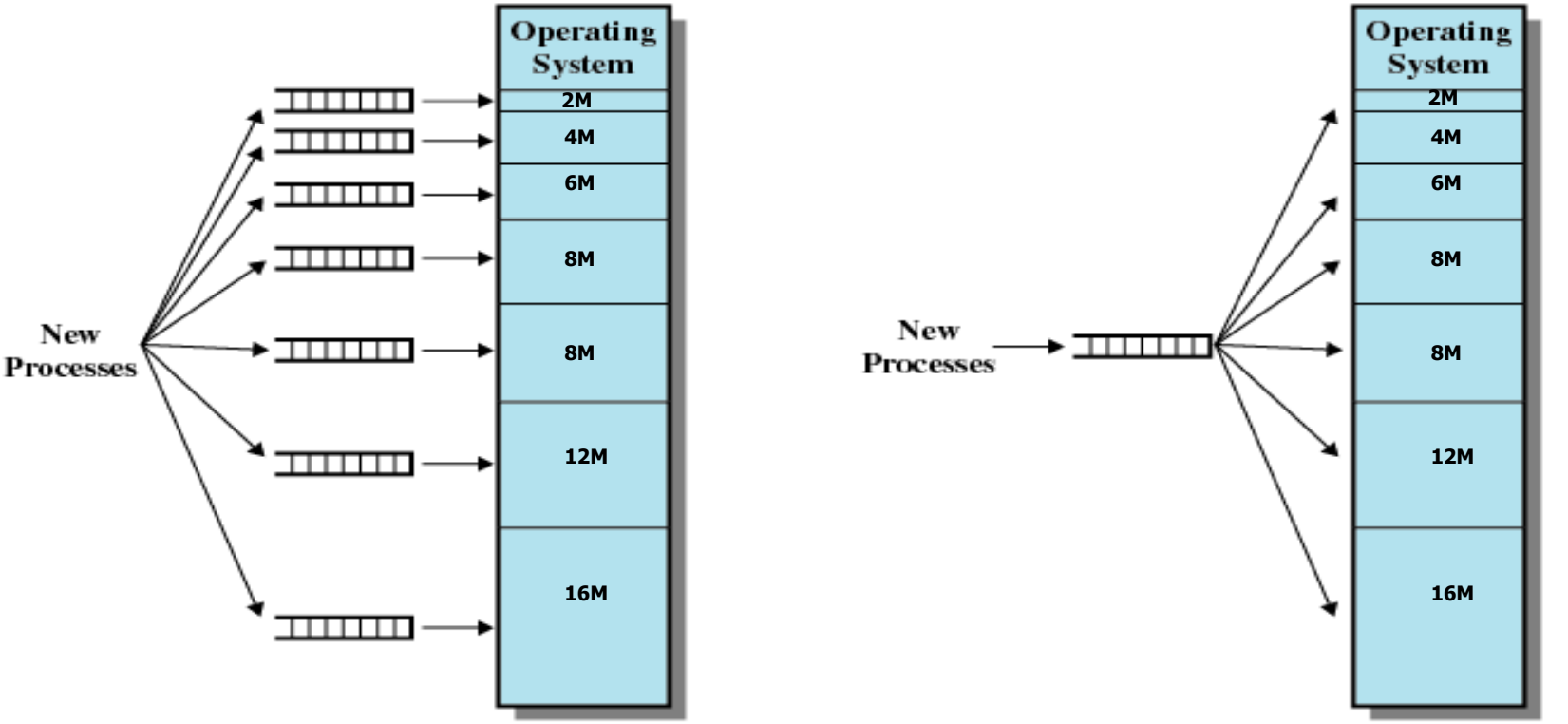


Figure 7.3 Memory Assignment for Fixed Partitioning

Multiple Queues

Advantage:

- Less Internal Fragmentation

Disadvantage:

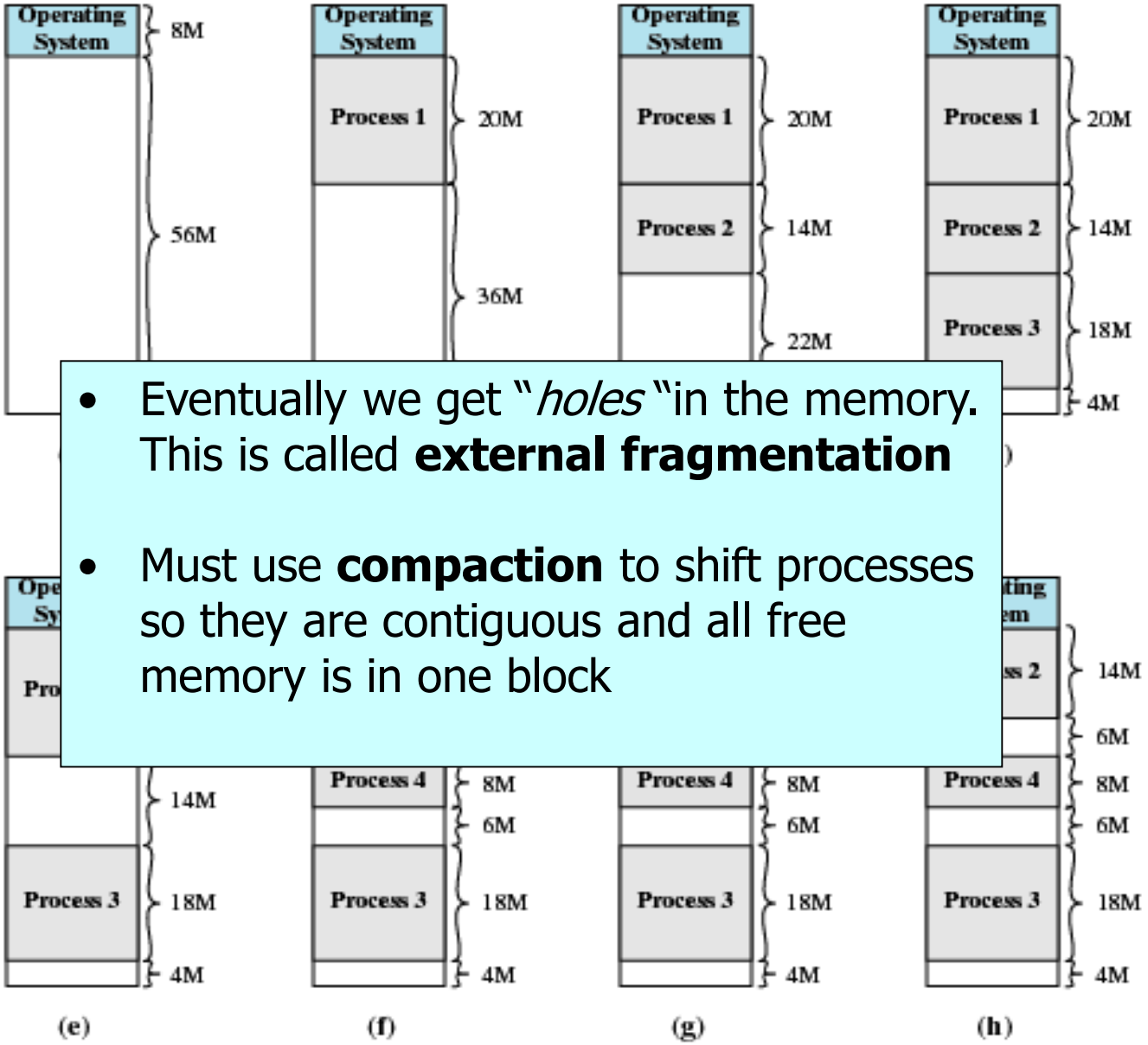
- Some 7M processes may be waiting in 8M Queue while 12M remains idle

A. Fixed Partitioning (cont..)

- Whether using equal or unequal fixed partitioning, these schemes have a lot of disadvantages:
  - The number of partitions specified at system generation time limits the number of active processes in the system
  - Because partition sizes are fixed without knowing the size of processes that will come, small processes will still waste a lot of memory space.

B. Dynamic Partitioning

- Partitions are of variable length and number
- Process is allocated exactly as much memory as required
- Eventually get holes in the memory. This is called external fragmentation
- Must use compaction to shift processes so they are contiguous and all free memory is in one block



- Eventually we get "*holes* "in the memory. This is called **external fragmentation**
- Must use **compaction** to shift processes so they are contiguous and all free memory is in one block

Figure 7.4 The Effect of Dynamic Partitioning

Dynamic Partitioning Placement Algorithm

- Operating system must decide which free block to allocate to a process
- **Best-fit algorithm**
  - Chooses the block that is closest in size to the request
  - Best performer overall
  - Since smallest block is found for process, the smallest amount of fragmentation is left
  - Memory compaction must be done more often

Dynamic Partitioning Placement Algorithm

First-fit algorithm

- Scans memory from the beginning and chooses the first available block that is large enough
- Fastest
- May have many process loaded in the front end of memory that must be searched over when trying to find a free block

Problems

- **It requires an expensive search of the entire free list to find the best hole.**
- **More importantly, it leads to the creation of lots of little holes that are not big enough to satisfy any requests. This situation is called *fragmentation*, and is a problem for all memory-management strategies, although it is particularly bad for best-fit.**

Dynamic Partitioning Placement Algorithm

- **Next-fit**
  - Scans memory from the location of the last placement
  - More often allocate a block of memory at the end of memory where the largest block is found
  - The largest block of memory is broken up into smaller blocks
  - Compaction is required to obtain a large block at the end of memory

Numerical

- Consider a swapping system in which memory consists of the following holes. Which holes will be taken for the following successive requests:
  - 12k
  - 10k
  - 9k

For Best-Fit, Worst-Fit and First-Fit.

10k
4k
20k
18k
7k
9k
12k
15k