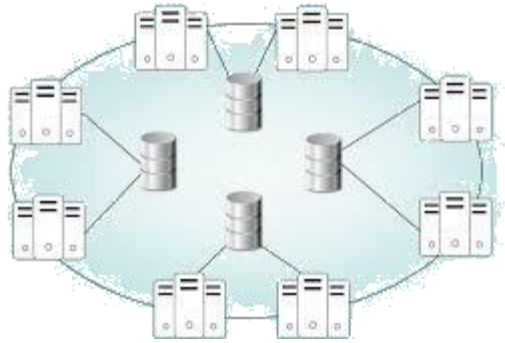
Distributed and Parallel Computer Systems

CSC 423

Spring 2021-2022

Lecture 11



Distributed Systems' Processes

INSTRUCTOR

DR / AYMAN SOLIMAN

➢ Contents

1) Design Issues for Threads Packages

2) Implementing a Threads Package

3) System Models

4) Allocation Models

❑ Design Issues for Processor Allocation Algorithms

- The major decisions the designers must make can be summed up in five issues:

1. Deterministic versus heuristic algorithms.
2. Centralized versus distributed algorithms.
3. Optimal versus suboptimal algorithms.
4. Local versus global algorithms.
5. Sender-initiated versus receiver-initiated algorithms.

❑ Deterministic and heuristic

- Deterministic algorithms are appropriate when everything about process behavior is known in advance.
- Imagine that you have a complete list of all processes, their computing requirements, their file requirements, their communication requirements, and so on.

❑ Deterministic and heuristic

- At the other extreme are systems where the load is completely unpredictable. Requests for work depend on who's doing what, and can change dramatically from hour to hour, or even from minute to minute.
- Processor allocation in such systems cannot be done in a deterministic, mathematical way, but of necessity uses ad hoc techniques called heuristics.

❑ Centralized and Decentralized

- Collecting all the information in one place allows a better decision to be made but is less robust and can put a heavy load on the central machine.
- Decentralized algorithms are usually preferable, but some centralized algorithms have been proposed for lack of suitable decentralized alternatives.

❑ Optimal and suboptimal

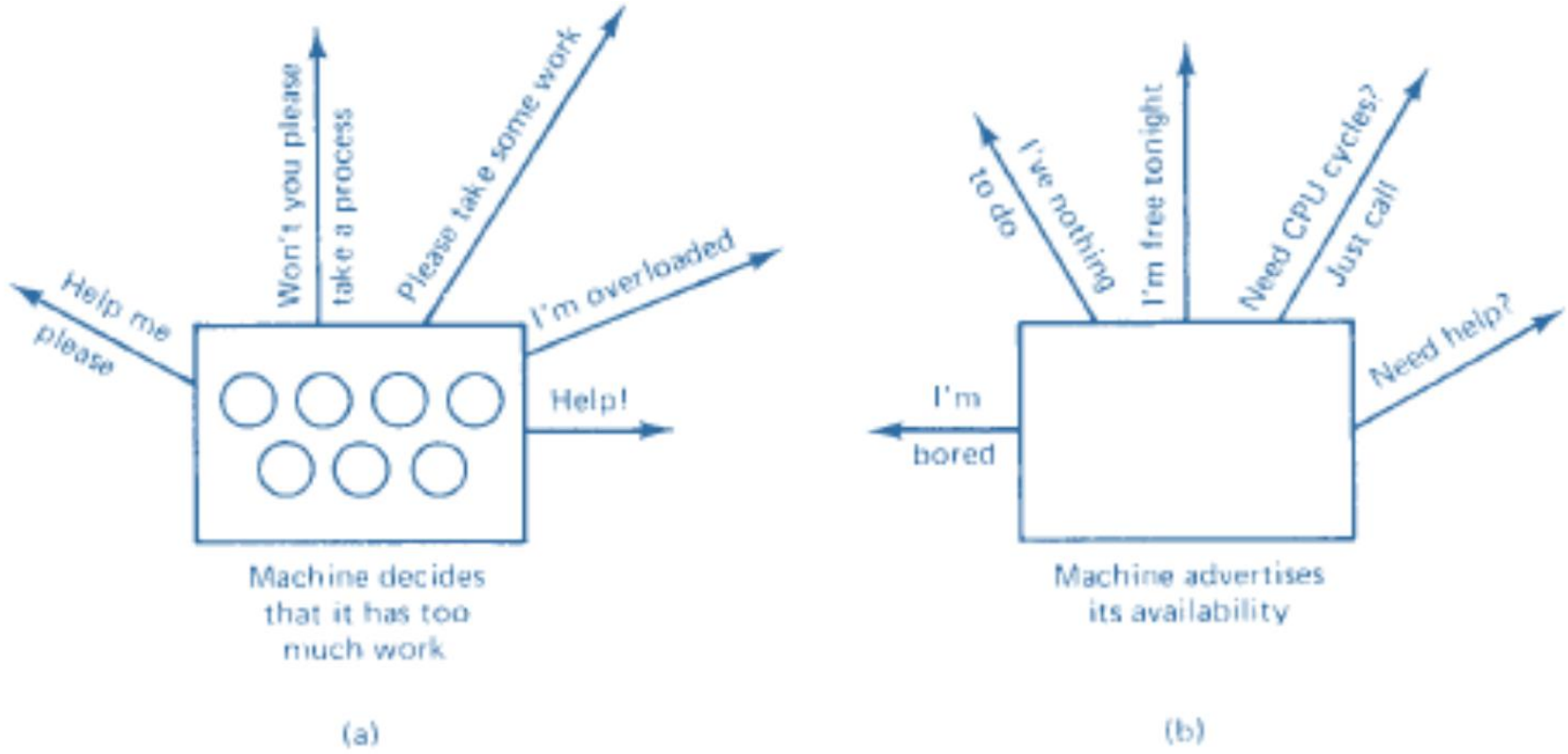
- Optimal solutions can be obtained in both centralized and decentralized systems but are invariably more expensive than suboptimal ones.
- In practice, most actual distributed systems settle for heuristic, distributed, suboptimal solutions because it is hard to obtain optimal ones.

❑ Local and global algorithms

- When a process is about to be created, a decision has to be made whether or not it can be run on the machine where it is being generated.
  - If that machine is too busy, the new process must be transferred somewhere else.

- The last issue in our list deals with location policy. Once the transfer policy has decided to get rid of a process, the location policy has to figure out where to send it. Clearly, the location policy cannot be local. It needs information about the load elsewhere to make an intelligent decision.

❑ Sender-initiated and receiver-initiated algorithms



(a) A sender looking for an idle machine
(b) A receiver looking for work to do.

❑ Implementation Issues for Processor Allocation Algorithms

- Virtually all the algorithms assume that machines know their own load, so they can tell if they are underloaded or overloaded and can tell other machines about their state.

- Measuring load is not as simple as it first appears. One approach is simply to count the number of processes on each machine and use that number as the load.

❑ Implementation Issues for Processor Allocation Algorithms

- The next step is to count only processes that are running or ready. After all, every running or runnable process puts some load on the machine,

  - even if it is a background process. However, many of these daemons wake up periodically, check to see if anything interesting has happened, and if not, go back to sleep. Most put only a small load on the system.

❑ Implementation Issues for Processor Allocation Algorithms

- A measurement is the fraction of time the CPU is busy.

- Clearly, a machine with a 20 percent CPU utilization is more heavily loaded than one with a 10 percent CPU utilization, whether it is running user or daemon programs.

❑ Processor Allocation Algorithms

- Whenever a new process is created, the creating machine checks to see if it is overloaded. If so, it seeks out a remote machine on which to start the new process. The three algorithms differ in how the candidate machine is located.

- Algorithm 1
  - picks a machine at random and just sends the new process there.
  - If the receiving machine itself is overloaded, it picks a random machine and sends the process off.
  - This process is repeated until either somebody is willing to take it, or a hop counter is exceeded

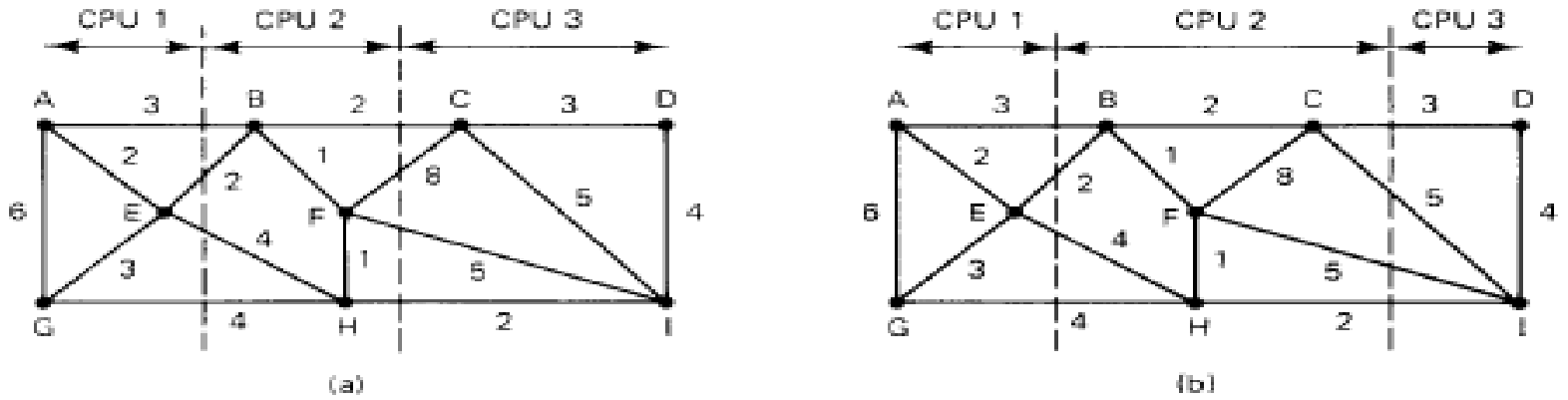
❑ Processor Allocation Algorithms

- Algorithm 2
  - picks a machine at random and sends it a probe asking if it is underloaded or overloaded.
  - If the machine admits to being underloaded
- Algorithm 3
  - probes k machines to determine their exact loads.
  - The process is then sent to the machine with the smallest load.

❑ Example Processor Allocation Algorithms

- Graph-Theoretic Deterministic Algorithm
  - If the number of CPUs, k, is smaller than the number of processes, several processes will have to be assigned to each CPU. The idea is to perform this assignment such as to minimize network traffic.
  - The system can be represented as a weighted graph
  - The goal is then to find the partitioning that minimizes the network traffic while meeting all the constraints.

❑ Example Processor Allocation Algorithms



- The total network traffic is the sum of the arcs intersected by the dotted cut lines, or 30 units. In Fig. (b) we have a different partitioning that has only 28 units of network traffic.
- Graph-theoretic algorithms of the kind we have just discussed are of limited applicability since they require complete information in advance

❑ Centralized Algorithm

- This algorithm is centralized in the sense that a coordinator maintains a usage table with one entry per personal workstation (i.e., per user), initially zero.
- When significant events happen, messages are sent to the coordinator to update the table.
- Allocation decisions are based on the table.
- it is concerned with giving each workstation owner a fair share of the computing power.

❑ Centralized Algorithm

- When a process is to be created, and the machine it is created on decides that the process should be run elsewhere

  - It asks the usage table coordinator to allocate it a processor.
  - If there is one available and no one else wants it, the request is granted.
  - If no processors are free, the request is temporarily denied, and a note is made of the request.

Thank you